Act-Then-Measure: Reinforcement Learning for Partially Observable Environments with Active Measuring

Merlijn Krale, Thiago D. Simão, Nils Jansen
Radboud University, Nijmegen
Institute for Computing and Information Sciences
{merlijn.krale, thiago.simao, nils.jansen}@ru.nl

Abstract

We study Markov decision processes (MDPs), where agents control when and how they gather information, as formalized by action-contingent noiselessly observable MDPs (ACNO-MPDs). In these models, actions have two components: a *control action* that influences how the environment changes and a *measurement action* that affects the agent's observation. To solve ACNO-MDPs, we introduce the *act-then-measure* (ATM) heuristic, which assumes that we can ignore future state uncertainty when choosing control actions. To decide whether or not to measure, we introduce the concept of *measuring value*. We show how following this heuristic may lead to shorter policy computation times and prove a bound on the performance loss it incurs. We develop a reinforcement learning algorithm based on the ATM heuristic, using a Dyna-Q variant adapted for partially observable domains, and showcase its superior performance compared to prior methods on a number of partially-observable environments.

Introduction

In recent years, partially observable Markov decision processes (POMDPs) have become more and more widespread to model real-life situations involving uncertainty (Kormushev, Calinon, and Caldwell 2013; Lei et al. 2020; Sunberg and Kochenderfer 2022). *Active measure* POMDPs are an interesting subset of these environments, in which agents have direct control over when and how they gather information, but gathering information has an associated cost (Bellinger et al. 2021). For example, maintenance of a sewer system might require regular inspections (Jimenez-Roa et al. 2022), or appropriate healthcare might require costly or invasive tests (Yu et al. 2023). In both cases, the risk or cost of gaining information needs to be weighted against the value of such information.

Reinforcement learning (RL) is a promising approach to handling problems where we must actively gather information. However, due to the complexity of POMDPs, successes with RL methods in partially observable settings are still limited (Dulac-Arnold et al. 2021). One may circumvent this by focusing on subsets of POMDPs which have certain exploitable properties. For example, Guo, Doroudi, and Brunskill (2016) proposed an efficient RL algorithm for small-horizon POMDPs, Simão, Suilen, and Jansen (2023) investigates offline RL where finite histories provide sufficient statistics. Similarly, this paper focuses on a subset of active measure POMDPs with complete and noiseless observations, called *action contingent noiselessly observable* MDPs (ACNO-MDPs; Nam, Fleming, and Brunskill 2021).

For ACNO-MDPs, two RL algorithms already exist. The first, *active measure RL* (AMRL-Q, Bellinger et al. 2021), is computationally inexpensive but uses a most-likely state approximation and always converges to non-measuring policies, causing poor performance in stochastic environments. The second, *observe-then-plan* (Nam, Fleming, and Brunskill 2021), performs well in smaller stochastic environments, but its reliance on general POMDP planners for policy optimization makes it computationally expensive. Therefore, *we investigate lightweight and high-performing RL methods in stochastic ACNO-MDPs*.

In this paper, we propose a method for stochastic ACNO-MDPs[1] in which we explicitly use knowledge of the setting for both learning and exploitation. To this end, we propose the *act-then-measure* heuristic, inspired by the Q_{MDP} approach (Littman, Cassandra, and Kaelbling 1995), which drastically decreases policy computation times. Since our method relies on a heuristic to compute a policy, we also investigate how much performance we can lose compared to the optimal policy, for which we prove an upper bound. We then describe an algorithm based on Dyna-Q, which uses this heuristic for RL in ACNO-MDPs. We compare it empirically to previous methods in both an environment designed to test whether algorithms can accurately determine the value of measuring and a standard RL environment. In both environments, our algorithm outperforms AMRL-Q and *observe-then-plan* while staying computationally tractable for much bigger environments than the latter.

Contributions. The main contributions of this work are: 1) identifying limitations of previous RL approaches for ACNO-MPDs, 2) introducing the *act-then-measure* (ATM) heuristic, 3) introducing the concept of *measuring value*, and 4) implementing Dyna-ATMQ, an RL algorithm for ACNO-MDPs following the ATM heuristic.

[1] Stochastic MDPs are the opposite of deterministic MDPs where all probability distributions are Dirac.

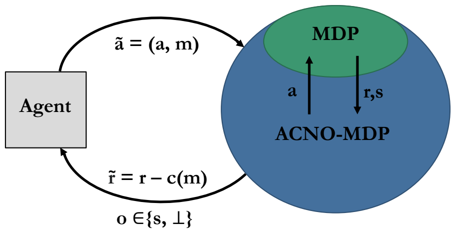

Figure 1: Agent-environment interaction in an ACNO-MDP. The agent performs a control action a and measurement m at each time step t. The internal environment state is defined by an MDP and affected only by control actions. After each step, the agent receives a scalarized reward $\tilde{r}{=}r-C(m)$ and observation $o \in \{s, \perp\}$ (with $o{=}s \iff m{=}1$).

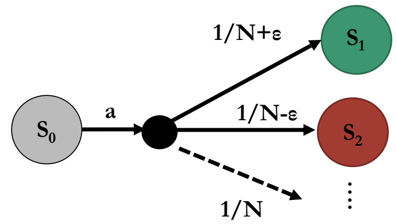

Figure 2: An ACNO-MDP where the value $Q(s_0, \langle a, 0\rangle)$ based on the most-likely successor state can be made arbitrarily inaccurate. In this example, using a most likely state means considering only s_1, even though the probability of reaching this state is only $1/N + \epsilon$, with N the number of successor states (which is only bounded by $|S|$), neglecting the probability of reaching the remaining successor states.

Background

This section gives a formal description of ACNO-MDPs, then describes and analyzes RL methods for the setting.

