# To the Max

## Reinventing Reward in Reinforcement Learning

Veviurko, Grigorii; Böhmer, Wendelin; de Weerdt, Mathijs

**Publication date**
2024

**Document Version**
Final published version

**Published in**
Proceedings of Machine Learning Research

**Citation (APA)**
Veviurko, G., Böhmer, W., & de Weerdt, M. (2024). To the Max: Reinventing Reward in Reinforcement Learning. *Proceedings of Machine Learning Research*, *235*, 49455-49470.

# To the Max: Reinventing Reward in Reinforcement Learning

**Grigorii Veviurko** [1] **Wendelin Böhmer** [1] **Mathijs de Weerdt** [1]

## Abstract

In reinforcement learning (RL), different reward functions can define the same optimal policy but result in drastically different learning performance. For some, the agent gets stuck with a suboptimal behavior, and for others, it solves the task efficiently. Choosing a good reward function is hence an extremely important yet challenging problem. In this paper, we explore an alternative approach for using rewards for learning. We introduce *max-reward RL*, where an agent optimizes the maximum rather than the cumulative reward. Unlike earlier works, our approach works for deterministic and stochastic environments and can be easily combined with state-of-the-art RL algorithms. In the experiments, we study the performance of max-reward RL algorithms in two goal-reaching environments from Gymnasium-Robotics and demonstrate its benefits over standard RL. The code is available at `https://github.com/veviurko/To-the-Max`.

## 1. Introduction

Reinforcement Learning (RL) is a learning paradigm where an intelligent agent solves sequential decision-making problems through trial and error. The main objective that an RL agent learns to optimize is the *cumulative return*, i.e., a discounted sum of the *rewards*. This makes the reward a crucial element of the problem, as it defines the optimal decision-making policy that the agent will try to learn.

It is well known (Ng et al., 1999) that there are infinitely many ways to define the reward function under which a desired policy is optimal. Practically, however, these rewards often result in drastically different learning processes. For example, many major successes of RL required meticulous engineering of the reward: by hand (Berner et al., 2019) or by learning it from a human example (Vinyals et al., 2019). Hence, designing a reward function that enables learning and corresponds to a certain optimal policy is a challenging problem in modern reinforcement learning.

In many RL problems, the true reward function is *sparse*, i.e., only successful completion of the task is rewarded. In particular, the sparse reward is characteristic to *goal-reaching* problems where the agent needs to enter the goal state (Plappert et al., 2018; Florensa et al., 2018; Ghosh et al., 2020). Sparse reward problems are notoriously hard to solve with standard RL. A popular and simple solution is to introduce a dense surrogate reward that represents some sort of distance between the agent and the goal (Towers et al., 2023; de Lazcano et al., 2023). However, this approach is very sensitive and should be carefully tailored to each problem individually, in order to not change the induced optimal policy. Specifically, this dense artificial dense reward should *a)* increase when the agent gets closer to the goal, and *b)* not distract the agent from the reaching the goal. Designing a function that satisfies both criteria can be tricky for a human expert, as it requires estimating the (discounted) cumulative returns in various states.

In this work, we propose *max-reward RL*, where the agent optimizes the maximum reward achieved in the episode rather than the cumulative return. This paradigms makes the reward design process much more intuitive and straightforward, as it only requires that "better" states correspond to larger rewards. Hence, as long as the goal-reaching action has the highest reward, the optimal policy does not change. Besides simplifying the reward design, the maximum reward objective can also be easier to optimize for. In standard RL, learning a value of a non-terminal state involves bootstrapping, and hence has a moving target. In max-reward RL, bootstrapping does not happen when the immediate reward is not smaller than the largest reward explored so far. Therefore, max-reward RL bootstraps less and hence, potentially, learns better.

One of the key properties of the cumulative return is that it satisfies the Bellman equation (Bellman, 1954) and hence can be efficiently approximated and optimized by iteratively applying the Bellman operator. To make the max-reward RL approach viable, an analogous learning rule is required. However, Cui & Yu (2023) prove that naively changing summation into a max operator in the standard Bellman

[1]Delft University of Technology. Correspondence to: Grigorii Veviurko <g.veviurko@tudelft.nl>.

*Proceedings of the 41st International Conference on Machine Learning*, Vienna, Austria. PMLR 235, 2024. 

update rule works *only* in a deterministic setting and hence cannot be used in most RL problems and algorithms.

Inspired by results from stochastic optimal control theory (Kröner et al., 2018), this paper introduces a theoretically justified framework for max-reward RL in the general stochastic setting. We introduce a Bellman-like equation, prove the stochastic and deterministic policy gradient theorems, and reformulate some of the state-of-the-art algorithms (PPO, TD3) for the max-reward case. Using the Maze environment (de Lazcano et al., 2023) with different surrogate dense rewards, we experimentally demonstrate that max-reward algorithms outperform their cumulative counterparts. Finally, experiments with a challenging Fetch environment (de Lazcano et al., 2023) show the promise of max-reward RL in more realistic goal-reaching problems.

## 2. Related work

The first attempt to formulate max-reward RL was made by Quah & Quek (2006), where the authors derived a learning rule for the maximum reward state-action value function. However, as it was shown later (Gottipati et al., 2020), that work made a technical error of interchanging expectation and maximum operators. Gottipati et al. (2020) corrected this error, but the value functions learned via their approach differ from the expected maximum reward if stochasticity is present. Independently, Wang et al. (2020) derived a similar method in the context of planning in deterministic Markov Decision Processes (MDPs). Later, Cui & Yu (2023) demonstrated that the presence of stochasticity poses a problem not only for the max-reward RL but also for other non-cumulative rewards.

There exists a parallel branch of research that (re)discovered maximum reward value functions in the context of safe RL for reach-avoid problems (Fisac et al., 2014). In their work, Fisac et al. (2019) considered a deterministic open-loop dynamic system, where the agent's goal is to avoid constraint violations. The authors derived a contraction operator, similar to the one by Gottipati et al. (2020), to learn the max-cost safe value function. Hsu et al. (2021) extended this approach to reach-avoid problems, where the goal is to reach the goal while not violating constraints. Later, max-cost value functions were utilized within the safe RL context to learn the best-performing policy that does not violate the constraints (Yu et al., 2022). The main limitation of the three aforementioned works is the same as for Gottipati et al. (2020) – their methods only apply to deterministic environments and policies.

Effective reward design is a long-standing challenge in reinforcement learning which dates back to at least as early as 1994 (Mataric, 1994). In this paragraph, we briefly summarize the existing work related to the reward design problem. For further reading, we refer the reader to Eschmann (2021). Some of the big successes of RL utilize a hand-designed reward function, e.g., in the game of DOTA (Berner et al., 2019) or robots playing soccer (Haarnoja et al., 2023). However, manually designed rewards often lead to undesirable behavior (Krakovna et al., 2020). Alternatively, the reward can be designed in an automated fashion. For example, based on state novelty to encourage exploration (Tang et al., 2017; Pathak et al., 2017; Burda et al., 2018), by learning it from the experiences (Trott et al., 2019), or by using human data (Ibarz et al., 2018).

To conclude, reward design and reward shaping remain challenging topics. In this work, we propose a new way to think about the reward – the max-reward RL framework. While self-sufficient in some cases, this approach can also be combined with various existing methods for reward design.

## 3. Background

We consider a standard reinforcement learning setup for continuous environments. An agent interacts with an MDP defined by a tuple $(\mathcal{S}, \mathcal{A}, R, P, p_0, \gamma)$, where $\mathcal{S}$ is the continuous *state space*, $\mathcal{A}$ is the continuous *action space*, and $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, \bar{R}]$ is a non-negative and bounded *reward function*.[1] For each state-action pair, $(s, a) \in \mathcal{S} \times \mathcal{A}$, the transition function $P(\cdot|s, a) \in \mathcal{P}(\mathcal{S})$ is a probability density function (PDF) of the next state $s'$ and $p_0(\cdot) \in \mathcal{P}(\mathcal{S})$ is the PDF of the initial state $s_0$. Scalar $0 \leq \gamma < 1$ is the *discount factor*. We use $\pi : \mathcal{S} \to \mathcal{P}(\mathcal{A})$ to denote a stochastic policy and $\mu : \mathcal{S} \to \mathcal{A}$ to denote a deterministic policy. The time is discrete and starts at zero, i.e., $t \in \mathbb{N} \cup \{0\}$. For each timestep $t$, the state is denoted by $s_t$, the action by $a_t$, and the reward by $r_{t+1} := R(s_t, a_t, s_{t+1})$. Everywhere in the text, the expectation over policy, $\mathbb{E}_\pi$, denotes the expectation over the joint distribution of $s_t, a_t, r_{t+1}$ for $t \in \mathbb{N} \cup \{0\}$ induced by $\pi, P$, and $p_0$. Sometimes, we use such notation as $\mathbb{E}_{x\sim\pi}$ (or just $\mathbb{E}_x$) to emphasize that the expectation is taken only over $x$.

In standard RL, the main quantity being optimized is the *cumulative return*, defined as $G_t = \sum_{i=0}^{\infty} \gamma^i r_{t+1+i}$. To maximize $\mathbb{E}_\pi\big[G_t\big]$, most RL algorithms learn state and/or state-action value functions defined as follows:

$$v^\pi(s) = \mathbb{E}_\pi\big[G_t \big| s_t = s\big], \quad v^*(s) = \max_\pi v^\pi(s).$$

$$q^\pi(s, a) = \mathbb{E}_\pi\big[G_t \big| \substack{s_t = s \\ a_t = a}\big], \quad q^*(s, a) = \max_\pi q^\pi(s, a).$$

Crucially, these functions are solutions to the corresponding *Bellman equations*:

$$v^\pi(s) = \mathbb{E}_{\substack{a_t \\ s_{t+1}}}\big[r_{t+1} + \gamma v^\pi(s_{t+1}) \big| s_t = s\big]$$

[1] Non-negativity of reward can be achieved in any MDP with bounded reward function.

