Monitored Markov Decision Processes

Simone Parisi
University of Alberta; Amii
Edmonton, Canada
parisi@ualberta.ca

Montaser Mohammedalamen
University of Alberta
Edmonton, Canada
mohmmeda@ualberta.ca

Alireza Kazemipour
University of Alberta
Edmonton, Canada
kazemipo@ualberta.ca

Matthew E. Taylor
University of Alberta; Amii
Edmonton, Canada
matthew.e.taylor@ualberta.ca

Michael Bowling
University of Alberta; Amii
Edmonton, Canada
mbowling@ualberta.ca

ABSTRACT

In reinforcement learning (RL), an agent learns to perform a task by interacting with an environment and receiving feedback (a numerical reward) for its actions. However, the assumption that rewards are always observable is often not applicable in real-world problems. For example, the agent may need to ask a human to supervise its actions or activate a monitoring system to receive feedback. There may even be a period of time before rewards become observable, or a period of time after which rewards are no longer given. In other words, there are cases where the environment generates rewards in response to the agent's actions but the agent cannot observe them. In this paper, we formalize a novel but general RL framework — *Monitored MDPs* — where the agent cannot always observe rewards. We discuss the theoretical and practical consequences of this setting, show challenges raised even in toy environments, and propose algorithms to begin to tackle this novel setting. This paper introduces a powerful new formalism that encompasses both new and existing problems and lays the foundation for future research.

KEYWORDS

Reinforcement learning, reward observability, active learning

ACM Reference Format:
Simone Parisi, Montaser Mohammedalamen, Alireza Kazemipour, Matthew E. Taylor, and Michael Bowling. 2024. Monitored Markov Decision Processes. In *Proc. of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2024), Auckland, New Zealand, May 6 – 10, 2024*, IFAAMAS, 9 pages.

1 INTRODUCTION

Reinforcement learning (RL) has developed into a powerful setting where agents can tackle a variety of tasks, including games [36], robotics [15], medical applications [45], and user engagement [11]. Autonomous agents trained with RL learn by trial and error: they are deployed in an environment, try different actions, and receive a numerical reward depending on the outcome of their actions. More interactions lead to more data as the agent tries to maximize its rewards. Traditionally, RL frames the environment-agent interaction as a Markov Decision Process (MDP), where rewards are assumed to be observable after every action. This is in stark contrast with many real-world situations, where the agent may need additional instrumentation (e.g., cameras or specialized sensors) or a human expert to observe the reward [46]. If the instrumentation breaks or the expert is unavailable, the agent does not observe any reward for its actions, even though the efficacy of its behavior is still important. Or even if the agent observes rewards, these could be imperfect due to human mistakes or faulty instrumentation [19]. *In other words, there will be situations where the agent cannot observe the exact rewards generated by the environment to judge its actions.* This paper argues that such circumstances should be part of the problem specification, suggesting an extension to MDPs is needed, as agents that ignore these complications can result in real-world failures.

Consider the situation shown in Figure 1. Here, an autonomous agent is tasked with household chores, and the quality of its behavior is provided through feedback from the homeowner and smart sensors. However, this reward feedback is not always observable, as the owner may not be present, the sensors may not have full coverage of the home, or even be malfunctioning. In such situations, the agent should not interpret the lack of reward as meaning that all behavior is equally desirable. Neither should it think that avoiding monitoring or intentionally damaging sensors is an effective way to avoid negative feedback. Further, the agent may need to reason about how to seek the most useful feedback, such as planning exploratory actions when the owner is home or in well-monitored rooms. Ideally, such an agent will eventually learn to judge its own actions without the need for human feedback or home sensors.

We argue that to autonomously learn to complete tasks in such real-world settings, RL agents need a comprehensive framework where 1) the agent cannot always observe rewards even though its behaviour should still seek to maximize the unobserved reward; 2) the agent may need to explicitly act to observe rewards, yet whether it observes the reward or not may not be fully under its control; 3) the process that determines the observation of rewards, namely *the monitor*, is itself something that can be learned or modelled by the agent; and 4) the reward provided to the agent may be imperfect. Most importantly, even if the agent does not receive explicit rewards through the monitor, its actions are still impactful: *the environment always generates rewards in response to the agent's actions, even when the agent cannot observe them because the monitor — that communicates rewards to the agent — could be unavailable or faulty.*

To the best of our knowledge, there is no framework that fully formalizes this problem setting in the RL literature. *Active RL* tackles

Proc. of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2024), N. Alechina, V. Dignum, M. Dastani, J.S. Sichman (eds.), May 6 – 10, 2024, Auckland, New Zealand. © 2024 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org).

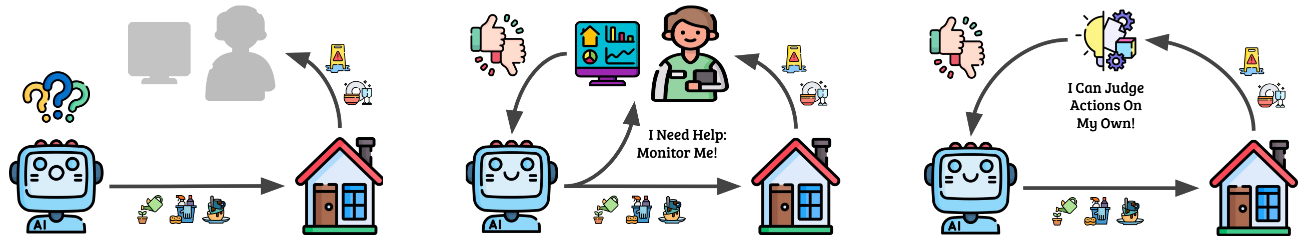


Figure 1: Example of Monitored MDP. The agent is tasked with household chores but needs the owner or home sensors to observe rewards. If the owner is not home or the sensors are unavailable *(left)*, the agent will not receive positive rewards for cleaning dishes or negative rewards for spilling water. Thus, the agent must learn how to seek monitoring — where to move for sensory feedback or when the owner is home *(center)* — and act appropriately even when monitoring is unavailable (e.g., be cautious when not being monitored). Eventually, the agent can judge actions on its own without any monitoring *(right)*.

similar problems [16, 37], but it is limited to cases where there are explicit binary actions that deterministically control when reward is observed. *Partial monitoring* addresses the problem of learning with limited feedback, but only in bandits [2, 3, 18, 20]. In *sparse-reward RL*, the agent receives a zero-reward after each action until eventually receiving a more informative reward [17], but rewards are always observable. In *cautious RL*, rewards may not be observable [26] but the agent has no means to control their observability.

This paper formalizes *Monitored Markov Decision Processes (Mon-MDPs)*, a novel RL framework that accounts for unobservable rewards by introducing the *monitor*, a separate MDP that dictates when and how the agent sees the rewards, and that the agent can affect with dedicated actions. We discuss the theoretical and practical consequences of unobservable rewards, present toy environments, and provide algorithms to illustrate the resulting challenges. We believe that Mon-MDPs allow to formalize the complexity of many real-world tasks, provide a unifying view of existing areas of research, and lay the foundation for new research directions.

