Metareasoning for Planning Under Uncertainty

Christopher H. Lin *
University of Washington
Seattle, WA
chrislin@cs.washington.edu

Andrey Kolobov, Ece Kamar, Eric Horvitz
Microsoft Research
Redmond, WA
{akolobov,eckamar,horvitz}@microsoft.com

Abstract

The conventional model for online planning under uncertainty assumes that an agent can stop and plan without incurring costs for the time spent planning. However, planning time is not free in most real-world settings. For example, an autonomous drone is subject to nature's forces, like gravity, even while it thinks, and must either pay a price for counteracting these forces to stay in place, or grapple with the state change caused by acquiescing to them. Policy optimization in these settings requires *metareasoning*—a process that trades off the cost of planning and the potential policy improvement that can be achieved. We formalize and analyze the metareasoning problem for Markov Decision Processes (MDPs). Our work subsumes previously studied special cases of metareasoning and shows that in the general case, metareasoning is at most polynomially harder than solving MDPs with any given algorithm that disregards the cost of thinking. For reasons we discuss, optimal general metareasoning turns out to be impractical, motivating approximations. We present approximate metareasoning procedures which rely on special properties of the BRTDP planning algorithm and explore the effectiveness of our methods on a variety of problems.

1 Introduction

Offline probabilistic planning approaches, such as policy iteration [Howard, 1960], aim to construct a policy for every possible state before acting. In contrast, online planners, such as RTDP [Barto *et al.*, 1995] and UCT [Kocsis and Szepesvári, 2006], interleave planning with execution. After an agent takes an action and moves to a new state, these planners suspend execution to plan for the next step. The more planning time they have, the better their action choices. Unfortunately, planning time in online settings is usually not free.

Consider an autonomous Mars rover trying to decide what to do while a sandstorm is nearing. The size and uncertainty of the domain precludes a-priori computation of a complete

policy, and demands the use of online planning algorithms. Normally, the longer the rover runs its planning algorithm, the better decision it can make. However, computation costs power; moreover, if it reasons for too long without taking preventive action, it risks being damaged by the oncoming sandstorm. Or consider a space probe on final approach to a speeding comet, when the probe must plan to ensure a safe landing based on new information it gets about the comet's surface. More deliberation time means a safer landing. At the same time, if the probe deliberates for too long, the comet may zoom out of range — a similarly undesirable outcome.

Scenarios like these give rise to a general *metareasoning decision problem*: how should an agent trade off the cost of planning and the quality of the resulting policy for the base planning task every time it needs to make a move, so as to optimize its long-term utility? Metareasoning about base-level problem solving has been explored for probabilistic inference and decision making [Horvitz, 1987; Horvitz *et al.*, 1989], theorem proving [Horvitz and Klein, 1995; Kautz *et al.*, 2002], handling streams of problems [Horvitz, 2001; Shahaf and Horvitz, 2009], and search [Russell and Wefald, 1991; Burns *et al.*, 2013]. There has been little work exploring generalized approaches to metareasoning for planning.

We explore the general metareasoning problem for Markov decision processes (MDPs). We begin by formalizing the problem with a general but precise definition that subsumes several previously considered metareasoning models. Then, we show with a rigorous theoretical analysis that optimal general metareasoning for planning under uncertainty is at most polynomially harder than solving the original planning problem with any given MDP solver. However, this increase in computational complexity, among other reasons we discuss, renders such optimal general metareasoning impractical. The analysis raises the issue of allocating time for *metareasoning itself*, and leads to an infinite regress of meta*reasoning (metareasoning, metametareasoning, etc.) problems.

We next turn to the development and testing of fast approximate metareasoning algorithms. Our procedures use the Bounded RTDP (BRTDP [McMahan *et al.*, 2005]) algorithm to tackle the base MDP problem, and leverage BRTDP-computed bounds on the quality of MDP policies to reason about the value of computation. In contrast to prior work on this topic, our methods do not require any training data, precomputation, or prior information about target domains.

We perform a set of experiments showing the performance of these algorithms versus baselines in several synthetic domains with different properties, and characterize their performance with a measure that we call the *metareasoning gap* — a measure of the potential for improvement from metareasoning. The experiments demonstrate that the proposed techniques excel when the metareasoning gap is large.

2 Related Work

Metareasoning efforts to date have employed strategies that avoid the complexity of the general metareasoning problem for planning via relying on different kinds of simplifications and approximations. Such prior studies include metareasoning for time-critical decisions where expected value of computation is used to guide probabilistic inference [Horvitz, 1987; Horvitz *et al.*, 1989], and work on the guiding of sequences of single actions in search [Russell and Wefald, 1991; Burns *et al.*, 2013]. Several lines of work have leveraged offline learning [Breese and Horvitz, 1990; Horvitz *et al.*, 2001; Kautz *et al.*, 2002]. Other studies have relied on optimizations and inferences that leverage the structure of problems, such as the functional relationships between metareasoning and reasoning [Horvitz and Breese, 1990; Zilberstein and Russell, 1996], the structure of the problem space [Horvitz and Klein, 1995], and the structure of utility [Horvitz, 2001]. In other work, [Hansen and Zilberstein, 2001] proposed a non-myopic dynamic programming solution for single-shot problems. Finally, several planners rely on a heuristic form of online metareasoning when maximizing policy reward under computational constraints in real-world time with no "conversion rate" between the two [Kolobov *et al.*, 2012; Keller and Geißer, 2015]. In contrast, our metareasoning model is unconstrained, with computational and base-MDP costs in the same "currency."

Our investigation also has connections to research on allocating time in a system composed of multiple sensing and planning components [Zilberstein and Russell, 1996; 1993], on optimizing portfolios of planning strategies in scheduling applications [Dean *et al.*, 1995], and on choosing actions to explore in Monte Carlo planning [Hay *et al.*, 2012]. In other related work, [Chanel *et al.*, 2014] consider how best to plan on one thread, while a separate thread processes execution.

