Active Measure Reinforcement Learning for Observation Cost Minimization

Colin Bellinger[†,*], Rory Coles[‡], Mark Crowley[◇], Isaac Tamblyn[†,**]
[†] National Research Council of Canada
Ottawa, Canada
[‡] University of Victoria, Victoria, Canada
[◇] University of Waterloo, Waterloo, Canada
Vector Institute for Artificial Intelligence, Toronto, Canada

Abstract

Markov Decision Processes (MDP) with explicit measurement cost are a class of environments in which the agent learns to maximize the costed return. Here, we define the costed return as the discounted sum of rewards minus the sum of the explicit cost of measuring the next state. The RL agent can freely explore the relationship between actions and rewards but is charged each time it measures the next state. Thus, an optimal agent must learn a policy without making a large number of measurements. We propose the active measure RL framework (Amrl) as a solution to this novel class of problem, and contrast it with standard reinforcement learning under full observability and planning under partially observability. We demonstrate that Amrl-Q agents learn to shift from a reliance on costly measurements to exploiting a learned transition model in order to reduce the number of real-world measurements and achieve a higher costed return. Our results demonstrate the superiority of Amrl-Q over standard RL methods, Q-learning and Dyna-Q, and POMCP for planning under a POMDP in environments with explicit measurement costs.

Keywords: Reinforcement Learning, Active Learning, Partial Observability, Sample Efficiency

1. Introduction

In sequential decision making, an agent learns a policy that maps states to actions with the objective of maximizing the discounted long-term reward. Policy learning is achieved via online interactions with the environment. Given the current state, the agent selects an action, receives a numeric reward and the next state from the environment, and then updates its policy.

We present a sub-class of this problem that is characterized by explicit, and potentially heterogeneous, state measurement costs. Specifically, at every time t there is an explicit cost associated with knowing the next state. This cost lowers the reward received from the environment. The agent, however, has the option to forgo the measurement cost at any time by not measuring the state of the environment. To achieve this, we utilize the concept of action pairs. At each time t, the agent selects an action pair that indicates which action to take (*e.g.*, move left) and whether or not to measure the next state of the environment. Problems of this nature occur in drug design and quantum control, for example, where at each time step the actor can choose to pay (with time and resources) for an accurate measurement of the current state, or forgo the cost and choose the next action according to its intuition or learned model.

To solve this sub-class of sequential decision making, we propose the **Active Measure Reinforcement Learning (Amrl)** framework. Similar to model-based RL [1–3], Amrl agents learn a policy π and a dynamics model M in parallel via interactions with the environment. In Amrl, the policy maps states to actions pairs, $\pi : s \rightarrow ap$, and the dynamics

[*] colin.bellinger@nrc-cnrc.gc.ca [**] isaac.tamblyn@nrc-cnrc.gc.ca

model is used to estimate the next state given the current state and an action, $M : s, a \rightarrow s'$. Whereas model-based RL utilizes the model to reduce the training time or the number of interactions with the environment, Amrl exploits its model to reduce the measurement costs. To this end, model-based RL agents update the policy, in part, based on state and reward trajectories simulated by the model. Alternatively, Amrl interacts with the environment at each time step. The true reward signal is always used and, when appropriate, model estimates of the next state are utilized in place of costly real-world measurements of the next state of the environment.

By choosing between measuring the next state of the environment at a cost or estimating it, Amrl utilizes an active learning strategy. Active learning is typically applied to supervised machine learning problems with the aim of reducing the cost of expensive labelling of training data [4]. The learner selects a limited number of samples from an unlabelled pool that are expected to most improve the classifier to be labelled by an oracle. Amrl views the environment as an oracle that can return the true state at a cost. During learning, Amrl actively shifts from initially paying to measure the next state to estimating the next state when it has more experience. This enables the agent to learn a good policy and dynamics model, and to reduce its costs over time.

Due to the agent shifting between measurements of the true state of the environment and the use of estimates, Amrl resides in the grey area between fully observable Markov Decision Processes (MDPs) and Partially Observable Markov Decision Processes (POMDPs). At the extremes, the agent can opt to operate exclusively in a fully observable world by paying for a state measurement at each time step or rely entirely on its learned model. The former enables standard model-free RL methods to be applied [5]. This ensures a good policy, but will not reach the optimal costed return. The latter requires model-based RL [1–3] or classical planning under POMDP methods [6] to be used, which require more state measurements for model learning and computationally complex planning. The problem can be solved more efficiently and with a higher return via Amrl, which dynamically shifts between measurements and estimates based on its past experience.

We analyze the suitability of POMDP solutions and demonstrate an implementation of Amrl using Q-learning and a statistics-based dynamics model (Amrl-Q)[1]. We compare Amrl-Q to Q-learning and Dyna-Q on four benchmark learning environments, including a new chemistry motivated environment; specifically, the junior scientist environment. The results show that Amrl-Q achieves a higher costed return than Q-learning, Dyna-Q and POMCP, whilst learning at an equivalent rate to Q-learning and Dyna-Q.

