# Leveraging Reinforcement Learning to Discover Algorithms for Computationally Efficient Hierarchical Planning

**Anonymous Author(s)**
Affiliation
Address
`email`

## Abstract

While many advances in artificial intelligence were enabled by Moore's law, a distinctive advantage of human intelligence is people's ability to use their limited computational resources extremely efficiently. To close this gap, research on metareasoning strives to equip machines with the ability to select computations which lead to the greatest increase in decision quality in the shortest amount of time. Reinforcement learning can be leveraged to learn reusable policies for selecting computations. In principle, this approach could be used to automate the process of algorithm discovery. But existing metalevel reinforcement learning methods are not sufficiently scalable. This is especially true for planning algorithms for large sequential decision problems with long temporal horizons. To develop a more scalable metalevel reinforcement learning method for discovering computationally efficient planning algorithms, we draw inspiration from cognitive science to decompose long-term planning into goal-setting and planning how to achieve the chosen goal. This makes it possible to efficiently learn and compose algorithms for both sub-problems. We leverage this approach to make two state-of-the-art strategy discovery methods more scalable so that they can be applied to a wider range of more complex problems. As a proof of concept, we show that the automatically discovered algorithms outperform existing planning algorithms across four increasingly more challenging sequential decision problems. Furthermore, we show that our hierarchical decomposition of the planning problem significantly increases the scalability of several state-of-the-art methods for automatic algorithm discovery without compromising their performance. These results suggest that future extensions of our cognitively-informed approach might make it possible to leverage reinforcement learning to discover new planning algorithms with human-level computational efficiency.

## 1 Introduction

Artificial Intelligence (AI) has seen tremendous growth in the last decade. Reinforcement learning (RL), in particular, has attracted a lot of interest in the scientific community after an RL agent reached human-level performance in playing Atari games [1] and AlphaGo [2] defeated the world champion in the game of Go. While RL agents have been able to perform well in specific tasks, they fail to adapt to small perturbations of the environment [3, 4] and are not able to generalize the skill learnt to different environments [5, 6]. Human learning, by contrast, generalizes more effectively to tasks both within and between domains [7]. The generalization problems of contemporary AI may partly stem from focusing on "what should the agent do?" (reasoning) rather than on "how should the agent decide what to do?" (metareasoning) [8]. This is understandable given that metareasoning is a harder computational problem than reasoning itself [9]. However, recent progress suggests it is possible to learn good policies for selecting computations that agents can reuse to efficiently reason about many different (decision) problems [10, 11]. Additionally, by taking the cost of performing computations into account, the discovered policies have to balance the benefit of gaining information about the environment with the cost associated with performing these computations. As a consequence, this approach can be used to automate the process of discovering computationally efficient algorithms.

Planning is integral for intelligent agents and has been a fundamental research topic in the field of problem solving and AI [12]. AI has yet to catch up to computational efficiency of human planning. Recent work in cognitive science has found that people's planning operations closely resemble the sequence of planning operations one would arrive at by optimal metareasoning about the resulting improvement in decision quality versus the cost of computation [10, 13, 14]. Unfortunately, the methods for discovering planning algorithms developed so far [10, 15] are not scalable to large environments because their run time grows exponentially with the size of the planning problem.

Research in cognitive science and neuroscience suggests that the brain decomposes long-term planning into goal-setting and planning at multiple hierarchically-nested timescales [16, 17]. Furthermore, Solway et al. [18] found that human learners spontaneously discover optimal action hierarchies. Inspired by these findings, we extend the near-optimal strategy discovery method proposed in Callaway et al. [10] by incorporating hierarchical structure into the space of possible planning strategies. Concretely, the planning task is decomposed into first selecting one of the possible final destinations as a goal solely based on its own value and then planning the path to this selected goal. We find that imposing hierarchical structure makes automatic strategy discovery methods significantly less computationally expensive than non-hierarchical methods without compromising the performance of the discovered algorithms. We prove that our hierarchical decomposition leads to a reduction in the computational complexity of the strategy discovery problem which manifests in substantial improvements even for small problem instances and makes it possible to scale up automatic algorithm discovery to planning problems that are too large for previous algorithm discovery methods.

The plan for this article is as follows: We start by introducing extant strategy discovery methods and the problem domain utilised throughout the paper. We, then, present a new reinforcement learning method for discovering hierarchical planning strategies. Next, we define a series of increasingly more challenging benchmark problems for automatic strategy discovery and use them to evaluate our method's performance and scalability against the state of the art. We close by discussing the implications of our findings for the development of more intelligent agents, understanding human planning [13], and improving human decision-making [19].

## 2 Background and related work

### 2.1 Discovering optimal planning strategies by solving metalevel MDPs

Callaway et al. [20] developed a method to automatically discover optimal planning strategies by modeling the optimal planning strategy as a solution to a metalevel Markov Decision Process (metalevel MDP). In general, a metalevel MDP $M = (\mathcal{B}, \mathcal{C}, T, r)$ is defined as an undiscounted MDP where $b \in \mathcal{B}$ represents the belief state, $T(b, c, b')$ is the probability of transitioning from belief state $b$ to belief state $b'$ by performing computation $c \in \mathcal{C}$, and $r(b, c)$ is a reward function that describes the costs and benefits of computation [21]. It is important to note that the actions in a metalevel MDP are computations which are different from object-level actions – the former are planning operations and the latter are physical actions that move the agent through the environment.

