Learning and planning in environments with delayed feedback

Thomas J. Walsh · Ali Nouri · Lihong Li · Michael L. Littman

Abstract This work considers the problems of learning and planning in Markovian environments with constant observation and reward delays. We provide a hardness result for the general planning problem and positive results for several special cases with deterministic or otherwise constrained dynamics. We present an algorithm, Model Based Simulation, for planning in such environments and use model-based reinforcement learning to extend this approach to the learning setting in both finite and continuous environments. Empirical comparisons show this algorithm holds significant advantages over others for decision making in delayed-observation environments.

Keywords Reinforcement learning · Delayed feedback · Markov decision processes

1 Introduction

In reinforcement learning (RL) [25] models, an agent's observations of its environment are almost universally assumed to be immediately available. However, as tasks and environments grow more complex, this assumption often falters. In such situations, the time it takes for an agent to receive an observation of its current environment could better be spent performing actions. Examples of this phenomena abound in the real world and continue to grow with our increased technological ability. For example, the Mars Rover program has tremendously

T. J. Walsh (✉) · A. Nouri · L. Li · M. L. Littman
Department of Computer Science, Rutgers University,
110 Frelinghuysen Rd., Piscataway, NJ 08854, USA
e-mail: thomaswa@cs.rutgers.edu

A. Nouri
e-mail: nouri@cs.rutgers.edu

L. Li
e-mail: lihong@cs.rutgers.edu

M. L. Littman
e-mail: mlittman@cs.rutgers.edu

broadened the theater of engagement available to roboticists, but direct control of these agents from Earth is limited by the vast communication latency. Delayed observations are also a challenge for agents that receive observations through terrestrial networks [1], such as the Internet or a multi-agent sensor network. Even solo agents that do advanced processing of observations (such as image processing) will experience delay between *observing* the environment, and the opportunity for *acting* based on this information. Such delay is not limited to a single timestep, especially when processing may occur in a pipeline of parallel processors. These scenarios involving delayed feedback have generated interest within the academic community, leading to the inclusion of a delayed version of the "Mountain Car" environment in the First Annual Reinforcement Learning Competition.[1]

One might be tempted to sidestep this problem by simply increasing the length of a "timestep" to synchronize an agent's actions with its delayed observations, but even when this approach is possible (in domains involving momentum, for example, it is not), the best possible policy for such a "wait agent" is often suboptimal—the agent can do better by acting before receiving the most recent observation. This paper considers practical solutions for dealing with constant observation and reward delays.

Prior work in the area of delayed-observation environments dates back over thirty years [7] and several important theoretical results have been developed, including the insight that action and observation delays are two sides of the same coin [13] and that planning can be performed for both finite- and infinite-horizon delayed Markov Decision Processes (MDPs) or Partially Observable MDPs (POMDPs) using algorithms for their undelayed counterparts in much larger state spaces constructed using the last observation and the actions that followed [3,4]. We cover this and several other approaches for learning and planning in delayed environments in Sect. 3. We then show that such augmented approaches can lead to an exponential state-space blowup and provide a hardness result for the planning problem in general delayed MDPs.

In light of these results, we develop algorithms for planning and learning in four practically motivated special cases of Markovian (if not for the delay) environments: finite and continuous environments with deterministic transitions, "mildly stochastic" finite environments, and continuous environments with bounded noise and smooth value functions. We provide loss bounds for our planning algorithm (Model Based Simulation) in these settings and show how to extend this approach to the learning setting using a model-based approach. In continuous state environments, this extension to the learning setting includes a new model-based reinforcement-learning algorithm (Model Parameter Approximation). In Sect. 5, we present empirical studies of learning agents in discrete and continuous delayed environments that demonstrate the advantages of our new techniques. These experiments include a study of the degradation of our planning algorithm with increased non-determinism in the environment dynamics. We assume throughout this work that the delay value is constant and provided to the planner or learner at initialization. Possible methods for relaxing both of these assumptions are briefly treated in Sect. 6, though we have found most delay values can be easily determined or derived before an agent's task begins.

2 Definitions

A *finite* discounted Markov Decision Process (MDP) [20] is defined as a 5-tuple $\langle S, A, P, R, \gamma \rangle$, where S is a set of states, A is a set of actions, and P is a mapping: $S \times A \times S \mapsto [0, 1]$ indicating the probability of the given action taking the agent from state $s \in S$ to state $s' \in S$.

[1] http://rlai.cs.ualberta.ca/RLAI/rlc.html.

R is a mapping: $S \mapsto \Re$, which governs the reward an agent receives in state s (similar results to those in this paper hold for $R : S \times A \mapsto \Re$ or if the range of the reward function is a distribution over reward values), and $\gamma \in (0, 1)$ is the discount factor. Without loss of generality, we assume throughout this work that the reward function is bounded between 0 and some constant $R_{\max}$. A deterministic Markov policy, $\pi : S \mapsto A$, maps states to actions. We refer to such policies as *memoryless*, as they depend only on the current state. The value function $V^{\pi}(s)$ represents the expected cumulative sum of discounted reward and satisfies the Bellman equation: $V^{\pi}(s) = R(s) + \gamma \sum_{s'} P(s, \pi(s), s') V^{\pi}(s')$. Every finite MDP has an optimal policy $\pi^*(s) = \operatorname{argmax}_{\pi} V^{\pi}(s)$ and a unique optimal value function $V^*(s)$. Given an MDP, techniques exist for determining $V^*(s)$ and $\pi^*(s)$ in time polynomial in the size of the MDP [20].

In this work, we will also consider *continuous* MDPs where $S \subseteq \Re^n$ and A may also be continuous ($A \subseteq \Re^m$). Computing value functions in this case often requires approximation methods, an issue we treat in Sects. 3.6 and 4.2.

We define a *constant-delay MDP* (CDMDP) as a 6-tuple $\langle S, A, P, R, \gamma, k \rangle$, where k is a non-negative integer indicating the number of timesteps between an agent occupying a state and receiving its feedback (the state observation and reward). When $k = 0$, the CDMDP becomes a regular MDP; otherwise we assume that k is bounded by a polynomial function of the size of the underlying MDP and the agent observes its initial state in response to each of its first k actions. This assumption is pragmatically grounded in that one would not expect an agent in a delayed environment to act before making at least a starting observation.

Since the current state is not revealed at the time an action is taken (when $k > 0$), one may think of a CDMDP *policy* as a mapping from previous state observations and actions (that is, histories) to actions. It is known that an optimal CDMDP policy can be determined using the so-called *information state* [4], denoted $I_k \in S \times A^k$, which consists of the last observation and the following k actions. In light of this fact, we formally define a CDMDP policy as $\pi : (S \times A^k) \mapsto A$. The *CDMDP planning problem* is defined as: given a CDMDP, initial state I_k^0, and a reward threshold θ, determine whether a policy exists that achieves an expected discounted reward (from the initial state) of at least θ. We note that the rewards counted do not include those observed on the first k steps, as these rewards are determined completely by I_k and are outside the control of the agent. Finding the optimal policy in a CDMDP can be quite difficult—the state-of-the-art technique is to construct a new "augmented" MDP with states I_k as described above. This technique is expounded upon in Sect. 3.3. In the *CDMDP learning problem*, an agent deployed in a delayed-feedback environment knowing only S, A, γ, and k is tasked with finding an optimal policy for the environment online. This problem has received little attention and we hope that the basic theoretical and empirical studies that follow motivate research in this area.

The positive results of this paper pertain to the following special cases for the underlying (undelayed) Markovian dynamics:

I *Deterministic finite*: The undelayed MDP is finite and deterministic; formally, $|S| < \infty$ and $\forall s \exists s' P(s, a, s') = 1$.

II *Deterministic continuous*: Same as Case I except S and A are continuous.

III *Mildly stochastic finite*: The undelayed MDP is finite and there is some $\delta \geq 0$ s.t. $\forall s \exists s' P(s, a, s') \geq 1 - \delta$. Case I is a degenerate case where $\delta = 0$.

IV *Bounded-noise continuous*: The underlying MDP is continuous, and transitions are governed by $s_{t+1} = T(s_t, a_t) + w_t$, where T is a deterministic transition function: $S \times A \mapsto S$, and w_t is bounded noise: $\|w_t\|_{\infty} \leq \Delta$ for some $\Delta \geq 0$. We further assume that the CDMDP's optimal value function is Lipschitz-continuous when the action sequences for two I_k's coincide; that is, $\left|V^*(s, a_1, \ldots, a_k) - V^*(s', a_1, \ldots, a_k)\right| \leq C_V \left\|s - s'\right\|$ for some constant $C_V > 0$. This property holds (for example) when the dynamics of the underlying MDP are smooth. We note that this case covers a wide class of nontrivial and important dynamical systems, including those with linear transitions and bounded white noise.