1.1. **Contributions**

The main contributions of this work are:

- Formalization of MDPs with explicit observation costs
- Definition of the Active Measure RL framework (Amrl)
- Presentation of an initial implementation of the framework, Amrl-Q, based on Q-learning
- Analysis of Amrl-Q on novel and benchmark RL environments
- Demonstration Amrl-Q's advantage over Q-learning, Dyna-Q and POMCP.

2. **Related Work**

Previous work on *active reinforcement learning* has focused on ameliorating the problem of defining a complete reward function over the state-action space [7–9]. In addition to selecting an action at each time step, the agents in these proposals actively decide to request

[1]See https://github.com/cbellinger27/CanAI2021_AMRL for code related to this paper.

a human expert to provide the reward for the state-action pair. To minimize reliance on human experts, there is a cost assigned to requesting a human-specified reward. The agent aims to minimize this cost whilst maximizing the discounted sum of rewards. Alternatively, Amrl is designed to lower the state measurement costs by actively shifting between measuring of the true state and estimating the state based on its learned dynamics model.

MDPs with explicit measurement costs described herein are partially related to Constrained Markov decision processes (CMDPs) [10]. Specifically, both problems have multiple objectives and charge costs to different actions. CMDPs, however, have explicit constraints and are generally solved via linear programming, whereas Amrl learns online and does not have explicit constraints. Active perception also relates to our work in that the agent takes actions to increase the information available [11]. The key distinction is that active adaptive perception applied to RL employs self-modification and self-evaluation, such as moving to a new location, to improve its perception [12]. Moreover, the active adaptive perception agent receives observations at each time step, whereas in the proposed class of problem, the agent has the choice to measure the true state at a cost or use an estimate at no cost.

The learning of the state transition dynamics of the Amrl framework is similar to the techniques employed in model-based RL [13–15] and solutions for POMDPs [16]. Model-based RL aims to improve learning efficiency by reducing the number of real-world training steps needed to obtain an optimal policy, whereas our work is focused on the cost of measuring the true state rather than the number of interactions with the environment. Amrl agents aim to minimize the associated measurement costs. State estimators are applied in POMDPs to reduce uncertainty that arises from partial observability. In contrast, in Amrl, the agent is learning an optimal policy under an MDP with observation costs. The agent chooses between paying the cost to measure the *true state* of the environment s_t or estimating it as $\hat{s}_t$. Thus, in Amrl, the state estimator is primarily a mechanism to increase the costed return, rather than manage partial observability.

3. Preliminaries

We define Amrl environments as a tuple: $(S, A, P, S', R, C, \gamma)$. These are the standard components of an MDP, where S is the state-space, A is the action-space, $P(s'|s, a)$ is the state transition probabilities, $R(s, a)$ is the reward function, and $\gamma \in [0, 1]$ is a discount factor. P and R are not known by the agent. $C(m)$ returns the cost of measuring the next state, where $m = 1$ indicates measure, and $m = 0$ indicates do not measure.

$$C(m) = \begin{cases} c > 0, & \text{if } m = 1 \\ 0, & \text{otherwise.} \end{cases} \tag{3.1}$$

This cost is subtracted from the reward return from the environment.

At each time step t the agent selects an action pair. In Amrl, the action pair $ap = \langle a_t, m_t \rangle$ consists of an atomic process $a_t \in A$ (*e.g.*, move left) and a measurement indicator $m_t \in [0, 1]$. If $m_t = 1$, the process a_t is applied to the environment, and the environment returns the reward minus the cost, along with the next state $(r_{t+1} - c_{t+1}, s_{t+1} = Env(a_t, m_t))$. Here, s_{t+1} results from the underlying, unknown transition dynamics $P(s_t, a_t)$. For $m_t = 0$, the process a_t is applied to the environment, but the environment only returns the reward $r_{t+1} = Env(a_t, m_t)$. In this case, the Amrl agent estimates the next state $\hat{s}_{t+1} \sim M(s_t, a_t)$, and selects its next action pair $\langle a_{t+1}, m_{t+1} \rangle$ based on this estimate, $\hat{s}_{t+1}$. The agent starts each episode with a true measurement of the environment's current state, s_0, and proceeds to sequentially select action pairs, ap_t, that determine the process to be applied and whether to measure s_{t+1} or estimate it. Importantly, the reward emitted from the environment is always a function of the process a_t and the true state of the environment s_t irrespective of whether the agent selected a_t based on s_t or an estimate $\hat{s}_t$.

In this work, we focus on episodic environments with discrete states, $S = \{1, ..., |S|\}$ and action sets $A = \{1, ..., |A|\}$, and stationary state-transition dynamics. In an MDP with measurement costs, the objective is to select a sequence of action pairs $\langle a_t, m_t \rangle$ that maximize the **costed return**, which is defined as the discounted sum of rewards minus the sum of measurement costs: $v(s) = E\Big[\sum_{t=0}^{\infty} \gamma^t \big(R(s_t, a_t) - C(m_t)\big) \mid s = s_0\Big]$. In Amrl, a policy, π, maps states S and action pairs $A \times M$ to a probability $\pi : S \times (A \times M) \to [0, 1]$, such that $\pi(s, \langle a_t, m_t \rangle)$ is the probability of selecting action pair $(a, m) \in A \times M$ while in state $s \in S$. The value function associated with policy π is:

$$v_\pi(s) = E\Big[\sum_{t=0}^{\infty} \gamma^t \big(R(s_t, ap_t)) - C(ap_t)\big) \mid s = s_0\Big], \tag{3.2}$$

where the actions are selected according to π. Since action pairs $(A \times M)$ can be thought of as a higher-level class of action, the standard RL theorems hold, provided the agent learns an accurate model. Thus, there is exists at least one policy π^* such that $V^\pi(s) \leq V^{\pi^*}(s)$, where π^* is an optimal policy and V^* is the corresponding value function.