3 Preliminaries

A key contribution of our work is formalizing the metareasoning problem for planning under uncertainty. We build on the framework of stochastic shortest path (SSP) MDPs with a known start state. This general MDP class includes finite-horizon and discounted-reward MDPs as special cases [Bertsekas and Tsitsiklis, 1996], and can also be used to approximate partially observable MDPs with a fixed initial belief state. An SSP MDP M is a tuple $\langle S, A, T, C, s_0, s_g \rangle$, where S is a finite set of states, A is a set of actions that the agent can take, $T : (S, A, S) \rightarrow [0, 1]$ is a transition function, $C : (S, A) \rightarrow \mathbb{R}$ is a cost function, $s_0 \in S$ is the start state, and s_g is the goal state. An SSP MDP must have a complete proper policy, a policy that leads to the goal from

any state with probability 1, and all improper policies must accumulate infinite cost from every state from which they fail to reach the goal with a positive probability. The objective is to find a Markovian policy $\pi_{s_0} : S \rightarrow A$ with the minimum expected cost of reaching the goal from the start state s_0 — in SSP MDPs, at least one policy of this form is globally optimal.

Without loss of generality, we assume an SSP MDP to have a specially designated NOP ("no-operation") action. NOP is an action the agent chooses when it wants to "idle" and "think/plan", and its semantic meaning is problem-dependent. For example, in some MDPs, choosing NOP means staying in the current state for one time step, while in others it may mean allowing a tidal wave to carry the agent to another state. Designating an action as NOP does not change SSP MDPs' mathematical properties, but plays a crucial role in our metareasoning formalization.

4 Formalization and Theoretical Analysis of Metareasoning for MDPs

The online planning problem of an agent, which involves choosing an action to execute in any given state, is represented as an SSP MDP that encapsulates the dynamics of the environment and costs of acting and thinking. We call this problem the *base problem*. The agent starts off in this environment with some default policy, which can be as simple as random or guided by an unsophisticated heuristic. The agent's metareasoning problem, then, amounts to deciding, at every step during its interaction with the environment, between improving its existing policy or using this policy's recommended action while paying a cost for executing *either of these options*, so as to minimize its expected cost of getting to the goal.

Besides the agent's state in the base MDP, which we call the *world state*, the agent's metareasoning decisions are conditioned on the algorithm the agent uses for solving the base problem, i.e., intuitively, on the agent's thinking process. To abstract away the specifics of this planning algorithm for the purposes of metareasoning formalization, we view it as a black-box MDP solver and represent it, following the Church-Turing thesis, with a Turing machine B that takes a base SSP MDP M as input. In our analysis, we assume the following about Turing machine B's operation:

- B is deterministic and halts on every valid base MDP M. This assumption does not affect the expressiveness of our model, since randomized Turing machines can be trivially simulated on deterministic ones, e.g., via seed enumeration (although potentially at an exponential increase in time complexity). At the same time, it greatly simplifies our theorems.

- An agent's thinking cycle corresponds to B executing a single instruction.

- A *configuration* of B is a combination of B's tape contents, state register contents, head position, and next input symbol. It represents the state of the online planner in solving the base problem M. We denote the set of all configurations B ever enters on a given input MDP M

as $X^{B(M)}$. We assume that B can be paused after executing y instructions, and that its configuration at that point can be mapped to an action for any world state s of M using a special function $f : S \times X^{B(M)} \to A$ in time polynomial in M's flat representation. The number of instructions needed to compute f is not counted into y. That is, an agent can stop thinking at any point and obtain a policy for its current world state.

- An agent is allowed to "think" (i.e., execute B's instructions) only by choosing the NOP action. If an agent decides to resume thinking after pausing B and executing a few actions, B re-starts from the configuration in which it was last paused.

We can now define metareasoning precisely:

Definition 1. *Metareasoning Problem. Consider an MDP $M = \langle S, A, T, C, s_0, s_g \rangle$ and an SSP MDP solver represented by a deterministic Turing machine B. Let $X^{B(M)}$ be the set of all configurations B enters on input M, and let $T^{B(M)} : X^{B(M)} \times X^{B(M)} \to \{0, 1\}$ be the (deterministic) transition function of B on $X^{B(M)}$. A metareasoning problem for M with respect to B, denoted $\mathtt{Meta}_B(M)$ is an MDP $\langle S^m, A^m, T^m, C^m, s_0^m, s_g^m \rangle$ s.t.*

- $S^m = S \times X^{B(M)}$

- $A^m = A$

- $T^m((s, \chi), a, (s', \chi'))$

$$= \begin{cases} T(s, a, s') & \text{if } a \neq \text{NOP}, \chi = \chi', \text{ and } a = f(s, \chi) \\ T(s, a, s') \cdot T^{B(M)}(\chi, \chi') & \text{if } a = \text{NOP} \\ 0 \text{ otherwise} \end{cases}$$

- $C^m((s, \chi), a, (s', \chi')) = C(s, a, s')$ if $T(s, a, s') \neq 0$, and 0 otherwise

- $s_0^m = (s_0, \chi_0)$, where χ_0 is the first configuration B enters on input M

- $s_g^m = (s_g, \chi)$, where χ is any configuration in $X^{B(M)}$

Solving the metareasoning problem means finding a policy for $\mathtt{Meta}_B(M)$ with the lowest expected cost of reaching s_g^m.

This definition casts a metareasoning problem for a base MDP as another MDP (a *meta-MDP*). Note that in $\mathtt{Meta}_B(M)$, an agent *must* choose either NOP or an action currently recommended by $B(M)$; in other cases, the transition probability is 0. Thus, $\mathtt{Meta}_B(M)$'s definition essentially forces an agent to switch between two "meta-actions": thinking or acting in accordance with the current policy.

Modeling an agent's reasoning process with a Turing machine allows us to see that at every time step the metareasoning decision depends on the combination of the current world state and the agent's "state of mind," as captured by the Turing machine's current configuration. In principle, this decision could depend on the entire history of the two, but the following theorem implies that, as for M, at least one optimal policy for $\mathtt{Meta}_B(M)$ is always Markovian.

Theorem 1. *If the base MDP M is an SSP MDP, then $\mathtt{Meta}_B(M)$ is an SSP MDP as well, provided that B halts on M with a proper policy. If the base MDP M is an infinite-horizon discounted-reward MDP, then so is $\mathtt{Meta}_B(M)$. If the base MDP M is a finite-horizon MDP, then so is $\mathtt{Meta}_B(M)$.*

Proof. Verifying the result for finite-horizon and infinite-horizon discounted-reward MDPs M is trivial, since the only requirement $\mathtt{Meta}_B(M)$ must satisfy in these cases is to have a finite horizon or a discount factor, respectively.