3 Strategies for dealing with observation delays

In this section, we cover several algorithms for learning and planning in delayed environments. Among those covered is the *augmented* MDP formalism espoused in prior works [13,4] for planning in delayed observation settings. We also cover a less burdensome strategy using eligibility traces, specifically Sarsa(λ) [21], which has been used in prior work to obtain memoryless policies in Partially Observable MDPs (POMDPs) [16]. We give a brief overview of the theoretical connections between the CDMDP setting and several popular representations, including factored MDPs and POMDPs. Finally, we introduce a new method for planning that can be used in the special cases of delayed environments covered above and that overcomes the exponential growth in the state space of augmented MDPs and has theoretical and empirical advantages over the often suboptimal Sarsa(λ).

3.1 Solution 1: The "wait" agent

The first solution we consider is the **wait agent**, which "waits" for k steps, until the current observation comes through, and then acts using the optimal action in the undelayed MDP. More formally, this approach corresponds to a CDMDP policy of $\pi(I_k) = \pi^*(s)$ if $I_k = (s, \emptyset^k)$, and $\emptyset$ otherwise. Here, $\emptyset$ is the "wait" action. While this approach may seem naive, in practice it is probably one of the most common, especially in environments where time is not particularly of the essence. Unfortunately, policies derived from this strategy will not, in general, provide satisfactory solutions to the CDMDP planning problem. Instead, the agent's resulting policy will likely be suboptimal, as it is essentially losing potential reward on every "wait" step. Furthermore, some environments, such as Mountain Car, where the agent is rarely at a standstill, will not provide a natural "wait" action, rendering this solution inapplicable. Still, when appropriate, the "wait" strategy provides the most cautious, and in limited cases (where a single misstep could be catastrophic and the agent has no failsafe mechanisms) the only practical solution.

3.2 Solution 2: Memoryless policies

Another intuitive planning approach is to just treat the CDMDP as an MDP and use a **memoryless** policy based on a policy (possibly the optimal one) for the undelayed MDP. That is, $\pi(I_k) = \pi'(s)$ if $I_k = (s, a_1, \ldots, a_k)$, where π' is a policy over S. In some environments, this simple solution can produce reasonable policies, especially if the delay is relatively small compared to the magnitude of the state transitions. For the CDMDP learning problem, searching for the best policy that ignores delay is intimately connected to the search for good memoryless policies in POMDPs. One known technique that has shown empirical success in the latter theater, without incurring an undue computational burden, is the use of eligibility traces [16], particularly in the online value-function-learning algorithm Sarsa(λ), which is an online model-free learning algorithm. At each timestep, having just reached state s_t, received reward r_t, and chosen action a_t, Sarsa(λ) performs the following operations, with $Q(s, a)$

being the state/action value function ($V(s) = \max_a Q(s, a)$) and α as the learning rate:

```
$e(s_{t-1}, a_{t-1}) \leftarrow 1$
$\delta \leftarrow r_t + \gamma Q(s_t, a_t) - Q(s_{t-1}, a_{t-1})$
for all $s \in S$ and $a \in A$
    $Q(s, a) \leftarrow Q(s, a) + \alpha\delta e(s, a)$
    $e(s, a) \leftarrow \gamma\lambda e(s, a)$
```

An action is then chosen based on the exploration policy and the process repeats. Using $\lambda > 0$ "blurs" together temporally proximate states and can mitigate the effect of partial observability (in our case, delayed observations). Intuitively, in the delayed setting, eligibility traces tie the value of a state more closely to states that appear soon after it, thus the effects of delay are dampened because the value of the observed (but delayed) state is more heavily dependent on the true (but unobserved) current state. As such, we include Sarsa(λ) in our empirical study (see Sect. 5) of the CDMDP Learning Problem. We further explore the connection between CDMDPs and POMDPs in Sect. 3.4.

3.3 Solution 3: The augmented approach

The traditional method for modeling MDPs with constant delay is the **augmented** approach [4], which involves explicitly constructing an MDP equivalent to the original CDMDP in the much larger state space $S \times A^k$. The formal construction of such an *augmented MDP* is covered in previous work [13], but we briefly recount it here:

The construction takes a CDMDP $C = \langle S, A, P, R, \gamma, k\rangle$ and produces a regular MDP $M = \langle \mathbf{I}_k, A, P', R', \gamma\rangle$. The new state space, $\mathbf{I}_k = S \times A^k$, contains information states in the form of $(s, a_1, \ldots, a_k)$ for $s \in S$ and $a_j \in A$. The new transition probabilities and reward function can be defined accordingly: for $I_k = (s, a_1, \ldots, a_k)$,

$$P'(I_k, a, I'_k) = \begin{cases} P(s, a_0, s'), & \text{if } I'_k = (s', a_2, \ldots, a_k, a) \\ 0, & \text{otherwise} \end{cases}$$
$$R'(I_k) = R(s).$$

We note that prior results [13] have shown that a variant of this construction can be performed in the presence of action delays, even if the observation, action, and reward delays have different values. Once the augmented MDP has been built, one can use any of the standard MDP planning algorithms to determine $V^*(I_k)$ for $I_k \in S \times A^k$. The corresponding optimal policy is known to be an optimal policy for the CDMDP [3]. While the augmented approach is sound and complete, in practice it presents several difficulties due to the exponential growth of the state-space. First, this expansion renders traditional MDP planning algorithms intractable for all but the smallest values of k. In Sect. 4.1, we show that the exponential state-space growth is unavoidable in general, but in Sect. 3.5, we describe an approach that averts this computational burden and provides optimal or near-optimal policies in the special cases from Sect. 2. Furthermore, extending the augmented approach to the learning setting naively could lead to an exponential sampling requirement as the algorithm attempts to experience all state-actions in $S \times A^k \times A$. In Sect. 4.3, we outline a more efficient way to learn the augmented MDP with a polynomial number of samples. However, we first mention a few more impractical solutions that suffer from similar state-space expansions as the augmented approach.

3.4 Other intractable methods

As mentioned in the discussion of eligibility traces, finding optimal policies in CDMDPs is intimately connected to determining optimal policies in POMDPs. The connection emerges because the states of the augmented MDP representing a CDMDP can be mapped directly to what in POMDPs are termed "belief states", that is probability distributions over the underlying states. Unfortunately, constructing a traditional POMDP transition function (where transitions are Markovian given the belief state) is problematic because belief states do not necessarily encode the last k actions (because two augmented states can map to the same belief state, losing the ability to recover the action k steps ago). One could create new "states" that encode the actions as well, but the result is exactly the augmented model. Thus, using POMDP planning techniques [11] in the CDMDP planning setting gains no benefit, in terms of worst-case analysis, over the augmented approach.

Representing the CDMDP as a factored MDP with one factor for each of the k previous actions, as well as factors for the most recent observation and current action, is another representational trick that is of no avail in the worst case. At first, this may seem surprising because the probability of the next observed state (from $k - 1$ steps prior) can be easily predicted given this model. Unfortunately, the problem of choosing the optimal next action given the current factor values (the CDMDP planning problem), is no easier in this representation, as the different permutations of values assigned to the factors each correspond to a specific state in the augmented MDP.

Finally, we mention an interesting connection between the CDMDP setting with a "wait" action and POMDPs with observation actions. Intuitively, in the delayed setting, simply being able to sit still for a number of timesteps provides the agent with information as a more contemporary observation is uncovered on each step. In such a setting, an optimal agent would need to decide how many steps to "wait" in every situation until it reached a belief state where it could act with high certainty on the distribution of outcomes. More generally, this can be seen as a variant of the only-costly-observable MDP (OCOMDP) [8] , which has been studied in a limited form in the learning setting. On the planning side, this representation requires reasoning about when to observe, a problem that quickly devolves into POMDP planning, and in practice algorithms in this setting yield only approximate solutions [27].

All the strategies discussed in this section lead to worst-case complexities at least comparable to the augmented approach. Furthermore, analyses of the loss in precision caused by using approximate solutions with these representations will often require environmental assumptions beyond the scope of this work. The focus of this paper is on *practical* solutions for environments with delayed dynamics under reasonable real-world assumptions. Hence, we do not further discuss these generally intractable solutions, comparing simply against the augmented approach.