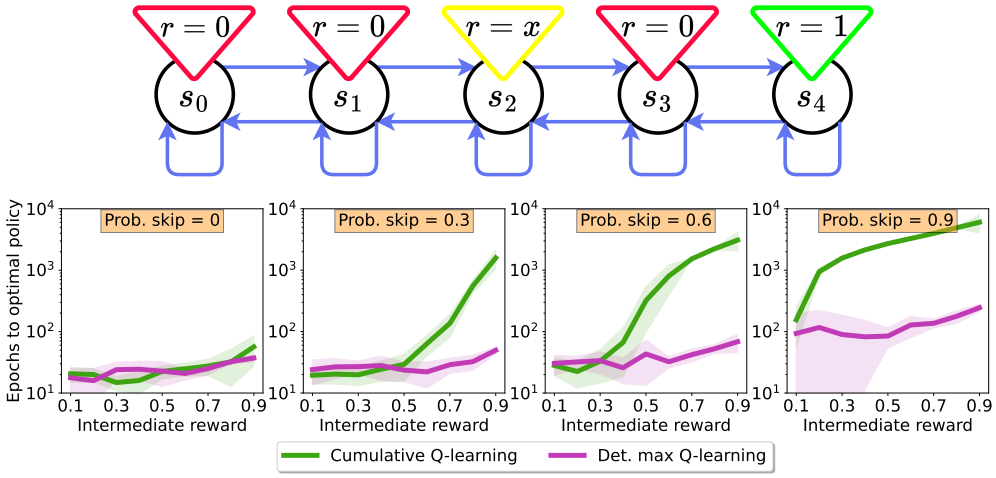


*Figure 1.* Five-state chain MDP with three actions (*left, stay, right*) available in each state and the training results for cumulative (in green) and max-reward (in violet) value iteration. The $y-$ axis is the number of training epochs to recover the optimal policy; the $x-$axis shows the values of the intermediate reward $x$. Four panels correspond to different probabilities of skipping transitions into $s_4$ during training.

$$q^\pi(s,a) = \mathbb{E}_{\substack{s_{t+1}\\a_{t+1}}}\left[r_{t+1} + \gamma q^\pi(s_{t+1}, a_{t+1})\middle|_{a_t=a}^{s_t=s}\right]$$

$$q^*(s,a) = \mathbb{E}_{s_{t+1}}\left[r_{t+1} + \gamma \max_{a'} q^*(s_{t+1}, a')\middle|_{a_t=a}^{s_t=s}\right]$$

The defining feature of these equations is that they can be solved by repeatedly applying *Bellman operators*. These operators are contractions and hence each of them has a unique fixed point that corresponds to one of the value functions above. For example, the optimal state-action value function $q^*(s,a)$ is the fixed point of the *Bellman optimallity operator* $\mathcal{T}^*$:

$$(\mathcal{T}^* q)(s,a) = \mathbb{E}_{s_{t+1}}\left[r_{t+1} + \gamma \max_{a'} q(s_{t+1}, a')\middle|_{a_t=a}^{s_t=s}\right] \quad (1)$$

The Bellman equation is foundational for all state-of-the-art RL algorithms as it allows training neural networks to approximate value functions. Therefore, for the max-reward framework to be useable, it is necessary to derive an analog of the Bellman equation. Below, we describe such an attempt made by Gottipati et al. (2020) and demonstrate that it is limited to purely deterministic problems.

### 3.1. Deterministic max-reward RL

Instead of cumulative return, max-reward RL aims at optimizing the *max-reward return*:

$$\hat{G}_t = \max\left\{r_{t+1}, \gamma r_{t+2}, \gamma^2 r_{t+3} \dots\right\} \quad (2)$$

Similarly to cumulative returns, $\hat{G}_t$ uses the discount factor $\gamma$ which is necessary for learning with Bellman-like updates, as we show later. To approximate $\mathbb{E}_\pi\left[\hat{G}_t\right]$, Gottipati et al. (2020) introduced the following definition of the state-action value functions:

$$\hat{q}^\pi_{det}(s,a) = \mathbb{E}_{\substack{s_{t+1}\\a_{t+1}}}\left[r_{t+1} \vee \gamma q(s_{t+1}, a_{t+1})\middle|_{a_t=a}^{s_t=s}\right]$$

$$\hat{q}^*_{det}(s,a) = \mathbb{E}_{s_{t+1}}\left[r_{t+1} \vee \gamma \max_{a'} q(s_{t+1}, a')\middle|_{a_t=a}^{s_t=s}\right]$$

where $\vee$ denotes the binary max operator, i.e., $a \vee b := \max\{a, b\}$. By construction, $\hat{q}^*_{det}$ and $\hat{q}^\pi_{det}$ satisfy Bellman-like recursive equations. In their work, Gottipati et al. (2020) proved that the following operator is a contraction:

$$(\hat{\mathcal{T}}^*_{det} q)(s,a) = \mathop{\mathbb{E}}_{s_{t+1}}\left[r_{t+1} \vee \gamma \max_{a'} q(s_{t+1}, a')\middle|_{a_t=a}^{s_t=s}\right] \quad (3)$$

Therefore, $\hat{q}^*_{det}$ is the unique fixed point of $\hat{\mathcal{T}}^*_{det}$ and can be learned, e.g., with Q-learning.

**Chain environment example.** Before going into the limitations of the approach above, we conduct a simple experiment to motivate the use of max-reward reinforcement learning. We show that max-reward RL is a better approach in a goal-reaching problem where the agent needs to learn to reach the goal state. Specifically, it dominates the standard cumulative RL when transitions into the goal state occur infrequently in the training data, which is often the case in larger-scale goal-reaching problems.

Consider the five-state chain environment in Figure 1. Transitions leading into $s_4$ have reward of 1, transitions into $s_2$ have a reward parametrized with $x \in (0,1)$, and other rewards are zero. Hence, the optimal policy, concerning both max-reward and cumulative returns, is to go to $s_4$ and stay there. We run tabular $Q-$value iteration algorithm using

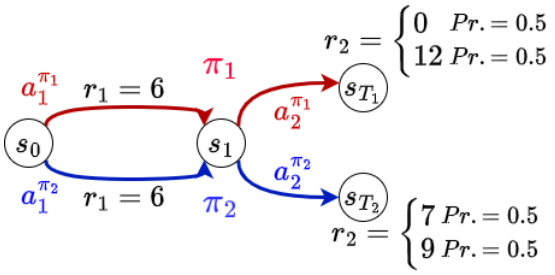


*Figure 2.* A three-state MDP with deterministic transitions and stochastic rewards. Two different policies, $\pi_1$ and $\pi_2$, share the same first action $a_1$, but then have different $a_2$, thereby resulting in different reward distributions.

standard (Eq. (1)) and max-reward (Eq. (3)) Bellman operators for different values of the intermediate reward $x$. In each training epoch, we iterate over all possible transitions. For each transition, we compute the target value using one of the Bellman operators and update the Q-table. Crucially, we *randomly skip some of the transitions into $s_4$ with a certain probability*. In the experiment, we consider four values for the skip probability – $p_{skip} \in \{0, 0.3, 0.6, 0.9\}$. During training, when a transition into $s_4$ is sampled, the Q-table is updated with probability $1 - p_{skip}$ and otherwise left unchanged. Transitions into other states are never skipped. In this way, we can control how often the agent is exposed to the transitions into the optimal state and thereby simulate problems where goal-reaching transitions are rarely encountered.

The results in Figure 1 indicate that for larger values of the skip probability, the max-reward approach converges to the optimal policy significantly faster than the cumulative approach. We believe that this phenomenon can be explained by differences in bootstrapping. In standard RL, the target for the $q-$value is a sum of the immediate reward and the $q-$value at the next timestep. Therefore, this target changes in each epoch until convergence. In the max-reward case, on the other hand, the target in the max-reward state is just the reward and does not change with time. This example suggests that the max-reward approach is a better choice in environments where the task of the agent is to reach the goal state.

**Issues when stochasticity is present**. Unfortunately, the max-reward approach described above has a serious theoretical drawback. Expanding the definition of $\hat{q}^*_{det}$ for more timesteps, we obtain a nested sequence of non-interchangeable $\vee$ and $\mathbb{E}$:

$$\hat{q}^*_{det}(s,a) = \mathbb{E}_{\pi^*}\left[r_{t+1} \vee \gamma \mathbb{E}_{\pi^*}\left[r_{t+2} \vee \dots\right] \Big|^{s_t=s}_{a_t=a}\right]$$

Using Jensen's inequality (Jensen, 1906), we conclude the following:

$$\hat{q}^*_{det}(s,a) \leq \mathbb{E}_{\pi^*}\left[\hat{G}_t \big|^{s_t=s}_{a_t=a}\right] \tag{4}$$

When both the policy and the transition model are deterministic, Eq. (4) becomes an equality. However, if stochasticity is present, the value of $\hat{q}^*_{det}(s,a)$ is merely a lower bound of the expected return. Hence, it can induce suboptimal policies.