4. **Amrl-Q**

Here we propose an initial implementation of the Amrl framework for a tabular learning environment and leave continuous state and action spaces for future work.

4.1. **Overview**

The Amrl-Q framework learns a value function Q, and a dynamics model $M(S_{t+1}|S_t, a_t)$ in parallel. Learning M and Q is essential to the active learning-based solution which enables the agent to reduce the total number of times it requests a true measurement. The theory behind this can be demonstrated with the pedagogical Markov chain.

Consider a five-state linear Markov chain with states named in order from 0 to 4 and two actions (left, right). This forms an episodic RL problem where the agent starts in state 0 and receives a reward of one upon entering the absorbing state, state 4, and 0 elsewhere. For temporal difference (TD) learning methods, such as Q-learning, applied to episodic problems such as this, the value of states and actions is refined over multiple episodes of training from the state closest to the absorbing state back to the start state.

If we assume a Q-table initialized to all zeros, after one episode of training is complete, only $Q(s = 3, a = \text{right})$ will have a value greater than zero; if we assume batch updating, after the second episode is complete, states 2 and 3 will have values greater than zero, and so on. In general, for an n-state chain of this nature and batch updating, the agent will require $n - 1$ episodes of training to start to improve the Q-values associated with the start state, state 0.

In standard MDP solutions, the number of times the agent visits each state per episode indicates how many true measurements of the environment it will make. We can estimate this by calculating the fundamental matrix N of the absorbing Markov chain P. The fundamental matrix is defined as $N = (I - Q)^{-1}$, where I is the identify matrix and Q is the $t \times t$ matrix representing the transient states in P. Based on this, the expected number of state visits before absorbing for an agent starting in state 0 and following a random policy is 8,6,4 and 2, respectively. Thus, in the first four episodes of training, the Q-agent is expected to take 46 measurements of the environment.

If we consider the dynamics model M learned by Amrl, according to the calculations above, in the first episode of training the agent is expected to have tried both actions in each state 4, 3, 2 and 1 times, respectively. For a deterministic P, the agent need only try each state-action pair once to have an accurate M. Thus, an Amrl agent acting optimally

will switch from actively measuring the next state to estimating it with M after the first episode of training. In this way, Amrl can improve measurement efficiency well beyond what can be achieved by standard RL methods and model-based RL (for deterministic environments at least.)

Algorithm 1 Amrl-Q Algorithm.

Parameters: step size $\alpha \in (0, 1]$, optimistic initializer $\beta > 0$, small $\epsilon > 0$
Algorithm:

Optimistically initialize Q-table of size $|S| \times |A| \times 2$ with β
Initialize |A| state-transition statistic table M_a of size $|S| \times |S|$ to zeros.
while more episodes of training **do**
    Get first state s_0 from the environment
    while not done episode **do**
        Select action pair (a, m) with ϵ greedy policy from Q table for state s
        Apply action a to environment
        if measure $m = 1$ **then**
            Measure next state s' in environment
            Update state transition model for action a $M_a[s, s'] \mathrel{+}= 1$
        else
            Sample next state $s' \sim M_a(s)$
        Get reward $r = R(s, a)$ and cost $c = C(m)$ from environment
        Update Q table for state s with tuple $(s, a, r - c, s')$ using Eq. (4.1)
        Set $s \leftarrow s'$

4.2. **Algorithm**

The Amrl-Q algorithm maintains $|A|$ count-based statistics tables of size $|S| \times |S|$ (*i.e.,* one dynamics model, M_a, per action.) The agent maintains an $|S| \times (|A| \cdot 2)$ dimensional Q-table, where $|A| \cdot 2$ is the number of action pairs. An environment with 2 atomic actions, for example, has 4 possible action pairs in each state. The Q-table is updated as:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[(r_{t+1} - c_{t+1}) \, \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right] \tag{4.1}$$

The agent employs an ϵ-greedy strategy to pick action pairs from the Q-table. If the action pair at time t includes $m_t = 1$, then the agent chooses to pay the cost c of measuring the next state from the environment. When the agent chooses to measure the true state, it updates $M_a(s_t, s_{t+1})$ for the corresponding action $a = a_t$. Otherwise, the agent estimates the next state from its model as $s_{t+1} \sim M_a(s_t, a_t)$ and proceeds to make decision at time $t + 1$ using the state estimate.

Much like a human learning a new task, the first few times an agent enters a state it must measure the result of taking an action. We define the β parameter that serves to optimistically initialize columns of the Q-table involving action pairs with $m = 1$[2]. In general, optimistic initialization is used to promote exploration in the earlier episodes of training [17]. Here, it is used to encode prior knowledge that the agent ought to initially pay for true state measurements and shift to estimating after it has gained enough experience. Larger β values cause the agent to gain more experience before shifting to use its dynamics model. This is necessary in environments with more stochasticity. The algorithm is outlined in Algorithm 1.