### 2.2 Methods for solving meta-level MDPs

In their seminal paper, Russell and Wefald [8] introduced the theory of rational metareasoning. In [22], they define the value of computation $\mathrm{VOC}(c, b)$ to be the expected improvement in decision quality achieved by performing computation $c$ in belief state $b$ and continuing optimally, minus the cost of computation $c$. Using this formalization, the optimal planning strategy is a selection of computations which maximises the value of computation (VOC), that is

$$\pi^*_{\mathrm{meta}} = \arg\max_c \mathrm{VOC}(c, b), \tag{1}$$

When the VOC is non-positive for all available computations, the policy terminates ($c = \perp$) and executes the object-level action that is best according to the current belief state. Hence, $\mathrm{VOC}(\perp, b) = 0$. In general, the VOC is computationally intractable but it can be approximated [10].Lin et al. [9] estimated VOC by the myopic value of computation ($\mathrm{VOI}_1$) which is the expected improvement in decision quality that would be attained by terminating deliberation immediately after the computation has been performed. Hay et al. [21] approximated rational metareasoning by solving multiple smaller metalevel MDPs that each define the problem of deciding between one object-level action and its best alternative. Callaway et al. [10] approximated the value of information by interpolating between the myopic VOI and the value of perfect information, that is

$$\widehat{\mathrm{VOC}}(c, b; \mathbf{w}) = w_1 \cdot \mathrm{VOI}_1(c, b) + w_2 \cdot \mathrm{VPI}(b) + w_3 \cdot \mathrm{VPI}_{\mathrm{sub}}(c, b) - w_4 \cdot \mathrm{cost}(c), \tag{2}$$

where $\mathrm{VPI}(b)$ denotes the value of perfect information, $\mathrm{VPI}_{\mathrm{sub}}(c, b)$ measures the benefit of having full information about the subset of parameters that the computation reasons about (e.g., the values of all paths that pass through the node evaluated by the computation), and $\mathrm{cost}(c)$ is the cost of the computation $c$.

### 2.3 The Mouselab-MDP paradigm

Previous methods for discovering near-optimal decision strategies [23, 20, 10] have been developed for and evaluated in a planning task known as the Mouselab-MDP paradigm [24]. The Mouselab-MDP paradigm was developed to make people's planning operations observable. This is achieved by recording which states people attend to in which order. More importantly for this article, we can draw on Callaway et al. [20]'s formalization of the problem of discovering optimal planning strategies for the Mouselab paradigm as a metalevel MDP. An example of a Mouselab-MDP environment used extensively in this paper is shown in Figure 1. It can be formalized by the notion of a directed acyclic graph (DAG), where each node is associated with a reward sampled from a probability distribution and each edge represents a direction of transition from one node to another. In this study, we model the reward structure of each node as a normal distribution centered at 0 with variance that increases proportionally with the graph depth[1]. The goal nodes are the terminal nodes in the DAG (represented in dark red and green diamonds in Figure 1). Therefore, the goal nodes have a higher variance when compared to the intermediate nodes, which is analogous to greater uncertainty while planning further in time. Initially, the agent starts at the root node and the reward of each node is concealed. The agent plans by performing computations that uncover node values (trading computational resources for information). At any time, the agent can stop planning and start navigating itself in the environment based on its belief state. The agent's objective is to reach a goal node by using a planning strategy that maximizes the net reward on the path to it.

Following the strategy discovery method by Lieder et al. [23], we model finding optimal planning strategy as a solution to a metalevel MDP

$$M = (\mathcal{B}, \mathcal{C}, T, R) \tag{3}$$

where each belief state $b \in \mathcal{B}$ encodes a normal distribution over the agent's belief of the reward for each node. The possible computations are $\mathcal{C} = \{\xi_1, ..., \xi_M, c_{1,1}, ..., c_{M,N}, \perp\}$, where $c_{g,n}$ reveals the reward at intermediate node $n$ in goal set $g$, $\xi_g$ reveals the reward at the goal node of goal set $h_g \in \mathcal{H}$. Furthermore, a goal set $h_g \in \mathcal{H}$ refers to all nodes which lie on all paths leading to a goal $g$ (including the goal node itself), and $\perp$ is the termination meta-action which stops planning and triggers execution of the planned strategy in which the agent selects a path to a goal with the highest expected sum of rewards according to the current belief state. The trade-off between the cost and the value of collecting information is necessary to establish a non-trivial solution.

## 3 Discovering hierarchical planning strategies

All existing strategy discovery algorithms evaluate and compare the utilities of all possible computations in each step. As a consequence these methods do not scale well to problems with large state spaces and long planning horizons. This is especially true of the BMPS algorithm whose run time is exponential in the number of nodes of the planning problem [10]. People, by contrast, would not even consider making detailed low-level motor plans for navigating a specific distant location (e.g.,

[1]Depth: longest path length starting from the root node.

