Tiered Reward: Designing Rewards for Specification and Fast Learning of Desired Behavior

Zhiyuan Zhou * UC Berkeley
Shreyas Sundara Raman Brown University
Henry Sowerby Brown University
Michael L. Littman Brown University

Abstract

Reinforcement-learning agents seek to maximize a reward signal through environmental interactions. As humans, our job in the learning process is to design reward functions to express desired behavior and enable the agent to learn such behavior swiftly. However, designing good reward functions to induce the desired behavior is generally hard, let alone the question of which rewards make learning fast. In this work, we introduce a family of a reward structures we call Tiered Reward that addresses both of these questions. We consider the reward-design problem in tasks formulated as reaching desirable states and avoiding undesirable states. To start, we propose a strict partial ordering of the policy space to resolve trade-offs in behavior preference. We prefer policies that reach the good states faster and with higher probability while avoiding the bad states longer. Next, we introduce Tiered Reward, a class of environment-independent reward functions and show it is guaranteed to induce policies that are Pareto-optimal according to our preference relation. Finally, we demonstrate that Tiered Reward leads to fast learning with multiple tabular and deep reinforcement-learning algorithms.

1 Introduction

Reinforcement learning (Sutton & Barto, 1998) (RL) is concerned with the problem of learning to behave to maximize a reward signal. In biological systems, this reward signal is considered to be the organism's motivational system, using pain and pleasure to modulate behavior (Porreca & Navratilova, 2017; Leknes & Tracey, 2010; Navratilova & Porreca, 2014). In engineered systems, however, rewards must be selected by the system designer (Nagpal et al., 2020; Dewey, 2014). We view rewards as a kind of programming language—a specification of the agent's target behavior (Littman et al., 2017; Zhou & Li, 2022). As arbiters of behavior correctness in the learning process, humans bear the responsibility of authoring this program. This is referred to as the reward-design problem (Dewey, 2014; Devidze et al., 2021; Sorg et al., 2010; Sowerby et al., 2022): *given a set of desired behavior, what kind of reward functions would efficiently express these behavior?* In this paper, we look at rewards that can correctly and efficiently express desirable states (goals and subgoals) and undesirable states (obstacles).

There are two essential steps in designing reward functions. First, one must decide what kind of behavior is desirable and should be conveyed. Then, there's the choice of reward function that induces such behavior. The first step is hard because there is no universal preference over behavior and having to explicitly write down all possible trade-offs is challenging. Even if the reward designer has a way of expressing preferences for all possible exchanges, it can be difficult, or even impossible, to design a reward function that captures them without prior knowledge of the environment.

First, let us consider the question of how to specify preference over a set of policies (Roy et al., 2021). In the goal–obstacle class of tasks we consider, preferences over policies are simple in deterministic

*Correspondence to: zhiyuan_zhou@berkeley.edu. Code for the paper can be found at https://github.com/zhouzypaul/tiered-reward

environments. We imagine all states are either goal states, obstacle states, or neither (background states), and all goals and obstacles are absorbing. Preferences in deterministic environments form a total order: reaching goals is better than reaching obstacles; if the policy reaches goals, faster is better; if the policy reaches obstacles, avoiding them longer is better. However, even in this simple setting, providing such precise trade-offs is difficult in stochastic environments. Is it better for an agent to increase the chance of getting to the goal by 5% if it also incurs 8% higher probability of hitting an obstacle? Is it better to increase by 50% the probability of getting to a goal if the expected time of getting there also increases by 20%? Preferences are less clear in a stochastic setting because there can be trade-offs between different outcomes and their probabilities. However, some comparisons are arguably clear cut. Informally, if one policy induces uniformly better outcomes than another—being more likely to reach a goal and doing so faster, being less likely to reach an obstacle and getting there more slowly—we prefer such a policy. If the policies can't be directly compared, we propose to be indifferent between them. Thus, we replace the standard reinforcement-learning notion of optimality with Pareto-optimality (Mornati, 2013) which is commonly adopted in multiobjective RL (Vamplew et al., 2011; Van Moffaert & Nowé, 2014); we seek a policy that is either preferred or incomparable to every other policy.

Even after resolving the issue of specifying behavior preference, policies are hard to express through reward functions in general (Zhang et al., 2009; Amodei & Clark, 2016), and some are even impossible to convey with a Markov (state–action-based) reward (Abel et al., 2021). Even when policies are expressible, designing bad reward functions can lead to undesirable or dangerous actions (Amodei & Clark, 2016), easy reward hacking (Amodei et al., 2016; Skalse et al., 2022), and more. We seek to design good reward functions, which can be characterized by many properties, such as interpretability and learning speed (Devidze et al., 2021). But the most important property a reward function must have is to guarantee the adoption of a desired policy. As we will show later in Section 5, even intuitively correct reward designs can lead to suboptimal policies. To hedge against this, we introduce a tiered reward structure that is guaranteed to induce Pareto-optimal policies. Intuitively, we partition the state space into several tiers, or goodness levels. States in the same tier are associated with the same reward, while states in a more desirable tier are associated with an exponentially higher reward. We prove that these tiered reward structures, with the proper constraints between reward values, induce Pareto-optimal behavior and empirically show that they can lead to fast learning.