3.5 Solution 4: A new approach, Model Based Simulation (MBS)

We now introduce a planning algorithm, Model Based Simulation (MBS), designed for the restricted CDMDP cases from Sect. 2. The theoretical difficulties of extending this approach to general CDMDPs are discussed in Sect. 4. The intuition behind MBS is that, in a deterministic or benignly stochastic environment, given I_k, one can use P to "simulate" the most likely single-step outcomes of the last k actions, starting from the last observed state, thus determining, or at least closely approximating, the current state of the agent. In the deterministic cases, this prediction is straightforward. In the other two cases, (mildly stochastic

Algorithm 1 Model Based Simulation

```
1: Input: A CDMDP $M = \langle S, A, P, R, \gamma, k \rangle$, and $I_k = (s, a_1, a_2, \ldots, a_k) \in S \times A^k$.
2: Output: The optimal action $a^* = \pi^*(I_k)$.
3: Construct a regular MDP $\bar{M} = \langle S, A, \bar{P}, R, \gamma \rangle$ where $\bar{P}(s, a, s') = 1$ for the most likely
   (when $S$ is finite) or expected (when $S$ is continuous) outcome of $a$ in $s$.
4: Find the optimal value function $\bar{V}^*$ and an optimal policy $\bar{\pi}^*$ for $\bar{M}$.
5: Compute the current (but unobserved) state $\bar{s}$ by applying action sequence $(a_1, \ldots, a_k)$
   to $s$ according to $\bar{P}$.
6: Return $\bar{\pi}^*(\bar{s})$.
```

and bounded noise) the algorithm will use the most likely or expected outcome, respectively, to simulate each step. Notice that when $k > 1$ the outcome of this sequence of simulations may not be the most likely current state given I_k, but we shall see that in the restricted cases we consider, the error in this approximation is bounded. The MBS algorithm appears in Algorithm 1.[2]

Extending MBS to the learning setting is fairly straightforward in the context of finite CDMDPs (Cases I and III). One needs only to employ a model-based RL algorithm such as R-max [6] to learn the parameters (P and R) of the underlying no-delay MDP. However, to extend MBS to *continuous* CDMDPs, simply discretizing the environment is not sufficient because this approach can easily turn deterministic (Case II) or slightly perturbed (Case IV) state transitions into far less benign dynamics, making the action simulations unsuitable. Instead, we require a method that trains a model of the transitions in the continuous space itself, but still plans in the discretized space (in order to make valid comparisons against the policies of the other finite-space algorithms). The next section defines such an algorithm.

Algorithm 2 Model parameter approximation

```
1: Input:
2:         A collection of $N$ sample instances $X = \{(s_i, a_i, r_i, s'_i) \mid i = 1, 2, \ldots, N\}$
3:         $S$, $A$, $\gamma$ and $R_{\max}$ from a continuous MDP and a set of discrete states $\hat{S}$
4:         Function approximators $T_A$ and $R_A$
5:         The current continuous observation $s$
6: Output: The action to be taken from $s$.
7: Train $T_A$ and $R_A$ using $X$.
8: Construct discrete MDP $\hat{M} = \langle \hat{S}, A, \hat{P}, \hat{R}, \gamma \rangle$; for any $\bar{s} \in \hat{S}$ and $a \in A$:
9: if we have enough samples in $X$ then
10:    use maximum-likelihood estimates
11: else if $T_A$ and $R_A$ have high confidence then
12:    generate an artificial sample set $X'$ using $T_A$ and $R_A$, build model using $X \cup X'$
13: else
14:    $\hat{P}(\bar{s}, a, \bar{s}) = 1$ and $\hat{R}(\bar{s}, a) = R_{\max}$.
15: end if
16: Find the optimal value function $\hat{V}^*$ and an optimal policy $\hat{\pi}^*$ for $\hat{M}$.
17: $\hat{s} = \text{Discretize}(s)$
18: Return $\hat{\pi}^*(\hat{s})$.
```

[2] Note: for continuous MDPs, some steps may require approximation, see Sect. 4.2.

3.6 Model parameter approximation

Model Parameter Approximation, or MPA, is a model-based RL algorithm designed for MDPs with bounded, continuous state and action spaces. It combines techniques from two other RL algorithms. First, MPA is closely related to Lazy Learning [2], which uses locally weighted regression to build approximations of the MDP dynamics and then plans in a discretized version of the MDP, using the trained regressor as a generative model in order to "sample" the transition and reward functions in areas of the discretized MDP where it lacks "real" experience. MPA performs a similar construction, but it can use any function approximator to model both the relative transition and reward functions. MPA also utilizes a model-based exploration strategy (not provided in Lazy Learning) based on the R-max algorithm. Specifically, MPA tags state/action pairs as "known" or "unknown" based on the amount of experience in each discrete state (in practice this threshold may be governed by the sampling requirements of the function approximators) and encouraging exploration of the unknown areas. As in Lazy Learning, because a discretized model is used, the value-iteration operation will provably converge, though we cannot guarantee the values will be identical to the true value function of the continuous MDP. The full MPA algorithm is presented in Algorithm 2.

We emphasize that MPA is a model-based reinforcement-learning algorithm for no-delay MDPs whose planning component is very similar to MBS without simulation. Therefore, we can use MBS and MPA together in the continuous CDMDP learning setting with only a few modifications. First, MPA's instance set, X, needs to contain one-step transitions, so we need to pair together the currently observed state and reward with the action from k steps ago before adding them to X. Also, we perform MBS's simulation *before* the discretization of the current state using MPA's transition function approximator, T_A, to apply the action sequence (using the expected one-step outcomes). We then discretize the outcome of that simulation and use the appropriate action. This CDMDP learning algorithm, MBS + MPA, produces a "discretized" policy, valid for comparison against the other algorithms we will investigate in Sect. 5. This comparability is one of the reasons MPA was chosen for testing with MBS over other suitable continuous model-based RL algorithms such as Kernel Based Prioritized Sweeping [10].

4 Theoretical analysis of delayed problems

In this section, we develop several theoretical properties of the CDMDP planning and learning problems for CDMDPs as described in Sect. 2. Our treatment includes a hardness result in the general case, positive results for the four special cases, and an efficient (in terms of sample complexity) way to learn augmented MDPs.

4.1 Planning results I: the general case

The first two solutions from Sect. 3 have polynomial-time bounds in the planning setting as they simply involve finding the optimal policy for the no-delay underlying MDP. However, examples are easily constructed where these approaches fail to solve the CDMDP planning problem correctly. Conversely, the expensive augmented approach represents a sound and complete method for finding an optimal policy. Although in certain cases it is unnecessary to fully expand the state space to $S \times A^k$, Theorem 1 below shows that converting the CDMDP representation to an equivalent regular MDP representation induces an exponential expansion over the size of the compact CDMDP model in the worst case. The proof is provided in Appendix A.

Theorem 1 *The smallest regular MDP $\bar{M} = \langle \bar{S}, A, \bar{P}, \bar{R}, \gamma \rangle$ induced by a finite CDMDP $M = \langle S, A, P, R, \gamma, k \rangle$ has a lower bound of $\left|\bar{S}\right| = \Omega(|A|^k)$.*

The exponential increase in the number of states suggests that this approach is intractable in general, and the next theorem establishes that it is unlikely the CDMDP planning problem can be solved in polynomial time. The proof is presented in Appendix B.

Theorem 2 *The general CDMDP planning problem is NP-Hard.*

A more complicated reduction from 3-SAT shows this problem is indeed *strongly* NP-Hard. We note that if P≠NP, then Theorem 1 would be a direct consequence of Theorem 2 since an MDP can be solved in time polynomial in the size of its representation. However, Theorem 1 gives a stronger result, showing an exponential blowup in representation is unavoidable when converting a CDMDP to an MDP, even if P = NP. The NP-Hardness result for CDMDP planning motivates the search for constrained cases where one can take advantage of special structure within the problem to avoid the worst case. We now provide theoretical results concerning the four special cases previously defined.

4.2 Planning results II: special cases

The following results provide bounds on $\left\|\bar{V}^* - V^*\right\|_\infty$, where $\bar{V}^*$ is the value function for $\bar{\pi}^*$ computed by MBS in its deterministic approximation $\bar{M}$ (*c.f.* Algorithm 1), and V^* is the true CDMDP value function. These bounds are also accuracy bounds for answering the CDMDP planning problem using $\bar{M}$ instead of M and can be used to derive the *actual* online performance bounds when using greedy policies w.r.t. $\bar{V}^*$ compared to the optimal CDMDP policy [22].

We begin with the finite-state cases, starting with the more general "mildly stochastic" setting (Case III) where MBS will assume that the last k transitions have each had the most likely one-step outcome.