[2]Q-values related to non-measurements, $m = 0$, are initialized to zero.

5. Experimental Setup

5.1. RL Environments

This section outlines the environments used to evaluate Amrl-Q. In each case, the environments are episodic and involve discrete states and actions. We consider both deterministic and stochastic state transition dynamics, and a wide range of measurement costs and reward structures.

Chain environments: We evaluate an 11-state, *standard chain environment* and a 20-state *even-odd chain environment*. In both cases, the episodes start in s_0 and they conclude in goal states s_{10} and s_{19} respectively. Upon entering goal state, the agent receives a reward of $r = 1$ in the standard chain and $r = 300$ in the even-odd chain. In all other states, the agent receives a reward of $r = -0.01$ in the standard chain and $r = -1$ in the even-odd chain. The agent selects from action pairs: (move left, don't measure) , (move left, measure), (move right, don't measure), (move right, measure). In the standard chain, measuring the next state has a cost of $c = 0.05$. Alternatively, we analyze the impact of measurement costs between 0.1 and 25 in the even-odd chain. In odd numbered states of the even-odd chain, the left and right actions are reversed to move the agent in the opposite direction. In general, state transitions include Gaussian additive noise: $s_{t+1} \sim P(s_{t+1}|s_t, a_t) + round(N(\sigma))$, where σ determines the extent of stochaticity.

Frozen Lake 8×8 ***environment***: This is a modified version of the openAI gym environment by the same name. In this version, the agent learns to navigate from a start location to a goal on either a slippery, which can cause the agent in move in unintended directions, or not slippery, 2-dimensional grid with holes. Each episode ends when the agent reaches the goal or falls through a hole. The agent receives a reward of $r = 1$ at the goal, $r = 0$ otherwise. The agent pays a cost of $c = 0.1$ for measuring the state of the environment. The action-space consists of actions pairs of move (move left, move right, move up, move down) and measure (0/1).

Taxi environment: This is a modified version of the openAI gym environment by the same name. The agent learns to navigate a 2-dimensional city grid world to pick up and drop off passengers at the appropriate location [18]. The agent receives a reward $r = 20$ for dropping off at the correct location, $r = -10$ for illegal pickup or drop-off and $r = -1$ at each time step. The agent is charged a cost of $c = 0.1$ for measuring the state of the environment. The action-space pairs of (move left, move right, move up, move down, pickup, drop-off) and measure (0/1).

Junior Scientist environment: This environment emulates a student learning to manipulate an energy source to produce a desired state change in a target material. Specifically, the agent starts with a sealed container of water composed of an initial h_0 percent ice, l_0 percent water and g_0 percent gas ($h_0 + l_0 + g_0 = 1$). The agent learns to sequentially and incrementally adjust a heat source in order to change the ratio of ice, liquid, gas from (h_0, l_0, g_0) to a goal ratio (h, l, g). The episode ends when the agent declares that it has reached the goal and it is correctly in the goal state. The action-space includes $A = \{$decrease, increase, done$\}$, where *decrease* and *increase* are fixed incremental adjustments in the energy source. The agent receives a reward of $r = 1$ when it reaches the goal and it correctly declares that it is done, and receives a reward of $r = -0.05$ at each time step. The agent is charged $c = 0.01$ for measuring the state of the environment. Measuring the state results in the environment returning the cumulative energy which has been added or removed from the system.

POMDP chain environment: This is a replica to the even-odd chain environment described above adopted for POMDP planning. In addition to states ($S \in \{0..19\}$), it has an observation space $O \in S \cup \{-1\}$. The agent select from the action pairs described above. As is standard in POMDPs, the agent receives an observation at o_t at each times step.

When the the agent selects action pairs with $m = 1$, the observation is equivalent to the true state, $o_{t+1} \leftarrow s_{t+1}$. When the agent does not measure, $m = 0$, $o_{t+1} = -1$.

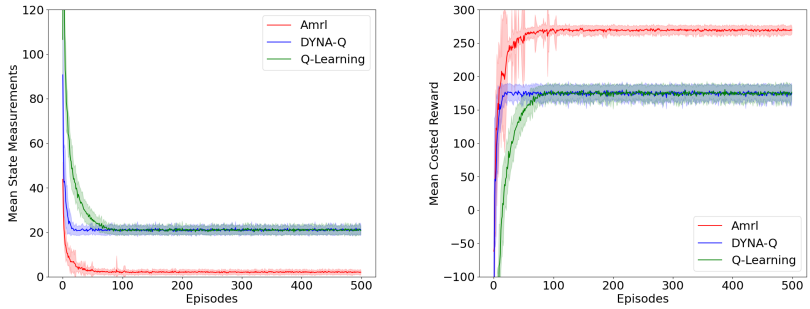

Figure 1. Left: Performance of Amrl-Q, DYNA-Q and Q-learning on the even-odd chain environment with deterministic state transitions ($N(\sigma = 0)$). The left plot shows mean number of state measurements per episode, and the right plot presents the mean costed reward. The red lines indicate that Amrl-Q achieves a higher costed reward and take fewer measurements on this environment.