2 PROBLEM FORMULATION

A Markov Decision Process (MDP) is a mathematical framework for sequential decision-making, defined by the tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P}, \gamma \rangle$. An agent interacts with an environment by repeatedly observing a state $s_t \in \mathcal{S}$, taking an action $a_t \in \mathcal{A}$, and observing a bounded reward $r_t \in \mathbb{R}$. The state dynamics are governed by the Markovian transition function $\mathcal{P}(s_{t+1} \mid a_t, s_t)$, while the reward is determined by the reward function $r_t \sim \mathcal{R}(s_t, a_t)$. Both functions are unknown to the agent, whose goal is to act to maximize the sum of discounted rewards $\sum_{t=1}^{\infty} \gamma^{t-1} r_t$, where $\gamma \in [0, 1)$ is the discount factor that describes the trade-off between immediate and future rewards.[1]

2.1 Monitored MDPs

In Monitored MDPs (Mon-MDPs) the observability of the reward is governed by the *monitor*, a separate Markovian decision process. Formally, a Mon-MDP is defined by the tuple

$$\langle \mathcal{S}^{\text{E}}, \mathcal{A}^{\text{E}}, \mathcal{P}^{\text{E}}, \mathcal{R}^{\text{E}}, \mathcal{M}, \mathcal{S}^{\text{M}}, \mathcal{A}^{\text{M}}, \mathcal{P}^{\text{M}}, \mathcal{R}^{\text{M}}, \gamma \rangle.$$

The tuple $\langle \mathcal{S}^{\text{E}}, \mathcal{A}^{\text{E}}, \mathcal{P}^{\text{E}}, \mathcal{R}^{\text{E}}, \gamma \rangle$ is the same as classic MDPs where the superscript E stands for "environment." However, the *environment reward* $r^{\text{E}}_t \sim \mathcal{R}^{\text{E}}(s^{\text{E}}_t, a^{\text{E}}_t)$ is not directly observable. Instead, the agent observes a *proxy reward* $\hat{r}^{\text{E}}_t \sim \mathcal{M}(r^{\text{E}}_t, s^{\text{M}}_t, a^{\text{M}}_t)$, where $\mathcal{M}$ is the *monitor function*, $s^{\text{M}} \in \mathcal{S}^{\text{M}}$ is the monitor state, and $a^{\text{M}} \in \mathcal{A}^{\text{M}}$ is the monitor action. Even so, the monitor function is not guaranteed to always show a reward and the agent may receive $\hat{r}^{\text{E}}_t = \bot$, i.e., "*unobservable reward*" (i.e., the monitor dictates what the agent sees about the environment reward according to its current state and action). The monitor state follows the Markovian transition function $\mathcal{P}^{\text{M}}(s^{\text{M}}_{t+1} \mid s^{\text{M}}_t, s^{\text{E}}_t, a^{\text{M}}_t, a^{\text{E}}_t)$,[2] and executing monitor actions yields a bounded *monitor reward* $r^{\text{M}}_t \sim \mathcal{R}^{\text{M}}(s^{\text{M}}_t, a^{\text{M}}_t)$.[3] We will use *monitor* to refer to the tuple $\langle \mathcal{M}, \mathcal{S}^{\text{M}}, \mathcal{A}^{\text{M}}, \mathcal{P}^{\text{M}}, \mathcal{R}^{\text{M}} \rangle$, and *monitor function* to refer to $\mathcal{M}$. The monitor together with the environment tuple $\langle \mathcal{S}^{\text{E}}, \mathcal{A}^{\text{E}}, \mathcal{P}^{\text{E}}, \mathcal{R}^{\text{E}}, \gamma \rangle$ is the *Mon-MDP*. Figure 2 shows a diagram of the Mon-MDP in contrast to the standard MDP framework.

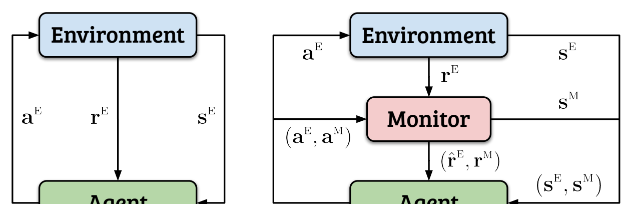


Figure 2: In classic MDPs (left), the agent directly observes environment rewards $r^{\mathbf{E}}$. In Mon-MDPs (right), instead, it receives proxy rewards $\hat{r}^{\mathbf{E}}$ via the *monitor*. Like the environment, the monitor is governed by a Markovian transition function and has its own rewards $r^{\mathbf{M}}$. At every step, the agent observes the environment and the monitor states, and executes actions affecting both. The goal is to maximize the cumulative sum of rewards $(r^{\mathbf{E}} + r^{\mathbf{M}})$, while observing $\hat{r}^{\mathbf{E}}$ instead of $r^{\mathbf{E}}$. If the reward is unobservable, the agent receives $\hat{r}^{\mathbf{E}} = \bot$.

[1] The constraint $\gamma < 1$ ensures the infinite sum is well-defined. Alternatively, one could allow $\gamma = 1$ but with restrictions on the MDP, e.g., absorbing states.

[2] The monitor states and actions may be as simple as "reward is (not) observable" and "do (not) ask for reward", or more heterogeneous. For example, the agent could ask for rewards by pushing buttons, collecting and using a device, or uncovering objects. Similarly, the monitor state may include the position of items, the battery of a device, or the distance of the agent from environment sensors. The next monitor state s^{M}_{t+1}, indeed, depends on both the current monitor pair $(s^{\text{M}}_t, a^{\text{M}}_t)$ and environment pair $(s^{\text{E}}_t, a^{\text{E}}_t)$. Consequently, the proxy reward also depends on both the environment and the monitor. In Figure 1, to receive rewards from the home sensors, the sensor has to be active (monitor state) and the agent near enough (environment state).

[3] The monitor reward can represent a cost (e.g., if monitoring consumes resources), but it is not constrained to be negative. Just like the environment reward, its design depends on the agent's desired behavior.

In Mon-MDPs, the agent repeatedly executes a joint action $(a^{\text{E}}_t, a^{\text{M}}_t)$ according to the joint state $(s^{\text{E}}_t, s^{\text{M}}_t)$. In turn, the environment and monitor states change and produce a joint reward $(r^{\text{E}}_t, r^{\text{M}}_t)$, but the agent observes $(\hat{r}^{\text{E}}_t, r^{\text{M}}_t)$. The agent's goal is to select joint actions to maximize $\sum_{t=1}^{\infty} \gamma^{t-1} \left(r^{\text{E}}_t + r^{\text{M}}_t\right)$ *even though it only observes* $\hat{r}^{\text{E}}_t$ instead of r^{E}_t. As shown in the remainder of the paper, not observing directly r^{E}_t (and possibly not observing any reward if $\hat{r}^{\text{E}}_t = \bot$) makes the optimization non-trivial, creating challenging — *and potentially impossible* — problems, which we discuss further in Section 3.

2.2 Why RL Needs Mon-MDPs

As in MDPs, the agent is always being judged, i.e., the environment always generates rewards in response to the agent's actions. In Mon-MDPs, however, the agent does not observe these rewards directly, but instead observes the proxy rewards given by the monitor. Most importantly, the monitor does not affect the environment reward — how the agent is judged — but only what the agent observes. In Figure 1, if the agent spills water on the floor, there is a clear "bad" feedback that the owner (i.e., the monitor) would give to the agent. However, if the owner is not home, the agent would not receive it. Nonetheless, the action that spilled water is still undesirable.

An alternative to the Mon-MDP framing of such a situation may be to formalize it as an MDP with delayed reward: the act of spilling water when the owner is not home causes no immediate feedback, but later when the owner returns there is negative feedback for the floor being wet. Such a framing either dictates a non-Markovian reward function or places a significant burden on the state representation to capture long sequences of actions, as well as stressing the agent's ability to do credit assignment. Further, suppose that the water dries before the owner returns, or that the agent purposefully only spills water where the owner will not notice. The agent may never receive any negative feedback for this behavior — a behavior that is undesirable. *The owner's lack of feedback should not be interpreted by the agent as indifference* to a behavior that would otherwise result in negative feedback when monitored. Instead, *the agent should understand that spilling water (an environment action) is undesirable regardless of the owner's presence (the monitor state).*

Another alternative to the Mon-MDP framing may be to avoid explicitly representing the monitor with its own states, actions, and rewards, as partially decoupled from the environment. However, this separation — where the monitor and its state do not affect the environment rewards — is the basis for the agent to know that hiding spilled water from the owner does not avoid (unobserved) negative rewards. An alternative that forces the environment and monitor state together into a single state variable (and similarly for environment and monitor actions) would only be able to achieve the same effect through carefully constructed generalization bias in the agent's representation, which is its own challenges.

There are further advantages for separating out the monitor process. First, whether the details of the monitor are known in advance by the agent is a design decision, and keeping them separate allows clarity to what is known and what must be learned through interaction. Second, explicitly reasoning about the monitor and environment separately facilitates better exploration and more advanced behavior. For example, if the outcome of some states or actions has not been monitored, the agent could either (a) avoid them, or (b) intentionally explore them when it knows it will gain monitoring feedback. For instance, if the agent learns that it can reliably observe rewards when the owner is home, it could try new actions for which it does not know the reward. In contrast, if the owner is not home, the agent may choose to act cautiously. Third, decoupling the monitor and the environment creates new opportunities for task transfer, where only the monitor is different or only the environment is. For example, the agent may first learn in a Mon-MDP and then be deployed in a similar — but entirely unmonitored — environment, or may be assigned new tasks under the same monitor. E.g., the agent in Figure 1 could be assigned new chores within the same house, and reasoning about the presence of the owner or home sensors would allow to learn more efficiently.

