Learning When to Observe: A Frugal Reinforcement Learning Framework for a High-Cost World

Colin Bellinger[1(✉)], Isaac Tamblyn[2,3], and Mark Crowley[4]

[1] National Research Council of Canada, Ottawa, Canada
colin.bellinger@nrc-cnrc.gc.ca
[2] Department of Physics, University of Ottawa, Ottawa, Canada
isaac.tamblyn@uottawa.ca
[3] Vector Institute for Artificial Intelligence, Toronto, ON, Canada
[4] Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, Canada
mark.crowley@uwaterloo.ca

Abstract. Reinforcement learning (RL) has been shown to learn sophisticated control policies for complex tasks including games, robotics, heating and cooling systems and text generation. The action-perception cycle in RL, however, generally assumes that a measurement of the state of the environment is available at each time step without a cost. In applications such as materials design, deep-sea and planetary robot exploration and medicine, however, there can be a high cost associated with measuring, or even approximating, the state of the environment. In this paper, we survey the recently growing literature that adopts the perspective that an RL agent might not need, or even want, a costly measurement at each time step. Within this context, we propose the Deep Dynamic Multi-Step Observationless Agent (DMSOA), contrast it with the literature and empirically evaluate it on OpenAI gym and Atari Pong environments. Our results, show that DMSOA learns a better policy with fewer decision steps and measurements than the considered alternative from the literature. The corresponding code is available at: https://github.com/cbellinger27/Learning-when-to-observe-in-RL.

Keywords: Reinforcement Learning · DQN · Sensing and Observation Costs · Noiselessly Observable Markov Decision Processes

1 Introduction

In many applications of reinforcement learning (RL), such as materials design, computational chemistry, deep-sea and planetary robot exploration and medicine [2,13,15], there is a high cost associated with measuring, or even approximating, the state of the environment. Thus, the RL system as a whole faces observation costs in the environment, along with processing and decision making costs in the

R. Meo and F. Silvestri (Eds.): ECML PKDD 2023, CCIS 2136, pp. 242–257, 2025.
https://doi.org/10.1007/978-3-031-74640-6_18

agent. On both sides, the costs result from a cacophony of factors including the use of energy, systems and human resources. In this work, we propose the Deep Dynamic Multi-Step Observationless Agent (DMSOA), the first RL agent in its class to reduce both measurement and decision making costs.

Since standard RL agents require a large number of *state-action-reward-next state* interactions with the environment during policy learning and application, the measurement and decision making costs can be very high. Traditionally, these underlying costs are hidden from the agent. Indeed, little consideration has been given to the idea that the agent might not need, or even want, a potentially costly observation at each time step. In the real-world, however, agents (animal or artificial) are limited by their resources. To save time and energy, decision making associated with common or predictable tasks is believed to be conducted re-actively or based on fast, low-resource systems. Only with deliberate cognitive intervention are the slower, resource-intensive planning systems used [8,22].

Recently, there have been a number of interesting conference papers, workshop discussions and theses discussing how to address this problem in RL [4–6,9,11,15,20]. The general approach is to augment RL by a) assign an intrinsic cost to measure the state of the environment, and b) provide the agent with the flexibility to decide if the next state should be measured. Together, these provide a mechanism and a learning signal to encourage the agent to reduce its intrinsic measurement costs relative the to the explicit control rewards it receives.

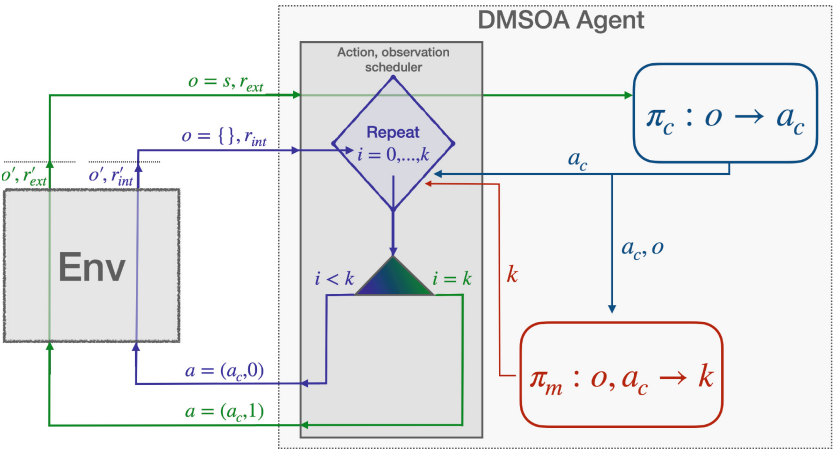

Fig. 1. Illustration of the DMSOA Framework.

When the agent opts not to measure the state of the environment, it must make its next control decision based on stale information or an estimate. Thus, at the highest level, this constitutes a partially observable Markov decision process (POMDP). Learning a POMDP, however, is much more difficult than learning a Markov decision process (MDP) with RL [17]. When designed as shown in

Fig. 1, the agent's experience is composed of fully observable measurements and partially observable estimates of the state of the environment. Thus, the problem is related to mixed observable Markov decision processes (MOMDPs), which are an easier sub-class of POMDPs [16]. The authors in [15] denoted this class of RL problem as action-contingent, noiselessly observable Markov decision processes (AC-NOMDPs). Distinct from an MOMDP, action-contingent in AC-NOMDP relates the fact that the agent explicitly chooses between measuring and not measuring the environment. "Noiselessly observable" relates to the fact that when the agent decides to measure at a cost, the state is fully observable. Although previous works have used different terms to refer to this class of problem, we believe that AC-NOMDP is the most descriptive of the underlying dynamics and use it throughout this paper.