5.2. **Algorithms for Comparison**

Because this work focuses on a new sub-class of problem, there are no clear direct competitors for comparison. The closest option is fully observable RL methods that pay the measurement cost at each time step. We analyze Q-learning [19] and Dyna-Q [1]. Dyna-Q particularly comparable because it learns a model of the environment and exploits it to improve sample efficiency. Alternatively, if we assume the availability of a model for planning, we can convert the problem to a POMDP. We consider this to be less desirable due to the need for a model and the complexity of planning under a POMDP. In addition, we compare Amrl-Q to Partially Observable Monte-Carlo Planning (POMCP) [6] in a POMDP setup.

5.3. **Evaluation Process**

We evaluate each algorithm based on the costed reward, number of steps and measurements per episode. The results show the mean and standard deviation calculated over 20 random trials. For each RL algorithm in our evaluation, we utilize a discount factor of $\gamma = 0.9$ and ϵ-greedy exploration $\epsilon = 0.1$. The Q-tables for both Q-learning and Dyna-Q are initialized to zeros, and Dyna-Q utilizes 5 planning steps after each real step. The β value in Amrl-Q is typically 0.1, however, we also explore the impact of higher optimistic initialization. For POMCP, we utilize random roll-outs, set the exploration constant to 200 and maximum depth to 5.

6. **Results**

6.1. **Comparison with RL Methods**

The mean performance of each agent on the even-odd chain environment with deterministic state transitions ($N(\sigma = 0)$) is shown in Figure 1. All three methods learn a policy that takes a similar number of steps to the goal (left plot). The learning curves show that Dyna-Q (blue line) learns slightly faster than Amrl-Q (red line) and significantly faster than Q-learning (green line). Amrl-Q quickly reduces it measurements and achieves a much higher costed return as a result. The number of steps demonstrates that whilst our method

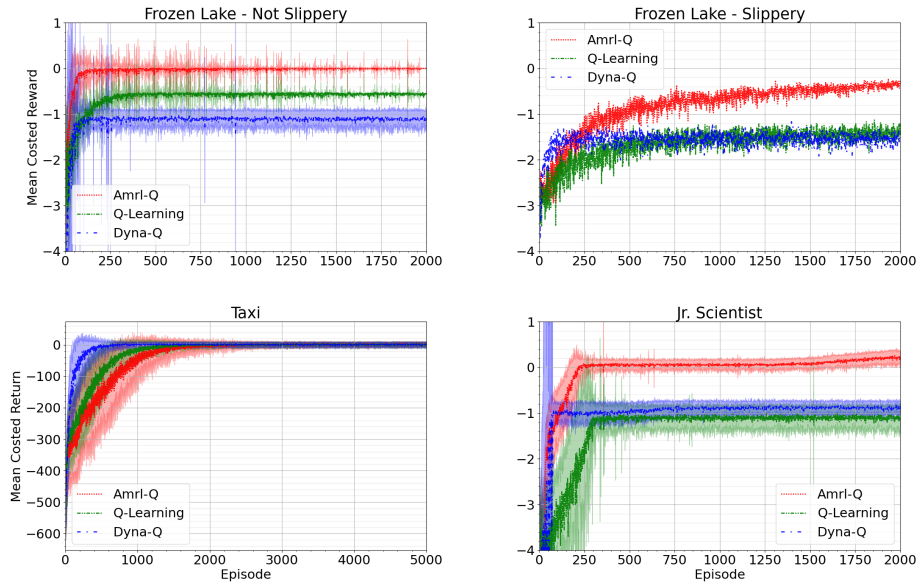

Figure 2. Comparison of mean costed rewards for Amrl-Q with Dyna-Q and Q-learning on frozen lake not slippery, frozen lake slipper, taxi and junior scientist. Amrl-Q achieves an equivalent or higher costed reward than the alternative methods.

learning an equally good policy at a similar rate, only Amrl-Q can increase the costed reward by actively shifting to estimate the next state.

Figure 2 shows the mean of the costed return for each algorithm on the Frozen Lake (slippery, and not slipper), Taxi and Junior Scientist environments. Amrl-Q has a clear advantage in terms of the mean costed reward on Frozen Lake slippery and not slippery, and Junior Scientist. This reemphasized in Table 1, which summarizes the mean number of measurements made by each algorithm.

Agent	Frozen Lake	Taxi
Random	31.95	31.95
Q-learning	15.45	14.83
Dyna-Q	13.99	14.67
Amrl-Q	10.5	12.13

Table 1. The mean number of measurements made by each algorithm per episode of training on frozen lake not slippery and taxi environments. In both environments, Amrl-Q requires fewer measurements than the alternative RL methods.

Amrl-Q learns faster than Q-learning and slightly slower than Dyna-Q on these environments. Dyna-Q has a slight advantage in terms of learning rate due to its use of model-based planning. Because Amrl is also a model-based method, Dyna-style planning can be integrated into it to further improve its convergence rate.