In Figure 2, we show an example where the policy maximizing $\hat{q}^*_{det}$ is suboptimal. The figure demonstrates a three-state MDP and two policies, $\pi_1$ (red arrows) and $\pi_2$ (blue arrows). Let $\gamma = 1$ for simplicity. For the state $s_0$, the expected max-reward return is higher for the policy $\pi_1$ :

$$\mathbb{E}_{\pi_1}[\hat{G}_0] = \mathbb{E}_{\pi_1}[r_1 \vee r_2] = 9 > \mathbb{E}_{\pi_2}[\hat{G}_0] = \mathbb{E}_{\pi_2}[r_1 \vee r_2] = 8$$

So $\pi_1$ is better in terms of the expected max-reward return, but the value functions have the following values:

$$\hat{q}^{\pi_1}_{det}(s_0) = \mathbb{E}_{\pi_1}\left[r_1 \vee \mathbb{E}_{\pi_1}[r_2]\right] = \mathbb{E}_{\pi_1}[r_1 \vee 6] = 6$$

$$\hat{q}^{\pi_2}_{det}(s_0) = \mathbb{E}_{\pi_2}\left[r_1 \vee \mathbb{E}_{\pi_2}[r_2]\right] = \mathbb{E}_{\pi_2}[r_1 \vee 8] = 8$$

Based on the values of $\hat{q}^{\pi}_{det}$, we would conclude that $\pi_2$ is better, which we already showed to be incorrect. This example demonstrates that even in a simple stochastic environment, the operator $\hat{T}^*_{det}$ can lead to incorrect policies. Therefore, it is an open question whether there exists a Bellman-like operator that would enable learning max-reward returns in the stochastic setting.

## 4. Max-reward RL

In this section, we introduce a novel approach to max-reward RL that is theoretically sound, works for both stochastic and deterministic cases, and can be combined with state-of-the-art RL algorithms. First, we expand the definition of the max-reward return given in Eq. (2):

$$\mathbb{E}_{\pi}[\hat{G}_t] = \mathbb{E}_{\pi}\left[r_{t+1} \vee \gamma \hat{G}_{t+1}\right] \tag{5}$$

Since $\mathbb{E}$ and $\vee$ do not commute, it is impossible to extract the term $\mathbb{E}_{\pi}\left[G_{t+1}\right]$ on the right-hand side of Eq. (5). Because of that, we cannot obtain an equation involving only $\mathbb{E}_{\pi}\left[G_t\right]$, $\mathbb{E}_{\pi}\left[G_{t+1}\right]$, and $r_{t+1}$. Instead, we will utilize an approach from stochastic optimal control theory (Kröner et al., 2018) and define the max-reward value function using an auxiliary variable that allows propagating information between timesteps:

**Definition 4.1.** Let $y \in \mathbb{R}$ be an auxiliary real variable. The *max-reward value functions* are defined as follows:

$$\hat{v}^{\pi}(s,y) = \mathbb{E}_{\pi}[y \vee \hat{G}_t \big|^{s_t=s}]$$

$$\hat{q}^{\pi}(s,a,y) = \mathbb{E}_{\pi}[y \vee \hat{G}_t \big|^{s_t=s}_{a_t=a}]$$

Since reward is lower-bounded, $r_{t+1} \geq 0$, we can always recover the expected max-reward return $\mathbb{E}_\pi\left[\hat{G}_t\right]$ by substituting $y = 0$ into the value functions:

$$\begin{aligned} \hat{v}^\pi(s,0) &= \mathbb{E}_\pi[\hat{G}_t \big| s_t{=}s] \\ \hat{q}^\pi(s,a,0) &= \mathbb{E}_\pi[\hat{G}_t \big| {}^{s_t=s}_{a_t=a}] \end{aligned} \tag{6}$$

Hence, if we find an efficient method of learning the max-reward value functions, we will be able to optimize $\mathbb{E}_\pi[\hat{G}_t]$.

The auxiliary variable $y$ is crucial when dealing with the max-reward returns. When we look at the value of the state $s'$ from the perspective of state $s$, we must consider the immediate reward $r = r(s,a,s')$. Specifically, we should treat low reward trajectories from $s'$ as if they still yield the reward of $r$. Expanding upon this observation, we conclude that maximization of the maximum reward requires propagating information about the past rewards. This is achieved via the auxiliary variable $y$.

By combining the definition of the max-reward value functions with Eq. (5), we obtain the following recursive equations:

**Lemma 4.2.** *Let $y \in \mathbb{R}$ and let $y' := \frac{R(s,a,s_{t+1}) \vee y}{\gamma}$. Then, the max-reward value functions are subject to the following Bellman-like equations:*

$$\hat{v}^\pi(s,y) = \gamma \mathbb{E}_{\substack{a_t \\ s_{t+1}}}\left[y' \vee \hat{v}^\pi(s_{t+1}, y') \big| s_t{=}s\right]$$

$$\hat{q}^\pi(s,a,y) = \gamma \mathbb{E}_{\substack{s_{t+1} \\ a_{t+1}}}\left[y' \vee \hat{q}^\pi(s_{t+1}, a_{t+1}, y') \big| {}^{s_t=s}_{a_t=a}\right]$$

Proof of this lemma, as well as all other proofs, can be found in Appendix A. The extra term $y' \vee$ might seem redundant, but it is important since it enforces the boundary conditions. Without it, the functions $v \equiv 0$ and $q \equiv 0$ would be solutions to these equations. Using Lemma 4.2, we can define Bellman-like operators for the max-reward value functions:

**Definition 4.3.** Let $v : \mathcal{S} \times \mathbb{R} \to \mathbb{R}$, $q : \mathcal{S} \times \mathcal{A} \times \mathbb{R} \to \mathbb{R}$ be real-valued functions and let $y' := \frac{R(s,a,s_{t+1}) \vee y}{\gamma}$. Then, the *max-reward Bellman operator* $\hat{\mathcal{T}}^\pi$ is defined as follows:

$$\hat{\mathcal{T}}^\pi v(s,y) := \gamma \mathbb{E}_{\substack{a_t \\ s_{t+1}}}\left[y' \vee v(s_{t+1}, y') \big| s_t{=}s\right]$$

$$\hat{\mathcal{T}}^\pi q(s,a,y) := \gamma \mathbb{E}_{\substack{s_{t+1} \\ a_{t+1}}}\left[y' \vee q(s_{t+1}, a_{t+1}, y') \big| {}^{s_t=s}_{a_t=a}\right]$$

In the following theorem, we prove that this operator is a contraction and that the max-reward state and state-action value functions are its fixed points.

**Theorem 4.4.** *$\hat{\mathcal{T}}^\pi$ is a $\gamma-$contraction with respect to the $L_\infty$ norm, and $\hat{v}^\pi$(or $\hat{q}^\pi$) is its fixed point.*

Theorem 4.4 implies that the max-reward value functions can be learned in the same way as the standard value functions – by sampling from the environment and applying Bellman operators. In the next section, we define the objective function of the max-reward RL problem and discuss how the presence of the auxiliary variable $y$ impacts the notion of optimal policy.

### 4.1. Max-reward objective

Similarly to standard RL, the main objective in the max-reward RL problems is to maximize the expected (max-reward) return from the initial state, defined as follows:

$$\hat{J}(\pi) = \mathbb{E}_{s_0 \sim p_0}\left[\hat{v}^\pi(s_0, 0)\right] \tag{7}$$

Then, the optimal policy is naturally defined as :

$$\pi^* = \arg\max_\pi \hat{J}(\pi). \tag{8}$$

To better understand the properties of the max-reward optimal policy, consider again the MDP in Figure 2. Let $\gamma = 1$. Then, the values of the objective function for $\pi_1$ and $\pi_2$ can be computed as follows:

$$\hat{J}(\pi_1) = \mathbb{E}_{\pi_1}[6 \vee r_2] = 9$$

$$\hat{J}(\pi_2) = \mathbb{E}_{\pi_2}[6 \vee r_2] = 8$$

Hence, $\pi_1$ is optimal. However, if we consider the max-reward return from $t = 1$, we have

$$\mathbb{E}_{\pi_1}[G_1] = \frac{12+0}{2} = 6 \quad \mathbb{E}_{\pi_2}[G_1] = \frac{9+7}{2} = 8$$

and hence $\pi_2$ obtains higher expected max-reward return starting at $s = s_1$. Seemingly, there is a contradiction: $\pi_1$ is optimal but $\pi_2$ is better from the state $s_1$. However, the explanation is simple: the maximum reward is the highest reward encountered anywhere along the trajectory. An optimal decision thus not only depends on the current state, as with the cumulative reward, but also on the maximum reward that has been acquired thus far. In the example, if we start from $s_1$, then we haven't encountered any reward yet. Hence, following $\pi_1$, we will have $r_2 = 0$ as the maximum reward half of the time. If we start from $s_0$, we receive a reward of $r_1 = 6$ when going to $s_1$. Then, the maximum reward will not be lower than 6, even if we get $r_2 = 0$. Thus, we conclude:

*In max-reward RL, the optimal policy $\pi^*$ maximizing $\hat{J}(\cdot)$ should depend not only on the current state, but also on the rewards obtained so far.*

To formalize this observation, we introduce additional notation. We define the *extended state space* as $\hat{\mathcal{S}} := \mathcal{S} \times \mathbb{R}$ and we denote extended states by $\hat{s} = (s,y)$, $s \in \mathcal{S}$, $y \in \mathbb{R}$. Then, for an extended state $(s,y) \in \hat{\mathcal{S}}$ and for an action $a \in \mathcal{A}$, the *extended transition model* $\hat{P}(\cdot,\cdot|s,y,a)$ is a PDF over $(s',y') \in \hat{\mathcal{S}}$, defined as

$$\hat{P}(s',y'|s,y,a) = P(s'|s,a)\,\delta\Big(y' - \frac{R(s,a,s') \vee y}{\gamma}\Big)$$

where $\delta(\cdot)$ is the Dirac delta function. The initial distribution of $(s_0, y_0)$ is given by $\hat{p}_0(s_0, y_0) = p(s_0)\delta(y_0)$ thereby ensuring $y_0 \equiv 0$. Combining everything, we introduce the following definition:

**Definition 4.5.** Let $M = (\mathcal{S}, \mathcal{A}, R, P, p_0, \gamma)$ be an MDP. Then, the *extended max-reward MDP* is an MDP $\hat{M}$ given by the tuple $(\hat{\mathcal{S}}, \mathcal{A}, R, \hat{P}, \hat{p_0}, \gamma)$.