In this paper, we propose Tiered Reward as a way to design reward functions that can correctly and swiftly express desired behavior. Our contribution is threefold: First, we define a preference over the entire policy space via a strict partial ordering on outcomes using the notion of Pareto-optimality. This addresses the question of behavior preference in stochastic environments. Then, we introduce a class of environment-independent tiered reward structure that provably induce Pareto-optimal policies with respect to this preference ordering. Finally, we demonstrate these tiered reward functions can lead to fast learning in both tabular and deep RL settings and is invariant to the choice of RL algorithms.

2 Related Work

Specifying behavior through rewards: Preference-based RL methods (Wirth et al., 2017; Brown et al., 2019; Liu et al., 2022) learn a reward function based on a dataset of preferences over trajectories. But, as we have argued, preferences over trajectory probabilities can be very difficult to specify. In addition, our reward scheme relieves the need for environment-specific preference datasets created by human experts. Multi-objective RL (Vamplew et al., 2011; Toro Icarte et al., 2018; Hayes et al., 2022) allows for different tasks to be specified through a set of reward functions. Our work proceeds in the orthogonal direction by designing a single reward function to trade off among multiple behaviors, instead of incentivizing all of them. Reward machines (Icarte et al., 2018; 2022) are finite state machines that compose reward functions and allow different rewards to be delivered dependent on the agent's trajectory. They reveal the structure of the reward function to the RL agent to support decomposition of complex tasks. Our focus on how to provide incentives for

specific outcomes is complementary and the two approaches can be used in concert. Temporal logic based languages (Littman et al., 2017; Camacho et al., 2017; Li et al., 2017; Camacho et al., 2019) have been used to specify behavior. Though these methods can be more expressive, they often lead to intractable planning and learning problems due to state-space explosion issues (Wongpiromsarn et al., 2010). We offer a different expressibility–tractability tradeoff.

Reward Design for fast learning: Mataric (1994) proposed to accelerate learning by incorporating domain knowledge and using a progress estimator, but does not provide a principled method of designing a reward function. Sowerby et al. (2022) showed that reward functions that maximize the action gap given a measure of horizon length lead to fast learning. However, designing such reward functions requires solving an optimization problem with detailed knowledge of the environment, making this approach impractical. Similarly, Devidze et al. (2021) formulated the reward-design problem as an optimization problem to maximize informativeness and sparsity. However, their method requires solving the MDP with ground truth transition dynamics and a reference reward function, which is often not available in practice.

3 Problem Setting

We view an RL environment as a Markov Decision Process (MDP), with state space S, action space A, transition model T, reward function R, and discount factor γ. A policy $\pi : S \times A \to [0, 1]$ is a mapping from the current state to a probability distribution of the action to be taken. The optimal policy starting from some initial state s_0 in the MDP is defined as any reward-maximizing policy $\pi^* \in \text{argmax}_\pi \mathbb{E}[\sum_t \gamma^t r_t | s_0, \pi]$. To make the reward-design problem as simple as possible for designers, we limit the reward function $R : S \to \mathbb{R}$ to be defined solely on states. In goal–obstacle tasks, we consider the goal states and obstacle states to be absorbing.

We are interested in the reward-design problem. In the common RL framework, tasks are specified by the reward function and the agent's objective is to maximize cumulative rewards. We take an alternative perspective: We specify tasks by prescribing a set of desirable policies and seek reward functions such that maximizing the reward will lead to the desirable policies. Formally, given a set of desirable (Pareto-optimal) policies Π, the reward-design problem is to create a reward function $R : S \to R$ such that the optimal policy $\pi^*_R \in \Pi$.

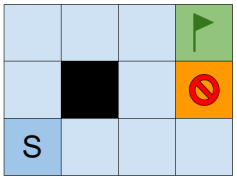
We imagine the state space S as exhibiting a tiered structure, where higher tiers are more desirable than lower tiers, and states within the same tier are equally desirable. Formally, we define:

Definition 3.1. k-Tier Markov Decision Process: A Tier MDP is an MDP with state space S, action space A, transition model $T : S \times A \times S \to \mathbb{R}$, reward function $R : S \to \mathbb{R}$, and discount factor γ. The state space is partitioned into k tiers, where $S = S_1 \cup S_2 \cup \ldots \cup S_k$ and $S_i \cap S_j = \emptyset, \forall i \neq j \in 1, 2, ..., k$. The reward function has the form $R(s) = r_i, \forall s \in S_i, i = 1, 2, ..., k$. In addition, $r_1 < r_2 < \ldots < r_k$.