Theorem 3 *In Case III, $\left\|\bar{V}^* - V^*\right\|_\infty \leq \frac{\gamma\delta R_{\max}}{(1-\gamma)^2}$. In other words, MBS solves the CDMDP planning problem for such CDMDPs with this accuracy in polynomial time.*

Proof We first bound the error on the one-step Bellman backup of the deterministic approximation. That is, we compare $R(s) + \gamma \max_a \sum_{s'} P(s, a, s')V^*(s')$ for arbitrary state s versus a backup in the deterministic approximation, $R(s) + \gamma \max_a V^*(T(s, a))$, where $T(s, a) = \arg\max_{s'} P(s, a, s')$ is the most likely outcome of a taken in s. Define $a' = \text{argmax}_a \sum_{s'} P(s, a, s')V^*(s')$ and $a'' = \text{argmax}_a V^*(T(s, a))$. On the one hand,

$$\left(R(s) + \gamma \max_a \sum_{s'} P(s, a, s')V^*(s')\right) - \left(R(s) + \gamma \max_a V^*(T(s, a))\right)$$
$$\geq \gamma \sum_{s'} P(s, a', s')V^*(s') - \gamma V^*(T(s, a'))$$

$$
\begin{aligned}
&= \gamma \sum_{s'} P(s,a',s')\left(V^*(s') - V^*(T(s,a'))\right)\\
&= \gamma \sum_{s' \neq T(s,a')} P(s,a',s')\left(V^*(s') - V^*(T(s,a'))\right)\\
&\geq \gamma \sum_{s' \neq T(s,a')} P(s,a',s')\left(-\frac{R_{\max}}{1-\gamma}\right)\\
&\geq -\frac{\gamma\delta R_{\max}}{1-\gamma},
\end{aligned}
$$

where the last inequality is due to our assumption that $\sum_{s' \neq T(s,a')} P(s,a',s') \leq \delta$. Similarly, we can show that

$$\left(R(s) + \gamma \max_a \sum_{s'} P(s,a,s')V^*(s')\right) - \left(R(s) + \gamma \max_a V^*(T(s,a))\right) \leq \frac{\gamma\delta R_{\max}}{1-\gamma}.$$

Therefore, we have

$$\left|\left(R(s) + \gamma \max_a \sum_{s'} P(s,a,s')V^*(s')\right) - \left(R(s) + \gamma \max_a V^*(T(s,a))\right)\right| \leq \frac{\gamma\delta R_{\max}}{1-\gamma}.$$

By a well-known fact about MDP approximation solutions (e.g., [17]), the two value functions differ by at most $\frac{1}{1-\gamma}$ times the single-step Bellman backup difference:

$$\left\|\bar{V}^* - V^*\right\|_\infty \leq \frac{\gamma\delta R_{\max}}{(1-\gamma)^2}.$$

Using MBS explicitly computes a value function for a deterministic approximation (Step 4 of MBS), which is subject to the bound above in relation to the true CDMDP value function. Answering the CDMDP planning problem within this accuracy can then be done by approximating the current state s through simulation and comparing $\bar{V}^*(s)$ to the reward bound θ. The major operation for MBS is the computation of $\bar{V}^*$ for a deterministic MDP $\bar{M}$, which can be done in $O(SA + S^3)$ [15]. □

We note that, by definition, $\bar{V}^*$ has taken the k-step prediction error into account; therefore, Theorem 3 provides a bound (indirectly) for the performance of MBS when it has to predict forward k steps using an inaccurate model. The bound above is only practically useful for small values of δ, because larger values could cause $\bar{M}$ to be a poor approximation of M. At the opposite extreme, setting $\delta = 0$, we arrive at the following result for Case I:

Corollary 1 *In Case I, MBS solves the CDMDP planning problem exactly in polynomial time.*

In the continuous cases (II and IV), computing $\bar{V}^*$ and its maximum, even in the undelayed case, requires approximation (e.g. discretization [17] or interpolation [18]) that will introduce an additional error, denoted ϵ, to $\bar{V}^*$ as compared to V^*. Computing $\bar{V}^*$ will also require some (possibly not polynomially bounded) time, T. In Case IV, we assume the magnitude of the noise is bounded by Δ and the optimal CDMDP value function is Lipschitz continuous with constant C_v, leading to the following result.

Theorem 4 *In Case IV, assuming an approximation algorithm for computing $\bar{V}^*$ within ϵ accuracy in time T, MBS solves the CDMDP planning problem with accuracy $\frac{2\gamma C_v \Delta}{1-\gamma} + \epsilon$ in time polynomial in the size of the input and T.*

Proof We establish an error bound on the one-step Bellman backup of the deterministic approximation using the Lipschitz continuity condition. We have assumed there exists $C_v > 0$ such that for any $I = (s, a_1, a_2, \ldots, a_k)$ and $I' = (s', a_1, a_2, \ldots, a_k)$, we have

$$\left|V^*(s, a_1, a_2, \ldots, a_k) - V^*(s', a_1, a_2, \ldots, a_k)\right| \leq C_v \left\|s - s'\right\|.$$

The backup error β_V is defined by:

$$\begin{aligned}
&\beta_V(s, a_1, a_2, \ldots, a_k)\\
&\quad = \left(R(s, a_1) + \gamma \max_a \left\{\int_S p(s' \mid s, a_1) V^*(s', a_2, \ldots, a_k, a) ds'\right\}\right)\\
&\qquad - \left(R(s, a_1) + \gamma \max_a \left\{V^*(s_0, a_2, \ldots, a_k, a)\right\}\right)\\
&\quad = \gamma \max_a \left\{\int_S p(s' \mid s, a_1) V^*(s', a_2, \ldots, a_k, a) ds'\right\} - \gamma \max_a \left\{V^*(s_0, a_2, \ldots, a_k, a)\right\}
\end{aligned}$$

where

$$s_0 = \int_S p(s' \mid s, a_1) s' ds'$$

is the expected next observation due to the transition from s by taking action a_1. Let

$$a' = \arg\max_a \int_S p(s' \mid s, a_1) V^*(s', a_2, \ldots, a_k, a) ds'$$

and

$$a'' = \arg\max_a V^*(s_0, a_2, \ldots, a_k, a).$$

We now show that $|\beta_V(s, a_1, a_2, \ldots, a_k)| \leq 2\gamma C_v \Delta$ for any $s, a_1, \ldots, a_k$. By definitions of a' and a'', we can rewrite $\beta_V(s, a_1, a_2, \ldots, a_k)$ as:

$$\gamma \int_S p(s' \mid s, a_1) V^*(s', a_2, \ldots, a_k, a') ds' - \gamma V^*(s_0, a_2, \ldots, a_k, a'').$$

On the one hand,

$$\begin{aligned}
&\beta_V(s, a_1, a_2, \ldots, a_k)\\
&\quad \geq \gamma \int_S p(s' \mid s, a_1) V^*(s', a_2, \ldots, a_k, a'') ds' - \gamma V^*(s_0, a_2, \ldots, a_k, a'')\\
&\qquad (\because \text{ definition of } a')\\
&\quad \geq \gamma \int_S p(s' \mid s, a_1) \left[V^*(s_0, a_2, \ldots, a_k, a'') - 2C_v\Delta\right] ds' - \gamma V^*(s_0, a_2, \ldots, a_k, a'')\\
&\qquad (\because \text{ any } s' \text{ with } p(s' \mid s, a_1) > 0 \text{ must be } 2\Delta\text{-close to } s_0)\\
&\quad = -2\gamma C_v \Delta.
\end{aligned}$$

Similarly, we can show that $\beta_V(s, a_1, a_2, \ldots, a_k) \leq 2\gamma C_v \Delta$. Combining these two results, we have $|\beta_V(s, a_1, \ldots, a_k)| \leq 2C_v\Delta$.

From here, the proof of the Theorem continues along the lines of Theorem 3. The bound on the error of the backup is extended to a bound on the error of the value function by introducing a factor of $\frac{1}{1-\gamma}$:

$$\left\| \bar{V}^* - V^* \right\|_\infty \leq \frac{2\gamma C_v \Delta}{1-\gamma}.$$

All the operations in MBS remain polynomial as in Theorem 3, except for step 4, which must be carried out by the approximation algorithm, introducing further error ϵ. Let $\tilde{V}^*$ be the approximate value function returned by the approximation algorithm. By assumption, $\left\| \bar{V}^* - \tilde{V}^* \right\|_\infty \leq \epsilon$. Therefore,

$$\left\| \tilde{V}^* - V^* \right\|_\infty \leq \left\| \tilde{V}^* - \bar{V}^* \right\|_\infty + \left\| \bar{V}^* - V^* \right\|_\infty \leq \epsilon + \frac{2\gamma C_V \Delta}{1-\gamma}.$$

□

Similarly to Case III, this bound is only of interest if Δ and ϵ are small. By setting $\Delta = 0$, we arrive at the following result that says planning in deterministic continuous CDMDPs is the same as in their equivalent undelayed ones:

Corollary 2 *In Case II, the MBS algorithm, using an approximation algorithm to compute $\bar{V}^*$ within ϵ accuracy in time T, can answer the CDMDP planning problem with accuracy ϵ in time polynomial in the size of the input and T.*

4.3 Efficiently learning an augmented MDP

A simplistic approach to the general CDMDP learning problem would be to apply standard RL algorithms in the augmented MDP state space. We will refer to this strategy as *naive augmented learning*. While theoretically sound, this tack requires gathering experience for every possible I_k (an exponential sampling requirement). A preferred alternative is to instead learn the one-step model from experience, then build the augmented model and use it to plan in conjunction with an algorithm, like R-max [6], that facilitates exploration. While this *compact augmented learning* still suffers in the worst case from the unavoidable exponential burden of planning (Theorems 1 and 2), its *sampling* requirement is polynomially bounded. Thus, compact augmented learning represents a practical (in terms of sample complexity) extension of the augmented approach to the CDMDP learning setting. Theorem 5 below makes this claim formal on finite MDPs.