ACNO-MDPs

We define our problem as an *action-contingent noiselessly observable MDP* (ACNO-MDP; Nam, Fleming, and Brunskill 2021). An ACNO-MDP is defined by a tuple $\mathcal{M} = (S, \tilde{A}{=}A{\times}M, P, R, C, \Omega, O, \gamma)$, where (S, A, P, R, γ) are the components of a standard MDP: S is the state space, A is the action space, $P(s' \mid s, a)$ is the transition function, $R(s, a)$ is the reward function and $\gamma \in [0, 1]$ is the discount factor. However, in the ACNO-MDP framework $\tilde{A}$ consists of pairs of *control actions* and *measurements*, taking the form $\tilde{a}{=}\langle a, m\rangle \in A \times M$, where $M{=}\{\text{not observe}, \text{observe}\}{=}\{0, 1\}$. A control action $a \in A$ affects the environment, while the measurement choice $m \in M$ only affects what the agent observes. Following the typical notation from POMDPs, Ω is the observation space and O the observation function, so $O(o \mid s', \langle a, m\rangle)$ is the probability of receiving observation $o \in \Omega$ when taking measurement m and action a, after transitioning to the state s'. In ACNO-MDPs all measurements are complete and noiseless, so we can define $\Omega{=}S \cup \{\perp\}$, where $\perp$ indicates an empty observation. Then, the observation function is defined as $O(o \mid s', \langle a, 1\rangle){=}1 \iff o{=}s'$, and 0 otherwise. Similarly, $O(o \mid s', \langle a, 0\rangle){=}1 \iff o{=}\perp$, and 0 otherwise. Measuring has an associated cost $C(0){=}0$ and $C(1){=}c$ (with $c \geq 0$), which gets subtracted from our reward, giving us a *scalarized-reward* $\tilde{r}_t{=}R(s_t, a_t) - C(m_t)$.

Agent-environment interactions for ACNO-MDPs are visualized in Figure 1. Starting in some initial state s_0, for each time-step t the agent executes an action-pair $\langle a_t, m_t\rangle$ according to a policy π. In general, these policies are defined for a belief state b_t, a distribution over the states representing the probability of being in each state of the environment, summarising all past interactions. After executing $\langle a_t, m_t\rangle$ in s_t, the environment transitions to a new state $s_{t+1} \sim P(\cdot \mid s_t, a_t)$, and returns to the agent a reward $r_t{=}R(s_t, a_t)$, a cost $c_t = C(m_t)$ and observation $o_{t+1} \sim O(\cdot \mid s_{t+1}, \langle a_t, m_t\rangle)$. The goal of the agent is to compute a policy π with the highest expected total discounted scalarized-reward $V(\pi, \mathcal{M}) = \mathbb{E}_{\pi,\mathcal{M}}\left[\sum_t \gamma^t \tilde{r}_t\right]$.

In this paper, we will mainly focus on reinforcement learning in ACNO-MDPs. We assume the agent only has access to the total number of states and the signals returned by the environment in each interaction, but otherwise has no prior information about the dynamics of the environment.

Q-learning for ACNO-MDPs

Bellinger et al. (2021) propose to solve the ACNO-MDP problem using an adaptation of Q-learning (Watkins and Dayan 1992). To choose the best action pair, the agent estimates both the transition probability function and value functions with tables $\hat{P}$ and Q of sizes $|S \times A \times S|$ and $|S \times \tilde{A}|$, respectively. Both are initialized uniformly, except that all actions with $m = 1$ are given an initial bias in Q to promote measuring in early episodes.

Beginning at the initial state, for every state s_t the agent executes an ϵ-greedy action-pair $\langle a_t, m_t\rangle$ according to Q. When $m_t = 1$, the successor state $s' = s_{t+1}$ is observed so the algorithm updates the transition probability $\hat{P}(\cdot \mid s_t, a_t)$. When $m_t = 0$, AMRL-Q does not update $\hat{P}$ and assumes the successor state is the *most likely next state* according to $\hat{P}$:

$$s' = \arg\max_{s \in S} \hat{P}(s \mid s_t, a_t).$$

Using the reward r_t and the (potentially estimated) successor state s', AMRL-Q updates both $Q(s_t, \langle a_t, 0\rangle)$ and $Q(s_t, \langle a_t, 1\rangle)$, as follows:

$$Q(s_t,\langle a_t, m\rangle) \leftarrow (1-\alpha)Q(s_t, \langle a_t, m\rangle) + \alpha\left[r_t - C(m) + \gamma \max_{a',m'} Q(s', \langle a', m'\rangle)\right]. \quad (1)$$

Although AMRL-Q is conceptually interesting and has very low computation times, in practice the algorithm has some considerable shortcomings:

AMRL-Q does not measure after convergence. Apart from its ϵ-greediness, for any state s AMRL-Q only takes a measuring action $\tilde{a}{=}\langle a, 1\rangle$ if $\tilde{a}$ has the highest Q-value.

In particular, this means that $Q(s, \langle a, 1\rangle) > Q(s, \langle a, 0\rangle)$ must hold. However, these Q-values get updated simultaneously with the same r_t and s', with $(r_t - C(m))$ always lower for m=1. Therefore, $Q(s, \langle a, 1\rangle)$ always converges to a value lower than $Q(s, \langle a, 0\rangle)$. This means AMRL-Q only converges to non-measuring policies, which may be suboptimal for stochastic environments where the optimal policy requires taking measurements.

AMRL-Q ignores the state uncertainty. As visualized in Figure 2, the most-likely successor state used in AMRL-Q can give arbitrarily inaccurate approximations of the value of the current state. Apart from sub-optimal action selection, this may also cause inaccuracies in the model in later steps, since AMRL-Q makes no distinction between measured and non-measured states for model updates.

Solving ACNO-MDP via POMDPs

Nam, Fleming, and Brunskill (2021) introduce two frameworks for solving tabular ACNO-MDPs. The first, named *observe-before-planning*, has an initial exploration phase in which the agent always measures to learn an approximated model. After this phase, a generic POMDP-solver computes a policy based on the approximated model. The second framework, named *observe-while-planning*, starts by using a POMDP-solver on some initial model and updates the model on-the-fly based on the measurements made. For both frameworks, a specific implementation is tested, using *episodic upper lower exploration in reinforcement learning* (EULER; Zanette and Brunskill 2019) for the exploration phase and *partially observable Monte-Carlo planning* (POMCP; Silver and Veness 2010) as a POMDP-solver. Both algorithms outperform the tested generic RL method for POMDPs, with *observe-before-planning* performing slightly better overall. We, therefore, focus on this framework in this paper. Apart from some more specific disadvantages of using POMCP for ANCO-MDPs (Krale, Simão, and Jansen 2023, Appendix B), we note one general shortcoming of this framework.

***Observe-before-planning* depends on a POMDP-solver.** While *observe-before-planning* uses the ACNO-MDP structure in its exploration phase, for exploitation, it relies only on a generic POMDP-solver. These solvers have high computational complexity, which limits in which environments they can be employed. In contrast, a method that uses the ACNO-MDP structure (where only control actions affect the underlying state) could, in principle, solve larger problems.