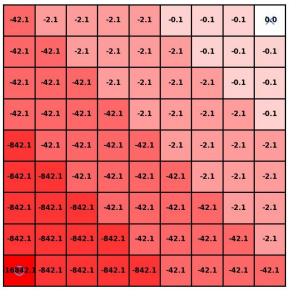
As an example, the grid world from Russell & Norvig (2010), as illustrated in Figure 1a, could be formulated as a 3-Tier MDP—the goal state is one tier (S_3), the lava state one tier (S_1), and all other states reside in the background tier (S_2). It is important to note that we put no constraints on how many states could be in each tier, nor how many tiers there can be. Therefore, the framework has a high degree of generality: any finite MDP with reward defined on states could be formulated as a Tier MDP by placing states with the same reward in the same tier. However, the Tier MDP is most useful when there are clear good and bad states in the state space, such as when there are goal and obstacle states, or even states of intermediate desirability such as subgoal states. In the following sections, we will show how to perform reward design in Tier MDPs.

4 Policy Ordering with Pareto-optimality

A policy induces a probability distribution over an infinite set of outcomes (specifically the probability of reaching each of the states after t steps, for all t). In goal–obstacle tasks, policies can be



(a) Russell/Norvig grid world. Objective is to reach the goal (green) without first visiting lava (orange) ($\gamma = 0.9$).



(b) A Tiered Reward in a grid world with 6 tiers. Start state in bottom left, goal state in top right. Darker colors correspond to more negative reward values.

Figure 1

characterized by statistics such as probability of reaching the goal and probability of avoiding the obstacle for each possible horizon length. Using these statistics, we will show below how the entire policy space can form a strict partial ordering to specify which policies are preferable.

We define o_t to be the probability of being in S_1 at timestep t, and g_t that of S_3. Given two policies π^A and π^B, we say π^A *dominates* π^B when both of these inequalities hold (and not both being strictly equal at all times):

$$\sum_{i=0}^{t} o_i^A \leq \sum_{i=0}^{t} o_i^B, \quad \sum_{i=0}^{t} g_i^A \geq \sum_{i=0}^{t} g_i^B, \qquad \forall t = 0, 1, 2, ..., \infty.$$

In words, one policy dominates another if it gets to the goal faster, while delaying encountering obstacles longer. The set of policies that are not dominated by any other policy is the set of *Pareto-optimal* policies. Because there is a finite number of policies and domination is transitive, the set of Pareto-optimal policies is non-empty. See a visualization of Pareto-optimal policies in Appendix C.

Pareto-optimal policies are interesting to consider for two main reasons. First, Pareto-optimal behavior always exists, even when policies that achieve other reasonable things do not. Secondly, Pareto-optimality addresses the preference problem by defining a strict partial ordering over the entire policy space. Although the policies on the Pareto frontier are incomparable among themselves, they are all better than the set of Pareto-dominated policies. We simply deem the set of Pareto-optimal policies to be the desirable behavior, and all others undesirable. Next, we show how to design rewards that guarantee Pareto-optimal policies.

5 Tiered Reward

In this section, we seek a sufficient condition on the reward function so that optimizing expected discounted reward will result in a Pareto-optimal policy with respect to our preference relation.

Definition 5.1. Pareto-optimal rewards: A reward function $R(s)$ is called *Pareto-optimal* if the policy it induces, $\pi_R \in \text{argmax}_\pi \, \mathbb{E}[\sum_t \gamma^t r_t | s_0, \pi]$, is Pareto-optimal.

Even some reasonable-sounding reward functions need not be Pareto-optimal. Going back to the Russell/Norvig grid example, an intuitive reward design would be requiring $r_{lava} < r_{background} < r_{goal}$. Consider three example reward functions in Table 1 that satisfy this constraint: Both R and G are Pareto-optimal, while B is Pareto-dominated (see Figure 8 in Appendix). Roughly, B doesn't encourage getting to the goal but is also not good at avoiding lava.

In fact, many of the reward functions that satisfy $r_{lava} < r_{background} < r_{goal}$ are not Pareto-optimal. Out of 1000 such rewards that we sampled randomly, 90.5% were Pareto-dominated (See Figure 11 in

Policy	r_{lava}	$r_{background}$	r_{goal}
R	-1	-0.1	$+1$
G	-1	0	$+0.5$
B	-1	-0.9	0

Table 1: Three intuitive reward functions of Russell/Norvig grid world. B is Pareto-dominated.

Appendix). Next, we present a simple rule that is sufficient to guarantee environment-independent Pareto-optimal reward functions.

5.1 The 3 Tiers Case

To facilitate understanding, we limit the problem space to 3-Tier MDPs for now, and generalize to k-Tier MDPs in Section 5.2. In a 3-Tier MDP, we will call the 3 tiers obstacles (S_1), background (S_2), and goals (S_3), in order of increasing desirability. States in S_1 and S_3 are absorbing.