Essentially, the extended MDP defined above tracks the (inversely) discounted maximum reward obtained so far. For example, if the maximum reward so far is $r_1$, then the extended state at timestep $t$ is $(s_t, \frac{r_1}{\gamma^t})$. Hence, to improve $\hat{J}$, we need $r_{t+1} > \frac{r_1}{\gamma^t}$.

Using the notion of extended MDP, we can redefine policy for the max-reward RL:

**Definition 4.6.** Let $M$ be an MDP and let $\hat{M}$ be its induced extended max-reward MDP. Then, any policy $\hat{\pi}$ in $\hat{M}$ is an *extended max-reward policy.*

After we have defined optimality in the max-reward sense, we can introduce the max-reward Bellman optimality operator:

**Definition 4.7.** Let $q : \mathcal{S} \times \mathcal{A} \times \mathbb{R} \to \mathbb{R}$ be a real-valued function and let $y' := \frac{R(s,a,s_{t+1}) \vee y}{\gamma}$. Then, the *max-reward Bellman optimality operator* $\hat{\mathcal{T}}^*$ is defined as follows:

$$\hat{\mathcal{T}}^* q(s, a, y) := \gamma \mathbb{E}_{s_{t+1}} \left[ y' \vee \max_{a'} q(s_{t+1}, a', y') \middle|_{a_t = a}^{s_t = s} \right]$$

Similarly to $\hat{T}^\pi$, this operator is also a contraction:

**Theorem 4.8.** *$\hat{\mathcal{T}}^*$ is a $\gamma-$contraction with respect to the $L_\infty$ norm, and $\hat{q}^*$ is its fixed point.*

We have most of the pieces of the max-reward RL framework. We established that it operates on the extended max-reward MDP $\hat{M}$, where the extended states preserve information about the past rewards. Then, both the max-reward optimal and on-policy value functions can be learned by sampling transitions from $\hat{M}$. Therefore, all DQN-based methods (Mnih et al., 2013) can be used under the max-reward RL paradigm directly. However, most state-of-the-art RL algorithms utilize policies parametrized by neural networks. This is possible due to the policy gradient theorems (Sutton et al., 1999; Silver et al., 2014), as they allow estimating the objective function gradient with respect to the policy parameters via sampling. In the next section, we formulate and prove max-reward policy gradient theorems for both deterministic and stochastic extended max-reward policies.

### 4.2. Policy gradient theorems

First, we define $\hat{p}_t^{\hat{\pi}}(s_0, y_0, s, y)$ – the probability measure of arriving in the extended state $(s, y)$ after $t$ timesteps, starting from $(s_0, y_0)$ and executing the extended policy $\hat{\pi}$. Let

$$\hat{P}^{\hat{\pi}}(s', y'|s, y) = \int_a \hat{\pi}(a|s, y) \hat{P}(s', y'|s, y, a) da$$

be the "on-policy" transition model. Then, $\hat{p}_t^{\hat{\pi}}(s_0, y_0, s, y)$ is defined as follows:

$$\hat{p}_0(s_0, y_0, s, y) = \delta(s - s_0)\delta(y - y_0)$$

$$\hat{p}_t^{\hat{\pi}}(s_0, y_0, s, y) = \int_{\tilde{s}, \tilde{y}} \hat{p}_{t-1}^{\hat{\pi}}(s_0, y_0, \tilde{s}, \tilde{y}) \, \hat{P}^{\hat{\pi}}(s, y|\tilde{s}, \tilde{y}) \, d\tilde{s} \, d\tilde{y}$$

The discounted stationary state distribution of an extended max-reward MDP is then given by

$$\hat{d}^{\hat{\pi}}(s, y) = \int_{s_0, y_0} \hat{p}_0(s_0, y_0) \sum_{t=0}^{\infty} \gamma^t \, \hat{p}_t^{\hat{\pi}}(s_0, y_0, s, y) \, ds_0 \, dy_0.$$

As such, $\hat{d}^{\hat{\pi}}$ is not a distribution. However, it can be normalized into one by dividing it by $C = \int_{s,y} \hat{d}^{\hat{\pi}}(s, y) \, ds \, dy$.

Finally, we can formulate and prove the max-reward policy gradient theorems. Consider a neural network with weights $\theta$ that represents a stochastic policy. Then, we have the following result:

**Theorem 4.9.** *Let $\hat{\pi}_\theta : \mathcal{S} \times \mathbb{R} \to \mathcal{P}(\mathcal{A})$ be a stochastic extended max-reward policy parameterized with $\theta$. Then, the following holds for $\nabla_\theta \hat{J}(\theta)$:*

$$\nabla_\theta \, \hat{J}(\theta) \propto \mathbb{E}_{\substack{(s,y) \sim \hat{d}^{\hat{\pi}} \\ a \sim \hat{\pi}_\theta}} \left[ \hat{q}^{\hat{\pi}_\theta}(s, a, y) \nabla_\theta \ln \hat{\pi}_\theta(a|s, y) \right]$$

The deterministic max-reward policy gradient follows from the stochastic version:

**Corollary 4.10.** *Let $\hat{\mu}_\theta : \mathcal{S} \times \mathbb{R} \to \mathcal{A}$ be a deterministic extended max-reward policy parameterized with $\theta$. Then $\nabla_\theta \hat{J}(\theta)$ can be computed as follows:*

$$\nabla_\theta J(\theta) \propto \mathbb{E}_{\hat{d}^{\hat{\mu}}} \left[ \nabla_\theta \hat{\mu}_\theta(s, y) \nabla_a \hat{q}^{\hat{\mu}_\theta}(s, a, y)|_{a = \mu_\theta(s,y)} \right]$$

The policy gradient theorems allow us to use various algorithms from standard RL, such as REINFORCE (Williams, 1992), A2C (Sutton et al., 1999), A3C (Mnih et al., 2016),

*Figure 3. Left:* Single-goal maze, where the goal (red ball) is always in the same location. *Right:* Two-goals maze with two spawn locations of the goal (red balls).

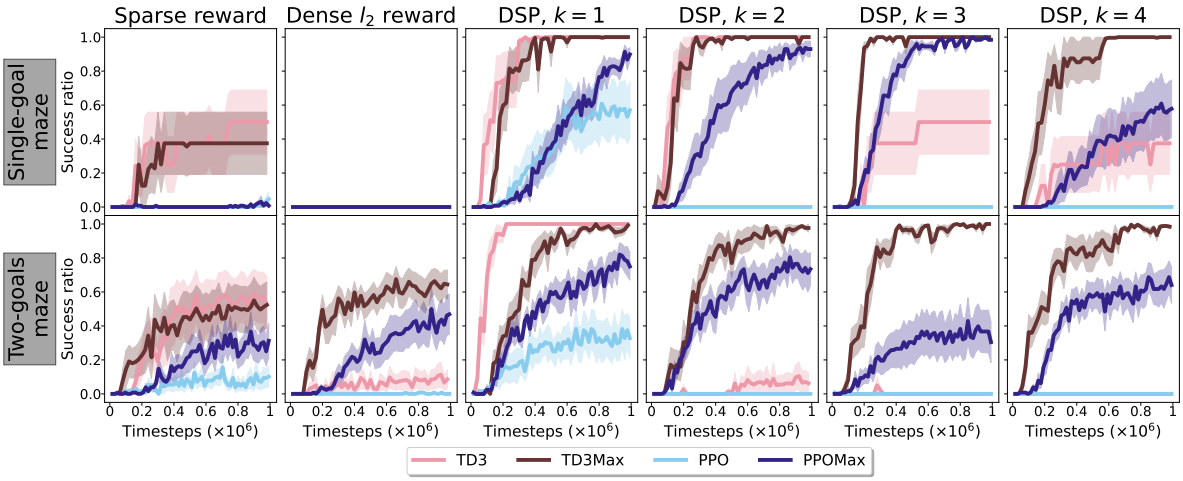


*Figure 4.* Learning curves of TD3, max-reward TD3, PPO, and max-reward PPO on two different mazes. The vertical axis is the success ratio, i.e., whether the goal was reached during the episode. The shaded area is the standard error of the mean. The horizontal axis is the total environmental timesteps in millions. For each maze, we present results for six different reward functions (columns).

TRPO (Schulman et al., 2015), PPO (Schulman et al., 2017), DDPG (Lillicrap et al., 2016), and TD3 (Fujimoto et al., 2018), to optimize maximum rewards. In this work, we focus on PPO and TD3, as they are considered to be the best-performing algorithms within their corresponding families. For max-reward PPO, the only difference compared to the standard version is that the advantage estimation uses max-reward returns. For max-reward TD3, the target value for the $Q$ functions is computed using the max-reward Bellman optimality operator (4.7). In Appendix, we provide descriptions of max-reward TD3 and PPO in pseudocode.