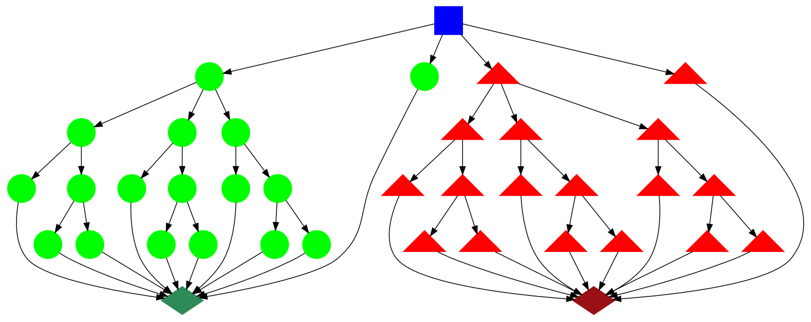

Figure 1: Mouselab-MDP environment with 2 goals. Nodes accompanied to each goal are denoted in green (circles) and red (triangles), respectively. The goal nodes have darker shades of green and red (diamonds), and the root node's color is blue (square).

Terminal C of San Juan Airport) until they arrive at a high-level plan that leads them to or through that location [25]. To efficiently plan over long horizons, people [17, 16, 25] and hierarchical planning algorithms [26, 27, 28, 29] decompose the problem into first setting goals and then planning how to achieve them . This two-stage process breaks large planning problems down to smaller problems that are easier to solve.

To discover hierarchical planning strategies automatically, our proposed strategy discovery algorithm therefore decomposes the problem of discovering planning strategies into the sub-problems of discovering a strategy for selecting a goal and discovering a strategy for planning the path to the chosen goal. Formally, this is accomplished by decomposing the metalevel MDP defining the strategy discovery problem (Equation 3) into two metalevel MDPs with smaller state and action spaces. Constructing metalevel MDPs for goal-setting and path planning is easy when there is a small set of candidate goals. Such candidate goals can often be identified based on prior knowledge or the structure of the domain [30, 18]. The metalevel MDP for goal-setting process (Section 3.1) only includes computations for estimating the values of a small set of candidate goal states $(V(g_1), \cdots, V(g_M))$. This means that goals are chosen without considering how costly it would be to achieve them. This makes sense when all goals are known to be achievable and the differences between the values of alternative goal states are substantially larger than the differences between the costs of reaching them. This is arguably true for many challenges people face in real life. For instance, when a high school student plans one's career, the difference between the long-term values of studying computer science versus becoming a janitor is likely much larger than the difference between the costs of achieving either goal. This is to be expected when the time it will take to achieve the goals is short relative to the agent's lifetime. The goal-achievement MDP (Section 3.2) only includes computations that update the estimated costs of alternative paths to the chosen goal by determining the costs or rewards of state-action pairs $(r(b, c))$ that lie on those paths. Decomposing the strategy discovery problem into these two components reduces the number of possible computations that the metareasoning method has to choose between from $M \cdot N$ to $M + N$, where $M$ is the number of possible final destinations (goals) and $N$ is the number of steps to the chosen goal. Perhaps the most-promising metareasoning method for automatic strategy discovery is the Bayesian Metalevel Policy Search algorithm (BMPS; [10, 15]). To solve the two types of metalevel MDPs introduced below more effectively, we also introduce an improvement of the BMPS algorithm in Section 3.3.

## 3.1 Goal-setting metalevel MDP

The optimal strategy for setting the goal can be formalized as the solution to the metalevel MDP $M_{\mathrm{H}} = (\mathcal{B}_{\mathrm{H}}, \mathcal{C}_{\mathrm{H}}, T_{\mathrm{H}}, R_{\mathrm{H}})$, where the belief state $b_{\mathrm{H}}(g) \in \mathcal{B}_{\mathrm{H}}$ denotes the expected cumulative reward that the agent can attain starting from the goal state $g \in \mathcal{G}$. The high level computations are $\mathcal{C}_{\mathrm{H}} = \{\xi_1, ..., \xi_M, \perp_{\mathrm{H}}\}$ where $\xi_g$ reveals the value $V(g)$ of the goal node $g$ of goal set $h_g \in \mathcal{H}$. $\perp_{\mathrm{H}}$ terminates the high-level planning leading to the agent to select the goal with the highest value according to its current belief state. The high-level metalevel reward function is $r_{\mathrm{H}}(b_{\mathrm{H}}, c) = -\lambda_{\mathrm{H}}$

for $c_{\mathrm{H}} \in \{\xi_1, ...\xi_M\}$ and

$$R_{\mathrm{H}}(b_{\mathrm{H}}, \perp_{\mathrm{H}}) = \max_{k \in \mathcal{G}} \mathbb{E}[b_{\mathrm{H}}(k)], \tag{4}$$