We argue that current RL frameworks do not capture the complexity of these — and many more — real-world tasks. In contrast, Mon-MDPs provide a comprehensive framework that can be applied to a large variety of challenging real-world complexities. In the next section, we discuss related work that capture some of these complexities, and show how Mon-MDPs can be seen as a general framework encompassing existing areas of research.

2.3 Related Work

In **partially observable MDPs (POMDPs)**, the environment state is not directly observable. The presence of an unobservable component may induce the reader to think that Mon-MDPs are related to POMDPs [1]. However, not observing the reward rather than the state is not just cosmetics. In Mon-MDPs, the agent cannot judge its own actions without rewards, but can still identify the environment state (and, thus, explore). In contrast, in POMDPs the agent must learn to identify the state from its observations to provide an appropriate input for its actions [12]. POMDPs and Mon-MDPs are not mutually exclusive, though, and we can formalize Mon-POMDPs where both rewards and states are not observable.

In **sparse-reward RL**, the agent receives meaningful rewards rarely and zero-rewards otherwise. This makes exploration hard, especially if the reward is given only at task completion. To compensate for the sparsity of rewards, intrinsic motivation relies on auxiliary rewards such as bonuses for hard-to-predict states [31, 35, 38], rarely-visited states [4, 30, 39], or impactful actions [29, 33]. At first it may seem that intrinsic motivation could be a complete solution to Mon-MDPs — the agent could use auxiliary rewards when environment rewards are unobservable. However, in Mon-MDPs the problem is not the sparsity of rewards, but their unobservability, i.e., receiving $\hat{r}^{\text{E}}_t = \bot$. While auxiliary rewards may improve exploration, they cannot replace r^{E}_t — that remains unobservable — and thus the agent cannot directly maximize the sum of environment rewards. Nonetheless, techniques for sparse-reward MDPs will likely still be valuable in Mon-MDPs. We return to this in Section 5.

In **cautious and risk-averse RL**, the agent faces some form of uncertainty, either aleatoric (inherent randomness of the environment) or epistemic (due to the agent's ignorance). In Mon-MDPs, the unobservability of the reward can be seen as epistemic uncertainty. Thus, cautious and risk-averse methods could be used to reason about rewards uncertainty. However, the general setting of cautious and risk-averse RL is fundamentally different from Mon-MDPs, as either the reward is always observable [47, 48] or never [26].

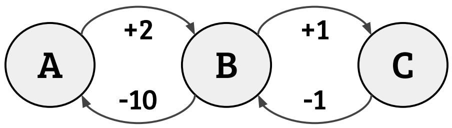


Figure 3: Example showing why learning with unobservable reward is non-trivial. The agent moves between states A, B, and C, but rewards can be observed only with monitoring. Otherwise, the agent observes $\hat{r}^{\text{E}}_t = \bot$. If the agent interprets the lack of reward as meaning that all actions are equally good, it could believe that transitioning from B to A is as good as transitioning from B to C.

In **human-in-the-Loop (HITL)**, a human helps the agent in making the correct decisions, e.g., by providing rewards or suggesting appropriate actions [8, 14, 21]. One may see the monitor as the human in HILT. However, in HITL, typically there is either (a) *never* an environment reward, e.g., the MDP has no reward function and all guidance comes from the human; or (b) *always* an environment reward, and the human provides additional guidance to the agent. In neither of these settings there is an environment reward that the agent can only sometimes see. Yet, many of these techniques deal with imperfect (human) rewards, which are a factor in Mon-MDPs.

In **Bayesian persuasion**, rewards depend on states, actions, and an external parameter [6, 10, 13, 44]. One agent (the *sender*) cannot influence the environment state, but its actions determine what another agent (the *receiver*) observes about the external parameter. This relationship recalls the monitor-agent's in Mon-MDPs, in the sense that one affects what the other observes. However, the two frameworks are fundamentally different. In Bayesian persuasion, there are two decision-makers (receiver and sender) with possibly conflicting goals — the sender affects the receiver's observations for its own good. In Mon-MDPs, the monitor is a fixed process like the environment, and the agent (the only decision-maker) has one goal — to maximize the sum of monitor and environment rewards.

In **active RL (ARL)**, the agent must pay a cost to observe either the state [5] or the reward [16, 37, 40]. ARL is perhaps the closest framework to Mon-MDPs but its setting is simpler. To the best of our knowledge, ARL considers only binary actions to request rewards, constant request costs, and perfect reward observations. By contrast, in Mon-MDPs (a) the observed reward depends on the monitor — a process with its own states, actions, and dynamics; (b) there may be no direct action to request rewards, and requests may fail; (c) the monitor reward is not necessarily a cost; and (d) the monitor can be imperfect and modify the reward. For these reasons, Mon-MDPs can be considered a more general form of ARL.

In **partial monitoring** for multi-armed bandits, the agent must maximize the payoffs of its actions, while unable to observe the exact payoffs [2, 3, 18, 20]. E.g., the agent may observe only payoffs for some bandit arms but not all, or payoffs within a range. Mon-MDPs can be considered a partial monitoring problem, as the agent has to maximize the cumulative sum of partially observable rewards. However, to the best of our knowledge, Mon-MDPs are the first example of partial monitoring in sequential decision-making.

3 MON-MDPS OPTIMALITY

In Mon-MPDs, the agent's goal is to maximize the sum of cumulative rewards $(r^{\text{E}}_t + r^{\text{M}}_t)$ while receiving $(\hat{r}^{\text{E}}_t, r^{\text{M}}_t)$. However, issues arise when $\hat{r}^{\text{E}}_t$ is *unobservable* ($\hat{r}^{\text{E}}_t = \bot$). On one hand, we cannot simply replace r^{E}_t with $\hat{r}^{\text{E}}_t$ as the sum would be ill-defined. On the other hand, we cannot replace $\hat{r}^{\text{E}}_t = \bot$ with an arbitrary value or even just ignore it (this could result in suboptimal or even dangerous agent's behavior, as we show in Section 4.2). To address this problem, we will first define optimality in Mon-MDPs, and then we discuss under what conditions convergence to optimality is guaranteed (despite the presence of unobservable rewards).

3.1 Policy Optimality

We define $a_t \coloneqq (a^{\text{E}}_t, a^{\text{M}}_t)$, $s_t \coloneqq (s^{\text{E}}_t, s^{\text{M}}_t)$, and $r_t \coloneqq r^{\text{E}}_t + r^{\text{M}}_t$ as the joint action, joint state, and joint reward, respectively. Although r_t may not be observable to the agent, it is well-defined — we can formalize the problem like a classic MDP with policy $\pi(a_t|s_t)$ and sum of discounted rewards $\sum_{t=1}^{\infty} \gamma^{t-1} r_t$. Similarly, we can define an optimal policy π^* as a policy maximizing the Q-function $Q^\pi(s, a)$, i.e.,

$$\pi^* \coloneqq \arg\max_\pi \, Q^\pi(s, a), \tag{1}$$

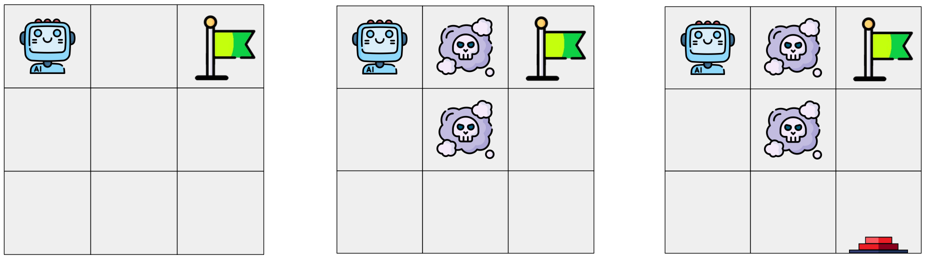
$$\text{where } Q^\pi(s_t, a_t) \coloneqq \mathbb{E}\Big[\sum\nolimits_{i=t}^{\infty} \gamma^{i-t} r_i \,\Big|\, \pi, \mathcal{P}, s_t, a_t\Big], \tag{2}$$

where $\mathcal{P}(s_{t+1} \mid s_t, a_t) = \mathcal{P}^{\text{M}}(s^{\text{M}}_{t+1} \mid s^{\text{M}}_t, a^{\text{M}}_t, s^{\text{E}}_t, a^{\text{E}}_t)\, \mathcal{P}(s^{\text{E}}_{t+1} \mid s^{\text{E}}_t, a^{\text{E}}_t)$. This problem is well-defined and the existence of at least one optimal policy is guaranteed under assumptions already satisfied by Mon-MDPs.[4] Notice that the monitor function $\mathcal{M}$ does not appear here — for every Mon-MDP, there *always* exists an optimal policy.