The Act-Then-Measure Heuristic

In this section, we propose the *act-then-measure* (ATM) heuristic for approximating optimal policies in ACNO-MDPs. Intuitively, this heuristic is based on the observation that control actions and measurements have very different effects, which implies it might be desirable to choose them using separate processes. Therefore, inspired by the Q_{MDP} heuristic (Littman, Cassandra, and Kaelbling 1995), our heuristic *chooses a control action, assuming all (state) uncertainty will be resolved in the next state(s).*

Following this heuristic, we do not need to consider measurements while deciding control actions, since measuring only affects state uncertainty. This means we can use a basic control loop (Figure 3), in which we choose control actions before measurements.

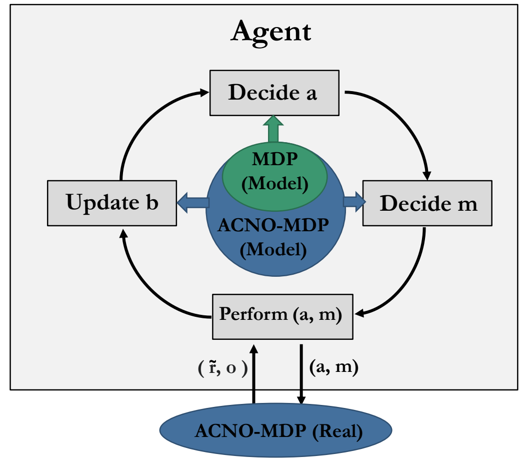

Figure 3: The control-loop for solving ACNO-MDPs using the *act-then-measure* heuristic. At each time step, a control action a is chosen according to the current belief state b, as though it is a belief over MDP states. Then, a measurement m is picked without ignoring the state uncertainty, (a, m) is executed, and the belief state b is updated accordingly.

Evaluating Control Actions

To choose control actions, we can approximate future returns using an MDP approximation:

$$Q(b, a) \approx \sum_{s \in S} b(s) Q_{\text{MDP}}(s, a), \tag{2}$$

where $Q_{\text{MDP}}(s, a)$ is the value of taking action a in state s and following the optimal policy of the underlying MDP afterward, and b denotes the current belief, so $b(s)$ is the probability that the current state is s. Since, in general, MDPs are more tractable than POMDPs, this approximation allows for a more efficient policy computation than POMDP-based methods like *observe-then-plan*. At the same time, in contrast to AMRL-Q, belief states are not approximated, which means current state uncertainty is fully considered and measurements can be made after convergence.

Evaluating Measurements

To use the ATM heuristic, we need a principled way to determine whether to take a measurement. Therefore, we require the ability to *estimate the value of a measurement*. For this, we start by defining the value function $Q_{\text{ATM}}(b, \tilde{a})$ as the value for executing $\tilde{a}$ in belief state b, assuming we follow the ATM-heuristic, i.e. that we choose control actions according to Equation 2. We will define $Q_{\text{ATM}}(b, \tilde{a})$ using Bellman equations. For readability, we first introduce the following notations:

$$b'(s'|b, a) = \sum_{s \in S} b(s) P(s'|s, a), \text{ and } \dot{\max_{\tilde{a} \in \tilde{A}}} = \max_{m \in M} \max_{a \in A},$$

where $b'(s'|b,a)$ represents the probability of transitioning to state s' when taking action a in the current belief state b, and $\dot{\max}$ describes the optimal action pair if the control action is decided before the measurement.

We note that the form of the Bellman equations for $Q_{\text{ATM}}(b,\tilde{a})$ depends on the current measuring action. If measuring, we can use the information we gain to choose the next optimal action to take, giving us the following:

$$Q_{\text{ATM}}(b,\langle a,1\rangle)=\hat{r}-c+\gamma\sum_{s'\in S}b'(s'|b,a)\dot{\max_{\tilde{a}\in\tilde{A}}}Q_{\text{ATM}}(s',\tilde{a}), \tag{3}$$

with $\hat{r}$ the expected reward of taking action a in belief state b and $Q_{\text{ATM}}(s,\tilde{a})$ the Q-value of a belief state with $b(s)=1$. If not measuring, we can only base our next action on the expected next belief. We may then define the *belief-optimal action* $\tilde{a}_b$ as follows:

$$\begin{aligned}\tilde{a}_b &= \arg\dot{\max_{\tilde{a}\in\tilde{A}}}Q_{\text{ATM}}(b_{\text{next}}(b,a),\tilde{a})\\ &= \arg\dot{\max_{\tilde{a}\in\tilde{A}}}\sum_{s'\in S}b'(s'|b,a)Q_{\text{ATM}}(s',\tilde{a}),\end{aligned} \tag{4}$$

where the second equality follows from the fact that control actions are chosen according to Equation 2, and is proven in Appendix C (Krale, Simão, and Jansen 2023). Using this, we find the following Bellman equation for $m=0$:

$$Q_{\text{ATM}}(b,\langle a,0\rangle)=\hat{r}+\gamma\sum_{s'\in S}b'(s'|b,a)Q_{\text{ATM}}(s',\tilde{a}_b). \tag{5}$$

Based on Equations 3 and 5, we define the *measuring value* $\text{MV}(b)$ as the difference between these two Q-values:

$$\begin{aligned}&\text{MV}(b,a)=Q_{\text{ATM}}(b,\langle a,1\rangle)-Q_{\text{ATM}}(b,\langle a,0\rangle)\\ &=-c+\gamma\sum_{s\in S}b'(s|b,a)\Big[\dot{\max_{\tilde{a}\in\tilde{A}}}Q_{\text{ATM}}(s,\tilde{a})-Q_{\text{ATM}}(s,\tilde{a}_b)\Big].\end{aligned} \tag{6}$$

To illustrate, suppose we predict a next belief state b' as given in Figure 4, and for simplicity assume $\gamma=1$. If we choose not to measure, the belief optimal action for b' is a_0, yielding a reward of 0.8 on average. If instead we do take a measurement, we can decide to take action a_0 if we reach state s_0 and action a_1 if we reach state s_1, yielding a return of $1-c$. Following Equation 6, the measuring value is thus $1-c-0.8=0.2-c$, meaning it is worth taking a measurement if $c\leq 0.2$. Generalising this example, we find the following condition for taking measurements:

$$m_{\text{MV}}(b,a)=\begin{cases}1 & \text{if } \text{MV}(b,a)\geq 0;\\ 0 & \text{otherwise},\end{cases} \tag{7}$$

and can define a policy following the ATM heuristic as:

$$\pi_{\text{ATM}}(b)=\langle\max_{a\in A}Q(b,a),m_{\text{MV}}(b,\max_{a\in A}Q(b,a))\rangle, \tag{8}$$

with $Q(b,a)$ as defined in Equation 2.