Definition 5.2. Tiered Reward: In a 3-Tier Markov Decision Process with discount factor $\gamma \in (0, 1)$, a reward function defined by

$$R(s) = \begin{cases} r_{obs} & \text{if } s \in S_1 \\ r_{background} & \text{if } s \in S_2 \\ r_{goal} & \text{if } s \in S_3 \end{cases}$$

is considered a *Tiered Reward* if

$$r_{obs} < \frac{1}{1-\gamma} r_{background} < r_{goal}.$$

and states in S_1 and S_3 are absorbing.

Theorem 5.3 (Pareto-optimal rewards in 3-Tier MDP)**.** *In a 3-Tier Markov Decision Process, a Tiered Reward is Pareto-optimal.*

We leave the proof in Appendix A but provide some intuition for Tiered Reward here. The middle term in Definition 5.2, $\frac{1}{1-\gamma} r_{background}$, is equal to the cumulative discounted return for infinitely getting a reward in the background tier ($(1 + \gamma + \gamma^2 + ...) r_{background}$). So, in a gross simplification, as long as the reward at the goal is more appealing than infinitely wandering in background states, and the obstacle less appealing, the reward induces behavior that arrives at the goal early and avoids the obstacles. Following this simple constraint, we as reward designers can easily create Pareto-optimal reward functions without requiring knowledge of the transition probabilities in the environment. Though environment-specific knowledge is needed partition the state space into tiers, Tiered Reward is generally applicable and environment-independent in the sense that the reward structure remain the same and the reward values for each tier are shared across different MDPs.

5.2 Generalizing to k Tiers

MDPs with more than 3 tiers can usefully model important problems such as those with well-defined subgoal states. Specifically, each subgoal region could be its own tier, instead of being grouped into one big background tier. Even though these problems could still be solved as a 3-Tier MDP, more knowledge about the environment could help design better reward functions and accelerate learning.

Definition 5.4. Tiered Reward: In a k-Tier($k > 3$) Markov Decision Process with discount factor $\gamma \in (0, 1)$ where the goal tier (k) is absorbing, the reward function R is a Tiered Reward if $R(s) = r_i, \forall s \in S_i, i = 1, 2, ..., k$, for reward values $r_1, r_2, ..., r_k \in \mathbb{R}$, that satisfy

$$r_1 < (\frac{1}{1-\gamma}) r_2 < (\frac{1}{1-\gamma})^2 r_3 < \cdots < (\frac{1}{1-\gamma})^{k-1} r_k \leq 0.$$

One such reward is visualized in Figure 1b. Notice that the k-Tiered Reward (Definition 5.4) uses a stricter condition than that of 3-Tiered Reward (Definition 5.2). In fact, Definition 5.4 is a sufficient condition for Definition 5.2 because 3 tiers is a special case with only one non-absorbing tier. One can also design a stricter 3-Tiered Reward with Definition 5.4 with $k = 3$. Definition 5.4 is stricter in that: first, all reward values are non-positive. This is a sufficient but not necessary condition to guarantee Pareto-optimality. We enforce this constraint not only to make it a sufficient condition, but also because step-wise penalty has been proved to support faster learning (Koenig & Simmons, 1993; Sowerby et al., 2022). Specifically, with a zero-initialized value function, step penalties create an incentive for the agent to try state–action pairs it has never experienced before, resulting in rapid exploration. Secondly, the reward values of higher tiers are exponentially greater than the lower ones. For adjacent tiers i and $i+1$, The reward values always satisfy $r_i < \frac{1}{1-\gamma} r_{i+1} < 0$.

This definition can be understood as a generalization of the 3-Tiered Reward. When the agent resides within tier $i \in \{2, 3, ..., k-1\}$, the k tiers could be partitioned into 3 groups to construct a 3-Tier MDP. In particular, S_1 will include tiers 1 through $i-1$, S_2 is just tier i, and S_3 is tiers $i+1$ to k. Note that we can generalize Theorem 5.3 to allow states in S_1 and S_3 to have any reward values as long as they satisfy the inequality in Definition 5.2 for a fixed reward value in S_2. Namely, denote $r_{low} = \max\{r_1, ..., r_{i-1}\}$ and $r_{high} = \min\{r_{i+1}, ..., r_k\}$, and as a k-Tiered Reward they satisfy $r_{low} < (\frac{1}{1-\gamma}) r_i < (\frac{1}{1-\gamma})^2 r_{high}$ And since $\gamma \in (0, 1)$, $r_{low} < (\frac{1}{1-\gamma}) r_i < (\frac{1}{1-\gamma})^2 r_{high} \leq r_{high}$. That is, (r_{low}, r_i, r_{high}) is a Tiered Reward function in the 3-Tier MDP with tiers S_1, S_2, and S_3, and therefore induces Pareto-optimal policies (Theorem 5.3). So, at tier i, the policy that optimizes the k-Tiered Reward will push agents to higher tiers as fast as possible and avoid lower tiers, as if they were goals and obstacles, respectively. In the special case that the agent resides within tier $i = 1$, the constraint from Definition 5.4 will treat tiers 2 through k as if they are all goals, pushing the agent towards them. In the case that $i = k$, the agent is already in the "goal tier". So overall, k-Tiered Reward will induce in a ratchet-like policy—go to the higher tiers as fast as possible while not fall back to the lower tiers—that makes learning fast. In fact, it has been shown that a similar increasing reward profile (Sowerby et al., 2022) leads to fast learning. Okudo & Yamada (2021) and Zhai et al. (2022) have also shown that intermediate rewards can accelerate learning and provably improve sample efficiency in goal–reaching tasks.