If M is an SSP MDP, then, per the SSP MDP definition [Bertsekas and Tsitsiklis, 1996], to ascertain the theorem's claim we need to verify that (1) $\mathtt{Meta}_B(M)$ has at least one proper policy and (2) every improper policy in $\mathtt{Meta}_B(M)$ accumulates an infinite cost from some state.

To see why (1) is true, recall that $\mathtt{Meta}_B(M)$'s state space is formed by all configurations Turing machine B enters on M. Consider any state (s_0', χ_0') of $\mathtt{Meta}_B(M)$. Since B is deterministic, as stated in Section 3, the configuration χ_0' lies in the linear sequence of configurations between the "designated" initial configuration χ_0 and the final proper-policy configuration that B enters according to the theorem. Thus, B can reach a proper-policy configuration from χ_0'. Therefore, let the agent starting in the state (s_0', χ_0') of $\mathtt{Meta}_B(M)$ choose NOP until B halts, and then follow the proper policy corresponding to B's final configuration until it reaches a goal state s_g of M. This state corresponds to a goal state (s_g, χ) of $\mathtt{Meta}_B(M)$. Since this construct works for any (s_0', χ_0'), it gives a complete proper policy for $\mathtt{Meta}_B(M)$.

To verify (2), consider any policy π^m for $\mathtt{Meta}_B(M)$ that with a positive probability fails to reach the goal. Any infinite trajectory of π^m that fails to reach the goal can be mapped onto a trajectory in M that repeats the action choices of π^m's trajectory in M's state space S. Since M is an SSP MDP, this projected trajectory must accumulate an infinite cost, and therefore the original trajectory in $\mathtt{Meta}_B(M)$ must do so as well, implying the desired result. $\square$

We now present two results to address the difficulty of metareasoning.

Theorem 2. *For an SSP MDP M and a deterministic Turing machine B representing a solver for M, the time complexity of $\mathtt{Meta}_B(M)$ is at most polynomial in the time complexity of executing B on M.*

Proof. The main idea is to construct the MDP representing $\mathtt{Meta}_B(M)$ by simulating B on M. Namely, we can run B on M until it halts and record every configuration B enters to obtain the set X. Given X, we can construct $S^m = S \times X$ and all other components of $\mathtt{Meta}_B(M)$ in time polynomial in $|X|$ and $|M|$. Constructing X itself takes time proportional to running time of B on M. Since, by Theorem 1, $\mathtt{Meta}_B(M)$ is an SSP MDP and hence can be solved in time polynomial in the size of its components, e.g., by linear programming, the result follows. $\square$

Theorem 3. *Metareasoning for SSP MDPs is P-complete under NC-reduction. (Please see the appendix for proof.)*

At first glance, the results above look encouraging. However, upon closer inspection they reveal several subtleties making optimal metareasoning utterly impractical. First, although both SSP MDPs and their metareasoning counterparts with respect to an optimal polynomial-time solver are in P, doing metareasoning for a given MDP M is appreciably more expensive than solving *that MDP itself*. Given that the additional complexity due to metareasoning cannot be ignored, the agent now faces the new challenge of allocating computational time between metareasoning and planning for the base problem. This challenge is a meta-metareasoning problem, and ultimately causes infinite regress, an unbounded nested sequence of ever-costlier reasoning problems.

Second, constructing $\text{Meta}_B(M)$ by running B on M, as the proof of Theorem 2 proceeds, may entail solving M in the process of metareasoning. While the proof doesn't show that this is the only way of constructing $\text{Meta}_B(M)$, without making additional assumptions about B's operation one cannot exclude the possibility of having to run B until convergence and thereby completely solving M even before $\text{Meta}_B(M)$ is fully formulated. Such a construction would defeat the purpose of metareasoning.

Third, the validity of Theorems 2 and 3 relies on an implicit crucial assumption that the transitions of solver B on the base MDP M are known in advance. Without this knowledge, $\text{Meta}_B(M)$ turns into a reinforcement learning problem [Sutton and Barto, 1998], which further increases the complexity of metareasoning and the need for simulating B on M. Neither of these is viable in reality.

The difficulties with optimal metareasoning motivate the development of approximation procedures. In this regard, the preceding analysis provides two important insights. It suggests that, since running B on M until halting is infeasible, it may be worth trying to *predict* B's progress on M. Many existing MDP algorithms have clear operational patterns, e.g., evaluating policies in the decreasing order of their cost, as policy iteration does [Howard, 1960]. Regularities like these can be of value in forecasting the benefit of running B on M for additional cycles of thinking. We now focus on exploring approximation schemes that can leverage these patterns.

5 Algorithms for Approximate Metareasoning

Our approach to metareasoning is guided by *value of computation* (VOC) analysis. In contrast to previous work that formulates VOC for single actions or decision-making problems [Horvitz, 1987; Horvitz *et al.*, 1989; Russell and Wefald, 1991], we aim to formulate VOC for online planning. For a given metareasoning problem $\text{Meta}_B(M)$, VOC at any encountered state $s^m = (s, \chi)$ is exactly the difference between the Q-value of the agent following $f(s, \chi)$ (the action recommended by the current policy of the base MDP M) and the Q-value of the agent taking NOP and thinking:

$$VOC(s^m) = Q^*(s^m, f(s, \chi)) - Q^*(s^m, \text{NOP}). \quad (1)$$

VOC captures the difference in long-term utility between thinking and acting as determined by these Q-values. An agent should take the NOP action and think when the VOC is positive. Our technique aims to evaluate VOC by estimating $Q^*(s^m, f(s, \chi))$ and $Q^*(s^m, \text{NOP})$. However, attempting to estimate these terms in a near-optimal manner ultimately runs into the same difficulties as solving $\text{Meta}_B(M)$, such as simulating the agent's thinking process many steps into the future, and is likely infeasible. Therefore, fast approximations for the Q-values will generally have to rely on simplifying assumptions. We rely on performing greedy metareasoning analysis as has been done in past studies of metareasoning [Horvitz *et al.*, 1989; Russell and Wefald, 1991]:

Meta-Myopic Assumption. *In any state s^m of the meta-MDP, we assume that after the current step, the agent will never again choose NOP, and hence will never change its policy.*

This meta-myopic assumption is important in allowing us to reduce VOC estimation to predicting the improvement in the value of the base MDP policy following a single thinking step. The weakness of this assumption is that opportunities for subsequent policy improvements are overlooked. In other words, the VOC computation only reasons about the current thinking opportunity. Nonetheless, in practice, we compute VOC at every timestep, so the agent *can* still think later. Our experiments show that our algorithms perform well in spite of their meta-myopicity.