In practice, calculating $Q_{\text{ATM}}(s,\tilde{a})$ in Equations 3 and 5 for all possible next belief states can be computationally intractable. An intuitive (over-)approximation to use is $Q_{\text{ATM}}(s,\langle a,m\rangle)\approx Q_{\text{MDP}}(s,a)$, in which case Equation 6 would likely give an overestimation of MV, leading to more measurements than required.

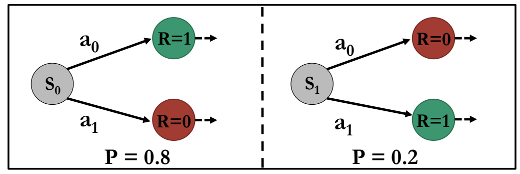

Figure 4: An example of a simple belief state.

Performance Regret of ATM

Now that π_{ATM} is fully defined, we are interested in its performance loss as compared to an optimal policy π^* not restricted by Equation 2. We first prove the following lemma:

Lemma 1. *Given a fully known ACNO-MDP $\mathcal{M}$. Define π_{ATM} as in Equation* 8*, and π'_{ATM} as: $\pi'_{ATM}(b)=\langle\max_{a\in A}Q(b,a),\psi(b)\rangle$, with $\psi: b\rightarrow m$. For any choice of ψ, the following holds:*

$$V(\pi_{ATM},\mathcal{M})\geq V(\pi'_{ATM},\mathcal{M}). \tag{9}$$

Intuitively, this lemma states that m_{MV} is the optimal way of deciding m when following the ATM heuristic. Appendix C (Krale, Simão, and Jansen 2023) provides the proof. Using this lemma, we can find an upper bound for the performance loss of π_{ATM}:

Theorem 1. *Given a fully known ACNO-MDP $\mathcal{M}$ with an optimal policy π^*. The performance loss for the policy following the act-then-measure heuristic π_{ATM} (Equation* 8*) has the following* minimal upper bound*:*

$$V(\pi^*,\mathcal{M})-V(\pi_{ATM},\mathcal{M})\leq\sum_t\gamma^t c. \tag{10}$$

Proof. We start by proving that Equation 10 is indeed an upper bound. For this, we introduce $\mathcal{M}_0$, an ACNO-MDP with the same dynamics and reward function as $\mathcal{M}$, but with $c=0$. In $\mathcal{M}_0$, always measuring and taking control actions in accordance to Q_{MPD} is an optimal policy. Let π_{Measure} be that policy, than the following holds:

$$V(\pi_{\text{Measure}},\mathcal{M}_0)=V(\pi^*,\mathcal{M}_0). \tag{11}$$

Since the behavior of π_{Measure} is independent of c, we can relate the expected return of this policy in $\mathcal{M}_0$ to that in $\mathcal{M}$:

$$V(\pi_{\text{Measure}},\mathcal{M})=V(\pi_{\text{Measure}},\mathcal{M}_0)-\sum_t\gamma^t c. \tag{12}$$

Furthermore, we notice π_{Measure} follows the control actions given by $\max_{a\in A}Q(b,a)$. Thus, via Lemma 1:

$$V(\pi_{\text{ATM}},\mathcal{M})\geq V(\pi_{\text{Measure}},\mathcal{M}). \tag{13}$$

Lastly, we note that for a given policy, the expected return in $\mathcal{M}_0$ can never be lower than that in $\mathcal{M}$. Then, in particular:

$$V(\pi^*,\mathcal{M})\leq V(\pi^*,\mathcal{M}_0). \tag{14}$$

Substituting Equations 12 and 14 into Equation 11, then substituting π^*_{ATM} for π_{Measure} following Equation 13, we find exactly our upper bound.

To prove the given bound is minimal, it suffices to show an ACNO-MDP where the bound is exact, which means no

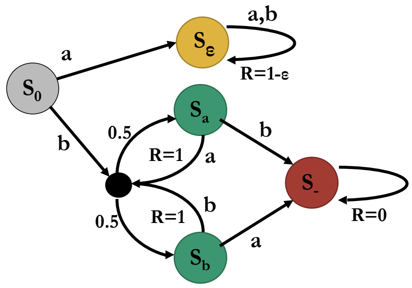

Figure 5: An example where the act-then-measure heuristic can fail for ACNO-MPDs. We assume $c \in [0, 0.5]$ and ϵ is infinitesimally small.

lower bound can exist. Such an ACNO-MDP is shown in Figure 5. Using the ATM heuristic, taking action b in state s_0 is optimal since both s_a and s_b yield an (infinitesimally) higher expected return than s_ϵ given full state information. However, after this action the optimal policy would be to measure every step, leading to a lost return of $\sum_t \gamma^t c$. □

In practice, the performance loss of using the ATM heuristic depends on the environment under consideration. We note the ATM assumption holds in deterministic environments with a single initial state, and has limited impact in environments where c is small relative to the episodic reward. In contrast, we recall that the AMRL-Q approach does not converge to policies that actively gather information. This means its performance loss with respect to the baseline policy is unbounded, even when c is small. *Observe-before-planning* does always converge to π^*, but in practice may be computationally intractable.

Dyna-ATMQ: an ATM-based RL Algorithm for ACNO-MDPs

To test both the ATM heuristic and measuring value, we implement *dynamic act-then-measure Q-learning* (Dyna-ATMQ), an RL algorithm specifically designed for ACNO-MDPs. A high-level version of the learning loop for an episode is given by Algorithm 1. The supplemental material (Krale, Simão, and Jansen 2023, Appendix A) presents the complete pseudo-code, while an explanation of all parts of the algorithm is given next.

Belief states. To deal with partially unknown states, we implement *discretized belief states* b_t, with $b_t(s)$ the estimated probability of being in state s at time t. After measuring, belief states are deterministic, i.e.

$$b_{t+1}(s) = \begin{cases} 1 & \text{if } s = s_{t+1}; \\ 0 & \text{otherwise.} \end{cases} \tag{15}$$

After a non-measuring action, we instead sample a new belief state, with $b_{t+1}(s) \sim \sum_{s' \in S} b_t(s') P(s|s', a)$.