Besides encouraging early visitation of good tiers, using Tiered Reward also guarantees maximum total visitation of all good tiers. This property is formalized in Theorem 5.5.

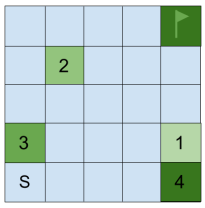
Theorem 5.5 (Tiered Reward and Cumulative Tier Visitation)**.** *In a k-Tier Markov Decision Process that has Tiered Reward $R(s)$, the induced optimal policy is π^*. Let $p_t^{*d} \in [0, 1]$ be the probability of being in tier $d \in \{1, 2, ..., k\}$ for the first time at timestep t following policy π^*. Then, there is no policy π, along with its induced probability distribution p_t^d, that satisfies both:*

$$\sum_{i=0}^{t} p_i^1 \leq \sum_{i=0}^{t} p_i^{*1}, \quad \forall t = 0, 1, 2, ..., \infty \qquad \sum_{i=0}^{t} p_i^d \geq \sum_{i=0}^{t} p_i^{*d}, \quad \forall d = [2..k], \forall t = 0, 1, 2, ..., \infty.$$

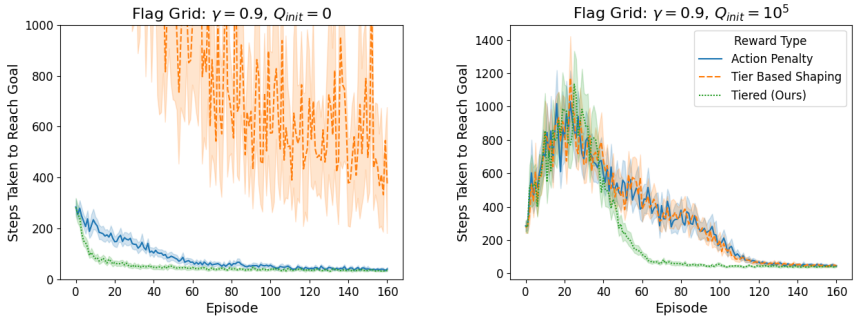
The proof is similar to that of Theorem 5.3 and can be found in Appendix B. To state the theorem in words, if a k-Tier MDP has a Tiered Reward structure, then the resulting policy will visit the worst tier (S_1) for as few times as possible, while visiting all the other good tiers (S_2, ..., S_k) as often as possible, respectively.

6 Experiments

Guaranteeing Pareto-optimal behavior is not the sole benefit of using Tiered Rewards; we find that it also leads to fast learning compared to two baseline reward functions. The first one, following Koenig & Simmons (1993), we call *action penalty*. This reward penalizes each step with -1, until the goal state is reached and the agent is awarded $+1$. To reiterate, such step-wise negative reinforcement encourages directed exploration assuming only knowledge about position of the goal. The second one uses reward shaping as Ng et al. (1999) showed that subgoals can be leveraged through shaping



(a) Flag Grid. The state space is the location of the agent plus the flag it has collected.



(b) Q Learning curves for Flag Grid. The two plots use different initial Q-values. $\gamma \in \{0.99, 0.95, 0.90, 0.85\}$ and $Q_{init} \in \{10^{10}, 10^5, 10^3\}$ all had similar performances, so we only report $\gamma = 0.90, Q_{init} = 10^5$ here. Error bar show standard deviation from 30 seeds.

Figure 2: Tiered Reward leads to fast learning on the Flag Grid.

rewards to guide the learning process. Reward shaping using an optimal value function is not a fair comparison both because it contains more information about the environment than tiers and it requires solving the environment with a pre-specified ground truth reward function, which induces a circular logic in reward design. For direct comparability, we use the Tiered Reward as a potential function $\Phi(s) = R_{tier}(s)$ to shape the action-penalty reward, resulting in what we call *tier-based shaping reward*: $R_{tbs}(s, a, s') = R_{penalty}(s) + \gamma\Phi(s') - \Phi(s)$.

There are many ways to design a Tiered Reward because it is a class of reward functions that is only constrained by an inequality (Definition 5.4), and not by specific reward values. In this section, for simplicity and clarity we use k-Tiered Reward defined by:

$$r_i = \begin{cases} 0 & \text{if } i = k \\ \frac{1}{1-\gamma} r_{i+1} - \delta & \text{if } \ i < k \end{cases}$$

where δ is a small constant used to satisfy the strict inequality constraint.