5.1 Implementing Metareasoning with BRTDP

We begin the presentation of our approximation scheme with the selection of B, the agent's thinking algorithm. Since approximating $Q^*(s^m, f(s, \chi))$ and $Q^*(s^m, \text{NOP})$ essentially amounts to assessing policy values, we would like an online planning algorithm that provides efficient policy value approximations, preferably with some guarantees. Having access to these policy value approximations enables us to design approximate metareasoning algorithms that can evaluate VOC efficiently in a domain-independent fashion.

One algorithm with this property is Bounded RTDP (BRTDP) [McMahan *et al.*, 2005]. It is an anytime planning algorithm based on RTDP [Barto *et al.*, 1995]. Like RTDP, BRTDP maintains a lower bound on an MDP's optimal value function V^*, which is repeatedly updated via Bellman backups as BRTDP simulates trials/rollouts to the goal, making BRTDP's configuration-to-configuration transition function $T^{B(M)}(\chi, \chi')$ stochastic. A key difference is that in addition to maintaining a lower bound, it also maintains an upper bound, updated in the same conceptual way as the lower one. If BRTDP is initialized with a *monotone* upper-bound heuristic, then the upper-bound decreases monotonically as BRTDP runs. The construction of domain-independent monotone bounds is beyond the scope of this paper, but is easy for the domains we study in our experiments. Another key difference between BRTDP and RTDP is that if BRTDP is stopped before convergence, it returns an action greedy with respect to the upper, not lower bound. This behavior guarantees that the expected cost of a policy returned at any time by a monotonically-initialized BRTDP is no worse than BRTDP's current upper bound. Our metareasoning algorithms utilize these properties to estimate VOC. In the rest of the discussion, we assume that BRTDP is ini-

tialized with a monotone upper-bound heuristic.

5.2 Approximating VOC

We now show how BRTDP's properties help us with estimating the two terms in the definition of VOC, $Q^*(s^m, f(s, \chi))$ and $Q^*(s^m, \text{NOP})$. We first assume that one "thinking cycle" of BRTDP (i.e., executing NOP once and running BRTDP in the meantime, resulting in a transition from BRTDP's current configuration χ to another configuration χ') corresponds to completing some fixed number of BRTDP trials from the agent's current world state s.

Estimating $Q^*(s^m, \text{NOP})$

We first describe how to estimate the value of taking the NOP action (thinking). At the highest level, this estimation first involves writing down an expression for $Q^*(s^m, \text{NOP})$, making a series of approximations for different terms, and then modeling the behavior of how BRTDP's upper bounds on the Q-value function drop in order to compute the needed quantities.

When opting to think by choosing NOP, the agent may transition to a different world state while *simultaneously* updating its policy for the base problem. Therefore, we can express $Q^*(s^m, \text{NOP}) =$

$$\sum_{s'} T(s, \text{NOP}, s') \sum_{\chi'} T^{B(M)}(\chi, \chi') V^*((s', \chi')). \quad (2)$$

Because of meta-myopicity, we have $V^*((s', \chi')) = V^{\chi'}(s')$ where $V^{\chi'}$ is the value function of the policy corresponding to χ' in the base MDP. However, this expression cannot be efficiently evaluated in practice, since we do not know BRTDP's transition distribution $T^{B(M)}(\chi, \chi')$ nor the state values $V^{\chi'}(s')$, forcing us to make further approximations. To do so, we assume $V^{\chi'}$ and $Q^{\chi'}$ are random variables, and rewrite $\sum_{\chi'} T^{B(M)}(\chi, \chi') V^{\chi'}(s') =$

$$\sum_{a} P(A_{s'}^{\chi'} = a) E[Q^{\chi'}(s', a) | A_{s'}^{\chi'} = a]. \quad (3)$$

where the random variable $A_{s'}^{\chi'}$ takes value a iff $f(s', \chi') = a$ after one thinking cycle in state (s, χ). Intuitively, $P(A_{s'}^{\chi'} = a)$ denotes the probability that BRTDP will recommend action a in state s' after one thinking cycle. Now, let us denote the Q-value upper bound corresponding to BRTDP's current configuration χ as $\overline{Q}^{\chi}$. This value is *known*. Then, let the upper bound corresponding to BRTDP's next configuration χ', be $\overline{Q}^{\chi'}$. Because we do not know χ', this value is *unknown*, and is a random variable. Because BRTDP selects actions greedily w.r.t. the upper bound, we follow this behavior and use the upper bound to estimate Q-value by assuming that $Q^{\chi'} = \overline{Q}^{\chi'}$. Since the value of $\overline{Q}^{\chi'}$ is unknown at the time of the VOC computation, $P(A_{s'}^{\chi'} = a)$ and $E[\overline{Q}^{\chi'}(s', a) | A_{s'}^{\chi'} = a]$ are computed by integrating over the possible values of $\overline{Q}^{\chi'}$. We have that $E[\overline{Q}^{\chi'}(s', a) | A_s^{\chi'} = a] =$

$$\int_{\overline{Q}^{\chi'}(s', a)} \overline{Q}^{\chi'}(s', a) \frac{P(A_{s'}^{\chi'} = a | \overline{Q}^{\chi'}(s', a)) P(\overline{Q}^{\chi'}(s', a))}{P(A_{s'}^{\chi'} = a)},$$

and $P(A_{s'}^{\chi'} = a) =$

$$\int_{\overline{Q}^{\chi'}(s', a)} P(\overline{Q}^{\chi'}(s', a)) \prod_{a_i \neq a} P(\overline{Q}^{\chi'}(s', a_i) > P(\overline{Q}^{\chi'}(s', a)).$$

Therefore, we must model the distribution that $\overline{Q}^{\chi'}$ is drawn from. We do so by modeling the change $\Delta \overline{Q} = \overline{Q}^{\chi} - \overline{Q}^{\chi'}$, due to a single BRTDP thinking cycle that corresponds to a transition from configuration χ to χ'. Since $\overline{Q}^{\chi}$ is known and fixed, estimating a distribution over possible $\Delta \overline{Q}$ gives us a distribution over $\overline{Q}^{\chi'}$.