Recently, [15] provided a theoretical analysis of the advantages of AC-NOMDP of over a general POMDP formulation and found a significant improvement in efficiency for RL with explicit observation costs and actions. All other previous works have carried out limited empirical evaluations in which the proposed algorithm is compared to a baseline MDP [5,9,11,20]. Moreover, the previous analyses primarily relied on just a few, or even single, experiments on OpenAI gym classic control environments and grid-worlds [5,11,20].

In this work, we compare DMSOA to the one-step memory-based observationless agent (OSMBOA) recently proposed in [5]. Our contribution serves to expand the state-of-the-art in AC-NOMDP algorithms and the understanding of how different classes of algorithms impact the observation behaviour. OSMBOA was selected for its demonstrated effectiveness and easy of use. At each time step, OSMBOA selects a control action and makes a decision about measuring the next state of the environment. If no measurement is made, the agent's next control action is selected based on its fixed-size internal memory of the last measured state(s). In contrast, DMSOA selects a control action and the number of times to apply the action before measuring the next state. Therefore, DMSOA learns to reduce both observation costs and decision making costs by dynamically applying its control action multiple times.

To facilitate fair comparison, we implement both agents as double DQN [25] with prioritized experience replay [19]. We evaluate the agents in terms of the accumulated extrinsic control reward and by the reduction in the number of observations and decision steps made on Atari Pong and OpenAI gym [7]. The results show that the proposed method learns a better control policy, requires fewer measurements of the environment and decision steps. Moreover, DMSOA has less variance across independent training runs.

The remainder of the paper is organized as follows. In the next section, Sect. 2, we outline the related work. Section 3 formalizes the AC-NOMDP problem and Sect. 4 presents the proposed algorithms. Section 5 provides the experimental setup. The results are shown in Sect. 6 and discussed in Sect. 7. The environmental impact of this work is described in Sect. 8 and our concluding remarks are in Sect. 9.

2 Related Work

This work fits into a small but growing sub-area of RL in which observations are optional at each time step and have an explicit cost to the agent when they are made. In the subsection immediately below, we provide an overview of methods recently applied to AC-NOMDP. Following that, we discuss the literature directly related to the proposed DMSOA algorithm.

2.1 AC-NOMDP Methods

In the existing work, the authors in [4,15,20] proposed tabular Q-learning-based algorithms for AC-NOMDPs and [5,9,11,15] proposed deep RL based methods. In [11], the authors modify TRPO, and in [9,15], actor-critic frameworks with a recurrent neural networks are used. In [5], the authors provide a wrapper class that modifies the underlying environment by expanding the observation and action spaces to facilitate any off-the-shelf deep RL algorithm to work in the AC-NOMDP setting.

Through our analysis of the literature, we have identified 4 key questions addressed when developing for AC-NOMDPs. These are: *1*) the mechanism by which the agent expresses its desire not to measure, *2*) how the observation is supplemented when no measurement is made, *3*) how the agent is encouraged to reduce its reliance of costly measurements, and *4*) how the agent is constructed.

The most common way to handle question *1*) is by expanding the action space. In the case of discrete actions, [4,5,9,15,20] expanded the action space to action tuples: $\langle \text{control actions} \rangle \times \langle \text{measure, don't measure} \rangle$. Alternatively, in [11], the agent specifies the control action plus a sample purity value $q \in \mathbb{R}^1$, where a larger q triggers a less accurate measurement with lower associated costs.

With respect to question *2*), when the agent does not request a fresh measurement in [4,9,11,15], the environment sends a Null state observation or an observation composed of zeros. In [4], the agent uses an internal statistical model to estimate the next state and in [9,15] the agents utilize a deep recurrent networks for estimating belief states and encoded states, respectively. In [5], the agent utilizes a fix-size memory of recent measurements when no measurement is made. To reduce partial observability, each observation is augmented with a flag indicating whether or not it is the result of a fresh measurement of the environment. Since the agent in [11] adjusts the noise level rather than turning on and off measurements, it makes its next action selection purely based on the noisy measurement returned.

Question *3*) relates to the rewards structure. This is generally divided into intrinsic rewards, which are used to encourage the agent to reduce its reliance on costly measurements and extrinsic rewards that push the agent to achieve the control objective. At each time step in [4,9,11,15], an intrinsic cost is subtracted from the extrinsic reward if the agent measures the state. Alternatively, in [21] a positive intrinsic reward is added when the agent foregoes a measurement. A critical point that remains unclear in the literature is how to acquire the extrinsic reward when no measurement is made. In most cases, this is simply

assumed to be available. We argue that if no measurement is made, the extrinsic reward cannot be known. As a result, in this work only the intrinsic portion of the reward is provided when no measurement is made.

The final question relates to the architecture of the agent. In [4,5,15], as single agent policy select both the control action and the measurement behaviour. Alternatively, in [9,11,20], separate policy are learned to determine the control actions and measurement actions. In addition, [4,9,15] also learn models for estimating the next state.

2.2 Works Related to DMSOA

The proposal of [20] is most algorithmically related to the DMSOA. In it, the author demonstrated the potential of dynamic action repetition for RL with observation costs using tabular q-learning. The agent learns to forego a sequence of one or more measurements in predictable regions of the state space by repeatedly applying the same action. Their proposed method is found to requires fewer measurement step to reach the goal than the MDP baseline. However, it is only suitable for discrete state and actions spaces, and was only evaluated on grid-world problems. In this work, we show how action repetition and measurement skipping can be implemented in deep RL for continuous and image-based observation spaces.