Theorem 5 *The compact learning approach, is PAC-MDP* [23]. *In particular, for any* $\epsilon, \delta > 0$, *the non-stationary policy of a compact learning agent is ϵ-optimal except in* [3]

$$\tilde{O}\left(\frac{|S|^2 \, |A| \, R_{\max}^3}{\epsilon^3 (1-\gamma)^6} \right)$$

time steps during the whole run of the algorithm, with probability at least $1 - \delta$.

Proof (Sketch) The proof is essentially identical to the proof for R-max [12].[4] To see why, we first observe that the key lemmas in the proof, including the implicit-explore-or-exploit

[3] $\tilde{O}$ is similar to the big-O notation except that it ignores the logarithmic factors.

[4] Although their proof is for undiscounted, finite-horizon RL algorithms, the same proof with minor changes holds valid for the discounted, infinite-horizon case.

lemma [12, Corollary 8.4.5] and the simulation lemma [12, Corollary 8.5.4], hold in the augmented MDP. We can thus define known-state augmented MDPs as usual, and define p_t as the probability of reaching an unknown augmented state within

$$H = O\left(\frac{\log \frac{R_{\max}}{\epsilon(1-\gamma)}}{1-\gamma}\right) = \tilde{O}\left(\frac{1}{1-\gamma}\right)$$

steps at time t. Consider two cases:

- When p is below a threshold $p^* = O\left(\frac{\epsilon(1-\gamma)}{R_{\max}}\right)$ (we call this an exploitation step), the implicit-explore-or-exploit lemma guarantees that the non-stationary policy at this time step is ϵ-optimal.
- Otherwise, an unknown state will be reached within H steps with nontrivial probability at least p^* (we call this an exploration step), then Hoeffding's inequality [9] can be used to bound the total number of exploration steps with high probability [12, Lemma 8.5.2]. Observe that although the augmented MDP has exponentially many states ($|\mathbf{I}_k| = |S| \cdot |A|^k$), it in fact has exactly the same set of parameters to learn as the underlying undelayed MDP (*c.f.*, Sect. 3.3); in other words, there are at most $O\left(m\,|S|^2\,|A|\right)$ steps in which the agent enters an unknown state, where m is the number of visits of a state–action pair to make it known. Consequently, we have exactly the same bound of the number of exploration steps as given in Lemma 8.5.2 of Kakade's proof [12].

Putting all these together, a *compact* augmented learner using R-max employs a policy that is ϵ-optimal except in at most

$$\tilde{O}\left(\frac{Hm\,|S|^2\,|A|\,R_{\max}}{\epsilon(1-\gamma)}\right) = \tilde{O}\left(\frac{m\,|S|^2\,|A|\,R_{\max}}{\epsilon(1-\gamma)^2}\right)$$

steps. Setting $m = \tilde{O}\left(\frac{S}{\epsilon^2(1-\gamma)^4}\right)$ suffices to estimate a model accurately enough so that the value functions are ϵ-close to the true value function. Using this value of m, we have the sample complexity bound of exploration as desired. $\square$

5 Empirical algorithm comparisons

To provide some concrete grounding to the theoretical observations, we now evaluate several of the methods discussed in Sect. 3 in the learning setting for each of the four cases. Agents were evaluated in episodic domains based on average cumulative reward over 200 episodes, with a cap of 300 steps per episode. Starting states for each episode were randomly selected but were consistent across different agents. All data points represent an average over 10 runs. Several of the strategies proposed thus far were originally designed for the delayed planning case, but we extended them to the learning setting in the following ways. The "wait agent" approach was used in environments where a "wait" action was permissible using R-max[5] in the finite-state setting and with MPA for continuous environments. Several variants of the memoryless-policy strategy were appraised, including model-based RL algorithms, R-max and MPA, as well as Sarsa(0)[6], Sarsa(.9), and "Batch" versions of Sarsa (B-Sarsa) that used

[5] Unless otherwise noted, all variants of R-max used a "known state threshold" of $m = 3$ samples when domains appeared stochastic to the learner.

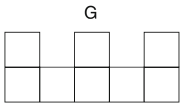
[6] Variants of Q(λ)-learning were also tried, yielding similar results to Sarsa(λ).

experience replay [14] every 1000 steps. The batch variants were included to increase the validity of comparisons to the model-based methods. The Sarsa learning rate was set to .3 (empirically tuned) and exploration in these cases was guided by optimistic initialization of the value function along with an ϵ-greedy [25] approach for picking actions, with ϵ initialized to .1 and decaying by a factor of .95 per episode. The learners using eligibility traces used "replacing" traces (in conformance with the presentation in Sect. 3.2), which fared better empirically then "accumulating traces". In the continuous domains, we used the same discretization scheme for all the memoryless learners and MBS + MPA. Due to the large number of variations, only the best and worst of these "memoryless" approaches are plotted for each environment. For the augmented approaches, we investigated both the *naive* and *compact* augmented learners described in Sect. 4.3, with planning taking place in the augmented MDPs using R-max. We also evaluated a *naive* augmented Sarsa(λ) learner in the augmented MDP state-space. Unfortunately, the computational burden of planning caused by the exponential state space expansion made these augmented approaches infeasible beyond delays of $k = 5$. Finally, for MBS, we again used R-max or MPA, as appropriate. All the variants of MPA (MPA, Wait + MPA, MBS + MPA) employed Locally Weighted Progression Regression (LWPR) [26] to approximate the transition function, and an averager to approximate the reward function. An LWPR confidence of 85% was used to separate "known" and "unknown" states.

5.1 Delayed W-maze I: a deterministic finite environment

We begin with a deterministic finite (Case I) domain, the "W-maze", as depicted in Fig. 1 (left). The agent starts in a random cell and its goal is to escape the maze through the top center square by executing the "up" action. All steps within the maze garner a reward of -1. The environment is designed to thwart memoryless approaches, which have trouble finding the right situation to begin going "up" and instead alternate between the extreme branches.

Figure 1 (right) shows the results of this experiment. The "wait" agent performs well in this environment, but sub-optimally for $k > 0$. In contrast, MBS + R-max quickly achieves optimality for all delay values. The best memoryless performer was B-Sarsa(.9), but its performance drops well below the random agent at higher delays. The worst memoryless learner was R-max, which fails to learn the transition function for $k>0$. The compact augmented learner performs comparably to MBS + R-max, but the planning for this method becomes



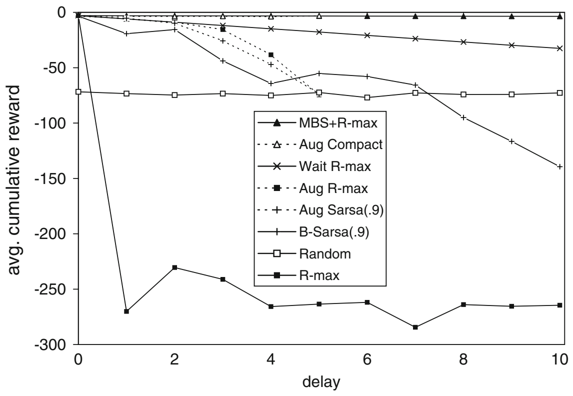


Fig. 1 Left: W-maze. Right: experimental results for deterministic W-maze

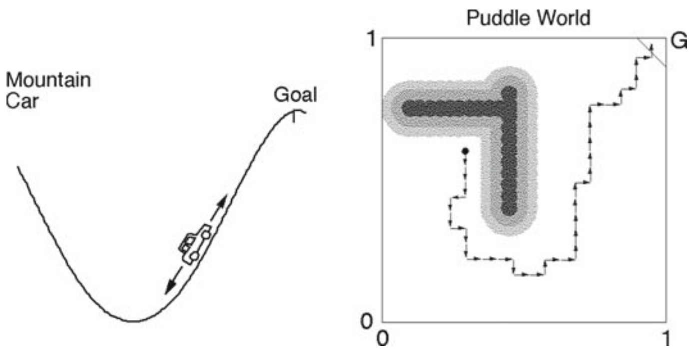


Fig. 2 Two benchmark problems in our empirical studies [24]

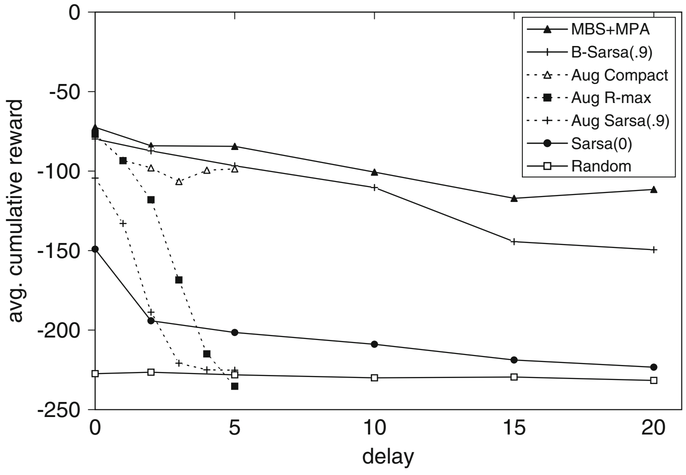


Fig. 3 Mountain Car Results, various strategies shown

intractable beyond a delay of $k = 5$. As expected, the naive augmented learners see a significant performance drop-off as delay increases. Unlike the memoryless approaches, which learn fast but can't represent the optimal policy, these learners are too slow to learn from the set of samples available to them.