## 5. Experiments

To empirically evaluate the benefits of using maximum instead of cumulative reward, we compare the max-reward TD3 and PPO with their cumulative counterparts using two goal-reaching environments from Gymnasium-Robotics (de Lazcano et al., 2023) under different dense reward functions.

### 5.1. Maze with shortest path rewards

First, we consider the Maze environment from Gym Robotics (de Lazcano et al., 2023) illustrated in Figure 3, where the agent controls a ball by applying acceleration in two dimensions. The objective is to reach the goal position in the maze. Episodes last 1000 timesteps and there are no terminal states. We use two mazes: *single-goal* maze, where the goal is always in the same location, and the *two-goals* maze where at each episode the goal location is chosen randomly from the two possible options. The main metric in this environment is *success ratio* – a binary value indicating whether the goal was reached during the episode.

We consider several reward functions that induce the same optimal policy of reaching the goal state:

1. *Sparse reward* – only reaching the goal is rewarded with $r = 1$.
2. *Dense $l_2$ reward* – default dense reward, defined as the exponent of the negative of the $l_2$-distance to the goal. Reaching the goal is rewarded with $r = 1$.
3. *Discrete shortest path (DSP)* – our custom reward that represents the true, topology aware distance to the goal. To compute it, the maze is split into $n \times m$ cells. Then, the distance matrix $D \in \mathbb{R}^{n \times m}$ is computed such that for each cell $(i, j)$, $D[i, j]$ is the number of cells between $(i, j)$ and the goal-containing cell. The DSP reward with parameter $k \in \mathbb{N}$ is then defined as

$$r^k_{dsp}(i, j) = \begin{cases} \beta^{D[i,j]+1}, & \text{if } D[i, j] = 0 \mod k \\ 0, & \text{otherwise} \end{cases}$$

where $\beta \in (0, 1)$ is a hyperparameter. The value of $k$ controls the sparsity of the reward, i.e., for larger $k$ fewer cells have a non-zero reward. Reaching the goal is rewarded with $r = 1$.

For the DSP reward, we first tune the value of $\beta$ by running standard TD3 and PPO on the single-goal maze. We set $k = 1$ and run 10 random seeds for each algorithm for $\beta \in \{0.65, 0.7, 0.75, 0.8, 0.85, 0.9, 0.95\}$. Additionally, we test the negative version of the DSP reward, $r^k_{dsp}(i, j) - 1$, which, in theory, should cause better exploration. For TD3, the best-performing reward was the negative DSP with $\beta_{TD3} = 0.9$,

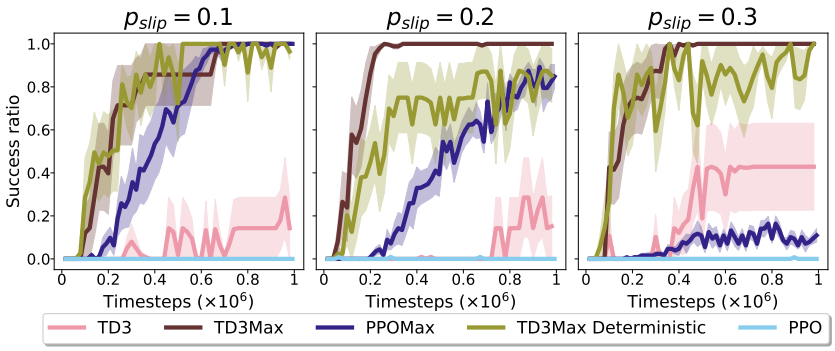


*Figure 5.* Learning curves of TD3, max-reward TD3, deterministic max-reward TD3, PPO, and max-reward PPO on a stochastic version of the single-goal maze with DSP reward, $k = 3$. The vertical axis is the success ratio, the shaded area is the standard error of the mean. The horizontal axis is the total environmental timesteps. The results confirm that our max-reward methods work in stochastic environments.

and for PPO – negative DSP with $\beta_{PPO} = 0.95$. In all other runs involving DSP reward we use these values of $\beta$.

Finally, we compare TD3, PPO, max-reward TD3, and max-reward PPO on the single-goal and two-goals mazes using sparse, dense $l_2$, and DSP reward for $k = 1, 2, 3, 4$ (for cumulative methods, negative DSP reward is used). Figure 4 demonstrates the learning curves. The sparse reward performs inconsistently due to insufficient exploration. Dense $l_2$ reward has local maximums (especially in the single-goal maze) and its performance greatly depends on the maze topology. The DSP reward, which represents the true distance to the goal, overall performs better.

Importantly, we see that the max-reward approaches work for all values of $k$, while the standard RL methods do not. For larger $k$, the reward becomes sparser, and cumulative approaches tend to converge to suboptimal policies. We believe that the nature of this phenomenon is the same as in the chain environment example discussed in Section 3.1. Specifically, the Maze environment can be seen as a larger chain with multiple intermediate rewards. During training, all methods quickly learn to stay in one of the cells with non-zero reward. Then, to update the policy, samples of transitions to a better state are needed. For larger $k$, these transitions become less frequent, as the cells with non-zero reward become further from one another. In line with the chain environment results, max-reward methods require fewer such transitions and therefore perform more efficiently.

Another potential reason for the superiority of max-reward methods lies in the way how they handle local optima. Since max-reward policy is conditioned on the discounted max-reward so far, $y$, it has no incentive to stay in the local optima. As $y$ "remembers" the reward at a local optimum, any trajectory leaving this optimum is at least as good as staying in the optimum. Combined with exploration techniques, e.g., entropy regularization in PPO, this causes the agent to leave local optima after visiting them.

**Stochastic Maze.** One of the strengths of our RL formulation is that it works with stochastic environments and/or policies. To experimentally verify that, we conduct an additional experiment using a stochastic variation of the single-goal maze. Specifically, we introduce a parameter $p_{slip}$ which regulates the level of stochasticity. Whenever the agent makes an action, it is replaced with a random action with probability $p_{slip}$. We compare max-reward and standard versions of TD3 and PPO on this environment. Additionally, we implement and test *deterministic* max-reward TD3 (Gottipati et al., 2020). In this experiment, we use the DSP reward with $k = 3$, as it is a case where the max-reward paradigm demonstrates improvement over standard RL in a deterministic Maze. The results presented in Figure 5 confirm the theory: our max-reward TD3 solves this stochastic environment while the deterministic max-reward TD3 is highly inconsistent. Therefore, we conclude that our method indeed can be used for stochastic environments.

### 5.2. Fetch environment

In the second experiment, we consider more challenging robotics problems. Specifically, we study the *Fetch-Slide* and *Fetch-Push* environments depicted in Figure 6. The agent controls a 7-DoF manipulator and its goal is to move the puck into the target location. In *Fetch-Slide*, the goal is located beyond agent's reach and hence it needs to slide the puck into the goal. In *Fetch-Push*, the agent needs to push the puck into the goal which can be anywhere on the platform. Each episode is truncated after 100 timesteps and there are no terminal states. We use the standard dense reward for this problem defined as negative of the $l_2-$distance between the puck and the goal. The performance metric for this environment is again the success ratio – a binary value that indicates whether the goal was reached during

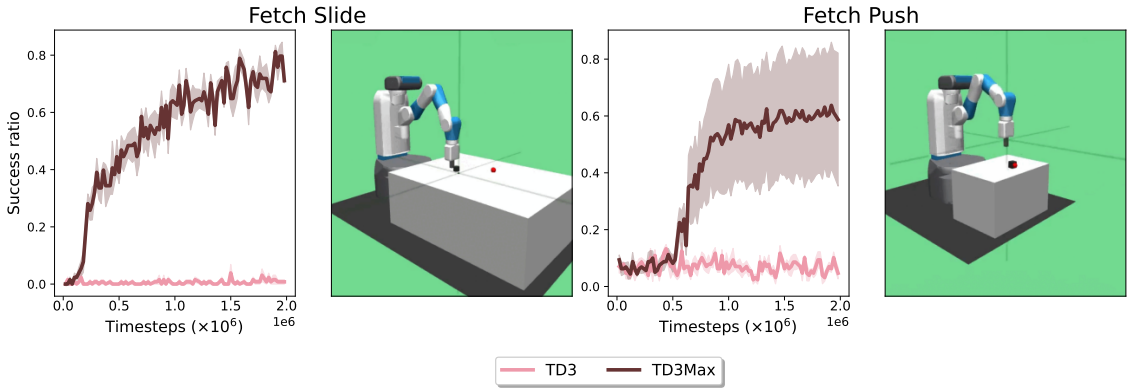


*Figure 6.* Success ratio for standard (light red) and max-reward (dark-red) TD3 in *Fetch Slide* (left) and *Fetch Push* (right) environments.

the episode. This environment is known to be challenging for standard RL and it cannot be solved without special approaches (Plappert et al., 2018).

The plot in Figure 6 demonstrates that the max-reward TD3 achieves a goal-reaching policy in both environments, while the standard version, in line with the prior work, fails to learn completely. We believe that this happens due to the difference in bootstrapping mentioned earlier. The environment is complex and multidimensional and the goal-reaching transitions are rare which makes the learning problem really hard for the standard methods. We believe that this experiment shows the great potential of max-reward RL in more realistic goal-reaching environments.