On Frozen Lake slippery, Amrl-Q requires a larger optimistic measure initialization (elevated from $\beta = 0.1$ to $\beta = 2.5$) to enable the agent to collect accurate statistics about the stochastic state transitions. We explore the relationship between β and stochastic transitions in next section. In the Taxi environment, Amrl-Q only acquires a slightly higher mean costed reward than the alternative methods after 2,500 episodes. This is due to the complexity of the environment, low measurement cost $c = 0.1$ and relatively large rewards, $r = 10$. As we demonstrate in the subsequent section, the relative size of cost and reward strongly influence the benefit of Amrl-Q.

6.2. **Comparison with POMDP Planning**

As an additional angle of comparison, we implemented a POMDP version of the 20-state active measure chain problem to compare POMCP with Amrl-Q[3]. Although Amrl-Q learned an optimal policy, POMCP was unable to effectively plan in this setting. This is irrespective of the fact that POMCP had access to the true model of the environment. Unlike Amrl, which actively shifted from paying for measurements to reliance on its model, POMCP myopically opted to save the cost by never measuring the true state. This resulted in the agent moving randomly in the environment. After 100 episodes of training, Amrl-Q received an average costed return of 263 and took 20.8 steps to the goal. Alternatively, POMCP achieved an average return of -268.69 and took 250 steps on average. Further experimentation revealed that, in order for POMCP to accurately plan, it had to be incentivised to measure the state by assigning negative measurement cost. This completely voids the purpose of the experiments.

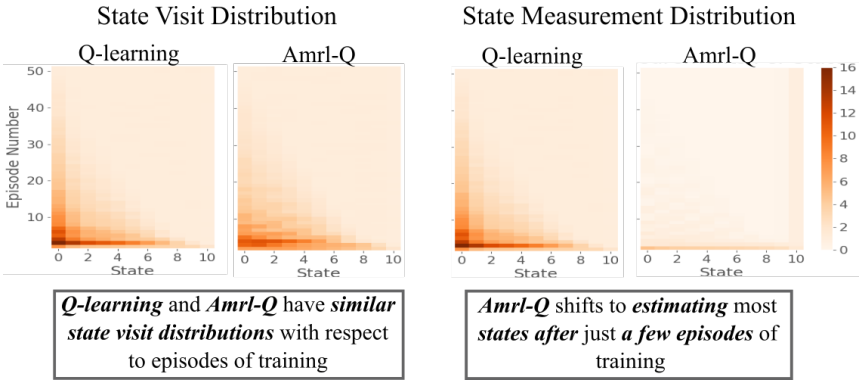

Figure 3. Two-dimensional histograms comparing of the number of state visits and measurements made by Q-learning versus Amrl-Q on the Chain environment. Amrl-Q visits the states with a similar frequency as Q-learning but makes significantly fewer measurements.

6.3. **Amrl-Q Analysis**

***Active Measurements*:** Figure 3 contains four 2-dimensional histograms. These depict the number of visits to each state (plots 1 and 2) and the number of measurements in each state (plots 3 and 4)[4] as a function of episodes of training. The x-axis specifies the state in the standard chain and the y-axis indicates the number of episodes of training completed. The darker orange cells indicate more visits / measurements, whilst the lighter colouring indicates a low number of visits. This highlights that Q-learning and Amrl-Q follow similar patterns in terms of the number of visits to each state across episodes of training. The state measurement distribution plots, however, emphasize that after just a few episodes of training, Amrl-Q shifts to estimated the next state. The max state measurement value for Amrl-Q (rightmost plot) is 6, in comparison to 16 for Q-Learning. In fewer than 30 episodes of training, the Amrl-Q is able to replace all measurements with its own estimate

***Amrl-Q Model Accuracy*:** Figure 4 displays an analysis of the relationship between performance, β-value in Amrl-Q and the amount of stochasticity in the even-odd chain environment. The rows of the figure show mean costed reward, cumulative number of observations made by the agent and the accuracy of Amrl-Q's model as a function of episodes

[3]This was implemented with the pomdp_py framework [20].
[4]Q-learning measures the state on each visit, therefore, plots 1 and 3 are the same.