DMSOA is a method that aims to improve the efficiency of RL. To this end, it is weakly related to other techniques to improve the sampling efficiency [10,18,19]. The classic sample efficiency work, however, aims to reduce the overall number of training steps needed to learn a suitable policy, rather than reducing the measurement or decision steps made by the agent.

DMSOA utilizes concepts from the RL literature on dynamic frame skipping to repeatedly apply the selected control action [12]. In DMSOA, however, the agent's measurement skipping policy is shaped by the intrinsic reward. Moreover, unlike frame skipping applications, which are concerned with processing speed not measurement costs, DMSOA does not have access to privileged extrinsic control rewards from intermediate steps. This making the problem more challenging.

In addition, DMSOA has a connection to the options framework [24], and particularly dynamic options [1]. Similar to the options framework, at each decision point the DMSOA agent chooses to apply a sequence of actions that will transition the agent through multiple states. In DMSOA, the agent's policy selects a single control action and the number of times to apply the action in order to reduce its measurement costs while still achieving the control objective. Through the incorporation of measurement costs, the agent is able to learn how many times the action should be applied in order to arrive at the next meaningful state. For DMSOA, a meaningful state is one for which the information provided by it is greater than the cost to measure it.

3 Problem Setup

An AC-NOMDP is defined by $\langle \mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{P}, \mathcal{R}_{ext}, \mathcal{R}_{int}, \mathcal{P}_{s_0}, \gamma \rangle$ where S is the state space, $\mathcal{A} = \langle A_c \times A_m \rangle$ is the set of action tuples composed of control actions a_c and binary measurement actions $a_m \in \{0, 1\}$ that specify if an observation of the next state is requested. The observation space $\mathcal{O}$ is related to $\mathcal{S}$ by the observation emission function $p(o|s', a)$ (more on this below). $\mathcal{P} : S \times A \times S \to \mathbb{R}$ denotes the transition probabilities, $\mathcal{R}_{ext} : S \times A_c \to \mathbb{R}$ denotes the extrinsic reward function, $\mathcal{R}_{int} : S \times A_m \to \mathbb{R}$ denotes the intrinsic reward function that encourages the agent to reduce the number of measurements it makes. The r_{int} value is typically set to slightly outweigh the r_{ext} value to achieve the balance between the need for information to solve to control problem and the cost of information. If r_{int} is very large or very small relative to r_{ext}, the agent may never measure at the cost of solving the control objective or always measure and fail to reduce the observation costs.

The function $\mathcal{P}_{s_0} : \mathcal{S} \to \mathbb{R}$ denotes the probability distribution over the initial state and $\gamma \in (0, 1]$ is the discount factor. The observation emission function $p(o|s', a)$ specifies the probability of observing $o \in \mathcal{O}$ given the action a in state s. Unlike the more general POMDP, the observation space in a AC-NOMDP is limited to $\mathcal{O} = \mathcal{S} \cup \{empty\}$, where *empty* is the missing measurement of the environment. In this setup, the potential probabilities of $p(o|s', a_m = 1) \in \{0, 1\}$, with $p(o|s', a_m = 1) = 1$ if and only if $o = s'$ and $p(o|s', a_m = 1) = 0$ for all $o \neq s'$. In contrast, $p(o|s', a_m = 0) = 1$ if and only if $o = stale$.

The agent learns a policy $\pi(o) : O \to A$ that maps observations to action tuples. The initial observation $o_0 = s_0$ contains a fresh measurement of the environment. The control action selected by the agent is applied in the environment and the underlying state transitions according to $\mathcal{P}$. At each time step, the reward, r_t is the intrinsic reward, $r_t = r_{(int,t)}$, if $a_m = 0$, otherwise the extrinsic reward, $r_t = r_{(ext,t)}$, is given in response to the state s_t and control action a_c selected by the agent. When the measurement action, $a_m = 1$, is selected, the agent receives a fresh measurement of the underlying state $o_{t+1} = s_{t+1}$ of the environment. Alternatively, when $a_m = 0$, the agent does not obtain a fresh measurement and the next action must be selected based the agent's internal mechanism, such as an internal memory or model.

The OSMBOA agent selects one control and one measurement action, $\langle a_{c,t}, a_{m,t} \rangle$ per time step, whereas the DMSOA agent moves from decision point to decision point with a frequency less than or equal to the environment's clock. At each decision point, the DMSOA agent selects a control action and the number of times to apply it, k. The state is only measured on the k^{th} application (e.g. $\langle a_c, a_m = 0 \rangle_t, ..., \langle a_c, a_m = 0 \rangle_{t+(k-1)}, \langle a_c, a_m = 1 \rangle_{t+k}$).

The agent's objective is to learn a policy π that maximizes the discounted expected costed return which incorporates both the intrinsic and extrinsic rewards:

$$J(\pi) = \mathop{\mathbb{E}}_{a_t \sim \pi, s_t \sim P} \left[\sum_t \gamma^t r(s_t, a_t) \right], \tag{1}$$

where $\gamma < 1$ is the discount factor. In this work, we focus on deep Q-learning based solutions [14] combined with standard improvements such as *n-step* DQN for better convergence [23], double DQN to improve stability [25] and prioritized replay to improve sample efficiency [19]. Although this work examines problems with discrete action spaces, the proposed algorithms can be modified for continuous action spaces.