## 6. Conclusions and future work

In this work, we provide a theoretical description of the max-reward reinforcement learning paradigm and verify it experimentally. Our theoretical contributions include a novel formulation of the max-reward value functions and a Bellman-like contraction operator that enables efficient learning. Besides, we prove the policy gradient theorems for max-reward policies and hence enable using the state-of-the-art RL algorithms in the context of max-reward RL.

In the experiments with two robotic environments, we show that max-reward RL works better for sparse reward problems with surrogate dense reward. This result confirm our intuition that maximum reward is a better choice for goal-reaching environments. Moreover, we demonstrate that our max-reward RL, unlike prior work, is also consistent in stochastic environments.

Qualitatively, we believe that the main strengths of the max-reward algorithms can be summarized as follows:

1. In max-reward RL, bootstrapping works differently than in the standard RL. Specifically, it allows for more efficient propagation of reward from the goal states.
2. Max-reward agents are more prone to getting stuck in local optima. Since the maximum reward obtained so far is a part of the extended state space, the agents do not have any incentive to stay in these optima.
3. Due to the auxiliary variable $y$, max-reward value functions are inherently distributional. As reported in prior work (Bellemare et al., 2017), learning distributional value functions can have positive impact even in deterministic problems.

In future work, we aim to study how max-reward RL can be combined with the existing methods for automated reward design and explore its potential in other problems.

## Acknowledgements

We would like to acknowledge Delft University of Technology for providing the resources and support necessary for this research. The collaborative environment greatly contributed to the development and success of this work. In particular, we are very thankful to Jinke He for reading the draft of this paper and providing helpful feedback.

Additionally, we our thank the anonymous reviewers for their thorough and constructive feedback. Their insightful comments and suggestions have significantly improved the quality of this paper.

## Impact Statement

This paper presents work whose goal is to advance the field of Reinforcement Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

## References


Bellemare, M. G., Dabney, W., and Munos, R. A distributional perspective on reinforcement learning, 2017.

Bellman, R. Some applications of the theory of dynamic programming—a review. *Journal of the Operations Research Society of America*, 2(3):275–288, 1954.

Berner, C., Brockman, G., Chan, B., Cheung, V., Debiak, P., Dennison, C., Farhi, D., Fischer, Q., Hashme, S., Hesse, C., et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.

Burda, Y., Edwards, H., Storkey, A., and Klimov, O. Exploration by random network distillation, 2018.

Cui, W. and Yu, W. Reinforcement learning with non-cumulative objective. *IEEE Transactions on Machine Learning in Communications and Networking*, 2023.

de Lazcano, R., Andreas, K., Tai, J. J., Lee, S. R., and Terry, J. Gymnasium robotics, 2023. URL http://github.com/Farama-Foundation/Gymnasium-Robotics.

Eschmann, J. Reward function design in reinforcement learning. *Reinforcement Learning Algorithms: Analysis and Applications*, pp. 25–33, 2021.

Fisac, J. F., Chen, M., Tomlin, C. J., and Sastry, S. S. Reach-avoid problems with time-varying dynamics, targets and constraints, 2014.

Fisac, J. F., Lugovoy, N. F., Rubies-Royo, V., Ghosh, S., and Tomlin, C. J. Bridging Hamilton-Jacobi safety analysis and reinforcement learning. In *2019 International Conference on Robotics and Automation (ICRA)*, pp. 8550–8556, May 2019.

Florensa, C., Held, D., Geng, X., and Abbeel, P. Automatic goal generation for reinforcement learning agents, 2018.

Fujimoto, S., van Hoof, H., and Meger, D. Addressing function approximation error in actor-critic methods. In *Proceedings of Machine Learning Research (ICML)*, volume 80, pp. 1587–1596, 2018.

Ghosh, D., Gupta, A., Reddy, A., Fu, J., Devin, C., Eysenbach, B., and Levine, S. Learning to reach goals via iterated supervised learning, 2020.

Gottipati, S. K., Pathak, Y., Nuttall, R., Sahir, Chunduru, R., Touati, A., Subramanian, S. G., Taylor, M. E., and Chandar, S. Maximum reward formulation in reinforcement learning, 2020.

Haarnoja, T., Moran, B., Lever, G., Huang, S. H., Tirumala, D., Wulfmeier, M., Humplik, J., Tunyasuvunakool, S., Siegel, N. Y., Hafner, R., Bloesch, M., Hartikainen, K., Byravan, A., Hasenclever, L., Tassa, Y., Sadeghi, F., Batchelor, N., Casarini, F., Saliceti, S., Game, C., Sreendra, N., Patel, K., Gwira, M., Huber, A., Hurley, N., Nori, F., Hadsell, R., and Heess, N. Learning agile soccer skills for a bipedal robot with deep reinforcement learning, 2023.

Hsu, K.-C., Rubies-Royo, V., Tomlin, C. J., and Fisac, J. F. Safety and liveness guarantees through Reach-Avoid reinforcement learning. December 2021.

Ibarz, B., Leike, J., Pohlen, T., Irving, G., Legg, S., and Amodei, D. Reward learning from human preferences and demonstrations in atari, 2018.

Jensen, J. L. W. V. Sur les fonctions convexes et les inégalités entre les valeurs moyennes. *Acta mathematica*, 30(1):175–193, 1906.

Krakovna, V., Uesato, J., Mikulik, V., Rahtz, M., Everitt, T., Kumar, R., Kenton, Z., Leike, J., and Legg, S. Specification gaming: the flip side of ai ingenuity. 2020.

Kröner, A., Picarelli, A., and Zidani, H. Infinite horizon stochastic optimal control problems with running maximum cost. *SIAM J. Control Optim.*, 56(5):3296–3319, January 2018.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2016.

Mataric, M. J. Reward functions for accelerated learning. In *Machine learning proceedings 1994*, pp. 181–189. Elsevier, 1994.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., and Kavukcuoglu, K. Asynchronous methods for deep reinforcement learning. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pp. 1928–1937. PMLR, 2016.

Ng, A. Y., Harada, D., and Russell, S. Policy invariance under reward transformations: Theory and application to reward shaping. In *Icml*, volume 99, pp. 278–287. Citeseer, 1999.

Pathak, D., Agrawal, P., Efros, A. A., and Darrell, T. Curiosity-driven exploration by self-supervised prediction, 2017.

Plappert, M., Andrychowicz, M., Ray, A., McGrew, B., Baker, B., Powell, G., Schneider, J., Tobin, J., Chociej, M., Welinder, P., et al. Multi-goal reinforcement learning: Challenging robotics environments and request for research. *arXiv preprint arXiv:1802.09464*, 2018.

Quah, K. and Quek, C. Maximum reward reinforcement learning: A non-cumulative reward criterion. *Expert Systems with Applications*, 31(2):351–359, 2006. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2005.09.054.

Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. Trust region policy optimization. In *Proceedings of Machine Learning Research (ICML)*, volume 37, pp. 1889–1897, 2015.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017.

Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., and Riedmiller, M. Deterministic policy gradient algorithms. In *International conference on machine learning*, pp. 387–395. Pmlr, 2014.

Sutton, R. S., McAllester, D., Singh, S., and Mansour, Y. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999.

Tang, H., Houthooft, R., Foote, D., Stooke, A., Chen, X., Duan, Y., Schulman, J., Turck, F. D., and Abbeel, P. exploration: A study of count-based exploration for deep reinforcement learning, 2017.

Towers, M., Terry, J. K., Kwiatkowski, A., Balis, J. U., Cola, G. d., Deleu, T., Goulão, M., Kallinteris, A., KG, A., Krimmel, M., Perez-Vicente, R., Pierré, A., Schulhoff, S., Tai, J. J., Shen, A. T. J., and Younis, O. G. Gymnasium, March 2023. URL https://zenodo.org/record/8127025.

Trott, A., Zheng, S., Xiong, C., and Socher, R. Keeping your distance: Solving sparse reward tasks using self-balancing shaped rewards, 2019.

Vinyals, O., Babuschkin, I., Czarnecki, W. M., Mathieu, M., Dudzik, A., Chung, J., Choi, D. H., Powell, R., Ewalds, T., Georgiev, P., et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575 (7782):350–354, 2019.

Wang, R., Zhong, P., Du, S. S., Salakhutdinov, R. R., and Yang, L. Planning with general objective functions: Going beyond total rewards. *Advances in Neural Information Processing Systems*, 33:14486–14497, 2020.

Williams, R. J. Simple statistical gradient algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256, 1992.

Yu, D., Ma, H., Li, S. E., and Chen, J. Reachability constrained reinforcement learning. May 2022.