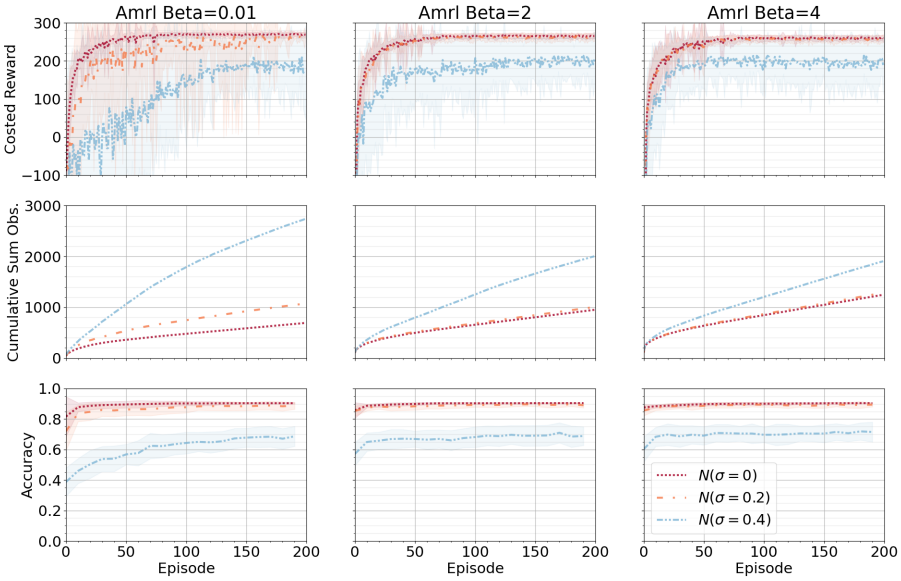

Figure 4. Relationship between stochastic state transition dynamics ($N(\sigma)$) and mean costed reward (top row), cumulative sum of observations (middle row) and model accuracy (bottom row) for Amrl-Q with β-values 0.01, 2 and 4.

of training. Each line in the plots specificies the amount of Gaussian noise in the state transitions dynamics ($N(\sigma = 0)$, $N(\sigma = 0.2)$, $N(\sigma = 0.4)$).

The results in the bottom row of Figure 4 demonstrate that for moderate levels of stochaticity ($N(\sigma = 0)$, $N(\sigma = 0.2)$), the agent's model quickly achieves prediction accuracy near 0.9 with $\beta = 0.01$. When the environment has more stochasticity in the transition dynamics ($N(\sigma = 0.4)$), setting the β-value too low ($\beta = 0.01$) causes model accuracy to start of low and take more time to improve. This has short- and medium-term implications of the learned policy and costed return.

Setting the β-value too high causes the agent to pay for more measurements than needed before shifting to use its model. We can see this by comparing the first and third cumulative sum of observations plots in the middle row. With $\beta = 4$ and $N(\sigma = 0)$ or $N(\sigma = 0.2)$, the agent has a much higher cumulative sum of observations, and lower than optimal costed return. Alternatively, these results also show that poor accuracy in the model causes the agent to make more active measurements per episode. Comparing the cumulative sum of observations for $\beta = 0.01$ and $N(\sigma = 0.4)$ to $\beta = 4$ and $N(\sigma = 0.4)$, the former makes significantly more measurements per episode. This is irrespective of the fact that latter is parameterized to take more measurements. This results from the fact that the agent is relying on an inaccurate model causing it to take more steps to get the to goal, and measurement more per episode.

Nonetheless, the results suggest that the model performance is robust to a wide range of β settings. For low stochasticity ($N(\sigma = 0)$, $N(\sigma = 0.2)$), the Amrl-Q model performs consistently well with $\beta = 0.01$ and $\beta = 2$. Model performance is only slightly worse $\beta = 0.01$ on $N(\sigma = 0.2)$ than with $\beta = 2$. Alternatively, for $N(\sigma = 0.4)$, model performance with $\beta = 2$ is only slightly less than with $\beta = 4$. Thus, although performance is sensitive to β, good performance can be achieve with a wide range settings

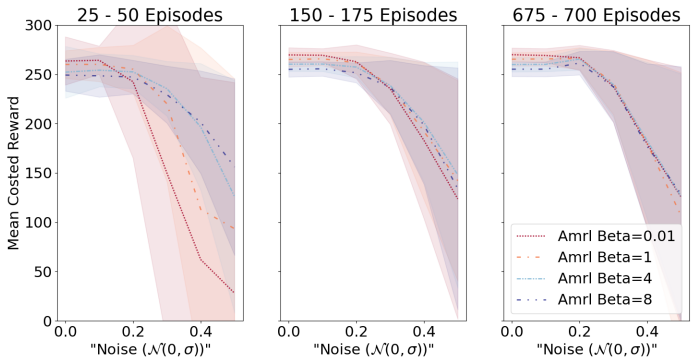

Figure 5. Relationship between the β parameter in Amrl and the amount of stochasticity in the state transition dynamics. The figures show the mean performance after 50-75, 150-75, and 675-700 episodes of training. A low β-value in environments with low stochasticity enables efficient policy learning. Large β-value perform better in the short-term when stochasticity is high. In the long-term, the costed returns of similar for low and high β-value.

***Relationship Between β and Stochasticity*:** Figure 5 plots the mean costed reward acquired by Amrl-Q with β-values of 0.01,1,4 and 8, versus the degree of stochasticity in the transition dynamics in the even-odd chain environment. The plots report the mean costed rewards averaged between 25-50, 150-175 and 675-700 episodes of training. The plots demonstrate that a low β-value enables the agent to quickly get a higher costed reward when the stochasticity is low to moderate. Thus, in environments with low to moderate stochasticities, a low β-value will enable the agent to quickly learn a good policy.

For higher levels of stochasticity ($N(\sigma > 0.2)$), however, an agent with a low β-value does poorly relative to those with higher β-values in the early episodes of training. Interestingly, the third plot demonstrates that in the even-odd chain environments with higher stochasticities, the agents with low β-values (0.01 and 1) recover the different in the costed reward in the later episodes of training. Once again, this suggests a degree of robustness to the setting of β.