4 Deep Dynamic Multi-step Observationless Agent

The Deep Dynamic Multi-Step Observationless Q-learning Agent (DMSOA) for noiselessly observable RL environments with explicit observation costs is presented in Fig. 1. The framework has three key components: the control policy $\pi_c : o \rightarrow a_a$ that maps the observation to a control action, the measurement skipping policy $\pi_m : o, a_c \rightarrow k$ that maps the observation and selected control action to $k \in \{1, ..., K\}$ the number of steps to apply a_c to the environment, and the action-observation scheduler. The action-observation scheduler applies the action pair $(a_c, 0)$ $k-1$ times and collects the intrinsic rewards r_{int} from the environment. On the k^{th} iteration, it applies the action pair $(a_c, 1)$, records the extrinsic reward r_{ext}, and passes the new observation to π_c and π_m. The extrinsic reward is equal to the control policy reward for applying a_c and arriving in the measured state after step $i = k$. The intrinsic reward is $r_{int} \in \{0, c\}$, where c is a bonus (i.e. "cost saving") given to the agent when it chooses not to measure. To ensure the agent is motivated to omit measurements whenever possible, we set $c \geq r_{ext}^{max}$. The optimal setting of c will depend on the application and the requirements of the domain.

In this work, the policies are implemented as deep Q networks (DQN), however, other forms of policy learning could be utilized. The agent's objective is to maximize the costed rewards $\sum_{t=0}^{\infty} \gamma^t r_t$. To achieve this we learn parameterized value functions $Q_c(o; \theta)$ and $Q_m(o, a; \zeta)$ as feed-forward deep neural networks. As described above, for an m-dimensional observation space and an n-dimensional action space, Q_c is a mapping from an m-dimensional observation to an n-dimensional vector of action values. The function Q_m is a mapping from an $m+1$-dimensional observation-action to a K-dimensional vector of measurement values. In the case of image data, each channel is augmented with the action details. The argmax of each output indicates the action to apply and the number of times to apply it.

During training, the experience tuples $(o_t, a_{(c,t)}, a_{(m,t)}, r_t, o_{t+1})$ are stored in a prioritized experience replay buffer. To improve stability, target networks θ^- and ζ^- for Q_c and Q_m are copied from θ and ζ every τ steps. In addition, we use the double DQN [25] to improve value estimates. The target for the control network is:

$$Y_i^{Q_c} \equiv r_t + \gamma \, Q_c\big(o_{t+1}, \text{argmax}_a Q_c(o_{t+1}, a; \theta_t); \theta_t^-\big). \tag{2}$$

For the same update step, the target for the measurement network is:

$$Y_i^{Q_m} \equiv r_t + \gamma \, Q_m\big((o_{t+1}, a_{(c,t+1)}), \text{argmax}_a Q_m((o_{t+1}, a_{(c,t+1)}), a; \zeta_t); \zeta_t^-\big). \tag{3}$$

The corresponding losses are:

$$\mathcal{L}_i^{Q_c}(\theta_i) = \mathbb{E}_{(o_t, a_{(c,t)}) \sim \mathcal{D}}\left[(Y_i^{Q_c} - Q_c(o_t, a_{(c,t)}; \theta_i)^2\right], \tag{4}$$

and

$$\mathcal{L}_i^{Q_m}(\zeta_i) = \mathbb{E}_{(o_t, a_{(c,t)}, a_{(m,t)}) \sim \mathcal{D}}\left[(Y_i^{Q_m} - Q_m((o_t, a_{(c,t)}), a_{(m,t)}; \zeta_i)^2\right] \tag{5}$$

5 Experimental Setup

In this section, we compare the performance of DMSOA to OSMBOA. In order to highlight the differences in the measurement behaviour of each method, we implement both with double DQN and a prioritized replay buffer. The hyperparameters were selected via grid search with 3 random trials. For the evaluation, we report the mean and standard deviation of the reward during training and the observation behaviour of the best policy. Each agent is reinitialized with 20 difference seeds and trained on the OpenAI gym environments Cartpole, Acrobot, Lunar Lander and Atari Pong. The experiments were run on CentOS with Intel Xeon Gold 6130 CPU and 192 GB memory. In addition, a NVIDIA V100 GPU was used in the training of the Atari agent. Clips of the DMSOA agents can be viewed here[1].

6 Results

Figure 2 shows the mean and standard deviation for each agent on the Cartpole and Acrobot environments. The aim in the Cartpole environment is for the agent to operate a cart such that a vertical pole remains balanced for as long as possible. The extrinsic reward is set to 1 and the intrinsic reward is set to 1.1. We truncate each episode at a maximum of 200 time steps. In the Acrobot environment, the objective is to apply torque to flip an arm consisting of two actuated links connected linearly above a target height in as few steps as possible. The agent receives an extrinsic reward of -1 or an intrinsic reward of -0.85 at each time step. The episode ends after 200 steps or when the arm is successfully flipped over the line.

DMSOA learns a policy for both environments that produces a higher costed reward than OSMBOA. This indicates that DMSOA requires fewer measurements whilst carrying out the control policy. In addition, the standard deviation is lower indicating more stability across independent training runs. The episode length plots on the right show that DMSOA learned policies to keep the Cartpole upright longer and flip the Acrobot over the goal faster.

The results for the Lunar Lander environment are presented in Fig. 3. The Lunar Lander environment is a rocket trajectory optimization problem [7]. The

[1] https://www.youtube.com/playlist?list=PLr6sWY5moZhFtTuCBbIjb4cZQOZkbjkOV.