An optimal policy exists, but can the agent actually learn it? The environment reward r^{E}_t is not always observable and the agent sees $\hat{r}^{\text{E}}_t$ instead. How should it treat unobservable rewards $\hat{r}^{\text{E}}_t = \bot$? This problem is non-trivial. Consider a Mon-MDP whose deterministic environment is shown in Figure 3. Every time the agent moves, it can ask to be monitored or not with $a^{\text{M}} \in$ {MONITOR ME, NO-OP}. The agent observes $\hat{r}^{\text{E}}_t = r^{\text{E}}_t$ when moving if $a^{\text{M}}_t =$ MONITOR ME, and $\hat{r}^{\text{E}}_t = \bot$ otherwise. The monitor reward is constant: $r^{\text{M}}_t = 0$. The optimal policy for sufficiently large γ moves between B and C for an undiscounted cumulative reward of 0. What should the agent think when it does not ask to be monitored and sees $\hat{r}^{\text{E}}_t = \bot$? Should it assume $\bot = 0$? If so, the agent will believe it can avoid negative rewards by not asking to be monitored, and choose to move between A and B for an (apparent, but incorrect) undiscounted cumulative reward of 2. Or, what if the monitor function clips the environment reward to $[-1, 1]$, a common practice in RL [25, 41]? In this case, the agent will conclude that moving between A and B yields the same rewards of moving between B and C.

This example shows that convergence to an optimal policy depends on the observability of proxy rewards, how the agent treats $\hat{r}^{\text{E}}_t = \bot$, and if the monitor function alters environment rewards. In the extreme case, solving a Mon-MDP may be *hopeless*, such as the case of a monitor function always returning $\bot$, where no agent could ever learn an optimal policy. The next section discusses some properties of "well-behaved" Mon-MDPs, sufficient to guarantee the existence of an algorithm that converges to an optimal policy. In Appendix B, we give a formal treatment of *solvable* and *unsolvable* Mon-MDPs as well as interesting settings between these two.

[4] These standard assumptions include stationary reward and transition functions, bounded rewards, and a discount factor $\gamma \in [0, 1)$ [32].



(a) ***Simple.*** **The reward r^{E}_t is 1 (goal) or 0 (otherwise). To observe $\hat{r}^{\mathrm{E}}_t = r^{\mathrm{E}}_t$, the agent must do a^{M}_t = `MONITOR ME` and pay $r^{\mathrm{M}}_t = -0.2$. Otherwise $\hat{r}^{\mathrm{E}}_t = \bot$.**

(b) ***Penalty.*** **Like the Simple version, but there are penalty cells that give $r^{\mathrm{E}}_t = -10$.**

(c) ***Button.*** **Monitoring is turned `ON` by hitting the red button with a^{E}_t = `DOWN`. When monitoring is `ON`, $r^{\mathrm{M}}_t = -0.2$ and $\hat{r}^{\mathrm{E}}_t = r^{\mathrm{E}}_t$ until the button is hit again.**

Figure 4: In our Mon-MDPs, the agent starts in the top-left cell and has to reach the goal avoiding penalty cell. There are nine states s^{E} (one for each cell) and four actions a^{E} (`LEFT`, `RIGHT`, `UP`, `DOWN`). Rewards r^{E} are 1 (goal), -10 (penalty cells), and 0 (otherwise). Episodes end when the agent reaches the goal or after 50 steps. We propose three levels of difficulty depending on the presence of penalty cells and on the type of monitor.

3.2 Convergence to an Optimal Policy

PROPERTY 1 (ERGODIC MON-MDP). *A Mon-MDP is ergodic if any joint state (s^E, s^M) can be reached by any other joint state $(s^E, s^M)'$ given infinite exploration. This implies that every state will be visited infinitely often given infinite exploration.*[5]

PROPERTY 2 (ERGODIC MONITOR FUNCTION). *A monitor function $\hat{r}^E \sim \mathcal{M}(r^E, s^M, a^M)$ is ergodic if for all environment pairs (s^E, a^E) the proxy reward will be observable ($\hat{r}^E \neq \bot$) given infinite exploration.*

PROPERTY 3 (TRUTHFUL MONITOR FUNCTION). *A monitor function $\hat{r}^E \sim \mathcal{M}(r^E, s^M, a^M)$ is truthful if $\forall t$ either $\hat{r}^E_t = r^E_t$ or $\hat{r}^E_t = \bot$.*

PROPOSITION 1 (SUFFICIENT CONDITIONS FOR CONVERGENCE TO AN OPTIMAL POLICY). *There exist an algorithm such that for any Mon-MDP with finite states and actions satisfying Properties 1, 2, and 3, the algorithm converges to an optimal policy of that Mon-MDP.*

We prove Proposition 1 in Section 4.3, after examining candidate algorithms in Section 4.2. For now, we remark that Properties 1, 2, and 3 guarantee that the agent will observe every environment reward infinitely often, even though not for every monitor state and action. In other words, the agent can still learn an optimal policy *even if there are situations where it cannot observe the rewards.*

4 EMPIRICAL ANALYSIS OF MON-MDPS

In this section, we show practical challenges an agent faces in Mon-MDPs, and why methods used in MDPs fail to converge to an optimal policy. We start by introducing toy environments and monitors: in some the agent can use a monitor action to immediately be monitored, while in others it must execute certain environment actions to activate monitoring. We then introduce algorithms that account for unobservable rewards $\hat{r}^{\mathrm{E}}_t = \bot$ in different ways. Source code is available at https://github.com/AmiiThinks/mon_mdp_aamas24.

4.1 The Environment and The Monitors

We study Mon-MDPs in the 3×3 gridworld shown in Figure 4, where the agent has to reach the goal ($r^{\mathrm{E}}_t = 1$) using actions $a^{\mathrm{E}} \in$ {`LEFT, DOWN, RIGHT, UP`} while avoiding penalties ($r^{\mathrm{E}}_t = -10$). However, rewards are not always observable. We consider three Mon-MDPs of increasing difficulty, that differ in the environment reward and the monitor dynamics (more details in Appendix D.1).

[5] This is a generalization of the definition of ergodic MDPs [32].

- **Simple grid (Simple).** Together with a^{E}, the agent selects $a^{\mathrm{M}} \in$ {`MONITOR ME, NO-OP`}. With a^{M}_t = `MONITOR ME`, the agent observes $\hat{r}^{\mathrm{E}}_t = r^{\mathrm{E}}_t$ and pays a cost ($r^{\mathrm{M}}_t = -0.2$). Otherwise it receives $\hat{r}^{\mathrm{E}}_t = \bot$ at no cost ($r^{\mathrm{M}}_t = 0$). The optimal policy brings the agent to the goal ($r^{\mathrm{E}}_t = 1$), never asking to be monitored.
- **Grid with penalties (Penalty).** Same monitor as Simple, but the grid now has cells with negative environment reward ($r^{\mathrm{E}}_t = -10$). The optimal policy brings the agent to the goal, avoiding penalty cells and never asking to be monitored.
- **Grid with penalties and button (Button).** There is no `MONITOR ME` action. Instead, monitoring is determined by the monitor state $s^{\mathrm{M}} \in$ {`ON, OFF`}. The agent can change s^{M} by hitting the red button in the rightmost state with a^{E}_t = `DOWN`. At the start of an episode, the monitor state is set randomly. Every time step s^{M}_t = `ON`, the agent pays a cost ($r^{\mathrm{M}}_t = -0.2$). The optimal policy brings the agent to the goal, avoiding penalty cells and turning `OFF` the monitor along the way if it was `ON` at the start of the episode.