Let $\hat{\Delta}\overline{Q}_{s,a}$ be the change in $\overline{Q}_{s,a}$ resulting from the most recent thinking cycle for some state s and action a. We first assume that the change resulting from an additional cycle of thinking, $\Delta \overline{Q}_{s,a}$, will be no larger than the last change, $\Delta \overline{Q}_{s,a} \leq \hat{\Delta}\overline{Q}_{s,a}$. This assumption is reasonable, because we can expect the change in bounds to decrease as BRTDP converges to the true value function. Given this assumption, we must choose a distribution D over the interval $[0, \hat{\Delta}\overline{Q}_{s,a}]$ such that for the next thinking cycle, $\Delta \overline{Q}_{s,a} \sim D$. Figure 1a illustrates these modeling assumptions for two hypothetical actions, a_1 and a_2.

One option is to make D uniform, so as to represent our poor knowledge about the next bound change. Then, computing $P(A_{s'}^{\chi'} = a)$ involves evaluating an integral of a polynomial of degree $O(|A|)$ (the product of $|A| - 1$ CDF's and 1 PDF), and computing $E[\overline{Q}^{\chi'}(s', a) | A_s^{\chi'} = a]$ also entails evaluating an integral of degree $O(|A|)$, and thus computing these quantities for all actions in a state can be computed in time $O(|A|^2)$. Since the overall goal of this subsection, approximating $Q^*(s^m, \text{NOP})$, requires computing $P(A_{s'}^{\chi'} = a)$ for all actions in all states where NOP may lead, assuming there are no more than $K << |A|$ such states, the complexity becomes $O(K|A|^2)$ for each state visited by the agent on its way to the goal.

A weakness of this approach is that the changes in the upper bounds for different actions are modeled independently. For example, if the upper bounds for two actions in a given state decrease by a large amount in the previous thinking step, then it is unlikely that in the next thinking step one of them will drop dramatically while the other drops very little. This independence can cause the amount of uncertainty in the upper bound at the next thinking step to be overestimated, leading to VOC being overestimated as well.

Therefore, we create another version of the algorithm assuming that the speed of decrease in Q-value upper bounds for all actions are perfectly correlated; all ratios between future drops in the next thinking cycle are equal to the ratios between the observed drops in the last thinking cycle. Formally, for a given state s, we let $\rho \sim \text{Uniform}[0, 1]$. Then, let $\Delta \overline{Q}_{s,a} = \rho \cdot \hat{\Delta}\overline{Q}_{s,a}$ for all actions a.

Now, to compute $P(A_{s'}^{\chi'} = a)$, for each action a, we represent the range of its possible future Q-values $\overline{Q}_{s,a}^{\chi'}$ with a line segment l_a on the unit interval [0,1] where $l_a(0) = \overline{Q}_{s,a}^{\chi}$ and
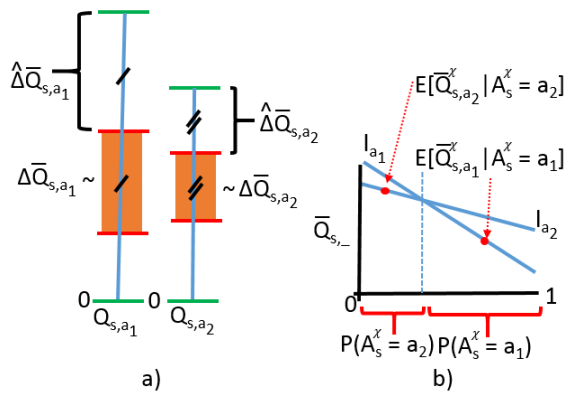
Figure 1: a) Hypothetical drops in upper bounds on the Q-values of two actions, a_1 and a_2. We assume the next Q-value drop resulting from another cycle of thinking, $\Delta \overline{Q}$, is drawn from a range equal to the last drop from thinking, $\hat{\Delta} \overline{Q}$ b) Assuming perfect correlation in the speed of decrease in the Q-value upper bounds, as the upper bounds of the two actions drop from an additional cycle of thinking, initially a_2 has a better upper bound, but eventually a_1 overtakes a_2.

$l_a(1) = \overline{Q}_{s,a}^{\chi} - \Delta \overline{Q}_{s,a}$. Then, $P(A_{s'}^{\chi'} = a)$ is simply the proportion of l_a which lies below all the other lines representing all other actions. We can naïvely compute these probabilities in time $O(|A|^2)$ by enumerating all intersections. Similarly, computing $E[\overline{Q}^{\chi'}(s', a)|A_s^{\chi'} = a]$ is also easy. This value is the mean of the portion of l_a that is beneath all other lines. Figure 1b illustrates these computations.

Whether or not we make the assumption of action independence, we further speed up the computations by only calculating $E[\overline{Q}^{\chi'}(s', a)|A_s^{\chi'} = a]$ and $P(A_{s'}^{\chi'} = a)$ for the two "most promising" actions a, those with the lowest expectation of potential upper bounds. This limits the computation time to the time required to determine these actions (linear in $|A|$), and makes the time complexity of estimating $Q^*(s^m, \text{NOP})$ for one state s be $O(K|A|)$ instead of $O(K|A|^2)$.

Estimating $\mathbf{Q^*(s^m, f(s, \chi))}$

Now that we have described how to estimate the value of taking the NOP action, we describe how to estimate the value of taking the currently recommended action, $f(s, \chi)$. We estimate $Q^*(s^m, f(s, \chi))$ by computing $E[\overline{Q}^{\chi'}(s, f(s, \chi))]$, which takes constant time, keeping the overall time complexity linear. The reason we estimate $Q^*(s^m, f(s, \chi))$ using future Q-value upper bound estimates based on a probabilistic projection of χ', as opposed to our current Q-value upper bounds based on the current configuration χ, is to make use of the more informed bounds derived at the future utility estimation. As the BRTDP algorithm is given more computation time, it can more accurately estimate the upper bound of a policy. This type of approximation has been justified before [Russell and Wefald, 1991]. In addition, using future utility estimates in both estimating $Q^*(s^m, f(s, \chi))$ and $Q^*(s^m, \text{NOP})$ provides a consistency guarantee: *if thinking leads to no policy change, then our method estimates VOC*

to be zero.