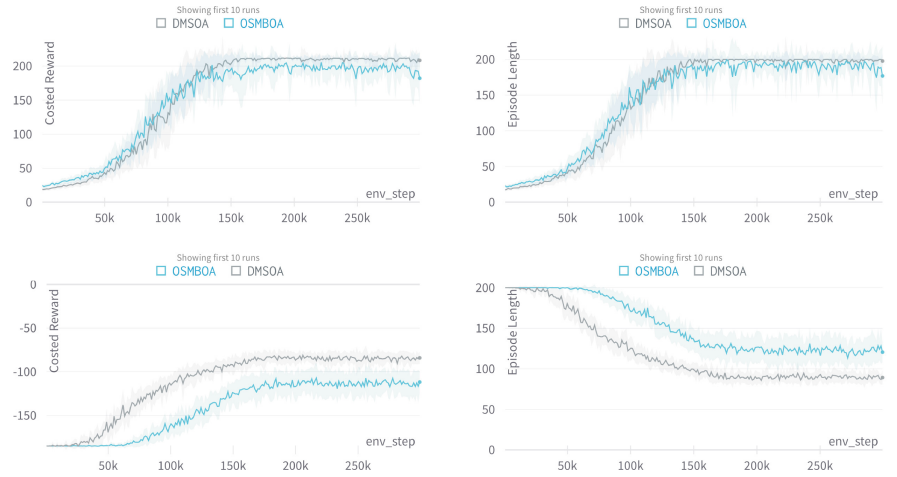

Fig. 2. Mean and standard deviation of performance on Cartpole (upper) and Acrobot (lower). Left: costed reward and right: episode length. In both cases, DMSOA has a higher mean costed reward, and is superior in terms of the control objective (longer episodes on Cartpole and shorter episodes on Acrobot.)

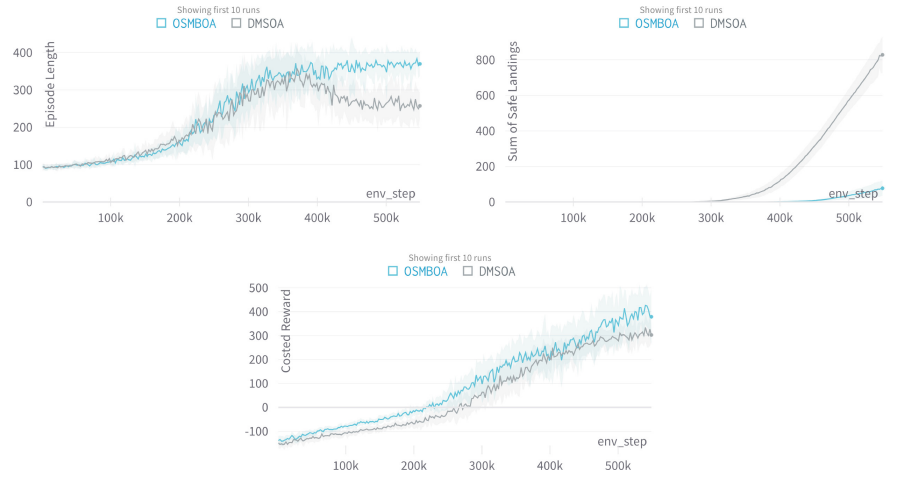

Fig. 3. Mean and standard deviation of the performance on the Lunar Lander environment. Top left: episode length, top right: sum of the number of successful landings, lower: costed reward. DMSOA and OSMBOA achieve similar mean costed rewards. DMSOA, however, learns to successfully land the ship more frequently and in fewer steps.

objective is to fire the lander's rockets such that it lands squarely in the target area. The fuel supply is infinite, but the best policy uses it sparingly. The environment has four discrete actions available: do nothing, fire left orientation

engine, fire main engine, fire right orientation engine. The intrinsic reward is 0.1. The extrinsic reward is -0.3 for firing the main engine and -0.03 for side engines, the reward is also scaled by the lander's distance from the landing pad. Ten points are added to the extrinsic reward for each leg that is in contact with the ground, and an additional 100 points are added for landing, while 100 points are subtracted for crashing. The episode ends when the agent lands or crashes, or is truncated after a maximum of 400 time steps.

The plot on the top left in Fig. 3 shows that mean episode length is longer for OSMBOA than DMSOA, and the top right plot shows that DMSOA has significantly more successful landings. This indicates that DMSOA learns a policy that quickly navigates the ship to a safe landing. The lower plot shows that OSMBOA has a slightly higher costed reward. As suggested by the first two plots, this is due to the fact that it takes longer to land and not because the policies is superior.

Table 1. Ratio of steps with measurements to steps without measurements of the converged policy during training.

Env.	DMSOA	OSMBOA
Cartpole	1:1.27	1:0.37
Acrobot	1:0.45	1:1.03
Lunar Lander	1:1.56	1:0.33

Table 1 shows the ratio of the number of steps made without measuring for each measurement made. On Cartpole and Lunar Lander, DMSOA makes more than one step without measuring for each measurement step, whereas on Acrobot it makes an average of 0.5 non-measuring steps for each measuring step. This suggests that the dynamics of Acrobot are less predictable, causing DMSOA to measure more frequently. Interestingly, Acrobot is the only environment where OSMBOA does better than a 1:1 ratio.