5.2 Delayed Mountain Car: a case II environment

In a progression towards more realistic domains, we further investigated these algorithms in a domain with deterministic *continuous* dynamics (Case II), a delayed version of "Mountain Car" [25], which was an event in the First Annual Reinforcement Learning Competition. This oft studied domain involves a car attempting to reach the top of a steep incline, as pictured in Fig. 2 (left). In order for the car to reach its goal, it must build up momentum, often by climbing the smaller slope to the left. The agent perceives the environment through two continuous variables, representing the car's location and velocity. The car has 3 actions (forward, neutral, reverse) and rewards of -1 for all steps and 0 at the top of the hill. We set the agents' initial states randomly, but used the same seeds across learners. For the "memoryless", and "augmented" approaches, we continued to use the algorithms described in the previous

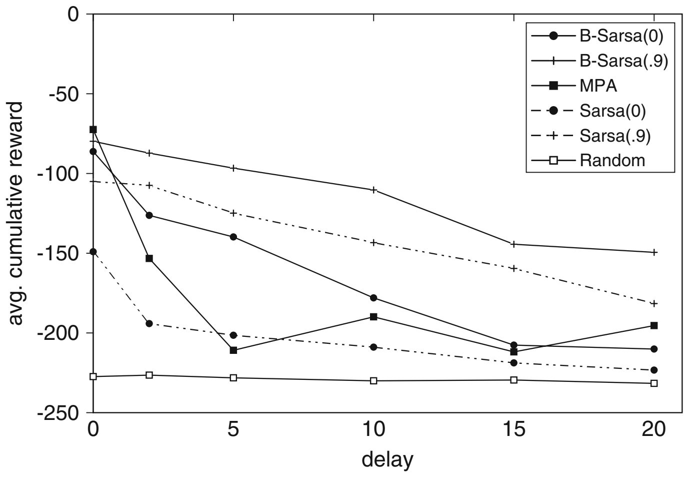


Fig. 4 Results for memoryless learners in the Mountain Car environment

section and overlaid a 10 × 10 (empirically tuned) grid for discretization. The "wait" agent strategy was not applicable because this domain has momentum. MPA was implemented using LWPR as previously described. The results are illustrated in Fig. 3. Again, the best performer was MBS + MPA, which has the advantage of modeling continuous actions and efficiently compensating for delay. However, for many delay values, Batch Sarsa(.9) performed almost as well, because action effects in Mountain Car are quite small. By focusing on the results of the memoryless learners (Fig. 4), we see the clear benefit of eligibility traces as both B-Sarsa(.9) and Sarsa(.9) outperform B-Sarsa(0), Sarsa(0) and MPA (without MBS) when $k > 0$. The augmented learners again falter, due to their inefficient use of samples, rarely making it to the top of the hill for $k > 2$.

5.3 Delayed W-maze II: a stochastic finite environment

We also considered a mildly stochastic (Case III) version of W-maze, where actions succeed with a probability of .7 and "slip" in one of the other three directions with probability .1 each. Because of the non-determinism, learners using R-max used a "known state threshold" of $m = 5$ samples in this case. The results of this experiment are illustrated in Fig. 5. Despite the non-determinism in the domain, MBS + R-max performed comparably to the compact augmented learner and outperformed all of the other approaches. The memoryless approaches all flounder with increasing delay, being outdone even by the naive augmented R-max and "wait" learners. As in the deterministic case, the failure of the memoryless learners stems from their inability to represent the optimal policy. Interestingly, the cautious "wait" learner outperforms both the naive augmented and memoryless learners in this domain as it neither tries to learn a complex model nor settles for suboptimal non-wait steps. But, the "wait" agent is still no match for MBS + R-max, even in this stochastic domain.

5.4 Delayed Puddle World: a case IV environment

Finally, we investigated a Case IV environment, stochastic Puddle World [5] where action outcomes were perturbed by bounded Gaussian noise. The 2D environment contains two

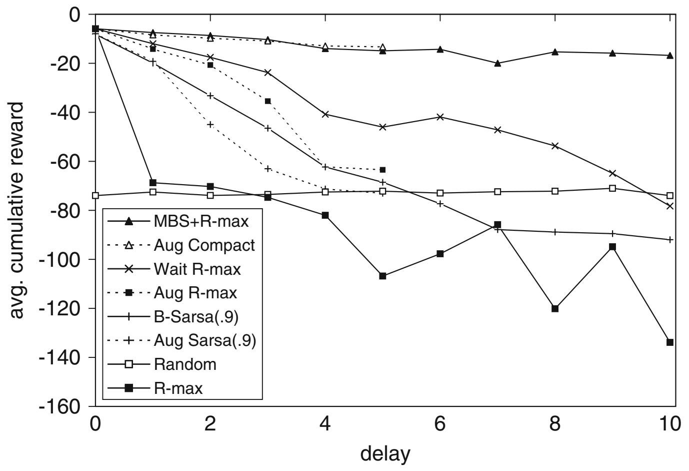


Fig. 5 Experimental results for stochastic W-maze

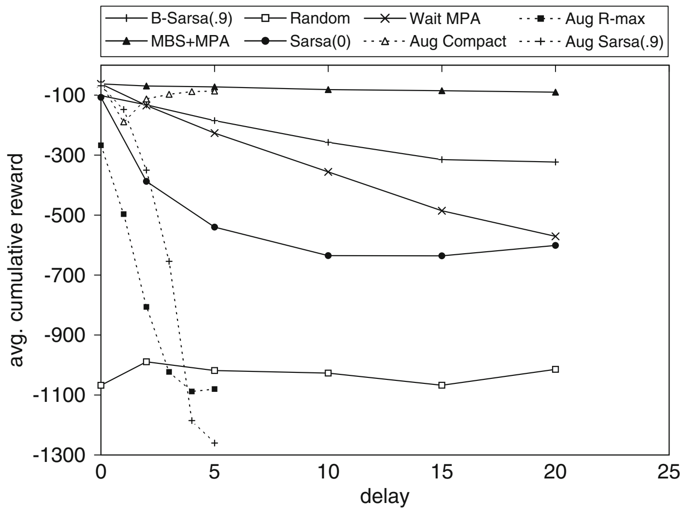


Fig. 6 Experimental results for stochastic Puddle World

puddles and a goal. A sample drawing of a puddle world appears in Fig. 2 (right). The observables at each step are the agent's X and Y coordinates (bounded in [0, 1]). The available actions are movement in the four compass directions, each with an expected magnitude of .05, but perturbed by Gaussian noise capped in magnitude by .01. Steps within the puddles garner large negative rewards while all other steps yield -1. We set the initial states randomly, though agents were not allowed to start in the puddles, and we again used the same seeds across experiments. In each of the 10 runs, the puddles were randomly placed and sized. A 10×10 tiling was used for discretization. The batch learners used experience replay every 2500 steps because of noise effects. The results are reported in Fig. 6. MBS + MPA clearly outperforms its memoryless counterparts, though eligibility traces help maintain performance with increasing delay. As with Mountain Car, MBS + MPA outperforms some augmented

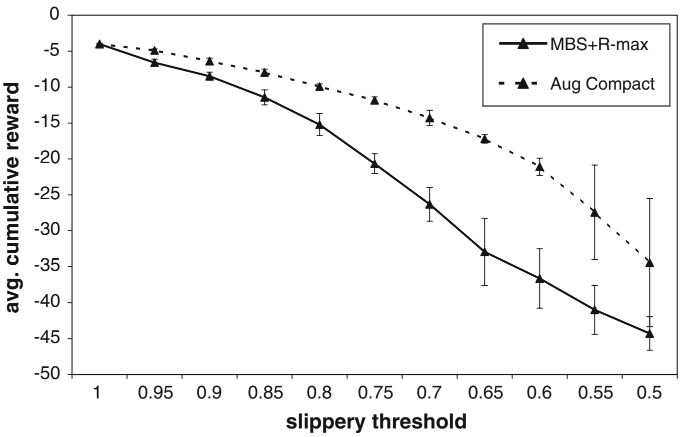


Fig. 7 Effect of noise on the performance of MBS + R-max and the compact augmented learner in the stochastic W-maze environment. While the performance of the latter is more consistent, it suffers from exponential time complexity w.r.t. k. Delay is set to $k = 5$ to keep the augmented MDP planning tractable

learners at $k = 0$ because MPA's function approximators quickly and accurately learn the domain dynamics. The "wait" agent, which loiters in the puddles during learning, performs poorly for large delays and is outperformed by the best memoryless learner, Batch-Sarsa(.9). This domain dramatically exhibits the benefits of the compact augmented approach over the naive ones as the latter learners dramatically falter with increased delay.