## 3.2 Goal-achievement metalevel MDP

Having set a goal $g \in \mathcal{G}$, the agent has to find the optimal planning strategy to achieve the goal. This planning strategy is formalized as the solution to the metalevel MDP $M_{\mathrm{L}} = (\mathcal{B}_{\mathrm{L}}, \mathcal{C}_{\mathrm{L}}, T_{\mathrm{L}}, R_{\mathrm{L}})$. The belief state $b \in \mathcal{B}_{\mathrm{L}}$ denotes the expected reward for each node. The agent can only perform a subset of meta-actions $\mathcal{C}_{g,\mathrm{L}} = \{c_{g,1}, ..., c_{g,N}, \perp_{\mathrm{L}}\}$, where $c_{g,n}$ reveals the reward at node $n$ in the goal set $h_g$. A goal set $h_g \in \mathcal{H}$ refers to all nodes, including the goal node, which lie on paths leading to goal $g \in \mathcal{G}$. $\perp_{\mathrm{L}}$ terminates planning and leads to the agent to select the path with the highest expected sum of rewards according to the current belief state. The low-level metalevel reward function is $r_{\mathrm{L}}(b, c_g) = -\lambda_{\mathrm{L}}$ for $c_g \in \{c_{g,1}, ..., c_{g,N}\}$ and

$$R_{\mathrm{L}}(b, \perp_{\mathrm{L}}) = \max_{p \in \mathcal{P}} \sum_{n \in p} \mathbb{E}[b_n], \tag{5}$$

where $\mathcal{P}$ is the set of all paths $p$ and $b_n$ is the belief of the reward for node $n$.

## 3.3 Hierarchical Bayesian Metalevel Policy Search

Building on the work by Callaway et al. [10], we extend the BMPS algorithm by introducing hierarchical structure as discussed above. BMPS approximates the value of computation (VOC) as per Equation 2 and the features used for this approximation for each level are discussed below.

The VOC for the high level policy is approximated using three features: (1) the myopic utility for performing a goal state evaluation ($\mathrm{VOI}_1^{\mathrm{H}}$), (2) the value of perfect information about all goals ($\mathrm{VPI}^{\mathrm{H}}$), and (3) the cost of the respective computation ($\mathrm{cost}_{\mathrm{H}}$).

$$\widehat{\mathrm{VOC}}_{\mathrm{H}}(c_{\mathrm{H}}, b_{\mathrm{H}}; \mathbf{w}^{\mathrm{H}}) = w_1^{\mathrm{H}} \cdot \mathrm{VOI}_1^{\mathrm{H}}(c_{\mathrm{H}}, b_{\mathrm{H}}) + w_2^{\mathrm{H}} \cdot \mathrm{VPI}^{\mathrm{H}}(b_{\mathrm{H}}) - w_3^{\mathrm{H}} \cdot \mathrm{cost}_{\mathrm{H}}(c_{\mathrm{H}}), \tag{6}$$

where $w_1^{\mathrm{H}}, w_2^{\mathrm{H}}$ are constrained to a standard simplex set, $w_3^{\mathrm{H}} \in \mathbb{R}_{[1,M]}$, and $M$ is the number of goals.

In a similar manner as for the high-level policy, the value of computation for the low-level policy is approximated by using a mixture of VOI features and the anticipated cost of the current computation and future computations, that is:

$$\begin{aligned} \widehat{\mathrm{VOC}}_{\mathrm{L}}(c, b, g; \mathbf{w}^{\mathrm{L}}) = {} & w_1^{\mathrm{L}} \cdot \mathrm{VOI}_1^{\mathrm{L}}(c, b, g) + w_2^{\mathrm{L}} \cdot \mathrm{VPI}^{\mathrm{L}}(b, g) \\ & + w_3^{\mathrm{L}} \cdot \mathrm{VPI}_{\mathrm{sub}}^{\mathrm{L}}(c, b, g) - w_4^{\mathrm{L}} \cdot \mathrm{cost}_{\mathrm{L}}(c, b, g, \mathbf{w}^{\mathrm{L}}) \end{aligned} \tag{7}$$

where $w_i^{\mathrm{L}}$ $(i = 1, 2, 3)$ are constrained to a standard simplex set, $w_4^{\mathrm{L}} \in \mathbb{R}_{[1,|h_g|]}$, and $|h_g|$ is the number of nodes in goal set $h_g$. The weight values for both levels are optimised in 100 iterations with Bayesian Optimization [31] using the GPyOpt library [32].

The cost feature of the original BMPS algorithm introduced by Callaway et al. [10] only considered the cost of a single computation whereas its VOI features consider the benefits of performing a sequence of computations. As a consequence, policies learned with the original version of BMPS are biased towards inspecting nodes that many paths converge on – even when the values of those nodes are irrelevant. To rectify this problem, we redefine the cost feature so that it considers the costs of all computations assumed by the VOI features. Concretely, to compute the low-level policy, we define the cost feature of BMPS as the weighted average of the costs of generating the information assumed by the VOI features $\mathcal{F} = \{\mathrm{VOI}_1^{\mathrm{L}}, \mathrm{VPI}^{\mathrm{L}}, \mathrm{VPI}_{\mathrm{sub}}^{\mathrm{L}}\}$, that is

$$\mathrm{cost}_{\mathrm{L}}(c, \mathbf{w}^{\mathrm{L}}) = \sum_{f \in \mathcal{F}} w_f^{\mathrm{L}} \cdot \sum_{n}^{|h_g|} \mathbb{I}(c, f, n) \cdot \mathrm{cost}(c) \tag{8}$$

where $\mathbb{I}(c, f, n)$ returns 1 if node $n$ is relevant when computing feature $f$ for computation $c$ and 0 otherwise.