In the Simple and Penalty Mon-MDPs, the agent can observe rewards with the explicit *monitor* action a^{M}_t = `MONITOR ME`. In the Button Mon-MDP, instead, the agent must use a sequence of *environment* actions a^{E}_t to start (or stop) observing rewards. This, and not being able to observe rewards for a period of time, has important consequences as we show in the next section.

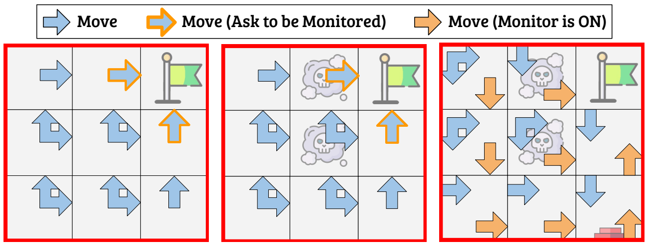
4.2 The Algorithms

We present algorithms based on Q-Learning [43]. Given samples (s_t, a_t, r_t, s_{t+1}), Q-Learning updates are

$$Q(s_t, a_t) \leftarrow (1-\alpha_t)Q(s_t, a_t) + \alpha_t(r_t + \gamma \overbrace{\max_a Q(s_{t+1}, a)}^{\text{greedy policy}}), \quad (3)$$

where α_t is the learning rate, and we wrote Q in place of Q^π for the sake of simplicity. In classic MDPs, Q-Learning is guaranteed to converge to an optimal greedy policy using ε-greedy exploration with an appropriate learning rate α_t and exploration ε_t schedules [9].

In Mon-MDPs, we observe $\hat{r}_t := (\hat{r}^{\mathrm{E}}_t, r^{\mathrm{M}}_t)$ — how do we update $Q(s_t, a_t)$ when $\hat{r}^{\mathrm{E}}_t = \bot$? We consider a set of Q-Learning variants that differ in how they treat $\hat{r}^{\mathrm{E}}_t = \bot$, and show that the resulting policies can be very different (and often suboptimal). For each variant, we show the *greedy policy* learned after 10,000 steps in each of our three Mon-MDPs. Figures show the action taken in every cell — including monitoring actions. Figures outlined in red are suboptimal policies, while figures outlined in green are optimal. Over 100



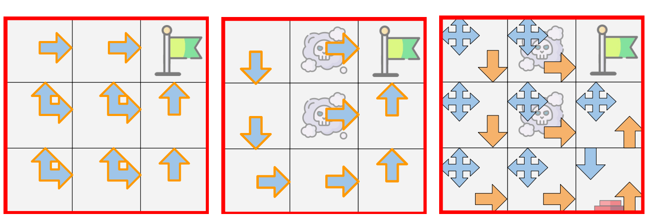
(a) Simple ✗ (b) Penalty ✗ (c) Button ✗

Figure 5: If $\perp = 0$, the agent asks to be monitored only when rewards are positive, "ignoring" negative rewards.

seeds, the algorithms always converged to these policies within 10,000 steps when $\gamma = 0.99$. For all details about the algorithms variants, please see Appendix C.

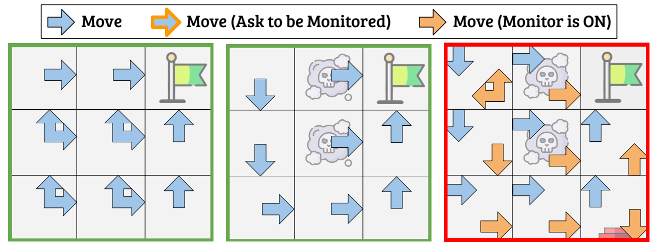
4.2.1 ***Algorithm 1: Assign $\perp = 0$.*** Our first algorithm assumes that unobservable rewards $\hat{r}^{\mathrm{E}}_t = \perp$ have a constant value of 0. This can be seen as treating the Mon-MDP as a sparse-reward MDP, where most rewards are 0. Figure 5 shows that the policy ends up ignoring negative rewards, asking to be monitored only when positive rewards can be observed. In the Simple Mon-MDP, the agent asks a^{M}_t = `MONITOR ME` as it moves to the goal ($r^{\mathrm{E}}_t = 1$). In the Penalty Mon-MDP, the agent does not avoid penalty cells ($r^{\mathrm{E}}_t = -10$), "pretending" that walking over them gives $r^{\mathrm{E}}_t = 0$ by not asking to be monitored. In the Button Mon-MDP, the agent learns to press the button — again, without avoiding penalty cells — and then goes to the goal. All of the learned policies across all three Mon-MDPs are suboptimal. In Appendix D.3, we show that this algorithm performs poorly for different values assigned to $\perp$.

4.2.2 ***Algorithm 2: Ignore $\hat{r}^{E}_t = \perp$.*** Instead of assigning an arbitrary value to unobservable rewards, the algorithm does not update the Q-function when $\hat{r}^{\mathrm{E}}_t = \perp$. This could be considered a safe strategy, as the agent disregards samples with incomplete information. However, as shown in Figure 6, the resulting policy ends up always seeking monitoring. In the Simple and Penalty Mon-MDPs, the agent executes a^{M}_t = `MONITOR ME` in every state. In the Button Mon-MDP, when the monitor is `ON` the agent walks to the goal without pressing the button. However, when the monitor is `OFF`, the policy acts randomly. This happens because the Q-function is never updated when the monitor is `OFF`, as receiving $\hat{r}^{\mathrm{E}}_t = \perp$ precludes any update. As a result, its learned policy depends only on the Q-function's initialization. When all Q-values initialized to the same value, the policy is random, as shown in Figure 6c.



(a) Simple ✗ (b) Penalty ✗ (c) Button ✗

Figure 6: Ignoring updates when $\hat{r}^{E}_t = \perp$, results in a policy that can navigate only when monitored.



(a) Simple ✓ (b) Penalty ✓ (c) Button ✗

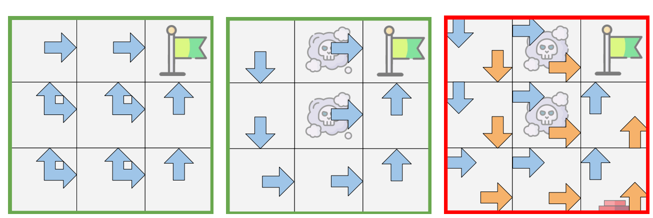
Figure 7: With two Q-functions and joint policy, the agent hits the wall in the top-left corner of the Button Mon-MDP.

4.2.3 ***Algorithm 3: Two Q-functions (Joint Greedy Policy).*** Ignoring samples when $\hat{r}^{\mathrm{E}}_t = \perp$ disregards useful information given by r^{M}_t. To fix this, we decouple the value of states and actions into two Q-functions: Q^{E} trained using proxy rewards (only when $\hat{r}^{\mathrm{E}}_t \neq \perp$) and Q^{M} using monitor rewards. This way, even if $\hat{r}^{\mathrm{E}}_t = \perp$ we can still update the latter. This begs the question: how should the algorithm greedily select actions when there are two Q-functions?

The first strategy we propose (the Algorithm we are describing) is to select them *jointly* with $(a^{\mathrm{E}}_t, a^{\mathrm{M}}_t) = \arg\max_{a^{\mathrm{E}},a^{\mathrm{M}}}\{Q^{\mathrm{E}} + Q^{\mathrm{M}}\}$. Intuitively, the agent would try to maximize the sum of both rewards simultaneously. As shown in Figure 7, while able to learn an optimal policy in the Simple and Penalty Mon-MDPs, this variant fails in the Button Mon-MDP. Interestingly, the policy correctly avoids penalty cells, turns `OFF` the monitor, and goes to the goal in all states *but in the top-left cell when the monitor is* `ON`. This is due to conflicting Q-values. Q^{E} wants to go `DOWN` and follow the safe path to the goal. On the contrary, Q^{M} wants to go `RIGHT`, step over the penalty cells (Q^{M} does not accumulate $r^{\mathrm{E}}_t = -10$), and end the episode (to stop receiving $r^{\mathrm{M}}_t = -0.2$). The sum of the Q-values, however, favors neither `DOWN` nor `RIGHT`, but `UP` and `LEFT`. After all, the `max` operator of the greedy policy is not linear,[6] thus summing the two Q-functions does not guarantee maximizing both (and, indeed, a single action may not maximize both Q-functions).