Transition model. To estimate our transition probabilities, we apply the *Bayesian MDP* approach as introduced by Dearden, Friedman, and Andre (1999). In this framework, a transition function $P(\cdot \mid s, a)$ is given by a

---

Algorithm 1: Dyna-ATMQ

Initialise transition model P, value function Q, belief state b_0;
while episode not completed **do**
  Choose control action a_t according to Q ▷ *Equation* 18
  Choose m_t according to a_t ▷ *Equation* 23
  Execute $\tilde{a}_t = \langle a_t, m_t \rangle$
  Receive reward r_t and observation o_t
  Determine next belief state b_{t+1} ▷ *Equation* 15
  Update P according to o_t ▷ *Equations* 16 *and* 17
  Update Q according to r_t and P ▷ *Equation* 19
  Update Q using model-based training
end while
return $\sum_t \gamma^t r_t$

Dirichlet distribution $D(s, a)$, as parameterised by $\vec{\alpha} = \{\alpha_{s,a,s_0}, \alpha_{s,a,s_1}, ...\}$. In the standard MDP-setting, $\alpha_{s,a,s'}$ is given by a (uniform) prior, plus the number of times a transition has already occurred. For the ACNO-MDP setting, we change this to the number of times it has been *measured*. Thus, at every step we update our model as follows:

$$\alpha_{s,a,s'} \leftarrow \begin{cases} \alpha_{s,a,s'} + 1 & \text{if } a_{t-1} = a, m_t = 1, \\ & \quad b_t(s) = 1, b_{t+1}(s') = 1; \\ \alpha_{s,a,s'} & \text{otherwise,} \end{cases} \tag{16}$$

and define estimated transition probabilities as:

$$P(s' \mid s, a) = \mathbb{E}[s' \mid D(s, a)] = \frac{\alpha_{s,a,s'}}{\alpha_{s,a}}, \tag{17}$$

where $\alpha_{s,a} = \sum_{s' \in S} \alpha_{s,a,s'}$.

Value function. To estimate the values of belief states, we make use of the *replicated Q-learning method*, as introduced in Chrisman (1992) and formalized by Littman, Cassandra, and Kaelbling (1995). In this method, we assume the optimal action for any belief state can be given as a linear function over all states. With this assumption, we choose a control action a in belief state b as follows:

$$a_t = \max_{a \in A} Q(b_t, a) = \max_{a \in A} \sum_{s \in S} b_t(s) Q(s, a). \tag{18}$$

To update the Q-values, we use the following update rule:

$$Q(s, a) \leftarrow (1 - \eta_s) Q(s, a) + \eta_s (\tilde{r} + \gamma \Psi(s, a)), \tag{19}$$

with $\eta_s = b(s)\eta$ the weighted learning rate and $\Psi(s, a)$ the estimated future return after state-action pair (s, a):

$$\Psi(s, a) = \sum_{s' \in S} P(s' \mid s, a) \max_{a'} Q(s', a'). \tag{20}$$

Lastly, to incentivize exploration, we create an *optimisitc* variant of Q. For this, we define an exploration bonus δ:

$$\delta(s, a) = \max \left[0, \frac{N_{\text{opt}} - \alpha_{s,a}}{N_{\text{opt}}} (R_{\max} - Q(s, a)) \right], \tag{21}$$

with $R_{\max}$ the maximum reward in the ACNO-MDP and N_{opt} a user-set hyperparameter. We use this metric to create an *optimistic value function* Q_{opt}:

$$Q_{opt}(s, a) = Q(s, a) + \delta(s, a), \tag{22}$$

which we use instead of the real Q-value in Equations 18 and 20. Inspired by R-Max (Brafman and Tennenholtz 2002), our metric initially biases all Q-values such that $Q(s,a) = R_{\text{max}}$, and removes this bias in a number of steps. However, instead of a binary change, δ makes this transition in N_{opt} (linear) steps. In practice, we found this gives a stronger incentive to explore all state-action pairs more uniformly, leading to a faster convergence rate.

Measurement condition. In an RL setting, we note there are two distinct reasons for wanting to measure your environment: *exploratory measurements* to improve the accuracy of the model, and *exploitative measurements* which improve the expected return. For the latter, we have already introduced *measuring value* (MV) as defined in Equation 6.

For the former, we again draw inspiration from R-Max (Brafman and Tennenholtz 2002) by introducing a parameter N_m, and measure the first N_m times a state-action pair is visited. We keep track of this number using $\vec{\alpha}$ as specified in Equation 16. Lastly, we specify to take exploratory measurements only if we are certain about the current state, since no model update is performed otherwise (Equation 16).

Combining both types of measurements, we construct the following condition for deciding when to measure:

$$m_t = \begin{cases} 1 & \text{if } \exists s : b_t(s)=1 \wedge \alpha_{s,a_t} < N_m; \\ m_{\text{MV}}(b_t, a_t) & \text{otherwise.} \end{cases} \tag{23}$$

Model-based training. Lastly, inspired by the Dyna-framework (Sutton 1991), at each step we perform an extra N_{train} *training steps*. For this, we pick a random state s and action a, create a *simulated reward*, and use this to perform a Q-update (Equation 19). For this simulated reward, we use the average reward received thus far $R_{s,a}$, which we initialise as 0 and update each step:

$$R_{s,a} \leftarrow \begin{cases} \frac{R_{s,a} \cdot \alpha_{s,a} + r_t}{\alpha_{s,a}+1} & \text{if } a_{t-1}=a, m_t=1, b_t(s)=1; \\ R_{s,a} & \text{otherwise.} \end{cases} \tag{24}$$

Although originally proposed to deal with changing environments, we mainly use the Dyna approach to speed up the convergence of the Q-table. This is especially relevant for our setting, where even the Q-values for actions never chosen by our policy need to be accurate to estimate $\text{MV}(b_t, a_t)$.

Empirical Evaluation

In this section, we report on our empirical evaluation of Dyna-ATMQ in a number of environments. We first describe the setup of both the algorithms and environments. Then, we show the results of our experiments and highlight some key conclusions. All used code and data can be found at https://github.com/LAVA-LAB/ATM.

Setup

We test the following algorithms:

Dyna-ATMQ: We implement Dyna-ATMQ as described in the previous section. We set γ=0.95, η=0.1, N_b=100 and N_{opt}=N_m=20. For offline training, we choose random states and update their current optimal action with probability ϵ_{train}=0.5, and a random different action otherwise. We use N_{train}=25, but also test a non-dynamic variant with N_{train}=0, which we'll refer to as ATMQ.

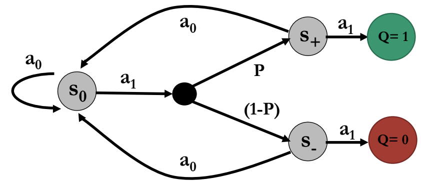

Figure 6: The *measuring value* environment used to test if an agent can determine the value of measuring.