Empirically, we show Tiered Reward provides faster learning on multiple tabular domains. We further extend our results to environments with high-dimensional image inputs and deep RL algorithms. Finally, we also explore the influence of the number of tiers and find that having more tiers can induce faster learning.

6.1 Fast Learning with Tiered Reward

The "Flag Grid" from Ng et al. (1999) is a natural 6-Tier MDP to study Tiered Reward. In this grid world (Figure 2a), the agent begins in the bottom-left corner and must learn to pick up four flags in sequence before reaching the goal. The agent can move in 4 directions with an 80% success rate, while acting randomly 20% of the time. All states in which the agent possesses the same number of flags constitute a tier, totaling 5 tiers. The goal constitutes the sixth and final tier.

We evaluate all three reward functions with Q-Learning (Watkins & Dayan, 1992), arguably the most well-understood and widely applicable RL algorithm. As Figure 2b (left) shows, Tiered Reward learns fastest of the three. Perhaps surprisingly, Tier Based Shaping performs orders of magnitudes worse than even action penalty. That is because, with discounting, the shaping function $F(s, a, s') = \gamma\Phi(s') - \Phi(s)$ becomes positive when s and s' belong to the same tier. As a result, zero-initialized Q-values are not optimistic with respect to these rewards, and so exploration with R_{tbs} is undirected and slow. For a fairer comparison, we also plot the learning curves where all Q-values are initialized to some arbitrary large value so that R_{tbs} also enjoys directed exploration as the two other rewards (Figure 2b, right). There is a noticeable spike in time taken to reach the goal early on during

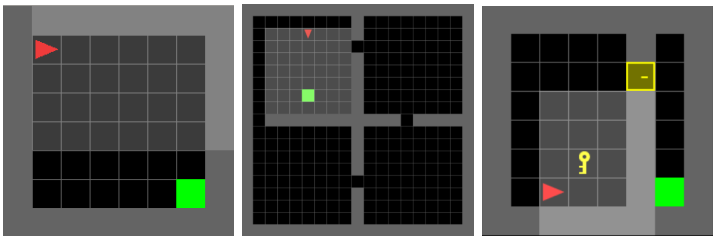


Figure 3: Illustration of EmptyGrid, FourRooms, and DoorKey from left to right. The agent is in red, and the goal in green. The objective is to navigate to the goal; in FourRooms, the agent has to find the gaps in the wall; in DoorKey, the agent must first pick up the key and unlock the yellow door. All three environments use visual observations.

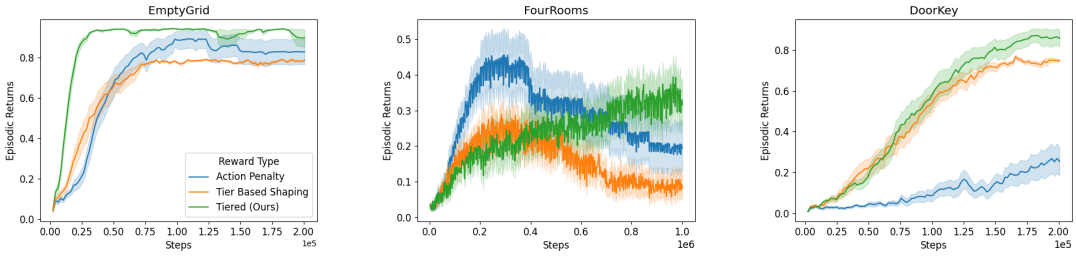


Figure 4: Learning curves of three reward functions on EmptyGrid, FourRooms, and DoorKey. Each agent is trained with the reward function labeled on the plot, but evaluated using the original MiniGrid reward ($1 - 0.9 * \text{step count}/\text{max steps}$ for success, and 0 for failure). Error bars show standard deviation from 30 random seeds.

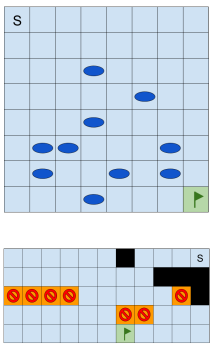
training. It is the result of optimistic Q-value initialization, which leads to more exploration and thus slower learning. Regardless of this trade off, Tiered Reward still consistently outperforms the two baselines.

It is important to note that our goal here is not to argue shaping is ineffective, nor to determine how to initialize Q-values for fast learning, but solely to demonstrate the usefulness of Tiered Reward in various different settings. To start, it makes learning faster than tier-based shaping reward and action penalty for different discount factors and Q-value initialization schemes. Moreover, it is simple to design and implement; there is no need to engineer environment-specific reward structures and initial Q-values to accelerate learning.