## A. Proofs

*Proof of Lemma 4.2.* First, we prove the equation for the state value function $\hat{v}^\pi$ :

$$\begin{aligned}\hat{v}^\pi(s,y) &= \mathbb{E}_\pi\big[y \vee \hat{G}_t \big| s_t{=}s\big] = \mathbb{E}_\pi\big[y \vee r_{t+1} \vee \gamma\hat{G}_{t+1}\big| s_t{=}s\big] \\ &= \gamma\mathbb{E}_\pi\big[y' \vee \hat{G}_{t+1}\big| s_t{=}s\big] = \gamma\mathbb{E}_\pi\big[y' \vee y' \vee \hat{G}_{t+1}\big| s_t{=}s\big] \\ &= \gamma\mathbb{E}_{\substack{a_t \\ s_{t+1}}}\big[y' \vee \hat{v}^\pi(s_{t+1},y')\big| s_t{=}s\big]\end{aligned}$$

Then, for the state-action value function $\hat{q}^\pi$ :

$$\begin{aligned}\hat{q}^\pi(s,a,y) &= \mathbb{E}_\pi\big[y \vee \hat{G}_t \big|_{a_t=a}^{s_t=s}\big] = \mathbb{E}_\pi\big[y \vee r_{t+1} \vee \gamma\hat{G}_{t+1}\big|_{a_t=a}^{s_t=s}\big] \\ &= \gamma\mathbb{E}_\pi\big[y' \vee \hat{G}_{t+1}\big|_{a_t=a}^{s_t=s}\big] = \gamma\mathbb{E}_\pi\big[y' \vee y' \vee \hat{G}_{t+1}\big|_{a_t=a}^{s_t=s}\big] \\ &= \gamma\mathbb{E}_{\substack{s_{t+1} \\ a_{t+1}}}\big[y' \vee \hat{q}^\pi(s_{t+1}a_{t+1},y')\big|_{a_t=a}^{s_t=s}\big]\end{aligned}$$

□

*Proof of Theorem 4.4.* First, we demonstrate a simple property of the $\vee$ operator that we will use later. Let $a, x, y \in \mathbb{R}$. Then, using equation $x \vee y = 0.5(x + y + |x - y|)$, we obtain the following:

$$\begin{aligned}a \vee x - a \vee y &= 0.5\big(a + x + |x-a| - a - y - |y-a|\big) \\ &= 0.5(x - y + |x-a| - |y-a|) \\ &\leq 0.5(x - y + |x-a-(y-a)|) \\ &= 0.5(x - y + |x-y|) \\ &\leq |x-y|\end{aligned} \tag{9}$$

Now, we can prove that $\hat{\mathcal{T}}^\pi$ is a contraction. We begin with the state-action case. Let $q, z : \mathcal{S} \times \mathcal{A} \times \mathbb{R}$ be two-real valued functions. Then, we can expand $\|\hat{\mathcal{T}}^\pi q - \hat{\mathcal{T}}^\pi z\|_\infty$ as follows:

$$\begin{aligned}\|\hat{\mathcal{T}}^\pi q - \hat{\mathcal{T}}^\pi z\|_\infty &= \gamma \sup_{s\in\mathcal{S},a\in\mathcal{A},y\in\mathbb{R}} \Big|\mathbb{E}_{\substack{s_{t+1} \\ a_{t+1}}}\Big[y' \vee q(s_{t+1},a_{t+1},y') - y' \vee z(s_{t+1},a_{t+1},y')\big|_{a_t=a}^{s_t=s}\Big]\Big| \\ &\leq \gamma \sup_{s\in\mathcal{S},a\in\mathcal{A},y\in\mathbb{R}} \mathbb{E}_{\substack{s_{t+1} \\ a_{t+1}}}\Big[\Big|y' \vee q(s_{t+1},a_{t+1},y') - y' \vee z(s_{t+1},a_{t+1},y')\Big|\big|_{a_t=a}^{s_t=s}\Big]| \\ &\overset{(*)}{\leq} \gamma \sup_{s_{t+1}\in\mathcal{S},a_{t+1}\in\mathcal{A},y'\in\mathbb{R}} \Big|y' \vee q(s_{t+1},a_{t+1},y') - y' \vee z(s_{t+1},a_{t+1},y')\Big| \\ &= \gamma \sup_{s\in\mathcal{S},a\in\mathcal{A},y\in\mathbb{R}} \Big|y \vee q(s,a,y) - y \vee z(s,a,y)\Big| \\ &\leq \gamma \sup_{s\in\mathcal{S},a\in\mathcal{A},y\in\mathbb{R}} \Big|q(s,a,y) - z(s,a,y)\Big| = \|q-z\|_\infty\end{aligned}$$

The first inequality follows from the fact that $\Big|\mathbb{E}_x[x]\Big| \leq \mathbb{E}_x\Big[|x|\Big]$ and the last inequality follows from Eq. (9). In $(*)$, we use the following property of the expectation: $\sup_y\big\{\mathbb{E}[x|y]\big\} \leq \sup_x\{x\}$. Now, we demonstrate the contraction property for the state value function: Let $v, u : \mathcal{S} \times \mathbb{R}$ be two-real valued functions. Then, we expand $\|\hat{\mathcal{T}}^\pi v - \hat{\mathcal{T}}^\pi u\|_\infty$ as follows:

$$\begin{aligned}\|\hat{\mathcal{T}}^\pi v - \hat{\mathcal{T}}^\pi u\|_\infty &= \gamma \sup_{s\in\mathcal{S},y\in\mathbb{R}} \Big|\mathbb{E}_{\substack{a_t \\ s_{t+1}}}\Big[y' \vee v(s_{t+1},y') - y' \vee u(s_{t+1},y')\big|_{s_t=s}\Big]\Big| \\ &\leq \gamma \sup_{s\in\mathcal{S},y\in\mathbb{R}} \mathbb{E}_{\substack{a_t \\ s_{t+1}}}\Big[\Big|y' \vee v(s_{t+1},y') - y' \vee u(s_{t+1},y')\Big|\big|_{s_t=s}\Big] \\ &\leq \gamma \sup_{s_{t+1}\in\mathcal{S},y'\in\mathbb{R}} \Big|y' \vee v(s_{t+1},y') - y' \vee u(s_{t+1},y')\Big| \\ &= \gamma \sup_{s\in\mathcal{S},y\in\mathbb{R}} \Big|y \vee v(s,y) - y \vee u(s,y)\Big| \\ &\leq \gamma \sup_{s\in\mathcal{S},y\in\mathbb{R}} \Big|v(s,y) - u(s,y)\Big| = \|v-u\|_\infty\end{aligned}$$

Therefore, the max-reward Bellman operator is a contraction. Hence, by the Banach fixed-point theorem, it has a unique fixed-point(s). From Lemma 4.2, we conclude that this is the max-reward value function(s). □

*Proof of Lemma 4.8.*

$$
\begin{aligned}
\|\hat{\mathcal{T}}^* q - \hat{\mathcal{T}}^* z\|_\infty &= \gamma \sup_{s\in\mathcal{S},a\in\mathcal{A},y\in\mathbb{R}} \Big|\mathbb{E}_{s_{t+1}}\Big[y' \vee \max_{a'} q(s_{t+1},a',y') - y' \vee \max_{a'} z(s_{t+1},a',y')\Big|_{a_t=a}^{s_t=s}\Big]\Big| \\
&\le \gamma \sup_{s\in\mathcal{S},a\in\mathcal{A},y\in\mathbb{R}} \mathbb{E}_{s_{t+1}}\Big[\Big|y' \vee \max_{a'} q(s_{t+1},a',y') - y' \vee \max_{a'} z(s_{t+1},a',y')\Big|_{a_t=a}^{s_t=s}\Big] \\
&\le \gamma \sup_{s\in\mathcal{S},a\in\mathcal{A},y\in\mathbb{R}} \mathbb{E}_{s_{t+1}}\Big[\max_{a'}\Big|y' \vee q(s_{t+1},a',y') - y' \vee z(s_{t+1},a',y')\Big|_{a_t=a}^{s_t=s}\Big] \\
&\le \gamma \sup_{s_{t+1}\in\mathcal{S},a_{t+1}\in\mathcal{A},y'\in\mathbb{R}} \Big|y' \vee q(s_{t+1},a_{t+1},y') - y' \vee z(s_{t+1},a_{t+1},y')\Big| \\
&= \gamma \sup_{s\in\mathcal{S},a\in\mathcal{A},y\in\mathbb{R}} \Big|y \vee q(s,a,y) - y \vee z(s,a,y)\Big| \\
&\le \gamma \sup_{s\in\mathcal{S},a\in\mathcal{A},y\in\mathbb{R}} \Big|q(s,a,y) - z(s,a,y)\Big| = \|q-z\|_\infty
\end{aligned}
\tag{10}
$$

□

*Proof of Theorem 4.9.* First of all, we notice that the max-reward Bellman equation implies another recursive equation for $\hat{v}^\pi$ and $\hat{q}^\pi$ :

$$
\hat{q}^\pi(s,a,y) = \gamma\mathbb{E}_{s_{t+1}}\big[y' \vee \hat{v}^\pi(s_{t+1},y')\big|_{s_t=s}\big] = \gamma\mathbb{E}_{\substack{a_t \\ s_{t+1}}}\big[\hat{v}^\pi(s_{t+1},y')\big|_{a_t=a}^{s_t=s}\big] \tag{11}
$$

As discussed in the main paper, we use the version with extra $\vee y'$ to enforce boundary conditions. However, we can still use the equation above as it is a property of the max-reward value function.