AMRL-Q: For AMRL-Q, we re-implement the algorithm as specified in Bellinger et al. (2021). We set γ=0.95 and α=0.1 to match those of Dyna-ATMQ, and use initial measurement bias β=0.1 as described in the paper. Lastly, we use ϵ=0.1 for the first 90% of all episodes but switch to a fully greedy approach for the last 10%.

ACNO-OTP: We implement the *observe-before-planning* algorithm specified in Nam, Fleming, and Brunskill (2021), using an altered version of the original code, which we refer to as ACNO-OTP (see Krale, Simão, and Jansen 2023, Appendix B, for more details). For the experiments, we use γ=0.95 and *ucb-coefficient* c=10. We perform 25.000 rollouts per step at a max search depth of 25, with between 1800 and 2000 particles. Since we are interested in results after convergence, we limit the exploitation phase to the last 50 episodes and only compare to these.

For our testing, we use the following environments:

Measuring value: As a simple environment to test measuring value, we convert our example from Figure 4 to a graph, as shown in Figure 6. This environment consist of three state S=$\{s_0, s_+, s_-\}$, with s_0 as the initial state. Our agent can choose actions from action space A=$\{a_0, a_1\}$, where a_0 always returns the agent to the initial state. From state s_0, taking action a_1 results in a transition to s_+ with probability p and a transition to s_- with probability $p-1$. Taking action a_1 in the states s_+ and s_- ends the episode and returns rewards r=1 and r=0, respectively.

For this environment, we can explicitly describe its optimal strategy and its expected value. We notice that depending on p and c, such strategies either try to measure the (otherwise indistinguishable) states s_+ and s_-, or they do not. When not measuring, our expected return is always p. When measuring, our expected return in s_+ is $1-c$, and in s_- it is the expected return of s_0 minus c. Combining this, we can calculate the expected return for s_0 with a measuring policy:

$$\mathbb{E}_\pi\left[\sum_t \gamma^t \tilde{r}_t\right] = \sum_{n=0} \gamma^{2n}\Big(p\cdot(1-p)^n\big(1-c(n+1)\big)\Big), \tag{25}$$

where n is the number of measurements required before the episode ends. For our experiments, we set γ=1 and p=0.8, which means measuring is profitable for $c \leq 0.16$.

Frozen lake: As a more complex toy environment, we use the standard *openAI gym* frozen lake environment (Brockman et al. 2016), which describes an $n \times n$ grid with a number of 'holes'. The goal of the agent is to walk from its initial

Algorithm	Measurement Cost 0.05 SR	M	0.10 SR	M	0.20 SR	M
ATMQ	0.94	1.30	0.76	0.50	0.78	0.11
Dyna-ATMQ	0.93	1.34	0.86	1.14	0.82	0.16
AMRL	0.82	0.00	0.80	0.00	0.78	0.00
ACNO-OTP	0.94	1.18	0.81	0.00	0.79	0.00

Algorithm	Variant: Deterministic SR	M	Semi-slippery SR	M	Slippery SR	M
ATMQ	1.00	0.00	0.75	2.65	0.02	0.76
Dyna-ATMQ	1.00	0.00	0.65	2.95	0.03	0.93
AMRL	1.00	0.00	0.41	0.00	0.03	0.00
ACNO-OTP	1.00	0.00	0.40	0.00	0.04	0.00

Table 1: Average scalarized return (SR) and the number of measurements (M) after training, in the measuring value (left) and frozen lake (right) environments. Results are gathered over 5 repetitions, and present the average over the last 50 episodes.

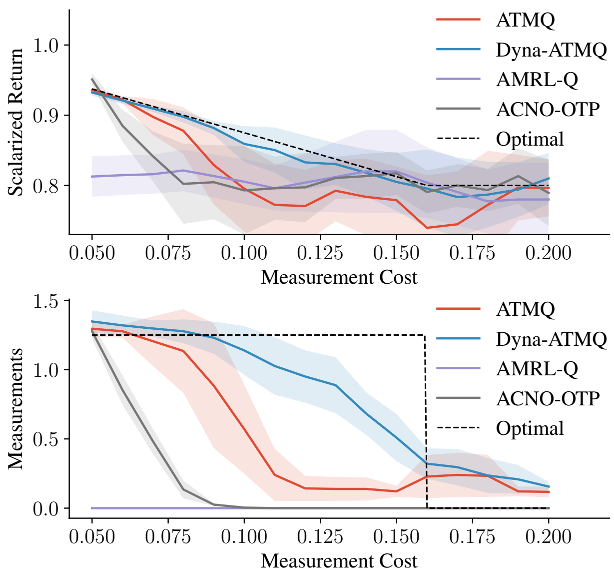

Figure 7: Scalarize returns and the number of measurements in the measuring value environment, with p=0.8 and varying measurement costs. Values are averages over 5 repetitions after convergence.

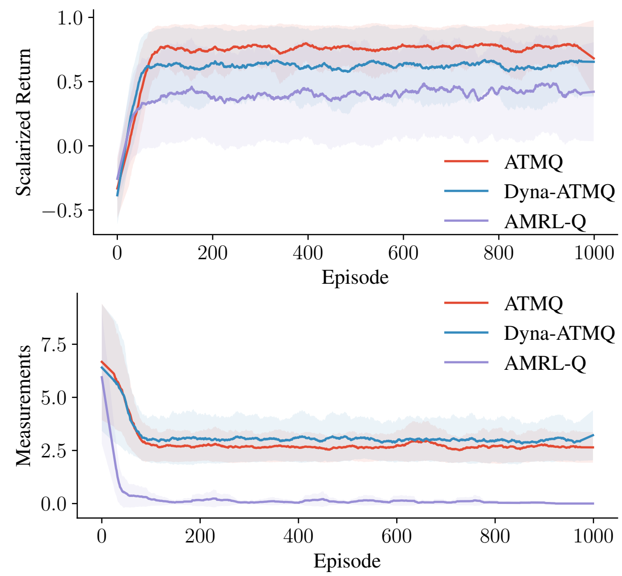

Figure 8: Empirical results on semi-slippery 4×4 frozen lake environment, gathered over 5 repetitions.

state to some goal state without landing on any hole spaces. The agent receives a reward r=1 if it reaches the goal and r=0 otherwise. The episode ends once the agent reaches the goal state or a hole tile. In our testing, we will use the predefined 4×4 and 8×8 map settings, as well as larger maps randomly generated, all with a measuring cost c=0.05. The agent has action space A={left, down, right, up}, but we consider three variations of their interpretation. Firstly, we use both the predefined deterministic and non-deterministic (or *slippery*) settings from the standard gym. In the deterministic case, the agent is always moved in the given direction, while in the slippery case it has an equal probability to move in the given or a perpendicular direction. We also implement and test a more predictable *semi-slippery* variant, where the agent always moves in the given direction, but has a 0.5 chance of moving two spaces instead of one.