6.1 Examination of Measurement Policies

Figure 4 shows the measurement behaviour of the best OSMBOA (left) and DMSOA (right) policies for Cartpole (top), Acrobot (middle) and Lunar Lander (lower) environments. Each row specifies a 1-episode roll-out of the best policy. Each column in the OSMBOA plots is the environment time step during the episode. For OSMBOA, the number of decision steps is equivalent to the number of steps in the environment. In contrast, each column in the DMSOA plots corresponds to a decision by the agent, with one or more environment time steps associated with it. In addition to highlighting the measurement efficiency, this also shows the decision efficiency. On Cartpole, DMSOA makes approximately 70 action selections (decisions) per episode of 200 environment steps (the mean steps per episode are shown in Fig. 2.

For OSMBOA, an orange cell indicates that a fresh measurement of the environment and blue specifies that no measurement was requested at corresponding time step. In the case of DMSOA, the colour indicates the number of consecutive steps that were taken without a fresh measurement. Blue indicates that a measurement is made after the control action is applied once, yellow indicates that a measurement is made after the control action is applied twice and red

indicates that a measurement is made after the control action is applied three times.

The distinct pattern in each plot suggests the different capabilities of each class of AC-NOMDP agent, along with the fact that each environment is unique in terms of its dynamics and complexity. The consistent measurement patterns for OSMBOA and DMSOA on Cartpole suggest that the environment has very regular dynamics. OSMBOA switches between selecting the next action from a freshly measured observation and selecting it from a stale observation. Alternatively, DMSOA learns to apply an action 3 times before measuring. This clearly demonstrates the potential of DMSOA to take more environment steps without measuring than OSMBOA.

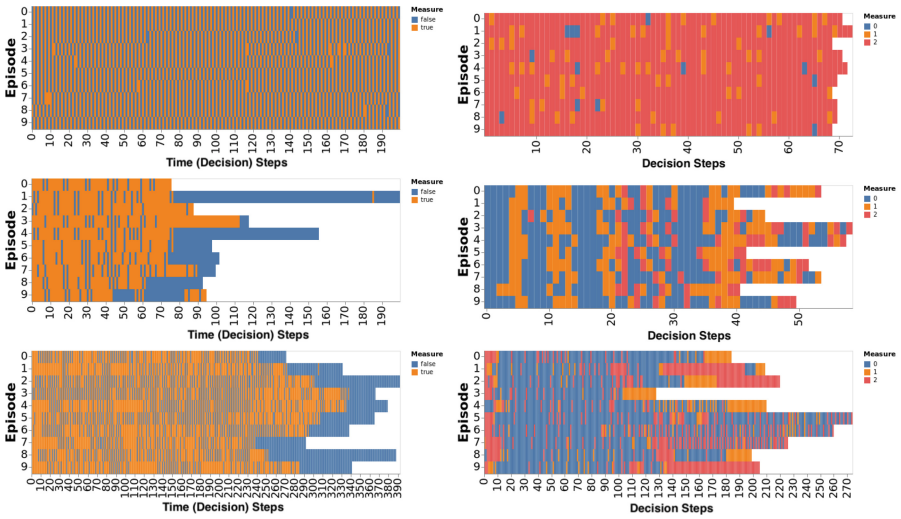

Fig. 4. Measurement behaviour of the best OSMBOA (left) and DMSOA (right) policies. Top: Cartpole, centre: Acrobot, lower: Lunar Lander. For OSMBOA, blue indicates that no measurement was made and orange indicates that a measurement was made. In the case of DMSOA, blue indicates that a measurement is made after one step, orange indicates that a measurement is made after two steps, and red indicates that a measurement is made after three steps. This figure reveals the very distinct measurement behaviour between the two classes of AC-NOMDP agents. (Color figure online)

Acrobot and Lunar Lander show much more complex measurement behaviour. For both OSMBOA and DMSOA on Acrobot during approximately the first 3/4 s of the each episode they display a pattern of frequently measuring followed by briefly not measuring. Our analysis finds that both OSMBOA and DMSOA skip measurements while the arcobot is in the lower left region of the observation space. This is roughly where the momentum of the Acrobot shifts from heading way from the goal, back towards to the goal. In this area, it is deemed safe to apply torque back towards the goal without observing. In the

last quarter of each episode, both agents take more steps without measuring. It is noteworthy that in most episodes OSMBOA takes significantly more steps without observing than DMSOA. However, this has a negative impact on the total number of action decisions made by OSMBOA on route to achieving to goal. This particularly visible in episodes 1 and 4. DMSOA does not to suffer from similar behaviour.

On the Lunar Lander environment both methods take few or no measurements near the end of the episode when the agent is close to landing. In addition, DMSOA repeatedly takes 2-3 steps before measuring at the beginning of each episode, whereas OSMBOA repeatedly measures early in each episode. Each method measures frequently during the middle of the episode as agent attempts to direct the lander safely towards the landing area. OSMBOA generally alternates between measuring and not measuring at each time step, whereas DMSOA typically takes many measurement steps followed by 1 to 2 steps without measuring before returning to measuring again. Similar to Acrobot, once OSMBOA estimates that it is on target to reach the goal it commits to never measuring again. When this estimate is erroneous, the leads to much longer episodes than necessary and the risk of crashing the ship.

6.2 Image-Based RL Results

The objective in the pong Atari game is to bounce the ball off of your paddle and past the opponents paddle into its goal [3]. The action space is 6-dimensional including do nothing, fire, move right, move left, fire right and fire left. The observation space is a (210, 160, 3) image. In the case of OSMBOA, a 210 by 1 vector of ones or zeros is added to each channel to indicate if the observation is fresh or stale. The agent gets an extrinsic reward of 1 for winning a match and 0 for each intermediate step. Each episode is composed of 21 matches and the intrinsic reward is 0.001.