5.3 Putting It All Together

The core of our algorithms involves the computations we have described, in every state s the agent visits on the way to the goal. In the experiments, we denote UnCorr Metareasoner as the metareasoner that assumes the actions are uncorrelated, and Metareasoner as the metareasoner that does not make this assumption. To complete the algorithms, we ensure that they decide the agent should think for another cycle if $\hat{\Delta} \overline{Q}_{s,a}$ isn't yet available for the agent's current world state s (e.g., because BRTDP has never updated bounds for this state's Q-value so far), since VOC computation is not possible without prior observations on $\hat{\Delta} \overline{Q}_{s,a}$. Crucially, all our estimates make metareasoning take time only linear in the number of actions, $O(K|A|)$, per visited state.

6 Experiments

We evaluate our metareasoning algorithms in several synthetic domains designed to reflect a wide variety of factors that could influence the value of metareasoning. Our goal is to demonstrate the ability of our algorithms to estimate the value of computation and adapt to a plethora of world conditions. The experiments are performed on four domains, all of which are built on a 100×100 grid world, where the agent can move between cells at each time step to get to the goal located in the upper right corner. To initialize the lower and upper bounds of BRTDP, we use the zero heuristic and an appropriately scaled (multiplied by a constant) Manhattan distance to the goal, respectively.

6.1 Domains

The four world domains are as follows:

- *Stochastic.* This domain adds winds to the grid world to be analogous to worlds with stochastic state transitions. Moving against the wind causes slower movement across the grid, whereas moving with the wind results in faster movement. The agent's initial state is the southeast corner and the goal is located in the northeast corner. We set the parameters of the domain as follows so that there is a policy that can get the agent to the goal with a small number of steps (in tens instead of hundreds) and where the winds significantly influence the number of steps needed to get to the goal: The agent can move 11 cells at a time and the wind has a pushing power of 10 cells. The next location of the agent is determined by adding the agent's vector and the wind's vector except when the agent decides to think (executes NOP), in which case it stays in the same position. Thus, the winds can never push the agent in the opposite direction of its intention. The prevailing wind direction over most of the grid is northerly, except for the column of cells containing the goal and starting position, where it is southerly. Note that this southerly wind direction makes the initial heuristic extremely suboptimal. To simulate stochastic state transitions, the winds have their prevailing direction in a given cell with 60% probability; with

40% probability they have a direction orthogonal to the prevailing one (20% easterly and 20% westerly).

We perform a set of experiments on this simplest domain of the set, to observe the effect of different costs for thinking and acting on the behaviors of algorithms. We vary the cost of thinking and acting between 1 and 15. When we vary the cost of thinking, we fix the cost of acting at 11, and when we vary the cost of acting, we fix the cost of thinking at 1.

- *Traps.* This domain modifies the *Stochastic* domain to resemble the setting where costs for thinking and acting are not constant among states. To simplify the parameter choices, we fix the cost of thinking and acting to be equal, respectively, to the agent's moving distance and wind strength. Thus, the cost of thinking is 10 and the cost of acting is 11. To vary the costs of thinking and acting between states, we make thinking and acting at the initial state extremely expensive at a cost of 100, about 10 times the cost of acting and thinking in the other states. Thus, the agent is forced to think outside its initial state in order to perform optimally.

- *DynamicNOP-1.* In the previous domains, executing a NOP does not change the agent's state. In this domain, thinking causes the agent to move in the direction of the wind, causing the agent to stochastically transition as a result of thinking. In this domain, the cost of thinking is composed of both explicit and implicit components; a static value of 1 unit and a dynamic component determined by stochastic state transitions as a result of thinking. The static value is set to 1 so that the dynamic component can dominate the decisions about thinking. The agent starts in cell $(98, 1)$. We change the wind directions so that there are easterly winds in the most southern row and northerly winds in the most eastern row that can push the agent very quickly to the goal. Westerly winds exist everywhere else, pushing the agent away from the goal. We change the stochasticity of the winds so that the westerly winds change to northerly winds with 20% probability, and all other wind directions are no longer stochastic. We lower the amount of stochasticity to better see if our agents can reason about the implicit costs of thinking. The wind directions are arranged so that there is potential for the agent to improve upon its initial policy but thinking is risky as it can move the agent to the left region, which is hard to recover from since all the winds push the agent away from the goal.

- *DynamicNOP-2.* This domain is just like the previous domain, but we change the direction of the winds in the northern-most row to be easterly. These winds also do not change directions. In this domain, as compared to the previous one, it is less risky to take a thinking action; even when the agent is pushed to the left region of the board, the agent can find strategies to get to the goal quickly by utilizing the easterly wind at the top region of the board.

6.2 The Metareasoning Gap

We introduce the concept of the *metareasoning gap* as a way to quantify the potential improvement over the initial heuristic-implied policy, denoted as Heuristic, that is possible due to optimal metareasoning. The metareasoning gap is the ratio of the expected cost of Heuristic for the base MDP to the expected cost of the optimal metareasoning policy, computed at the initial state. Exact computation of the metareasoning gap requires evaluating the optimal metareasoning policy and is infeasible. Instead, we compute an upper bound on the metareasoning gap by substituting the cost of the optimal metareasoning policy with the cost of the optimal policy for the *base* MDP (denoted OptimalBase). The metareasoning gap can be no larger than this upper bound, because metareasoning can only add cost to OptimalBase. We quantify each domain with this upper bound (MG^{UB}) in Table 1 and show that our algorithms for metareasoning provide significant benefits when MG^{UB} is high. We note that none of the algorithms use the metareasoning gap in its reasoning.

	Heuristic	OptimalBase	MG^{UB}
Stochastic (Thinking)	1089	103.9	10.5
Stochastic (Acting)	767.3	68.1	11.3
Traps	979	113.5	8.6
DynamicNOP-1	251.4	66	3.8
DynamicNOP-2	119.4	66	1.8

Table 1: Upper bounds of metareasoning gaps (MG^{UB}) for all test domains, defined as the ratio of the expected cost of the initial heuristic policy (Heuristic) to that of an optimal one (OptimalBase) at the initial state.