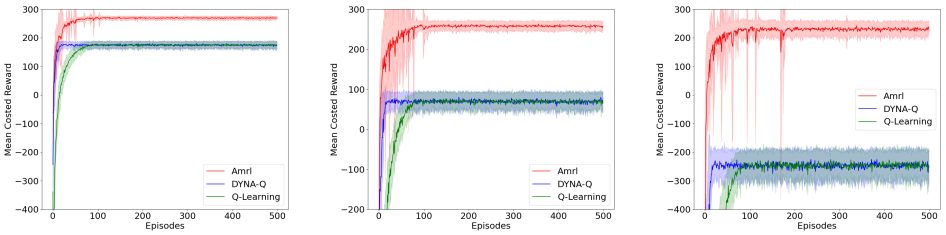

Figure 6. Mean costed reward for Amrl-Q, Dyna-Q and Q-learning on the even-odd chain with $N(\sigma = 0)$ and measurement costs, c, equal to 5 (left), 10, (middle) and 25 (right). Amrl-Q's relative advantage grows with increasing measurement costs.

***Measurement Costs*:** Figure 6 depicts the mean costed return for Amrl-Q, Dyna-Q and Q-learning on the even-odd chain with $N(\sigma = 0)$ and measurement costs, c, equal to 5 (left), 10, (middle) and 25 (right). At the lower extreme of $c = 0$ (not shown here), each method converges to the same mean costed reward. As demonstrated in Figure 6, and our previous results, Amrl-Q has an advantage in terms of the costed reward when $c > 0$, and this advantage grows with c.

7. Conclusion

We introduced a sequential decision-making framework, Amrl, in which the agent selects an action and whether or not to measure the next state at a cost at each time step. We formulate our solution in terms of active learning, and empirically show that Amrl-Q learns to shift from relying on costly measurements to using its learned dynamics to increase the costed reward. Our results demonstrate the superiority of Amrl-Q over standard RL methods, Q-learning and Dyna-Q, POMCP for planning under a POMDP. This has the potential to expand the applicability of RL to important applications in operational planning, scientific discovery, and medical treatments. To achieve this, additional research is required to develop Amrl methods for continuous state and action environments, and function approximation methods, such as deep learning.

References

[1] R. S. Sutton. "Integrated architectures for learning, planning, and reacting based on approximating dynamic programming". In: *Machine learning proceedings 1990*. Elsevier, 1990, pp. 216–224.

[2] S. Gu et al. "Continuous deep q-learning with model-based acceleration". In: *International Conference on Machine Learning*. PMLR. 2016, pp. 2829–2838.

[3] I. Clavera et al. "Model-based reinforcement learning via meta-policy optimization". In: *Conference on Robot Learning*. PMLR. 2018, pp. 617–629.

[4] B. Settles. "Active learning". In: *Synthesis Lectures on Artificial Intelligence and Machine Learning* (2012), pp. 1–114.

[5] V. Mnih et al. "Human-level control through deep reinforcement learning". In: *Nature* 518.7540 (2015), p. 529.

[6] D. Silver and J. Veness. "Monte-Carlo planning in large POMDPs". In: *Advances in neural information processing systems*. 2010, pp. 2164–2172.

[7] R. Akrour, M. Schoenauer, and M. Sebag. "April: Active preference learning-based reinforcement learning". In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer. 2012, pp. 116–131.

[8] D. Krueger et al. "Active reinforcement learning: Observing rewards at a cost". In: *Future of Interactive Learning Machines, NIPS Workshop*. 2016.

[9] S. Schulze and O. Evans. "Active reinforcement learning with monte-carlo tree search". In: *arXiv preprint arXiv:1803.04926* (2018).

[10] E. Altman. *Constrained Markov decision processes*. Vol. 7. CRC Press, 1999.

[11] J. J. Gibson. "The senses considered as perceptual systems." In: Houghton Mifflin, 1966.

[12] D. Bossens, N. C. Townsend, and A. Sobey. "Learning to learn with active adaptive perception". In: *Neural Networks* 115 (2019), pp. 30–49.

[13] M. Deisenroth and C. E. Rasmussen. "PILCO: A model-based and data-efficient approach to policy search". In: *Proceedings of the 28th International Conference on machine learning (ICML-11)*. 2011, pp. 465–472.

[14] V. Kumar, E. Todorov, and S. Levine. "Optimal control with learned local models: Application to dexterous manipulation". In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2016, pp. 378–383.

[15] Y. Gal, R. McAllister, and C. E. Rasmussen. "Improving PILCO with Bayesian neural network dynamics models". In: *Data-Efficient Machine Learning workshop, ICML*. Vol. 4. 2016, p. 34.

[16] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. "Planning and acting in partially observable stochastic domains". In: *Artificial intelligence* 101.1-2 (1998), pp. 99–134.

[17] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[18] T. G. Dietterich. "Hierarchical reinforcement learning with the MAXQ value function decomposition". In: *Journal of artificial intelligence research* 13 (2000), pp. 227–303.

[19] C. J. Watkins and P. Dayan. "Q-learning". In: *Machine learning* 8.3-4 (1992), pp. 279–292.

[20] K. Zheng and S. Tellex. "pomdp_py: A Framework to Build and Solve POMDP Problems". In: *arXiv preprint arXiv:2004.10099* (2020).