The results in Fig. 5 show that DMSOA wins significantly more matches than OSMBOA (top left), achieves a higher costed reward (top right) and more intrinsic reward (lower). Thus, DMSOA learns to be a better Pong player and requires fewer measurements. Due the longer episodes and training times, a measurement behaviour plot similar to Fig. 4 is not feasible within the confines of this paper. However, recordings of each agent and its measurement behaviour are available in the paper Github repository.

From our analysis for the measurement policies of each agent, we found that both learn to measure less frequently when the ball is travelling away from their paddle. Alternatively, if the ball is near their paddle or the opponents paddle, each agent measures more frequently. Inline with the observations on Acrobot and Lunar Lander, when OSMBOA reaches a state from which it expects to win the match, it switch to not measuring for the remainder of the match. If the prediction is correct, it can achieve a greater reduction in measurements than DMSOA. If it is wrong, however, OSMBOA general loses the match. An erroneous prediction of this nature is particularly risky in a complex and dynamic environment.

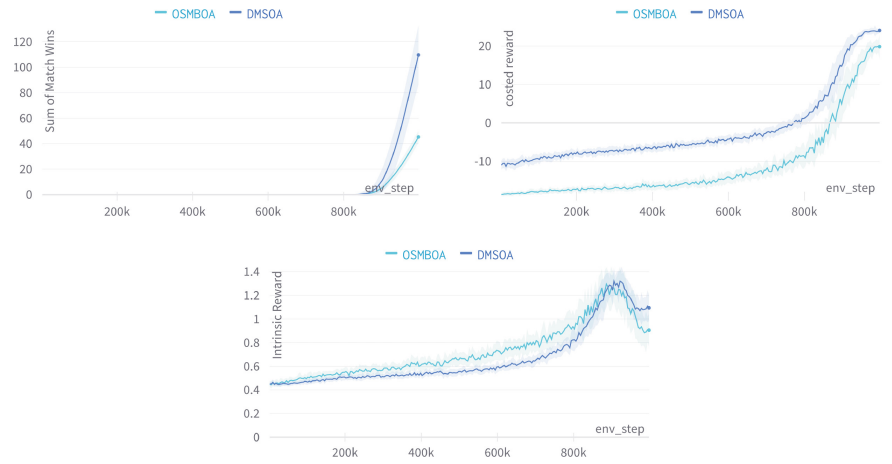

Fig. 5. Mean and standard deviation of the performance on the Atari Pong environment. Top left: sum of the number of wins, top right: costed reward, lower: intrinsic reward. DMSOA learns a policy that wins more games with a better costed reward and fewer measurements.

7 Discussion

The results indicate that DMSOA has a clear advantage over OSMBOA in terms of its convergence rate and the reduction in measurements and decision steps. We believe that the control action repetition capabilities of DMSOA improve its exploration of the environment and its understanding of the implications (positive and negative) of taking multiple steps without measuring. This helps it to quickly converge to good control and measurement policies. In addition, the fact that DMSOA's multi-step action sequences always ends with a measurement of the final state provides it with a good grounding from with to select the next control action. On the other hand, because the extrinsic reward for intermediate steps is not available, there is the potential for more noise in the reward signal for longer DMSOA action repetition trajectories. Due to the fact that OSMBOA is limited to one-step action, noise in the reward is less of a concern. Although, DMSOA appears to handle the noisy reward signal, future work should examine this in more detail.

For unshaped (or uniform) reward environments, such as Cartpole, Acrobot and Pong, setting the intrinsic reward is simple and the agent is insensitive to the value so long as it is slightly larger than the extrinsic reward. Alternative, the intrinsic reward requires fine tuning on environments with complex reward shaping such as Lunar Lander. As heuristic, we suggest starting the fine tuning from the mean of the extrinsic reward collected over multiple random walks in the environment.

In multiple environments, we found that OSMBOA commits to not measuring towards the end of each episode. This is surprising since if OSMBOA takes more

than one step without measuring, it enters a partially observable state. This is akin to playing the game with its eyes closed. The agent is, thus, unaware if any unexpected event occurs. On Acrobot and pong, this resulted in it not achieving the goal, or it taking much longer than otherwise necessary. An example of this is seen in episodes 1 and 4 of the Acrobot plot in Fig. 4.

8 Environmental Impact

This work aims to strike a balance between scientific understanding and energy consumption. To do this we have selected a number of OpenAI gym classic control environments from which RL policies can be efficiently learned, along with one large image-based RL environment. Although the set is relatively small the dynamics are diverse enough to illustrate the differences between the two classes of AC-NOMDP algorithms considered in this work. We also note that the work was conducted in a jurisdiction in which the majority of the electric comes from sources such as hydro-electric and nuclear.

In addition to reducing the measurement costs, this work can lead to a reduction in the associated carbon footprint for RL in this area. Unlike OSMBOA, the multi-step capabilities of DMSOA may robustly lower the number of forward passes through the network for decision making, offer savings in terms of communication with the environment and lower latency. Moreover, it can help with exploration, thereby reduce the time to policy convergence.