## 4 Benchmarks and baselines

To benchmark our algorithms for hierarchical strategy discovery, we evaluated the performance of various algorithms in four increasing challenging environment structures with 2-5 candidate goals. Each goal set consisted of 17 intermediate nodes, forming 10 possible paths that reach the goal state in maximum 5 steps (see Figure 1). The reward for each node was sampled from a normal distribution centered at 0. The distribution variance of non-goal nodes was 5 for nodes reachable within a single step (level 1) and doubled from each level to the next. The distribution variance from which the reward associated with the goal node was sampled starts from 100 and increases by 20 for every additional goal node. The cost for a computation on both levels was set to 1.

The benchmarks were selected carefully to first compare performance between strategy discovery algorithms and established search algorithms. To estimate an upper bound on the performance of existing planning algorithms on our benchmark problems, we selected Backward Search and Bidirectional Search [33] because – unlike most planning algorithms – they start by considering potential final destinations which is optimal for planning in our benchmark problems (see Results). These search algorithms terminate when they find a path whose expected return exceeds a threshold, called its aspiration value. To obtain an upper bound on the performance of those planning algorithms, we determined their aspiration values by applying Bayesian Optimisation to maximise their estimated net returns. We also evaluated the performance of a random-search algorithm, which chooses uniformly at random from the set of metalevel operations that have not been performed yet.

In addition to those planning algorithms, our baselines also include the state-of-the-art methods for automatic strategy discovery: the greedy myopic VOC strategy discovery algorithm [9], which approximates the VOC by its myopic utility ($\mathrm{VOI}_1$), BMPS [10], and the Adaptive Metareasoning Policy Search algorithm (AMPS) [34] which uses approximate metareasoning to decide when to terminate planning. Our implementation of the AMPS algorithm uses a deep Q-network to estimate the value of stopping versus continuing to plan from the expected termination reward of the best path. The planning operations are selected by maximizing the myopic value of information ($\mathrm{VOI}_1$). Finally, to illustrate the versatility of our hierarchical problem decomposition, we also applied it to the the greedy myopic VOC strategy discovery algorithm.

## 5 Results

We compared the performance of the strategies discovered by hierarchical BMPS and the hierarchical greedy myopic VOC method against the performance of the strategies discovered by the two state-of-the-art methods and two standard planning algorithms on the benchmark problems described above. We measured each strategy's performance by its net returns[2] and we additionally compared the strategy discovery methods in terms of their runtimes and scalability. As shown in Figure 2a and Table 1, the algorithms discovered by our new hierarchical strategy discovery methods outperform extant planning algorithms and the strategies discovered by the AMPS algorithm across all of our benchmark problems ($p < .01$ for all pairwise two-tailed t-tests). Critically, while imposing hierarchical constraints on the strategy space of BMPS and the greedy myopic VOC method had no negative effect on the performance of the resulting strategies ($p > .770$ for all pairwise t-tests), this substantially increased the scalability of both methods. This is evident from the reduced run time and time complexity of both strategy discovery methods (see Figure 2b and Table 2). The improved run time profile reflects a reduction in the asymptotic upper bound on the algorithms' run times when hierarchical structure is imposed on the strategy space. For example, for the greedy myopic VOC method, the upper bound decreases from $O(M^2N^2)$ to $O(M^2 + N^2)$, where $M$ is the number possible final destinations as a goal and $N$ is the number of steps to the chosen goal. The reduction in the upper bound for BMPS is even more pronounced and is analyzed in the Appendix which means that algorithms that use hierarchical structure are scalable to more complex environments.

The planning algorithm discovered by the hierarchical BMPS algorithm in the benchmark environments is qualitatively different from all existing planning algorithms[3]. It first evaluates the goal nodes

[2]The net return is the difference between the sum of rewards and the cost of planning.

[3]While this strategy was discovered assuming that the cost of evaluating a potential goal node is the same as the cost of evaluating an intermediate node, we found that the discovered strategy remained the same as we increased the cost of evaluating goal nodes to 2, 5, or 10.

Table 1: Net returns of various strategy discovery methods ($S$) and existing planning algorithms ($P$). The best algorithms and the best net returns for each environment setting (column) are formatted in **bold**. The four best methods performed significantly better than the other methods but the differences between them are not statistically significant.

| Type | Name | 2 Goals | 3 Goals | 4 Goals | 5 Goals |
|---|---|---|---|---|---|
| $S$ | **Hierarchical BMPS** | 108.79 | **150.63** | 178.98 | 206.45 |
| $S$ | **Non-hierarchical BMPS** | **111.53** | 148.01 | **182.38** | 204.37 |
| $S$ | **Hierarchical greedy myopic VOC** | 108.48 | 150.13 | 178.81 | **206.57** |
| $S$ | **Non-hierarchical greedy myopic VOC** | 107.98 | 150.41 | 180.35 | 205.40 |
| $S$ | Adaptive Metareasoning Policy Search | 77.08 | 109.39 | 127.01 | 141.34 |
| $P$ | Bidirectional Search | 88.59 | 115.07 | 134.08 | 154.24 |
| $P$ | Backward Search | 87.85 | 114.29 | 134.43 | 156.56 |
| $P$ | Random Policy | 52.73 | 80.05 | 89.31 | 101.15 |