5.5 Decaying performance with noise

While the theoretical results we have presented provide worst case bounds on the performance of MBS in stochastic environments, one may wonder about the empirical correlation between non-determinism and algorithm performance. In an effort to better understand this relationship, we performed experiments evaluating agents under different levels of stochasticity, while keeping all the other parameters fixed. Stochastic W-maze and Puddle World were chosen as our discrete and continuous domains, respectively. Although one anticipates a performance drop-off in general as the stochasticity increases, our goal was to quantify the loss solely from MBS's approximation of transitions (the simulated steps). To this end, we evaluated two algorithms. First, was MBS + R-max, which enjoys the fast learning of R-max in the undelayed MDP and uses MBS approximation for fast planning. This was compared to the compact augmented learner, which also enjoys the fast learning of R-max, but has to construct the exponentially large (but more exact) augmented MDP for planning. Figures 7 and 8 illustrate the decaying performance of the two algorithms in stochastic W-maze and Puddle World, respectively. In both environments, the delay is set to $k = 5$, as larger delays become intractable for the compact augmented learner. As the graphs illustrate, increased stochasticity is more detrimental to MBS + R-max than to the compact augmented learner, as MBS's simulations become less accurate. However, considering MBS + R-max's huge computational and space advantage, the performance is still in an acceptable range.

6 Conclusions and future work

In this article, we evaluated algorithms for environments with constant observation and reward delay. We showed the general CDMDP planning problem is NP-Hard, but planning can be

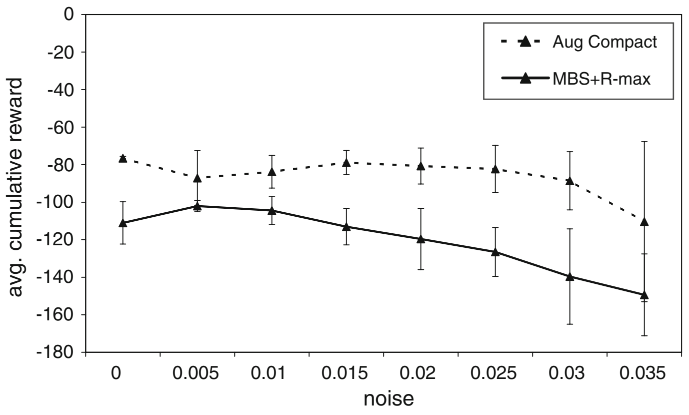


Fig. 8 Effect of noise on the performance of MBS + R-max and the compact augmented learner in the stochastic Puddle World environment. Delay is again set to $k = 5$

done in polynomial time in the deterministic finite setting, and we provided loss bounds on efficient planning in three other settings. We introduced Model Based Simulation (MBS) for planning in CDMDPs, and Model Parameter Approximation (MPA) to extend MBS for learning in continuous environments. We also proposed a method that provably learns a finite-state CDMDP efficiently, despite the exponential size of the learned model. We conducted experiments that reinforced our claims, showing that augmented MDPs can be learned efficiently, but are costly to store or plan with, and that MBS outperformed this and other natural alternatives in several benchmark delayed MDPs, though heavily noisy transitions can fell MBS.

Several open research topics in this area remain. In the learning setting, one could relax the assumption that the delay is known. We have experimented with several methods for learning delay, including techniques from the clustering literature. While these approaches seem to work well in practice, theoretical results regarding such learning have not been produced. A related problem is variable delay, or *jitter*, which is common when dealing with network latency and has been studied in prior work on augmented MDPs [13]. Also, though we covered two important *stochastic* special cases, there may be more conditions that facilitate efficient planning. A related open question is whether an algorithm that exploits structure within the belief space (for instance, if the number of reachable belief states from any start state within k steps is small) could plan in time not influenced by the potential exponential expansion. We note that MBS is an extreme case of such an algorithm, which considers only $|S|$ reachable belief states, all of them pure.

Acknowledgements This work was supported in part by NSF IIS award 0329153. We thank the First Annual Reinforcement Learning Competition and Adam White.

A Appendix: Proof of Theorem 1

In an undelayed MDP, applying action a from state s produces a probability distribution over the next states. It follows from the Markov assumption that in an MDP with $|S|$ states, there can be at most $|S|$ distinct probability distributions over states after performing any possible action sequence. We now describe a method which, given $|A|$ and k, constructs a CDMDP with two states such that performing each action results in $\Theta(|A|^k)$ different probability distributions based on different k-step histories $(s, a_1, \ldots, a_k)$. It follows from this construction that any regular MDP has to have $\Omega(|A|^k)$ states to represent these probability distributions.

Let $p(s'|\vec{a}, s)$ denote the probability of going to state s' after applying action sequence $\vec{a}$ from state s. For any action $i \in \{1, \ldots, |A|\}$ in a two-state MDP, the corresponding transition matrices P_i can be written as follows.

$$P_i = \begin{pmatrix} x_i & 1 - x_i \\ y_i & 1 - y_i \end{pmatrix}, \tag{1}$$

where $x_i = p(s_1|i, s_1)$ and $y_i = p(s_1|i, s_2)$. We can compute the probability distribution after performing action sequence $\vec{a} = (a^1, a^2, \ldots, a^l)$ by multiplying their individual transition matrix:

$$T_{\vec{a}} = P_{a^1} P_{a^2} \cdots P_{a^l} \tag{2}$$

Define $d_i = x_i - y_i$. With trivial algebra one can show that for an action sequence $\vec{a} = (i, j)$ of length 2,

$$\begin{aligned} p(s_1|\vec{a}, s_1) &= x_i d_j + y_j \\ p(s_1|\vec{a}, s_2) &= y_i d_j + y_j. \end{aligned}$$

This observation can be easily extended to multiple-step cases. For any action sequence $\vec{a} = (a^1, a^2, \ldots, a^k)$ with $a^i \in \{1, \ldots, |A|\}$

$$\begin{aligned} p(s_1|\vec{a}, s_1) &= (((x_{a^1} d_{a^2} + y_{a^2}) d_{a^3} + y_{a^3}) \cdots) d_{a^k} + y_{a^k} \\ &= x_{a^1} d_{a^2} d_{a^3} \cdots d_{a^k} + y_{a^2} d_{a^3} d_{a^4} \cdots d_{a^k} + \cdots + y_{a^k}. \end{aligned}$$

We now construct a delayed MDP such that $D = d_1 = \cdots = d_k$. Then,

$$\begin{aligned} p(s_1|\vec{a}, s_1) &= x_{a^1} D^{k-1} + y_{a^2} D^{k-2} + y_{a^3} D^{k-3} \cdots y_{a^k} \\ p(s_1|\vec{a}, s_2) &= y_{a^1} D^{k-1} + y_{a^2} D^{k-2} + y_{a^3} D^{k-3} \cdots y_{a^k}. \end{aligned}$$

Let $x_i = (2i - 1)D$ and $y_i = 2(i - 1)D$. This choice guarantees that we can generate $|A|^k$ distinct coefficients for each of the probabilities above. In order to get uniqueness for the polynomials, we set $D = \frac{1}{2|A|}$. In fact, any $D < \frac{1}{(2|A|-1)}$ guarantees that:

$$(SG) D^i > (BG) \sum_{j=i+1}^{k-1} D^j, \quad \forall i \in [0..k-2].$$

SG and BG are the smallest and biggest differences in the coefficients of the polynomials respectively; that is, $SG = \min |c_i - c_j|, i \neq j$ and $BG = \max |c_i - c_j|, i \neq j$, where each c is a coefficient. The condition above is necessary and sufficient for uniqueness of the polynomial given the uniqueness of the coefficients. In fact, with x_i and y_i defined as above, $\frac{p(s_1|\vec{a}, s_i)}{D^k}$ is a well-formed representation in base $\frac{1}{D}$. In this case, each of the $2(|A|^k)$ probabilities correspond to unique numbers in this representation. Therefore, any regular MDP has to have $\Omega(|A|^k)$ states to represent these distinct probability distributions. □