6.3 Experimental Setup

We compare our metareasoning algorithms against a number of baselines. The Think*Act baseline simply plans for n cycles at the initial state and then executes the resulting policy, without planning again. We also consider the Prob baseline, which chooses to plan with probability p at each state, and executes its current policy with probability $1 - p$. An important drawback of these baselines is that their performance is sensitive to their parameters n and p, and the optimal parameter settings vary across domains. The NoInfoThink baseline plans for another cycle if it does not have information about how the BRTDP upper bounds will change. This baseline is a simplified version of our algorithms that does not try to estimate the VOC.

For each experimental condition, we run each metareasoning algorithm until it reaches the goal 1000 times and average the results to account for stochasticity. Each BRTDP trajectory is 50 actions long.

6.4 Results

In *Stochastic*, we perform several experiments by varying the costs of thinking (NOP) and acting. We observe (figures can be found in the appendix) that when the cost of thinking is low or when the cost of acting is high, the baselines do

well with high values of n and p, and when the costs are reversed, smaller values do better. This trend is expected, since lower thinking cost affords more thinking, but these baselines don't allow for predicting the specific "successful" n and p values in advance. `Metareasoner` does not require parameter tuning and beats even the best performing baseline for all settings. Figure 2a compares the metareasoning algorithms against the baselines when the results are averaged over the various settings of the cost of acting, and Figure 2b shows results averaged over the various settings of the cost of thinking. `Metareasoner` does extremely well in this domain because the metareasoning gap is large, suggesting that metareasoning can improve the initial policy significantly. Importantly, we see that `Metareasoner` performs better than `NoInfoThink`, which shows the benefit from reasoning about how the bounds on Q-values will change. `UnCorr Metareasoner` does not do as well as `Metareasoner`, probably because the assumption that actions' Q-values are uncorrelated does not hold well.

We now turn to *Traps*, where thinking and acting in the initial state incurs significant cost. Figure 2c again summarizes the results. `Think*Act` performs very poorly, because it is limited to thinking only at the initial state. `Metareasoner` does well, because it figures out that it should not think in the initial state (beyond the initial thinking step), and should instead quickly move to safer locations. `Uncorr Metareasoner` also closes the metareasoning gap significantly, but again not as much as `Metareasoner`.

We now consider *DynamicNOP-1*, a domain adversarial to approximate metareasoning, because winds almost everywhere push the agent away from the goal. There are only a few locations from which winds can carry the agent to the destination. Figure 2d shows that our algorithms do not achieve large gains. However, this result is not surprising. The best policy involves little thinking, because whenever the agent chooses to think, it is pushed away from the goal, and thinking for just a few consecutive time steps can take the agent to states where reaching the goal is extremely difficult. Consequently, `Think*Act` with 1-3 thinking steps turns out to be near-optimal, since it is pushed away from the goal only slightly and can use a slightly improved heuristic to head back. `Metareasoner` actually does well in many individual runs, but occasionally thinks longer due to VOC computation stochasticity and can get stuck, yielding higher average policy cost. In particular, it may frequently be pushed into a state that it has never encountered before, where it must think again because it does not have any history about how BRTDP's bounds have changed in that state, and then subsequently get pushed into an unencountered state again. In this domain, our approximate algorithms can diverge away from an optimal policy, which would plan very little to minimize the risk of being pushed away from the goal.

DynamicNOP-2 provides the agent more opportunities to recover even if it makes a poor decision. Figure 2e demonstrates that our algorithms perform much better in *DynamicNOP-2* than in *DynamicNOP-1*. In *DynamicNOP-2*, even if our algorithms do not discover the jetstreams that can push it towards the goal from initial thinking, they are provided more chances to recover when they get stuck. When

thinking can move the agent on the board, having more opportunities to recover reduces the risk associated with making suboptimal thinking decisions. Interestingly, the metareasoning gap is decreased at the initial state by the addition of the extra jetstream. However, the metareasoning gap at many other states in the domain is increased, showing that the metareasoning gap at the initial state is not the most ideal way to characterize the potential for improvement via metareasoning in all domains.

7 Conclusion and Future Work

We formalize and analyze the general metareasoning problem for MDPs, demonstrating that metareasoning is only polynomially harder than solving the base MDP. Given the determination that optimal general metareasoning is impractical, we turn to approximate metareasoning algorithms, which estimate the value of computation by relying on bounds given by BRTDP. Finally, we empirically compare our metareasoning algorithms to several baselines on problems designed to reflect challenges posed across a spectrum of worlds, and show that the proposed algorithms are much better at closing large metareasoning gaps.

We have assumed that the agent can plan only when it takes the `NOP` action. A generalization of our work would allow varying amounts of thinking as part of any action. Some actions may consume more CPU resources than others, and actions which do not consume all resources during execution can allocate the remainder to planning. We also can relax the meta-myopic assumption, so as to consider the consequences of thinking for more than one cycle. In many cases, assuming that the agent will only think for one more step can lead to underestimation of the value of thinking, since many cycles of thinking may be necessary to see significant value. This ability can be obtained with our current framework by projecting changes in bounds for multiple steps. However, in experiments to date, we have found that pushing out the horizon of analysis was associated with large accumulations of errors and poor performance due to approximation upon approximation from predictions about multiple thinking cycles. Finally, we may be able to improve our metareasoners by learning about and harnessing more details about the base-level planner. In our `Metareasoner` approximation scheme, we make strong assumptions about how the upper bounds provided by BRTDP will change, but learning distributions over these changes may improve performance. More informed models may lead to accurate estimation of non-myopic value of computation. However, learning distributions in a domain-independent manner is difficult, since the planner's behavior is heavily dependent on the domain and heuristic at hand.

References

[Barto *et al.*, 1995] Andrew G. Barto, Steven J. Bradtke, and Satinder P. Singh. Learning to act using real-time dynamic programming. *Artificial Intelligence*, 72:81–138, 1995.