Table 2: Average time to evaluate an environment represented in seconds.

| Strategy Discovery Algorithm | 2 Goals | 3 Goals | 4 Goals | 5 Goals |
|---|---|---|---|---|
| **Hierarchical BMPS** | **4.18** | **6.45** | **7.45** | **9.30** |
| Non-hierarchical BMPS | 16.09 | 44.81 | 117.46 | 203.18 |
| Hierarchical greedy myopic VOC | 7.35 | 8.52 | 10.03 | 14.64 |
| Non-hierarchical greedy myopic VOC | 10.56 | 29.02 | 121.53 | 101.97 |
| Adaptive Metareasoning Policy Search | 6.45 | 13.39 | 24.43 | 64.36 |

Table 3: Comparison in performance of BMPS with and without hierarchical structure on 2-goal environment with various variance ratios. $\sigma_\Sigma$: cumulative standard deviation of the longest path to a goal node; $\sigma_1$: standard deviation of the first goal node; $\sigma_2$: standard deviation of the second goal node; $\Delta$: absolute difference between net returns; $\%\Delta$: relative difference between net returns.

| $\sigma_\Sigma$ | $\sigma_1$ | $\sigma_2$ | Hierarchical | Non-hierarchical | $\Delta$ | $\%\Delta$ |
|---|---|---|---|---|---|---|
| 46.1 | 100 | 120 | 108.79 | 111.53 | 2.74 | 2.46 |
| 46.1 | 75 | 75 | 89.41 | 90.62 | 1.21 | 1.34 |
| 46.1 | 50 | 60 | 76.84 | 79.52 | 2.68 | 3.37 |
| 46.1 | 25 | 30 | 60.62 | 64.95 | 4.33 | 6.67 |
| 46.1 | 12 | 15 | 53.63 | 59.63 | 6.00 | 10.06 |

until it finds a goal node with a sufficiently high reward. Then, it plans backward from the chosen goal to the current state. In evaluating candidate paths from the goal to the current state, it discards each path from further exploration as soon as a high negative reward is found on it.[4] Unlike the discovered strategies, most extant planning algorithms plan forward and the few planning algorithms that plan backward (e.g., Bidirectional Search and Backward Search) do not preemptively terminate the exploration of a path. These properties are not an artifact of the imposed hierarchical structure. Rather, they reflect the nature of the optimal planning algorithm for environments where goal values are more variable than immediate and intermediate rewards. In fact, we found that even though the hierarchical decomposition eliminated a large number of possible planning algorithms from the hypothesis space, it retained the optimal planning algorithm for the studied environment.

To determine the range of planning problems for which our hierarchical decomposition can be used to discover high-performing planning algorithms at scale, we varied the variance structure of the 2-goal environment and compared the algorithmic performance of BMPS with and without hierarchical structure (see Table 3). We found that the usefulness of the hierarchical decomposition depends on the ratio of the variances of goal values versus path costs. Concretely, the performance loss of the algorithm utilising the hierarchical structure is below $5\%$ when the variance of goal values is at least as high as the variance of the path costs and increases to $10\%$ as the variance of goal values drops to only one third of the variance of the path costs.

[4]Interestingly, this phenomenon has also been observed in human planning [35].

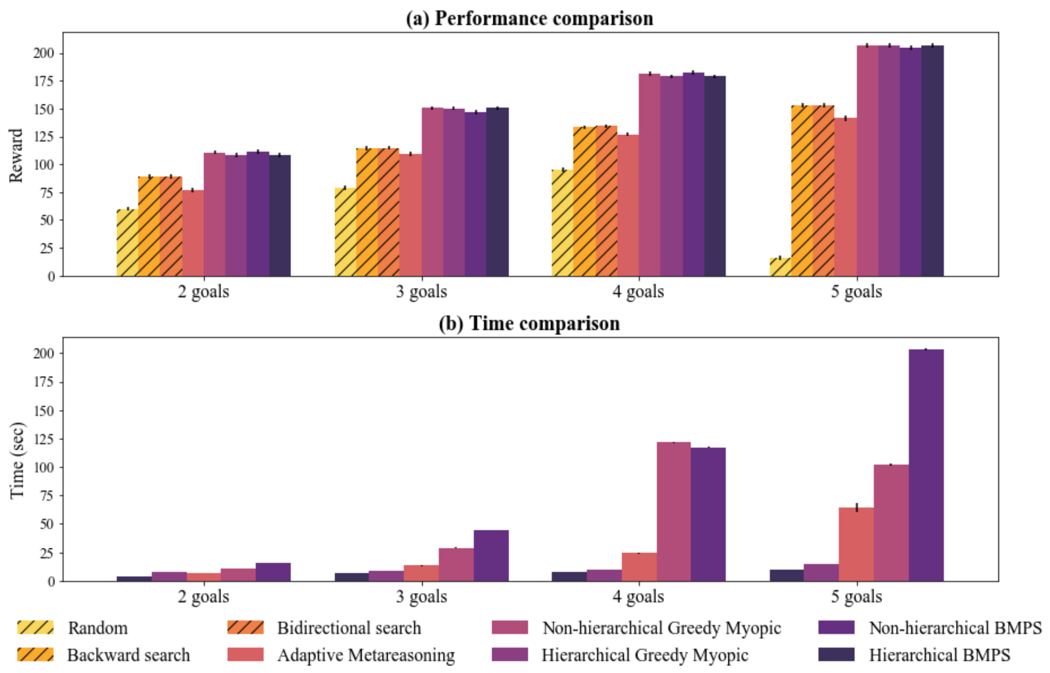

Figure 2: (a) Net returns of existing planning algorithms (striped bars) versus planning strategies discovered by various strategy discovery methods (bars without stripes). (b) Comparison of the mean time for various strategy discovery methods (in seconds).