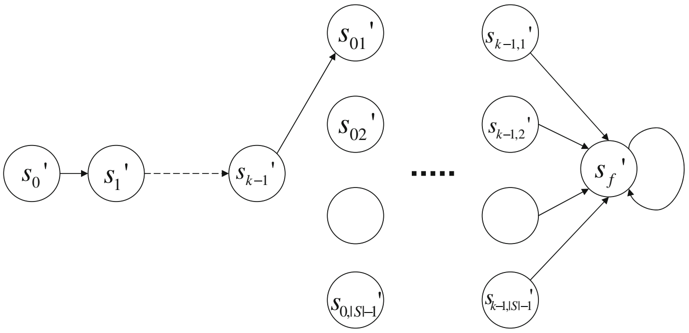


Fig. 9 The induced MDP for the NP-hardness proof of CDMDP planning

B Appendix: Proof of Theorem 2

The proof is by reduction from the problem of planning in a finite-horizon unobservable MDP (UMDP). Consider an arbitrary finite-horizon UMDP $U = \langle S, A, P, R, k\rangle$ where k is the horizon, with an initial state s_1 (a similar construction is possible when the initial state s_1 is not fixed but drawn from a distribution). Let $n = |S|$. We now create a CDMDP $C = \langle S', A, P', R', \gamma, k\rangle$ where $\left|S'\right| = k + kn + 1$. For ease of presentation, states in S' are grouped into three categories (*c.f.*, Figure 9):

1. States s_1' through s_k'
2. States s_{ti}' for $1 \le t \le k$ and $1 \le i \le n$
3. A final state, s_f'.

The transitions and rewards for the three categories are as follows.
$P'(s_t', a, s_j') = 1$ if $j = t + 1$ and 0 otherwise, for $1 \le t \le k - 1$.
$P'(s_k', a, s_{11}') = 1$, else 0.[7]
$R'(s_t', a) = 0$ for $1 \le t \le k$ and $a \in A$.

$P'(s_{ti}', a, s_{t+1,j}') = P(s_i, a, s_j)$ for $1 \le t \le k - 1$, $1 \le i, j \le n$, and $a \in A$.
$P'(s_{ki}', a, s_f') = 1$ for $1 \le i \le n$ and $a \in A$.
$R'(s_{ti}', a) = R(s_i, a)/\gamma^{t+k-1}$ for $1 \le i \le n$, $1 \le t \le k$, and $a \in A$.

$P'(s_f', a, s_f') = 1$ for $a \in A$.
$R'(s_f', a) = 0$ for $a \in A$.

The initial state of the CDMDP for the planning problem is defined as $I_k^0 = (s_1', a_1 \cdots a_1)$, that is, the initial state s_1' followed by action a_1 (arbitrary) k times.

Intuitively, the first set of k states are merely "dummy" states needed to define I_k^0. The next kn states represent one of the UMDP states at a specific timestep t, and the rewards for these states are scaled based on the timestep they represent in order to equate the value functions for U and C. The final state is a sink state. Remembering that the first k rewards are always zero, the value function for C can be written as:

$$V^*(I_k^0) = R(s_1) + \gamma \max_a \sum_i P'(s_{11}', a, s_{2i}')V^*(s_i),$$

[7] Note: if we had a stochastic initial state distribution, this would be a stochastic transition.

where $V^*(s_i)$ is the optimal value of state s_i in the UMDP U. Since $P'(s'_{11}, a, s'_{2i}) = P(s_1, a, s_i)$ we get

$$\begin{aligned} V^*(I_k^0) &= R(s_1) + \gamma \max_a \sum_i P(s_1, a, s_i) \frac{R(s_i)}{\gamma} + \cdots \\ &= R(s_1) + \max_a \sum_i P(s_1, a, s_i) R(s_i) + \cdots, \end{aligned}$$

which equals $V^*(s_1)$, the optimal value function for U at the start state s_1.

Since U is an arbitrary finite-horizon UMDP, answering the CDMDP planning problem for C provides an answer for whether any policy from a given start state in a finite-horizon UMDP can have a value of at least θ, which is known to be NP-Complete [19]. The construction is polynomial as C is bounded in size by a polynomial function of the parameters of U. $\square$

References

1. Altman, E., & Nain, P. Closed-loop control with delayed information. In *Proceedings of the ACM SIGMETRICS and Performance* 1–5, pp. 193–204.
2. Atkeson, C. G., Moore, A. W., & Schaal, S. (1997). Locally weighted learning for control. *Artificial Intelligence Review, 11*(1–5), 75–113.
3. Bander, J. L., & White C. C. III (1999). Markov decision processes with noise-corrupted and delayed state observations. *Journal of the Operational Research Society, 50*, 660–668.
4. Bertsekas, D. P. (2001). *Dynamic programming and optimal control* (2nd ed., Vol. 1/2). Athena Scientific.
5. Boyan, J. A., & Moore, A. W. (1995). Generalization in reinforcement learning: Safely approximating the value function. In *Advances in neural information processing systems*: *Proceedings of the 1994 conference* (pp. 369–376). Cambridge, MA: MIT Press.
6. Brafman, R. I., & Tennenholtz, M. (2002). R-max—A general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research, 3*, 213–231.
7. Brooks, D. M., & Leondes, C. T. (1972). Markov decision processes with state-information lag. *Operations Research, 20*(4), 904–907.
8. Fox, R., & Tennenholtz, M. (2007). A reinforcement learning algorithm with polynomial interaction complexity for only-costly-observable MDPs. In *Proceedings of the 22nd Conference on Artificial Intelligence*, pp. 553–558.
9. Hoeffding, W. (1963). Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association, 58*(301), 13–30.
10. Jong, N. K., & Stone, P. (2006). Kernel-based models for reinforcement learning. In *Proceedings of the 2006 ICML Kernel Machines and Reinforcement Learning Workshop*.
11. Kaelbling, L. P., Littman, M. L., & Cassandra, A. R. (1998). Planning and acting in partially observable stochastic domains. *Artificial Intelligence, 101*(1–2), 99–134.
12. Kakade, S. (2003). *On the Sample Complexity of Reinforcement Learning*. PhD thesis, University College London, UK.
13. Katsikopoulos, K. V., & Engelbrecht, S. E. (2003). Markov decision processes with delays and asynchronous cost collection. *IEEE Transactions on Automatic Control, 48*, 568–574.
14. Lin, L.-J. (1993). *Reinforcement Learning for Robots using Neural Networks*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA.
15. Littman, M. L. (1996). *Algorithms for sequential decision making*. PhD thesis, Brown University, Providence, RI, 1996.
16. Loch, J., & Singh, S. (1998). Using eligibility traces to find the best memoryless policy in partially observable Markov decision processes. In *Proceedings of the 15th International Conference on Machine Learning*, pp. 323–331.
17. Munos, R., & Moore, A. W. (2000). Rates of convergence for variable resolution schemes in optimal control. In *Proceedings of the 17th International Conference on Machine Learning*, pp. 647–654.
18. Ormoneit, D., & Sen, Ś. (2002). Kernel-based reinforcement learning. *Machine Learning, 49*, 161–178.
19. Papadimitriou, C. H., & Tsitsiklis, J. N. (1987). The complexity of Markov decision processes. *Mathematics of Operations Research, 12*(3), 441–450.

20. Puterman, M. L. (1994). *Markov decision processes: Discrete stochastic dynamic programming*. New York: Wiley.
21. Singh, S. P., & Sutton, R. S. (1996). Reinforcement learning with replacing eligibility traces. *Machine Learning, 22*(1–3), 123–158.
22. Singh, S. P., & Yee, R. C. (1994). An upper bound on the loss from approximate optimal-value functions. *Machine Learning, 16*(3), 227–233.
23. Strehl, A. L., Li, L., Wiewiora, E., Langford, J., & Littman, M. L. (2006). PAC model-free reinforcement learning. In *Proceedings of the 23rd International Conference on Machine Learning*, pp. 881–888.
24. Sutton, R. S. (1996). Generalization in reinforcement learning: Successful examples using sparse coarse coding. In D. S. Touretzky, M. C. Mozer, & M. E. Hasselmo (Eds.), *Advances in neural information processing systems 8*, (pp. 1038–1045). Cambridge, MA: MIT Press.
25. Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning: An introduction*. Cambridge, MA: MIT Press.
26. Vijayakumar, S., & Schaal, S. (2000). Locally weighted projection regression: An O(n) algorithm for incremental real time learning in high dimensional space. In *Proceedings of the 17th International Conference on Machine Learning*, pp. 1079–1086.
27. Zubek, V. B., & Dietterich, T. G. (2000). A POMDP approximation algorithm that anticipates the need to observe. In *Proceedings of the Pacific Rim International Conference on Artificial Intelligence*, pp. 521–532.