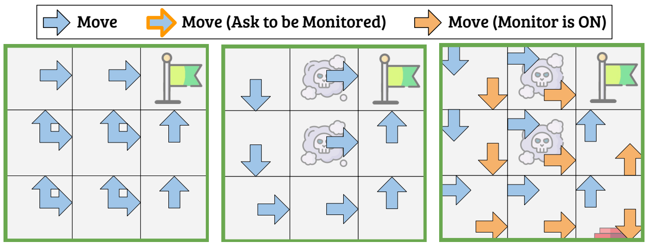
4.2.4 ***Algorithm 4: Two Q-functions (Sequential Greedy Policy).*** To avoid conflicting Q-values, we modify the action selection so that the agent selects *first* $a^{\mathrm{E}}_t = \arg\max_{a^{\mathrm{E}}} Q^{\mathrm{E}}$ and *then* $a^{\mathrm{M}}_t = \arg\max_{a^{\mathrm{M}}} Q^{\mathrm{M}}\,|_{a^{\mathrm{E}}_t}$. Thus, this agent prioritizes Q^{E}, as maximizing Q^{M} is subject to the greedy environment action. The policy in Figure 8 still fails in the Button Mon-MDP, as the agent does not turn `OFF` the monitor on its way to the goal. This happens because

[6] Given two functions $f(x), g(x)$, $\max_x(f(x)+g(x)) \neq \max_x f(x) + \max_x g(x)$.



(a) Simple ✓ (b) Penalty ✓ (c) Button ✗

Figure 8: Using a sequential policy, instead, the agent does not turn `OFF` monitoring in the Button Mon-MDP.



(a) Simple ✓ (b) Penalty ✓ (c) Button ✓

Figure 9: With a reward model, the agent learns the optimal policy in all three Mon-MDPs.

there are no explicit monitor actions and the agent must use environment actions to turn it `OFF`. Yet, going `DOWN` in the bottom-right cell to press the button is not optimal for Q^{E}. Since a^{E} is selected greedily using Q^{E}, the agent goes to the goal ignoring the button.

4.2.5 ***Algorithm 5: Learn a Reward Model.*** The agent replaces $\hat{r}^{\mathrm{E}}_t = \bot$ with the reward predicted by a model. In discrete Mon-MDPs, the model is a table like the Q-function that stores the running mean of the environment rewards as it observes $\hat{r}^{\mathrm{E}}_t = r^{\mathrm{E}}_t$ (see Section 4.3 for more details). This algorithm converges to an optimal policy in all Mon-MDPs, as shown in Figure 9. Intuitively, the reward model allows the agent to know the current reward r^{E}_t even without observing it. We note, however, that this algorithm works because all three Mon-MDPs satisfy the conditions of Proposition 1. In Section 4.3, we formally prove the convergence to an optimal policy of Algorithm 5 according to Proposition 1.

4.2.6 ***Remarks.*** We emphasize that Algorithms 3, 4, and 5 (Joint, Sequential, Reward Model) solve the Penalty Mon-MDP because they can reason about the environment independently from the monitor. As discussed in Section 2.2, because the monitor and the environment are decoupled, the agent learns that the monitor does not change the environment reward, and that walking on penalty cells is undesirable even if not monitored. If the agent walks on penalty cells while monitored (a similar action to spilling water with the owner at home) and observes a negative reward, it will learn that the reward would still be negative even when the monitor (the owner) is not providing it. Joint and Sequential decouple environment and monitor with two Q-functions, Reward Model with a reward model that depends only on the environment. However, Joint and Sequential fail in the Button Mon-MDP. In Appendix C, we discuss about stricter conditions of convergence for Sequential, and about the lack of guarantees of convergence for Joint.

4.3 Proof of Proposition 1

We return now to prove Proposition 1 from Section 3.2, and use Q-Learning with a reward model as the candidate algorithm that can solve all finite Mon-MDPs satisfying Properties 1, 2, and 3.

Proof 1. *Consider Q-Learning on a finite Mon-MDP that replaces the observed reward $\hat{r}^E_t$ with the running average of observed proxy reward stored in a table $\widehat{R}(s^E, a^E)$, i.e.,*

$$\widehat{Q}(s_t, a_t) := \mathbb{E}\big[\textstyle\sum_{i=t}^{\infty} \gamma^{i-t}\big(\widehat{R}(s^E_i, a^E_i) + r^M_i\big) \mid \pi, \mathcal{P}, s_t, a_t\big]$$

$$N_{k+1}(s^E_t, a^E_t) \leftarrow N_k(s^E_t, a^E_t) + 1 \qquad \text{if } \hat{r}^{\mathrm{E}}_t \neq \bot$$

$$\widehat{R}_{k+1}(s^E_t, a^E_t) \leftarrow \frac{\big(N_{k+1}(s^E_t, a^E_t) - 1\big)\,\widehat{R}_k(s^E_t, a^E_t) + \hat{r}^E_t}{N_{k+1}(s^E_t, a^E_t)} \qquad \text{if } \hat{r}^{\mathrm{E}}_t \neq \bot$$

$$\widehat{Q}_{k+1}(s_t, a_t) \leftarrow (1 - \alpha_t)\widehat{Q}_k(s_t, a_t) + \alpha_t\big(\widehat{R}_{k+1}(s^E_t, a^E_t) + r^M_t + \gamma \max_a \widehat{Q}_{k+1}(s_{t+1}, a)\big) \quad (4)$$

where k denotes the k-th update, and $N(s^E_t, a^E_t)$ is a count that increases every time the agent observes a reward, i.e., if $\hat{r}^E_t \neq \bot$. Then, this algorithm converges to an optimal policy in Eq. (1) *if (a) the policy is greedy in the limit with infinite exploration (GLIE), and (b) the learning rate α_t satisfies the Robbins-Monro conditions [34].*

(I) Under a GLIE policy, the agent will visit every state infinitely often (Property 1), will observe a reward for every state (Property 2), and the observed reward will be the environment reward (Property 3). Therefore, the agent will observe the environment reward for every environment state-action pair infinitely often.

(II) Under (I) and by the central limit theorem,

$$\lim_{k\to\infty} \widehat{R}_k(s^E, a^E) = \mathbb{E}\left[r^E \mid \pi, \mathcal{P}^E\right]$$

(III) Given (II) and because of linearity of expectation, maximizing the Q-function learned using rewards from $\widehat{R}(s^E, a^E)$ approaches the original optimization problem of Eq. (2), *i.e.,*

$$\begin{aligned}
\widehat{Q}_k(s_t, a_t) \;&:=\; \mathbb{E}\big[\textstyle\sum_{i=t}^{\infty} \gamma^{i-t}\big(\widehat{R}_k(s^E_i, a^E_i) + r^M_i\big) \mid \pi, \mathcal{P}, s_t, a_t\big] \\
&\underset{k\to\infty}{=} \mathbb{E}\big[\textstyle\sum_{i=t}^{\infty} \gamma^{i-t}\big(\mathbb{E}\left[r^E_i \mid \pi, \mathcal{P}^E\right] + r^M_i\big) \mid \pi, \mathcal{P}, s_t, a_t\big] \\
&=\; \mathbb{E}\big[\textstyle\sum_{i=t}^{\infty} \gamma^{i-t}\big(r^E_t + r^M_i\big) \mid \pi, \mathcal{P}, s_t, a_t\big] \\
&=:\; Q(s_t, a_t)
\end{aligned}$$

(IV) Given (III), Eq. (3) *and* (4) *are equivalent in the limit. Under a GLIE policy and if the learning rate α_t satisfies the Robbins-Monro conditions, Eq.* (3) *converges to the Q-function of an optimal greedy policy [7, 9, 24].*[7]

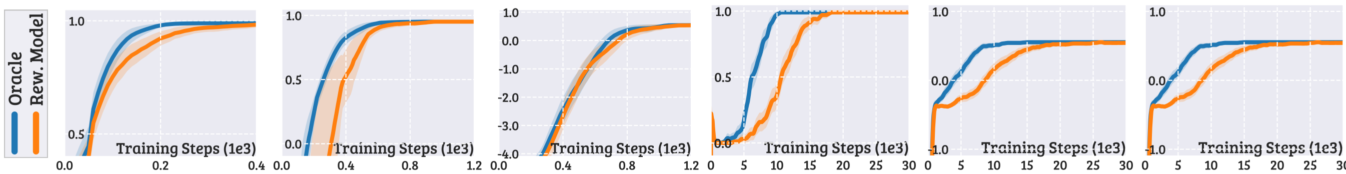
4.4 Empirical Rate of Convergence

One assumption needed in Proof 1 is that Q-Learning uses a GLIE policy. This, together with Mon-MDP ergodicity (Property 1) guarantees that the agent will visit every state-action pair and observe every reward. But how hard is exploration in Mon-MDPs? The agent will not observe the environment reward all the time, and cannot learn an optimal policy until it has seen them all sufficiently often. How does this affect the rate of convergence to an optimal policy? Intuitively, if some rewards are unobservable learning will be slower, more so if the environment reward function is noisy.