[Bertsekas and Tsitsiklis, 1996] Dimitri P. Bertsekas and John Tsitsiklis. *Neuro-dynamic Programming*. Athena Scientific, 1996.
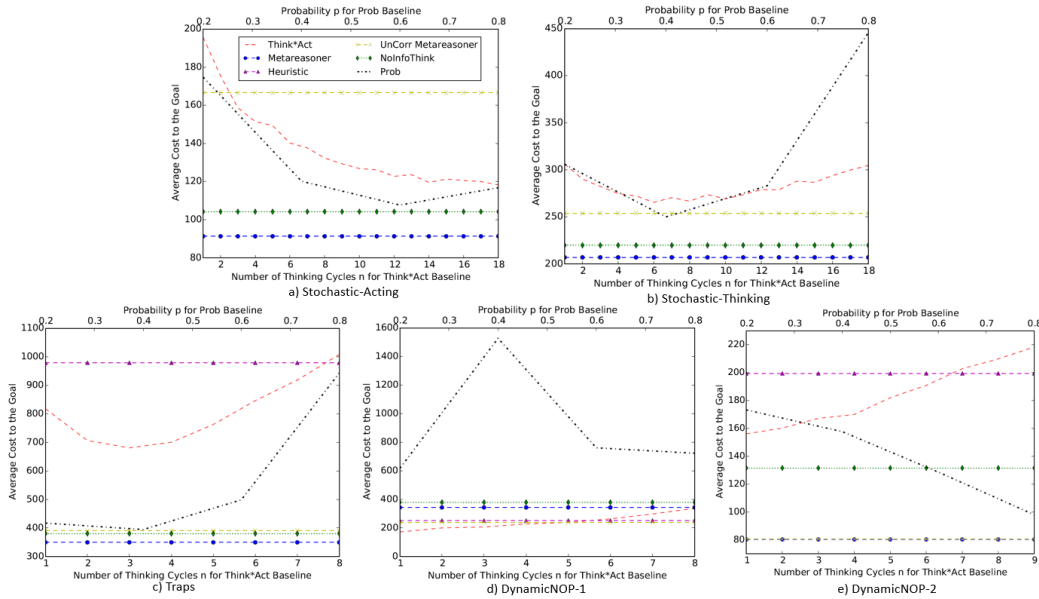
Figure 2: Comparison of `Metareasoner` and `Uncorr Metareasoner` with baselines on experimental domains. Some figures do not include `Heuristic` when it performs especially poorly for readability.

[Breese and Horvitz, 1990] John S. Breese and Eric Horvitz. Ideal reformulation of belief networks. In *UAI*, 1990.

[Burns *et al.*, 2013] Ethan Burns, Wheeler Ruml, and Minh B. Do. Heuristic search when time matters. *Journal of Artificial Intelligence Research*, 47:697–740, 2013.

[Chanel *et al.*, 2014] Caroline P. Carvalho Chanel, Charles Lesire, and Florent Teichteil-Königsbuch. A robotic execution framework for online probabilistic (re)planning. In *ICAPS*, 2014.

[Dean *et al.*, 1995] Thomas Dean, Leslie Pack Kaelbling, Jak Kirman, and Ann Nicholson. Planning under time constraints in stochastic domains. *Artificial Intelligence*, 76:35–74, 1995.

[Hansen and Zilberstein, 2001] Eric A Hansen and Shlomo Zilberstein. Monitoring and control of anytime algorithms: A dynamic programming approach. *Artificial Intelligence*, 126(1):139–157, 2001.

[Hay *et al.*, 2012] Nick Hay, Stuart Russell, David Toplin, and Solomon Eyal Shimony. Selecting computations: Theory and applications. In *UAI*, 2012.

[Horvitz and Breese, 1990] Eric J. Horvitz and John S. Breese. Ideal partition of resources for metareasoning. Technical Report KSL-90-26, Stanford University, 1990.

[Horvitz and Klein, 1995] Eric Horvitz and Adrian Klein. Reasoning, metareasoning, and mathematical truth: Studies of theorem proving under limited resources. In *UAI*, 1995.

[Horvitz *et al.*, 1989] Eric J. Horvitz, Gregory F. Cooper, and David E.Heckerman. Reflection and action under scarce resources: Theoretical principles and empirical study. In *IJCAI*, 1989.

[Horvitz *et al.*, 2001] Eric Horvitz, Yongshao Ruan, Carla P. Gomes, Henry Kautz, Bart Selman, and David M. Chickering. A bayesian approach to tackling hard computational problems. In *UAI*, 2001.

[Horvitz, 1987] Eric Horvitz. Reasoning about beliefs and actions under computational resource constraints. In *UAI*, 1987.

[Horvitz, 2001] Eric Horvitz. Principles and applications of continual computation. *Artificial Intelligence*, 126:159–196, 2001.

[Howard, 1960] R.A. Howard. *Dynamic Programming and Markov Processes*. MIT Press, 1960.

[Kautz *et al.*, 2002] Henry Kautz, Eric Horvitz, Yongshao Ruan, Carla Gomes, and Bart Selman. Dynamic restart policies. In *AAAI*, 2002.

[Keller and Geißer, 2015] Thomas Keller and Florian Geißer. Better be lucky than good: Exceeding expectations in mdp evaluation. In *AAAI*, 2015.

[Kocsis and Szepesvári, 2006] Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In *ECML*, 2006.

[Kolobov *et al.*, 2012] Andrey Kolobov, Mausam, and Daniel S. Weld. Lrtdp vs uct for online probabilistic planning. In *AAAI*, 2012.

[McMahan *et al.*, 2005] H. Brendan McMahan, Maxim Likhachev, and Geoffrey J. Gordon. Bounded real-time dynamic programming: Rtdp with monotone upper bounds and performance guarantees. In *ICML*, 2005.

[Russell and Wefald, 1991] Stuart Russell and Eric Wefald. Principles of metareasoning. *Artificial intelligence*, 49(1):361–395, 1991.

[Shahaf and Horvitz, 2009] Dafna Shahaf and Eric Horvitz. Ivestigations of continual computation. In *IJCAI*, 2009.

[Sutton and Barto, 1998] Richard S. Sutton and Andrew G. Barto. *Introduction to Reinforcement Learning*. MIT Press, 1998.

[Zilberstein and Russell, 1993] Shlomo Zilberstein and Stuart J. Russell. Anytime sensing, planning and action: A practical model for robot control. In *IJCAI*, 1993.

[Zilberstein and Russell, 1996] Shlomo Zilberstein and Stuart Russell. Optimal composition of real-time systems. *Artificial Intelligence*, 82:181–213, 1996.