Before proving the theorem, we introduce simplified notation to improve readability – for all functions, we use subscripts to denote the input variables. For example, $\hat{v}_t := \hat{v}^{\hat{\pi}}(s_t,y_t)$. The proof follows the one for the standard policy gradient theorem. We begin by obtaining a recurrent equation for $\nabla_\theta \hat{v}_0$ :

$$
\begin{aligned}
\nabla_\theta \hat{v}_0 = \nabla_\theta\Big(\int_{a_0} \hat{\pi}_0 \hat{q}_0 da_0\Big) &= \underbrace{\int_{a_0} (\nabla_\theta \hat{\pi}_0)\hat{q}_0 da_0}_{\phi_0} + \int_{a_0} \hat{\pi}_0 (\nabla_\theta q)_0 da_0 \\
&\overset{\text{Eq.(11)}}{=} \phi_0 + \gamma \int_{a_0} \hat{\pi}_0 \Big(\nabla_\theta \int_{s_1,y_1} \hat{p}(s_1,y_1|s_0,y_0,a_0)\hat{v}_1 ds_1 dy_1\Big) da_0 \\
&= \phi_0 + \gamma \int_{s_1,y_1}\int_{a_0} \hat{\pi}_0 \hat{p}(s_1,y_1|s_0,y_0,a_0)(\nabla_\theta \hat{v}_1) ds_1 dy_1 da_0 \\
&= \phi_0 + \gamma \int_{s_1,y_1} \hat{p}_1^{\hat{\pi}}(s_0,y_0,s_1,y_1)(\nabla_\theta \hat{v}_1) ds_1 dy_1 \,,
\end{aligned}
$$

where we introduced the shorthand $\phi_t = \phi(s_t,y_t) = \int_a \nabla_\theta \hat{\pi}(a|s_t,y_t)\, q(s_t,a,y_t)\, da$. Expanding this recurrence further, we obtain

$$
\begin{aligned}
\nabla_\theta \hat{v}_0 &= \sum_{t=0}^{\infty} \int_{s_t,y_t} \gamma^t \hat{p}_t^{\hat{\pi}}(s_0,y_0,s_t,y_t)\phi(s_t,y_t) ds_t dy_t \\
&= \int_{s,y} \Big(\sum_{t=0}^{\infty} \gamma^t \hat{p}_t^{\hat{\pi}}(s_0,y_0,s,y)\Big)\phi(s,y) ds dy \\
&\propto \int_{s,y} \hat{d}^{\hat{\pi}}(s,y|s_0,y_0)\phi(s,y) ds dy \\
&= \int_{s,y} \hat{d}^{\hat{\pi}}(s,y|s_0,y_0)\Big(\int_a (\nabla_\theta \hat{\pi}(a|s,y))\hat{q}(s,a,y) da\Big) ds dy
\end{aligned}
$$

Above, $\hat{d}^{\hat{\pi}}(s, y|s_0, y_0)$ is the discounted stationary distribution of $s, y$ for policy $\pi$ given $s_0, y_0$. Finally, we substitute this formula for $\nabla_\theta \hat{v}_0$ into the definition of $\hat{J}(\theta)$ and conclude the proof:

$$\begin{aligned}\nabla_\theta \hat{J}(\theta) &= \int_{s_0,y_0} \hat{p}_0(s_0, y_0) \int_{s,y} \hat{d}^{\hat{\pi}}(s, y|s_0, y_0) \Big( \int_a \big(\nabla_\theta \hat{\pi}(a|s, y)\big) \hat{q}(s, a, y) da \Big) ds dy ds_0 dy_0 \\ &= \int_{s,y} \hat{d}^{\hat{\pi}}(s, y) \Big( \int_a \big(\nabla_\theta \hat{\pi}(a|s, y)\big) \hat{q}(s, a, y) da \Big) ds dy \\ &= \mathbb{E}_{\substack{s,y\sim\hat{d}^{\hat{\pi}} \\ a\sim\hat{\pi}(\cdot|s,y)}} \big[\hat{q}^{\hat{\pi}}(s, a, y) \nabla_\theta \ln \hat{\pi}(a|s, y)\big]\end{aligned} \tag{12}$$

□

## B. Experimental details

For all experiments, we used our implementation of TD3 and PPO that we verified on several MuJoco domains. The implementation of the max-reward algorithms is similar to their cumulative versions except for the following differences:

1. The input layer of all neural networks has an extra dimension to work with the extended states $(s, y)$.
2. The output layer of the value networks uses *Tanh* activation and is rescaled to $u \in [0, \bar{R}]$. Then, it is transformed with $ReLU(u - y) + y$ to enforce $\hat{v}^\pi(s, y) \geq y$.

Hyperparameters of all runs are reported in Tables 1-2.

| Parameter | PPO | PPOMax | TD3 | TD3Max |
|---|---|---|---|---|
| Parallel environments | 16 | 16 | 16 | 16 |
| Discount factor $\gamma$ | 0.99 | 0.999 | 0.99 | 0.995 |
| Learning rate | 3e-4 | 3e-4 | 3e-4 | 3e-4 |
| Lr. annealing | No | No | No | No |
| Entropy weight | 5e-2 | 5e-2 | | |
| Value loss weight | 0.5 | 0.5 | | |
| Clip coef. | 0.2 | 0.2 | | |
| GAE $\lambda$ | 0.95 | 1 | | |
| Policy update freq. | | | 2 | 2 |
| Target soft update $\tau$ | | | 0.005 | 0.005 |
| Expl. noise type | | | pink | pink |
| Expl. noise std | | | 0.7 | 0.7 |
| Expl. noise clip | | | 0.5 | 0.5 |
| Target noise scale | | | 0.2 | 0.2 |
| Initial expl. steps | | | 25000 | 25000 |
| Tr. epochs per rollout | 10 | 10 | | |
| Rollout length | 1024 | 2048 | | |
| Minibatch size | 32 | 32 | 256 | 256 |

*Table 1.* Hyperparameters for the experiments with Maze environment.

## C. Algorithms

| Parameter | TD3 | TD3Max |
|---|---|---|
| Parallel environments | 16 | 16 |
| Discount factor $\gamma$ | 0.99 | 0.995 |
| Learning rate | 3e-4 | 3e-4 |
| Lr. annealing | No | No |
| Policy update freq. | 2 | 2 |
| Target soft update $\tau$ | 0.005 | 0.005 |
| Expl. noise type | pink | pink |
| Expl. noise std | 0.1 | 0.1 |
| Expl. noise clip | 0.5 | 0.5 |
| Target noise scale | 0.2 | 0.2 |
| Initial expl. steps | 25000 | 25000 |
| Minibatch size | 256 | 256 |

*Table 2.* Hyperparameters for the experiments with Fetch environment.

**Algorithm 1** Max-reward TD3

Initialize critic networks $\hat{q}_{\phi_1}, \hat{q}_{\phi_2}$ and actor network $\mu_\theta$
Initialize target networks $\phi'_1 \leftarrow \phi_1$, $\phi'_2 \leftarrow \phi_2$, $\theta' \leftarrow \theta$
Initialize replay buffer $\mathcal{D}$
**for** $episode = 1, 2, \ldots$ **do**
  Initialize $s_0, y_0 \sim \hat{p}_0$
  **for** $t = 0, 1, \ldots, T-1$ **do**
    Sample exploration noise $\epsilon_t$
    Execute $a_t = \mu(s_t, y_t) + \epsilon_t$ and get $s_{t+1}, r_{t+1}$
    Update $y_{t+1} = (y_t \vee r_{t+1})/\gamma$
    Save $(s_t, y_t, a_t, s_{t+1}, r_{t+1}, y_{t+1})$ into $\mathcal{D}$
    **if** initial exploration is over **then**
      Sample a mini-batch of size $N$ from $\mathcal{D}$
      Sample target actions noise $\eta$
      $\tilde{a} \leftarrow \mu_{\theta'}(s', y') + \eta$
      $z \leftarrow y' \vee \gamma \min_{i=1,2} \hat{q}_{\phi'_i}(s', \tilde{a}, y')$
      Critic loss $L_c = \frac{1}{N} \sum_{i=1}^{2} (z - \hat{q}_{\phi_i}(s, a, y))^2$
      Perform gradient update step on $L_c$
      **if** time to update policy **then**
        $L_a \leftarrow \mathbb{E}\big[\hat{q}_{\phi_1}(s, \mu_\theta(s, y), y)\big]$
        Perform gradient update step on $L_a$
        $\phi'_i \leftarrow \tau\phi_i + (1-\tau)\phi'_i,\ i = 1, 2$
        $\theta' \leftarrow \tau\theta + (1-\tau)\theta$
      **end if**
    **end if**
  **end for**
**end for**

**Algorithm 2** Max-reward PPO

Initialize actor $\hat{\pi}_\theta$ and critic $\hat{v}_\phi$
**for** $iteration = 1, 2, \ldots$ **do**
  Initialize trajectories buffer $\mathcal{D}$
  **for** $actor = 1, 2, \ldots, N$ **do**
    Initialize $s_0, y_0 \sim \hat{p}_0$
    **for** $t = 0, 1, \ldots, T-1$ **do**
      Execute $a_t \sim \pi_\theta(\cdot|s_t, y_t)$ and get $s_{t+1}, r_{t+1}$
      Update $y_{t+1} = (y_t \vee r_{t+1})/\gamma$
      Save $(s_t, y_t, a_t, s_{t+1}, r_{t+1}, y_{t+1})$ into $\mathcal{D}$
    **end for**
    $\hat{G}_t^n \leftarrow \gamma^n \hat{v}_\phi(s_{t+n}, y_{t+n}),\ n = 1, \ldots, T-t$
    $\hat{G}_t(\lambda) = (1-\lambda)\sum_{n=1}^{T-t} \lambda^{n-1} \hat{G}_t^n$
    Compute advantages $\hat{A}_t = \hat{G}_t(\lambda) - \hat{v}_\phi(s_t, y_t)$
  **end for**
  **for** k=$1, 2 \ldots K$ **do**
    Critic loss: $L_c \leftarrow \frac{1}{T}\sum_t (\hat{G}_t^{T-t} - \hat{v}_\phi(s, y))^2$
    Actor loss: $L_a \leftarrow L_{PPO}(\pi_\theta, \{\hat{A}_t\}_{t=1}^T)$
    Perform gradient update step on $L_a + L_c$
  **end for**
**end for**