In this section, we empirically investigate the rate of convergence of "Q-Learning with Reward Model" presented in Section 4.2.5 against an "Oracle" Q-Learning. The Oracle executes monitor actions and receives monitor rewards, but always observes $\hat{r}^{\mathrm{E}}_t = r^{\mathrm{E}}_t$. For each Mon-MDP, we consider also a version where the environment reward has Gaussian noise with standard deviation 0.05. For all details about the experiments, more plots and table, and an additional evaluation on harder Mon-MDPs, refer to Appendix D.2.

Figure 10 shows that the Oracle always converges faster to an optimal policy, up to ×2 faster with noisy rewards. While $\widehat{R}$ compensates for the unobservability of rewards, the agent still needs to

[7] A greedy optimal policy always exists for MDPs with finite states and actions, stationary reward and transition functions, bounded rewards, and $\gamma \in [0, 1)$ [32].



(a) Simple (b) Penalty (c) Button (d) Simple (Noisy) (e) Penalty (Noisy) (f) Button (Noisy)

Figure 10: Episode return $\sum_{t=1}^{T} \gamma^{t-1}(r_t^{\mathbf{E}} + r_t^{\mathbf{M}})$ of greedy policies averaged over 100 seeds (shades denote 95% confidence interval).

observe the rewards sufficiently often — especially if they are noisy — for the model to be accurate. In Appendix D.2, we show results on Mon-MDPs with larger monitor spaces and richer dynamics, where the gap between the Oracle and Reward Model is even larger.

5 FUTURE WORK

Throughout the paper, we discussed how Mon-MDPs relate to prior work, and our empirical study has highlighted important challenges that would benefit from existing RL techniques. Below, we describe some of the most interesting directions of future research that this work opens up, and connect them to existing areas of research such as meta RL, model-based RL, cautious RL, and intrinsic motivation.

Convergence, Bounds, and Connection to Partial Monitoring. Mon-MDPs are a new framework and therefore open to further theoretical analysis. First and foremost, we have presented a set of *sufficient* conditions for convergence, but these may not be *necessary*. For instance, the monitor may not need to be truthful, as suggested by prior work on reward shaping [28]. Relaxing these conditions will likely pose an additional challenge that could be addressed by having a belief over the reward [22]. Furthermore, it would be interesting to investigate the convergence bounds of monitored algorithms. For example, similar research proved regret bounds for many partial monitoring bandits [2, 3, 18].

Generalization, Train-And-Deploy, and Meta RL. In this work, we considered finite Mon-MDPs and assumed properties on the Mon-MDP that may not hold in real-world problems, e.g., monitor ergodicity and truthfulness. Can the agent learn an optimal policy even when these properties are not satisfied?

Consider the agent in Figure 1, but this time it can *never* be monitored while watering plants and — if it spills water it will never receive a negative feedback. However, the agent can be monitored when cleaning dishes. Can it learn that spilling water is undesirable by receiving negative feedback for spilling water while cleaning dishes? This requires 1) reasoning over the monitor and the environment independently — spilling water is undesirable regardless of the monitor state — and 2) generalization across environment states and actions — spilling water in the kitchen and spilling water is equally undesirable. In Section 2.2, we argued that the Mon-MDP framework already allows the former reasoning. For the latter, we need to incorporate generalization and go beyond finite Mon-MDPs.

More generally, Mon-MDPs can be further extended to consider situations where the agent must act in unmonitored environments — where rewards are *never* observable — after being trained in a monitored environment. This is closely related to train-and-deploy and meta RL settings [23, 42], and requires the ability to generalize knowledge about rewards across states — possibly of different environments — to compensate for their unobservability.

Unsolvable Mon-MDPs. What if the agent cannot learn an optimal policy because some rewards are *never* observable? While it may be impossible to act optimally with respect to environment rewards, the agent should still act "optimally" according to what it can observe. In this regard, it is interesting to consider algorithms that can tackle *unsolvable Mon-MDPs*, i.e., can learn "useful" policies in Mon-MDPs where it is impossible to learn an optimal policy due to unobservability of the rewards. In Appendix B, we formally discuss the notion of solvability from a theoretical point of view and set the stage for future directions of research. For example, the best way to act in situations of uncertainty is still a matter of dispute in RL and relates to cautious and risk-averse RL [26, 47].

Exploration. In Section 4.4, we showed that unobservable rewards make exploration significantly harder. Clearly, naive ϵ-greedy exploration does not exploit the complexity of Mon-MDPs, and we believe there are exciting potential improvements. In particular, as discussed in Section 2.2, explicitly reasoning on monitor and environment separately facilitates better exploration and more advanced behaviors. For example, the agent could use intrinsic motivation [27, 29] to prefer environment states for which it has not observed the reward yet. At the same time, it could try new actions in states where it knows it will be monitored.

6 CONCLUSION

MDPs offer a framework to tackle decision-making problems, but the assumption of reward observability is not descriptive of all real-world problems. To account for situations where the agent cannot observe the rewards generated by the environment to judge its actions, we presented *Monitored MDPs*. We discussed the theoretical and practical consequences of unobservable rewards, and presented toy environments and algorithms to illustrate subsequent challenges. While prior work on active RL and partial monitoring has addressed partially observable rewards, this is — to the best of our knowledge — the first work that presents a generic formalism allowing for sequential decision-making without requiring the monitor to have explicit binary monitoring actions.

In the same way that RL built its foundation starting from theoretical analyses on discrete MDPs and the empirical investigations of chainworlds and gridworlds, with this work we aim to set the stage for future research ranging from theoretical analysis of stronger guarantees of convergence, development of better algorithms, and practical applications of Mon-MDPs to real-world problems.

ACKNOWLEDGEMENTS

This research was supported by grants from the Alberta Machine Intelligence Institute (Amii); a Canada CIFAR AI Chair, Amii; Digital Research Alliance of Canada; Huawei; Mitacs; and NSERC.

REFERENCES

[1] Karl J Astrom et al. 1965. Optimal control of Markov processes with incomplete state information. *J. Math. Anal. Appl.* 10, 1 (1965), 174–205.

[2] Peter Auer, Nicolo Cesa-Bianchi, Yoav Freund, and Robert E Schapire. 2002. The Nonstochastic Multiarmed Bandit Problem. *SIAM J. Comput.* 32, 1 (2002), 48–77.

[3] Gábor Bartók, Dean P Foster, Dávid Pál, Alexander Rakhlin, and Csaba Szepesvári. 2014. Partial monitoring – classification, regret bounds, and algorithms. *Mathematics of Operations Research* 39, 4 (2014), 967–997.

[4] Marc G Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. 2016. Unifying count-based exploration and intrinsic motivation. In *Advances in Neural Information Processing Systems (NIPS).*

[5] Colin Bellinger, Rory Coles, Mark Crowley, and Isaac Tamblyn. 2021. Active Measure Reinforcement Learning for Observation Cost Minimization. In *Canadian Conference on AI.*

[6] Martino Bernasconi, Matteo Castiglioni, Alberto Marchesi, Nicola Gatti, and Francesco Trovò. 2022. Sequential information design: Learning to persuade in the dark. In *Advances in Neural Information Processing Systems (NeurIPS).*

[7] Dimitri P. Bertsekas and John N. Tsitsiklis. 1996. *Neuro-Dynamic Programming.* Athena Scientific.

[8] Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2017. Deep reinforcement learning from human preferences. In *Advances in Neural Information Processing Systems (NIPS).*

[9] Peter Dayan. 1992. The convergence of TD(λ) for general λ. *Machine Learning* 8 (1992), 341–362.