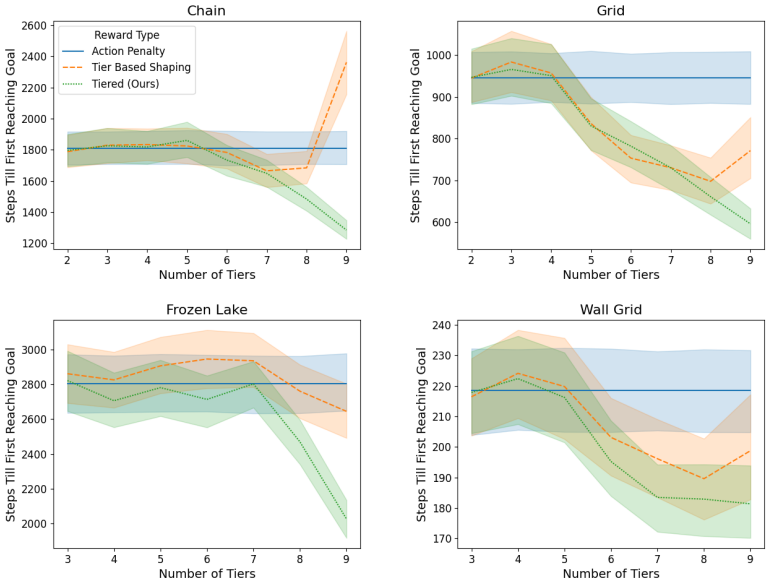
6.2 Tiered Reward in Deep RL

We further show that Tiered Reward also make learning faster in Deep RL with image observations. We choose three goal–obstacle environments from MiniGrid (Chevalier-Boisvert et al., 2023): EmptyGrid, FourRooms, and DoorKey. In all three environments (Figure 3), the agent aims to learn a policy to navigate to the goal using image observations. FourRooms is a long-horizon problem; the DoorKey environment has a complicated transition function and action space, and is hard to solve using classical RL algorithms with a sparse reward. In all three environments, we design Tiered Reward with three tiers. For EmptyGrid and FourRooms, tiers are assigned based on each state's $L1$ distance to the goal; for DoorKey, tiers are assigned based on the sub-goals the agent has completed (getting key, opening door, and reaching goal).

We use Proximal Policy Optimization (Schulman et al., 2017) (PPO) as the RL algorithm. To provide numerical stability, deep RL methods often employ reward scaling and clipping (Henderson et al., 2018). To follow, we linearly scale the tiered reward values to be between -1 and 0. More experiment details are included in Appendix F.



(a) Top: Frozen lake environment with holes in dark blue.
Bottom: Wall Grid environment with walls in black and lava in orange.



(b) Q-Learning results with $\gamma = 0.9$. Since there are 3 types of states in the latter two environments, we start at 3 tiers instead of 2. Error bar show standard deviation from 300 seeds.

Figure 5: Influence of more tiers on four grid worlds with Q learning. RMAX results in Figure 9.

For evaluation, we plot the episodic return with respect to the original reward function from MiniGrid (Chevalier-Boisvert et al., 2023). The original reward functions in MiniGrid are designed by human experts and express the task of reaching the goal quickly and avoiding obstacles. Thus, performance on the original reward function shows how quickly the agent has learned the specified behavior. Figure 4 shows the learning performance of PPO trained on different reward functions on MiniGrid. Across the three environments, Tiered Reward show the fastest learning, outperforming the Action Penalty and Tier Based Shaping baselines. Tier Based Shaping performs differently according to how hard the exploration problem is: as discussed before, when s and s' are in the same tier, the shaping function $F(s, a, s') = \gamma\Phi(s') - \Phi(s) > 0$, and therefore encourages the agent to exploit rather than explore. For FourRooms, Tiered Reward initially learns slower than Tier Based Shaping, but later catches on and eventually outperforms its counterpart. All three reward functions suffer from large standard deviation in FourRooms likely because this environment is a hard exploration problem; different exploration during learning leads to wildly different outcomes.

6.3 Influence of More Tiers

Finally, we explore Tiered Reward with a varying number of tiers. First, we choose for simplicity and clarity four grid-world domains with absorbing goals and obstacles that are suited to a flexible number of tiers:

1. Chain: a 90–state 1D environment with left and right actions. Starting from one end, the agent tries to reach the other end with actions of success rate 80%; failed actions transition to the opposite direction.

2. Grid: a 9×9 grid where the agent starts in one corner and aims for the opposite corner. The agent can move in four cardinal directions with a 80% success rate, while slipping to either side with a 10% chance.

3. Frozen Lake: a slippery grid with holes that will swallow the agent (Figure 5a top). The objective is to reach the goal without falling into any holes. Each of the 4 directional actions succeed 1/3 of the time, and slip to either side with probability 1/3.

4. Wall Grid: a grid world with multiple lava states and wall states (Figure 5a bottom). The agent has to circle around the walls while avoiding the lava to get to the goal. Transition dynamics same as Grid.

For Chain and Grid, tiers are decided based on their $L1$ distance to the goal; for Frozen Lake and Wall Grid, tiers are based on the sum of $L1$ distance to the goal and start state, weighted $2:1$.