Results

To test the *measuring value* metric, we run Dyna-ATMQ on the measuring value environment for a range of different measurement costs. The results can be found in Table 1 (left) and Figure 7. We notice that both Dyna-ATMQ variants, as well as ACNO-OTP, can find close-to-optimal measuring and non-measuring policies. However, as clearly seen in Figure 7 (bottom), all algorithms use non-measuring policies for costs where measuring would still be optimal. The Dyna-variant of ATMQ performs slightly better than both others, but the difference is minimal, especially in terms of rewards. In contrast, AMRL-Q always converges to a non-measuring policy, regardless of measurement cost.

To test how the *act-then-measure*-heuristic effects performance for varying amounts of non-determinism, we run tests on all three variants of the 4×4 frozen lake environment. Results are given in Table 1 (right). For both the deterministic and slippery variants, both versions of ATMQ perform about on par with both of its predecessors. For the former, it converges to an optimal non-measuring policy, and for the latter none of the algorithms get a significantly positive result. However, in the semi-slippery environment, both variants significantly outperform both ACNO-OTP and AMRL-Q, with the non-training variant performing slightly better. To visualize, training curves for our algorithm and AMRL-Q in this environment are shown in Figure 8.

To test the scalability of algorithms using the *act-then-*

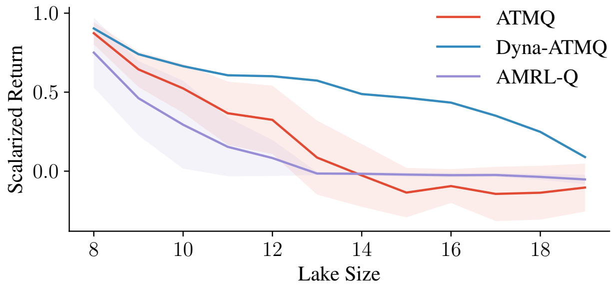

Figure 9: Average scalarized return (after convergence) for semi-slippery frozen lake environment, for different sizes. Results for ATMQ and AMRL-Q averaged over 5 repetitions, for Dyna-ATMQ over 1.

measure-heuristic, we test the performance of Dyna-ATMQ on a number or larger semi-slippery frozen lake Environments. Results of both ATMQ variants and AMRL-Q are shown in Figure 9 [2] . Although the performance of both variants drops quickly with the size of the environment, they are able to achieve above-zero returns for far bigger environments than AMRL-Q. The Dyna-variant performs better for larger environments, even after convergence.

Discussion

Based on our results, we make the following claims:

Measuring value is a suitable metric. In Table 1(right), we notice Dyna-ATMQ converges to a non-measuring policy in the deterministic environments, as expected. For stochastic environments, we note it makes more measurements than our baselines but gets better or equal returns. This suggests it correctly identifies when taking measurements is valuable. We notice suboptimal measuring behavior only when the difference in return between measuring and non-measuring is small, but note that this could be caused by slight errors in our Q-table.

Dyna-ATMQ performs well in small environments. In both the measuring value and small frozen lake environments, we find Dyna-ATMQ performs better than the bound given by Theorem 1. Moreover, it outperforms or equals all baseline algorithms while staying computationally tractable.

Dyna-ATMQ is more scalable than current methods. Dyna-ATMQ stays computationally tractable for larger environments than ACNO-OTP, while yielding higher returns than AMRL-Q. More generally, we note that our current implementation of the ATM-heuristic approximates the Q-values of states in a way that is known to lead to errors for highly uncertain settings (Littman, Cassandra, and Kaelbling 1995). This suggests a more sophisticated algorithm using the ATM heuristic could improve scalability.

[2]Because of high computation times, we were unable to obtain results for ACNO-OTP in these environments.

Related Work

For the tabular ACNO-MDP setting, three RL algorithms already exist: the AMRL-Q (Bellinger et al. 2021), and the *observe before planning* and the ACNO-POMCP algorithms (Nam, Fleming, and Brunskill 2021). The latter is shown to perform worse than *observe before planning* so is not considered in this paper, the other two are analysed in detail in this paper and used as baselines in our experiments.

Another closely related work is that of Doshi-Velez, Pineau, and Roy (2012). They introduce a framework in which agents explore a POMDP, but have the additional option to make 'action queries' to an oracle. The method used is comparable to ours and their concept of *Bayesian Risk* resembles the concept of measuring value introduced here. However, since their method relies on action queries instead of measurements, results cannot easily be compared.

We also note some related papers which explore active measure learning in different contexts. Yin et al. (2020) propose a method for AMRL which relies on a pre-trained neural network to infer missing information. Ghasemi and Topcu (2019) propose a method to choose near-optimal measurements on a limited budget per step, which can be used to improve pre-computed 'standard' POMDP policies. Bernardino et al. (2022) investigate diagnosing patients using an MDP approach, in which the action themselves correspond to taking measurements. Mate et al. (2020) consider a restless multi-armed bandit setting where executing an action also resolves uncertainty for the chosen arms. Lastly, Araya-López et al. (2011) study how to approximate an MDP without a reward function.

Conclusion

In this paper, we proposed the *act-then-measure* heuristic for ACNO-MDPs and proved that the lost return for following it is bounded. We then proposed *measuring value* as a metric for the value of measuring in ACNO-MDPs. We describe Dyna-ATMQ as an RL algorithm following the ATM heuristic, and show empirically it outperforms prior RL methods for ACNO-MDPs in the tested environments.

Future work could focus on improving the performance of Dyna-ATMQ, for example, by implementing more sophisticated action choices and Q-updates, or by taking epistemic uncertainty more into account for exploration. To improve scalability, an interesting line of research is to adapt an already existing method to use the ATM-heuristic. Model-based methods, such as MBPO (Janner et al. 2019), are most suitable for such adaptations. Another possible direction is to investigate the ATM-heuristic in the more general active measure POMDP setting, in which we lose the assumption of complete and noiseless measurements. Lastly, our approach could be considered in different multiobjective settings, such as one where the preference function for reward and measurement cost is not known a-priori (Marler and Arora 2004), or where the measuring cost is used as a constraint (Ghasemi and Topcu 2019).

Acknowledgments

This research has been partially funded by NWO grant NWA.1160.18.238 (PrimaVera) and the ERC Starting Grant 101077178 (DEUCE).