## 6 Discussion

The design of efficient algorithms could, in principle, be partly automated by using machine learning methods for automatic strategy discovery. But previous strategy discovery methods lacked scalability. To tackle this issue, we have introduced a new approach for discovering planning algorithms which is more scalable than existing metareasoning methods. The central idea is to decompose the strategy discovery problem into discovering goal-setting strategies and discovering goal achievement strategies. We found that this drastically reduces the time complexity of automatic strategy discovery (Figure 2b) without compromising on the quality of the discovered algorithms (Figure 2a). Our method discovered a new planning algorithm that is qualitatively different from and performs significantly better than conventional planning algorithms. The proposed hierarchical decomposition is versatile and can be used to scale up other strategy discovery algorithms. Future work will evaluate the performance of automatically discovered hierarchical planning strategies against some of the best hierarchical planning algorithms designed by AI researchers [26, 29]. We found that our hierarchical decomposition is useful for a wide range of planning problems – especially when the values of candidate goals are more variable than the costs of the paths that lead there. This is true of many real-world problems, including the challenge of choosing a career. Going back to the student's dilemma of becoming a computer scientist or a janitor, the likely economic return of being a computer scientist is much greater than that of a janitor. In comparison, the difference in the return in the selection of subjects in school would be smaller. Hence, the variance of the goal nodes would, most likely, be higher than the variance of the possible paths to reach the goal. This suggests that incorporating cognitively-inspired hierarchical structure into the definition of algorithm discovery problems is a promising step toward enabling AI to discover algorithms with human-level computational efficiency. In addition, our method also enables cognitive scientists to scale up the resource-rational analysis methodology for understanding the cognitive mechanisms of decision-making [13] to increasingly more naturalistic models of the decision problems people face in real life. Finally, making automatic strategy discovery more scalable is also an important step towards leveraging AI to improve human decision-making through intelligent cognitive tutors that discover and teach optimal decision strategies [19].

## Broader impact

The work presented in this paper impacts research in both computer science and cognitive science. This study fundamentally tackles the quest to enable general artificial intelligence. More tangibly, this work can be used to develop cognitive tutors [19, 36] that improve human decision-making by teaching the automatically discovered strategies. The optimal strategies discovered by our algorithm can be taught to humans to enable them to improve their decision making ability. Our work is an important step to increase the scalability of strategy discovery algorithms which is required for discovering optimal planning strategies for large and complex decision problems that people face in the real-world. If the algorithm was deployed in a cognitive tutor, the failure of the algorithm would mean that a near-optimal planning strategy would not be found and could result in the tutor teaching a sub-optimal strategy. While the development of cognitive tutors is still in the research phase, the success of the discovery algorithm is vital for the usefulness of such tutors. We believe that the data accumulated is free from any human bias since it is artificially generated.

## References

[1] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[2] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359, 2017.

[3] Vahid Behzadan and Arslan Munir. Vulnerability of deep reinforcement learning to policy induction attacks. In *International Conference on Machine Learning and Data Mining in Pattern Recognition*, pages 262–275. Springer, 2017.

[4] Sandy Huang, Nicolas Papernot, Ian Goodfellow, Yan Duan, and Pieter Abbeel. Adversarial attacks on neural network policies. *arXiv preprint arXiv:1702.02284*, 2017.

[5] Karl Cobbe, Oleg Klimov, Chris Hesse, Taehoon Kim, and John Schulman. Quantifying generalization in reinforcement learning. *arXiv preprint arXiv:1812.02341*, 2018.

[6] Charles Packer, Katelyn Gao, Jernej Kos, Philipp Krähenbühl, Vladlen Koltun, and Dawn Song. Assessing generalization in deep reinforcement learning. *arXiv preprint arXiv:1810.12282*, 2018.

[7] Brenden M Lake, Tomer D Ullman, Joshua B Tenenbaum, and Samuel J Gershman. Building machines that learn and think like people. *Behavioral and brain sciences*, 40, 2017.

[8] Stuart Jonathan Russell and Eric Wefald. *Do the right thing: studies in limited rationality*. MIT press, 1991.

[9] Christopher H Lin, Andrey Kolobov, Ece Kamar, and Eric Horvitz. Metareasoning for planning under uncertainty. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[10] Frederick Callaway, Sayan Gul, Paul M Krueger, Thomas L Griffiths, and Falk Lieder. Learning to select computations. *Uncertainty in Artificial Intelligence*, 2018.

[11] Nicholas James Hay. *Principles of Metalevel Control*. PhD thesis, UC Berkeley, 2016.