[10] Jiarui Gan, Rupak Majumdar, Goran Radanovic, and Adish Singla. 2022. Bayesian persuasion in sequential decision-making. In *AAAI Conference on Artificial Intelligence.*

[11] Jason Gauci, Edoardo Conti, Yitao Liang, Kittipat Virochsiri, Yuchen He, Zachary Kaden, Vivek Narayanan, Xiaohui Ye, Zhengxing Chen, and Scott Fujimoto. 2018. Horizon: Facebook's Open Source Applied Reinforcement Learning Platform. arXiv:arXiv:1811.00260

[12] Noah Golowich, Ankur Moitra, and Dhruv Rohatgi. 2022. Learning in observable POMDPs, without computationally intractable oracles. *Advances in Neural Information Processing Systems (NeurIPS)*, 1458–1473.

[13] Emir Kamenica and Matthew Gentzkow. 2011. Bayesian persuasion. *American Economic Review* 101, 6 (2011), 2590–2615.

[14] W Bradley Knox and Peter Stone. 2009. Interactively shaping agents via human reinforcement: The TAMER framework. In *International Conference on Knowledge Capture (K-CAP).*

[15] Jens Kober, J Andrew Bagnell, and Jan Peters. 2013. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research* 32, 11 (2013), 1238–1274.

[16] David Krueger, Jan Leike, Owain Evans, and John Salvatier. 2020. Active Reinforcement Learning: Observing Rewards at a Cost. arXiv:arXiv:2011.06709

[17] Pawel Ladosz, Lilian Weng, Minwoo Kim, and Hyondong Oh. 2022. Exploration in deep reinforcement learning: A survey. *Information Fusion* 85 (2022), 1–22.

[18] Tor Lattimore and Csaba Szepesvári. 2019. An information-theoretic approach to minimax regret in partial monitoring. In *Conference on Learning Theory.*

[19] Jianxiong Li, Xiao Hu, Haoran Xu, Jingjing Liu, Xianyuan Zhan, Qing-Shan Jia, and Ya-Qin Zhang. 2023. Mind the Gap: Offline Policy Optimization for Imperfect Rewards. In *International Conference on Learning Representations (ICLR).*

[20] Nick Littlestone and Manfred K Warmuth. 1994. The weighted majority algorithm. *Information and computation* 108, 2 (1994), 212–261.

[21] James MacGlashan, Mark K Ho, Robert Loftin, Bei Peng, Guan Wang, David L Roberts, Matthew E Taylor, and Michael L Littman. 2017. Interactive learning from policy-dependent human feedback. In *International Conference on Machine Learning (ICML).*

[22] Ofir Marom and Benjamin Rosman. 2018. Belief reward shaping in reinforcement learning. In *AAAI Conference on Artificial Intelligence.*

[23] Tatsuya Matsushima, Hiroki Furuta, Yutaka Matsuo, Ofir Nachum, and Shixiang Gu. 2020. Deployment-Efficient Reinforcement Learning via Model-Based Offline Optimization. arXiv:arXiv:2006.03647

[24] Francisco S Melo. 2001. Convergence of Q-learning: A simple proof. *Institute Of Systems and Robotics, Tech. Rep* (2001), 1–4.

[25] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (2015), 529–533.

[26] Montaser Mohammedalamen, Dustin Morrill, Alexander Sieusahai, Yash Satsangi, and Michael Bowling. 2021. Learning to Be Cautious. arXiv:arXiv:2110.15907

[27] Mirco Mutti, Lorenzo Pratissoli, and Marcello Restelli. 2021. Task-agnostic exploration via policy gradient of a non-parametric state entropy estimate. In *AAAI Conference on Artificial Intelligence.*

[28] Andrew Y Ng, Daishi Harada, and Stuart Russell. 1999. Policy invariance under reward transformations: Theory and application to reward shaping. In *International Conference on Machine Learning (ICML).*

[29] Simone Parisi, Victoria Dean, Deepak Pathak, and Abhinav Gupta. 2021. Interesting Object, Curious Agent: Learning Task-Agnostic Exploration. In *International Conference on Neural Information Processing Systems (NeurIPS).*

[30] Simone Parisi, Davide Tateo, Maximilian Hensel, Carlo D'Eramo, Jan Peters, and Joni Pajarinen. 2022. Long-Term Visitation Value for Deep Exploration in Sparse Reward Reinforcement Learning. *Algorithms* 15, 3 (2022).

[31] Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell. 2017. Curiosity-driven Exploration by Self-supervised Prediction. In *International Conference on Machine Learning (ICML).*

[32] Martin L. Puterman. 1994. *Markov Decision Processes: Discrete Stochastic Dynamic Programming.* Wiley-Interscience. 694 pages.

[33] Roberta Raileanu and Tim Rocktäschel. 2020. RIDE: Rewarding Impact-Driven Exploration for Procedurally-Generated Environments. In *International Conference on Learning Representations (ICLR).*

[34] Herbert Robbins and Sutton Monro. 1951. A stochastic approximation method. *The Annals of Mathematical Statistics* 22 (1951), 400–407.

[35] Jürgen Schmidhuber. 1991. A Possibility for lmplementing Curiosity and Boredom in Model-Building Neural Controllers. In *International Conference on Simulation of Adaptive Behavior (SAB).*

[36] Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. 2020. Mastering Atari, Go, chess and shogi by planning with a learned model. *Nature* 588, 7839 (2020), 604–609.

[37] Sebastian Schulze and Owain Evans. 2018. Active Reinforcement Learning with Monte-Carlo Tree Search. arXiv:arXiv:1803.04926

[38] Bradly C Stadie, Sergey Levine, and Pieter Abbeel. 2015. Incentivizing exploration in reinforcement learning with deep predictive models. In *NIPS Workshop on Deep Reinforcement Learning.*

[39] Alexander L Strehl and Michael L Littman. 2008. An analysis of model-based interval estimation for Markov decision processes. *Journal of Computer and System Sciences (JCSS)* 74, 8 (2008), 1309–1331.

[40] Aaron D Tucker, Caleb Biddulph, Claire Wang, and Thorsten Joachims. 2023. Bandits with costly reward observations. In *Conference on Uncertainty in Artificial Intelligence (UAI).*

[41] Hado P van Hasselt, Arthur Guez, Matteo Hessel, Volodymyr Mnih, and David Silver. 2016. Learning values across many orders of magnitude. In *Advances in Neural Information Processing Systems (NIPS).*

[42] Jane X Wang, Zeb Kurth-Nelson, Dharshan Kumaran, Dhruva Tirumala, Hubert Soyer, Joel Z Leibo, Demis Hassabis, and Matthew Botvinick. 2018. Prefrontal cortex as a meta-reinforcement learning system. *Nature Neuroscience* 21, 6 (2018), 860–868.

[43] Christopher JCH Watkins and Peter Dayan. 1992. Q-learning. *Machine Learning* 8, 3-4 (1992), 279–292.

[44] Jibang Wu, Zixuan Zhang, Zhe Feng, Zhaoran Wang, Zhuoran Yang, Michael I. Jordan, and Haifeng Xu. 2022. Sequential Information Design: Markov Persuasion Process and Its Efficient Reinforcement Learning. In *ACM Conference on Economics and Computation.*

[45] Chao Yu, Jiming Liu, Shamim Nemati, and Guosheng Yin. 2021. Reinforcement learning in healthcare: A survey. *ACM Computing Surveys (CSUR)* 55, 1 (2021), 1–36.

[46] Fabio Massimo Zanzotto. 2019. Human-in-the-loop artificial intelligence. *Journal of Artificial Intelligence Research (JAIR)* 64 (2019), 243–252.

[47] Jesse Zhang, Brian Cheung, Chelsea Finn, Sergey Levine, and Dinesh Jayaraman. 2020. Cautious adaptation for reinforcement learning in safety-critical settings. In *International Conference on Machine Learning (ICML).*

[48] Luisa Zintgraf, Kyriacos Shiarlis, Maximilian Igl, Sebastian Schulze, Yarin Gal, Katja Hofmann, and Shimon Whiteson. 2020. VariBAD: A Very Good Method for Bayes-Adaptive Deep RL via Meta-Learning. In *International Conference on Learning Representations (ICLR).*