References

Araya-López, M.; Buffet, O.; Thomas, V.; and Charpillet, F. 2011. Active Learning of MDP Models. In *EWRL*, volume 7188 of *Lecture Notes in Computer Science*, 42–53. Springer.

Bellinger, C.; Coles, R.; Crowley, M.; and Tamblyn, I. 2021. Active Measure Reinforcement Learning for Observation Cost Minimization. In *Canadian Conference on AI*. Canadian Artificial Intelligence Association.

Bernardino, G.; Jonsson, A.; Loncaric, F.; Castellote, P. M.; Sitges, M.; Clarysse, P.; and Duchateau, N. 2022. Reinforcement Learning for Active Modality Selection During Diagnosis. In *MICCAI (1)*, volume 13431 of *Lecture Notes in Computer Science*, 592–601. Springer.

Brafman, R. I.; and Tennenholtz, M. 2002. R-MAX - A General Polynomial Time Algorithm for Near-Optimal Reinforcement Learning. *J. Mach. Learn. Res.*, 3: 213–231.

Brockman, G.; Cheung, V.; Pettersson, L.; Schneider, J.; Schulman, J.; Tang, J.; and Zaremba, W. 2016. Openai gym. *arXiv preprint arXiv:1606.01540*.

Chrisman, L. 1992. Reinforcement Learning with Perceptual Aliasing: The Perceptual Distinctions Approach. In *AAAI*, 183–188. AAAI Press / The MIT Press.

Dearden, R.; Friedman, N.; and Andre, D. 1999. Model based Bayesian Exploration. In *UAI*, 150–159. Morgan Kaufmann.

Doshi-Velez, F.; Pineau, J.; and Roy, N. 2012. Reinforcement learning with limited reinforcement: Using Bayes risk for active learning in POMDPs. *Artif. Intell.*, 187: 115–132.

Dulac-Arnold, G.; Levine, N.; Mankowitz, D. J.; Li, J.; Paduraru, C.; Gowal, S.; and Hester, T. 2021. Challenges of real-world reinforcement learning: definitions, benchmarks and analysis. *Mach. Learn.*, 110(9): 2419–2468.

Emani, P.; Hamlet, A. J.; and Crane, C. 2015. POMDPy: An Extensible Framework for Implementing POMDPs in Python. *arXiv preprint arXiv:2004.10099*.

Ghasemi, M.; and Topcu, U. 2019. Online Active Perception for Partially Observable Markov Decision Processes with Limited Budget. In *CDC*, 6169–6174. IEEE.

Guo, Z. D.; Doroudi, S.; and Brunskill, E. 2016. A PAC RL Algorithm for Episodic POMDPs. In *AISTATS*, volume 51 of *JMLR Workshop and Conference Proceedings*, 510–518. JMLR.org.

Janner, M.; Fu, J.; Zhang, M.; and Levine, S. 2019. When to Trust Your Model: Model-Based Policy Optimization. In *NeurIPS*, 12498–12509.

Jimenez-Roa, L. A.; Heskes, T.; Tinga, T.; Molegraaf, H. J.; and Stoelinga, M. 2022. Deterioration modeling of sewer pipes via discrete-time Markov chains: A large-scale case study in the Netherlands. In *32nd European Safety and Reliability Conference, ESREL 2022: Understanding and Managing Risk and Reliability for a Sustainable Future*, 1299–1306.

Kormushev, P.; Calinon, S.; and Caldwell, D. G. 2013. Reinforcement Learning in Robotics: Applications and Real-World Challenges. *Robotics*, 2(3): 122–148.

Krale, M.; Simão, T. D.; and Jansen, N. 2023. Act-Then-Measure: Reinforcement Learning for Partially Observable Environments with Active Measuring. *arXiv preprint arXiv:2303.08271*.

Lei, L.; Tan, Y.; Zheng, K.; Liu, S.; Zhang, K.; and Shen, X. 2020. Deep Reinforcement Learning for Autonomous Internet of Things: Model, Applications and Challenges. *IEEE Commun. Surv. Tutorials*, 22(3): 1722–1760.

Littman, M. L.; Cassandra, A. R.; and Kaelbling, L. P. 1995. Learning Policies for Partially Observable Environments: Scaling Up. In *ICML*, 362–370. Morgan Kaufmann.

Marler, R.; and Arora, J. 2004. Survey of Multi-Objective Optimization Methods for Engineering. *Structural and Multidisciplinary Optimization*, 26: 369–395.

Mate, A.; Killian, J. A.; Xu, H.; Perrault, A.; and Tambe, M. 2020. Collapsing Bandits and Their Application to Public Health Intervention. In *NeurIPS*.

Nam, H. A.; Fleming, S. L.; and Brunskill, E. 2021. Reinforcement Learning with State Observation Costs in Action-Contingent Noiselessly Observable Markov Decision Processes. In *NeurIPS*, 15650–15666.

Silver, D.; and Veness, J. 2010. Monte-Carlo Planning in Large POMDPs. In *NIPS*, 2164–2172. Curran Associates, Inc.

Simão, T. D.; Suilen, M.; and Jansen, N. 2023. Safe Policy Improvement for POMDPs via Finite-State Controllers. In *AAAI*. ArXiv preprint arXiv:2301.04939.

Sunberg, Z.; and Kochenderfer, M. J. 2022. Improving Automated Driving Through POMDP Planning With Human Internal States. *IEEE Trans. Intell. Transp. Syst.*, 23(11): 20073–20083.

Sutton, R. S. 1991. Dyna, an Integrated Architecture for Learning, Planning, and Reacting. *SIGART Bull.*, 2(4): 160–163.

Watkins, C. J. C. H.; and Dayan, P. 1992. Technical Note Q-Learning. *Mach. Learn.*, 8: 279–292.

Yin, H.; Li, Y.; Pan, S. J.; Zhang, C.; and Tschiatschek, S. 2020. Reinforcement Learning with Efficient Active Feature Acquisition. *arXiv preprint arXiv:2011.00825*.

Yu, C.; Liu, J.; Nemati, S.; and Yin, G. 2023. Reinforcement Learning in Healthcare: A Survey. *ACM Comput. Surv.*, 55(2): 5:1–5:36.

Zanette, A.; and Brunskill, E. 2019. Tighter Problem-Dependent Regret Bounds in Reinforcement Learning without Domain Knowledge using Value Function Bounds. In *ICML*, volume 97 of *Proceedings of Machine Learning Research*, 7304–7312. PMLR.