[12] Allen Newell, Herbert Alexander Simon, et al. *Human problem solving*, volume 104. Prentice-Hall Englewood Cliffs, NJ, 1972.

[13] Falk Lieder and Thomas L Griffiths. Resource-rational analysis: understanding human cognition as the optimal use of limited computational resources. *Behavioral and Brain Sciences*, 43, 2020.

[14] Thomas L Griffiths, Frederick Callaway, Michael B Chang, Erin Grant, Paul M Krueger, and Falk Lieder. Doing more with less: meta-reasoning and meta-learning in humans and machines. *Current Opinion in Behavioral Sciences*, 29:24–30, 2019.

[15] A Kemtur, Y Jain, Aashay Mehta, Frederick Callaway, Saksham Consul, Jugoslav Stojcheski, and Falk Lieder. Leveraging machine learning to automatically derive robust planning strategies from biased models of the environment. In *CogSci 2020*. CogSci, 2020.

[16] Charles S Carver and Michael F Scheier. *On the self-regulation of behavior*. Cambridge University Press, 2001.

[17] Matthew M Botvinick. Hierarchical models of behavior and prefrontal function. *Trends in cognitive sciences*, 12(5):201–208, 2008.

[18] Alec Solway, Carlos Diuk, Natalia Córdova, Debbie Yee, Andrew G Barto, Yael Niv, and Matthew M Botvinick. Optimal behavioral hierarchy. *PLoS computational biology*, 10(8), 2014.

[19] F. Lieder, F. Callaway, Y.R. Jain, P.M. Krueger, P. Das, S. Gul, and T.L. Griffiths. A cognitive tutor for helping people overcome present bias. In *RLDM 2019*, 2019.

[20] F. Callaway, F. Lieder, P. Das, S. Gul, P. Krueger, and T.L. Griffiths. A resource-rational analysis of human planning. In C. Kalish, M. Rau, J. Zhu, and T. Rogers, editors, *CogSci 2018*, 2018.

[21] Nicholas Hay, Stuart Russell, David Tolpin, and Solomon Eyal Shimony. Selecting computations: Theory and applications. *arXiv preprint arXiv:1408.2048*, 2014.

[22] Stuart Russell and Eric Wefald. ‘principles of metareasoning. *Artificial intelligence*, 49(1-3):361–395, 1992.

[23] Falk Lieder, Paul M Krueger, and Tom Griffiths. An automatic method for discovering rational heuristics for risky choice. In *CogSci*, 2017.

[24] F. Callaway, F. Lieder, P. M. Krueger, and T. L. Griffiths. Mouselab-MDP: A new paradigm for tracing how people plan. In *The 3rd Multidisciplinary Conference on Reinforcement Learning and Decision Making, Ann Arbor, MI*, 2017. URL `https://osf.io/vmkrq/`.

[25] Momchil S Tomov, Samyukta Yagati, Agni Kumar, Wanqian Yang, and Samuel J Gershman. Discovery of hierarchical representations for efficient planning. *PLoS computational biology*, 16(4):e1007594, 2020.

[26] Leslie Pack Kaelbling and Tomás Lozano-Pérez. Hierarchical planning in the now. In *Workshops at the Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010.

[27] Earl D Sacerdoti. Planning in a hierarchy of abstraction spaces. *Artificial intelligence*, 5(2):115–135, 1974.

[28] Bhaskara Marthi, Stuart J Russell, and Jason Andrew Wolfe. Angelic semantics for high-level actions. In *Seventeenth International Conference on Automated Planning and Scheduling*, pages 232–239, 2007.

[29] Jason Wolfe, Bhaskara Marthi, and Stuart Russell. Combined task and motion planning for mobile manipulation. In *Twentieth International Conference on Automated Planning and Scheduling*, 2010.

[30] Anna C Schapiro, Timothy T Rogers, Natalia I Cordova, Nicholas B Turk-Browne, and Matthew M Botvinick. Neural representations of events arise from temporal community structure. *Nature neuroscience*, 16(4):486, 2013.

[31] Jonas Mockus. *Bayesian approach to global optimization: theory and applications*, volume 37. Springer Science & Business Media, 2012.

[32] The GPyOpt authors. GPyOpt: A bayesian optimization framework in python. `http://github.com/SheffieldML/GPyOpt`, 2016.

[33] Stuart Russell and Peter Norvig. Artificial intelligence: a modern approach. 2002.

[34] Justin Svegliato and Shlomo Zilberstein. Adaptive metareasoning for bounded rational agents. In *CAI-ECAI Workshop on Architectures and Evaluation for Generality, Autonomy and Progress in AI (AEGAP), Stockholm, Sweden*, 2018.

[35] Quentin JM Huys, Neir Eshel, Elizabeth O’Nions, Luke Sheridan, Peter Dayan, and Jonathan P Roiser. Bonsai trees in your head: how the pavlovian system sculpts goal-directed choices by pruning decision trees. *PLoS computational biology*, 8(3), 2012.

[36] F. Lieder, F. Callaway, Y. R. Jain, P. Das, G. Iwama, S. Gul, P.M. Krueger, and T. L. Griffiths. Leveraging artificial intelligence to improve people’s planning strategies, 2020. Manuscript in revision.