9 Conclusion

In this work, we consider the problem of RL for environments where agent's decision making and measuring of the state of the environment have explicit costs, namely AC-NOMDPs. We provide the first survey of methods recently proposed for AC-NOMDPs. Building on the existing work, we propose DMSOA, an RL algorithm learns a control and a measurement policy to reduce measurement and decision steps. Our empirical results confirm the previously published results for OSMBOA on Cartpole, Acrobot and Lunar Lander, and show that OSMBOA is also capable on the more complex, image-based Atari Pong environment. However, we find that our proposed method DMSOA learns a *better control policy* than OSMBOA, and *requires fewer costly measurement and decision steps.*

This demonstrates the great potential to reduce measurement and decision costs associated with RL by allowing the agent to take control of its action and observation behaviour. We expect this to be a necessary capability of RL agents applied in many real-world applications. The next steps that we envision are developing more sophisticated loss functions for DMSOA, incorporating recurrency into the network to deal with time, expanding the analysis to additional methods and more realistic setting such as self-driving chemistry where observation can be costly and potentially destructive [2].

Acknowledgments. This work was supported with funding from the National Research Council of Canada's AI for Design Program.

References

1. Barto, A.G., Mahadevan, S.: Recent advances in hierarchical reinforcement learning. Discr. Event Dyn. Syst. **13**(1–2), 41–77 (2003)
2. Beeler, C., et al.: Chemgymrl: An interactive framework for reinforcement learning for digital chemistry. arXiv preprint arXiv:2305.14177 (2023)
3. Bellemare, M.G., Naddaf, Y., Veness, J., Bowling, M.: The arcade learning environment: an evaluation platform for general agents. J. Artif. Intell. Res. **47**, 253–279 (2013)
4. Bellinger, C., Coles, R., Crowley, M., Tamblyn, I.: Active measure reinforcement learning for observation cost minimization. In: Proceedings of the Canadian Conference on Artificial Intelligence. Canadian Artificial Intelligence Association (CAIAC) (Jun 8 2021). https://caiac.pubpub.org/pub/3hn8s5v9
5. Bellinger, C., Drozdyuk, A., Crowley, M., Tamblyn, I.: Balancing information with observation costs in deep reinforcement learning. In: Proceedings of the Canadian Conference on Artificial Intelligence. Canadian Artificial Intelligence Association (CAIAC) (may 27 2022). https://caiac.pubpub.org/pub/0jmy7gpd
6. Bellinger, C., Drozdyuk, A., Crowley, M., Tamblyn, I.: Scientific discovery and the cost of measurement – balancing information and cost in reinforcement learning. In: ICML 2nd Annual AAAI Workshop on AI to Accelerate Science and Engineering (AI2ASE) (Feb 13 2023)
7. Brockman, G., et al.: Openai gym. arXiv preprint arXiv:1606.01540 (2016)
8. Daniel, K.: Thinking fast and slow. United States of America (2011)
9. Fu, Y.: The Cost of OPS in Reinforcement Learning. Master's thesis, University of California, Berkeley (2021)
10. Gal, Y., McAllister, R., Rasmussen, C.E.: Improving pilco with bayesian neural network dynamics models. In: Data-Efficient Machine Learning workshop, ICML. vol. 4, p. 34 (2016)
11. Koseoglu, M., Özcelikkale, A.: How to miss data?: Reinforcement learning for environments with high observation cost. In: 2020 International Conference on Machine Learning (ICML) Workshop, Wien, Österrike, 12-18 juli (2020)
12. Lakshminarayanan, A., Sharma, S., Ravindran, B.: Dynamic action repetition for deep reinforcement learning. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 31 (2017)
13. Mills, K., Ronagh, P., Tamblyn, I.: Finding the ground state of spin Hamiltonians with reinforcement learning. Nature Mach. Intell. **2**(9), 509–517 (2020). https://doi.org/10.1038/s42256-020-0226-x
14. Mnih, V., et al.: Human-level control through deep reinforcement learning. Nature **518**(7540), 529–533 (2015)
15. Nam, H.A., Fleming, S., Brunskill, E.: Reinforcement learning with state observation costs in action-contingent noiselessly observable markov decision processes. Adv. Neural. Inf. Process. Syst. **34**, 15650–15666 (2021)
16. Ong, S.C., Png, S.W., Hsu, D., Lee, W.S.: Planning under uncertainty for robotic tasks with mixed observability. Int. J. Robot. Res. **29**(8), 1053–1068 (2010)
17. Papadimitriou, C.H., Tsitsiklis, J.N.: The complexity of Markov decision processes. Math. Oper. Res. **12**(3), 441–450 (1987)
18. Pathak, D., Agrawal, P., Efros, A.A., Darrell, T.: Curiosity-driven exploration by self-supervised prediction. In: International Conference on Machine Learning, pp. 2778–2787. PMLR (2017)

19. Schaul, T., Quan, J., Antonoglou, I., Silver, D.: Prioritized experience replay. arXiv preprint arXiv:1511.05952 (2015)
20. Shann, T.Y.A.: Reinforcement learning in the presence of sensing costs. Master's thesis, University of British Columbia (2022). https://doi.org/10.14288/1.0413129, https://open.library.ubc.ca/collections/ubctheses/24/items/1.0413129
21. Sharma, S., Srinivas, A., Ravindran, B.: Learning to repeat: Fine grained action repetition for deep reinforcement learning. arXiv preprint arXiv:1702.06054 (2017)
22. Simon, H.A.: Bounded rationality. Utility and probability, pp. 15–18 (1990)
23. Sutton, R.S., Barto, A.G.: Reinforcement learning: An introduction. MIT press (2018)
24. Sutton, R.S., Precup, D., Singh, S.: Between mdps and semi-mdps: a framework for temporal abstraction in reinforcement learning. Artif. Intell. **112**(1–2), 181–211 (1999)
25. Van Hasselt, H., Guez, A., Silver, D.: Deep reinforcement learning with double q-learning. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 30 (2016)