In absence of a "correct" reward function that expresses the tasks, we measure the learning speed by recording the steps required for the agent to reach the goal for the first time. For fair comparison, we optimistically initialize $Q_{init} = 10^5$ so that R_{tbs} also enjoys the benefit of directed exploration. The results are presented in Figure 5b. We repeat the same experiments with a model-based RL algorithm, RMAX (Brafman & Tennenholtz, 2002). The results are similar to that of Q-learning (Figure 9 in Appendix E). As expected, Tiered Reward makes learning faster as the number of tiers increases because more information about the environment aids reward design. Tiered Reward consistently beats action penalty and is at least as good as tier-based shaping reward, and often much better. Even when Tiered Reward performs the same as shaped reward, it provides the added benefit of simplicity and better interpretability—it is based only on states, and not (s, a, s') triples.

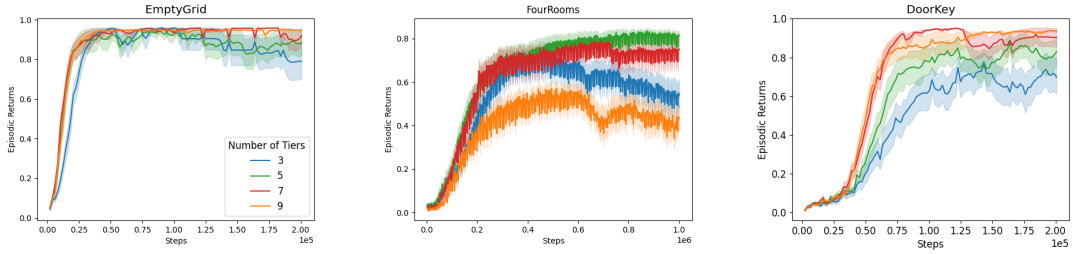


Figure 6: Learning curves on EmptyGrid, FourRooms, and DoorKey with different number of tiers in Tiered Reward. Error bars show standard deviation from 30 random seeds.

Adding more tiers achieves similar effects in deep RL settings under numerical constraints. Figure 6 shows the learning performance of Tiered Reward with 3, 5, 7, and 9 tiers on MiniGrid environments. In EmptyGrid, having more tiers makes learning faster and more stable. In DoorKey, 7 and 9 tiers produce similar results, both significantly faster than 3 or 5 tiers. In FourRooms, 5 tier leads to the fastest learning result, and using more than 5 tiers *monotonically* decreases learning performance. We believe this is because of numerical issues during scaling of Tiered Reward: we scale reward values to $[-1, 0]$ for stability in learning, but it also decreases the difference of reward value among tiers, especially among higher (more desirable) tiers. Reward values for all states in tiers higher than 3 gets normalized to very close to 0 (in the range 10^{-6} to 10^{-15}), and therefore tiers become indistinguishable under some numerical accuracy. Having more tiers suffers more from this problem because more states become indistinguishable, and as a result have worse performance. [1] We hypothesize that this issue can be resolved with a smarter reward scaling or clipping method, and leave that for future work. In practice, the optimal number of tiers can be determined a priori with MDP-specific information or empirically. Though more tiers sometimes leads to slower learning than 3 tiers, all Tiered Rewards provide faster learning than the two reward baselines in Figure 4.

[1] To alleviate the numerical issues caused by scaling, we set discount factor to 0.5 in these experiments to obtain smaller original Tiered Reward. See Appendix H for further details

7 Conclusion

In contrast to standard reward-design solutions that are environment-dependent, we presented Tiered Rewards—a class of environment-independent reward structures that provably leads to (Pareto) optimal behavior and empirically leads to fast learning. Tiered Reward can be defined in both tabular and high dimensional environments, and is RL-algorithm agnostic. Interesting future work includes getting theoretical guarantees that Tiered Reward lead to asymptotically faster learning and addressing the numerical issues with more tiers.

Acknowledgments

This research is supported in part by the Office of Naval Research (ONR) award N00014-20-1-2115.

References

David Abel, Will Dabney, Anna Harutyunyan, Mark K Ho, Michael Littman, Doina Precup, and Satinder Singh. On the expressivity of markov reward. *Advances in Neural Information Processing Systems*, 34:7799–7812, 2021.

Dario Amodei and Jack Clark. Faulty reward functions in the wild. *URL: https://blog. openai. com/faulty-reward-functions*, 2016.

Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*, 2016.

Ronen I. Brafman and Moshe Tennenholtz. R-MAX—a general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research*, 3:213–231, 2002.

Daniel Brown, Wonjoon Goo, Prabhat Nagarajan, and Scott Niekum. Extrapolating beyond suboptimal demonstrations via inverse reinforcement learning from observations. In *International conference on machine learning*, pp. 783–792. PMLR, 2019.

Alberto Camacho, Oscar Chen, Scott Sanner, and Sheila A McIlraith. Non-markovian rewards expressed in ltl: guiding search via reward shaping. In *Tenth annual symposium on combinatorial search*, 2017.

Alberto Camacho, Rodrigo Toro Icarte, Toryn Q Klassen, Richard Anthony Valenzano, and Sheila A McIlraith. Ltl and beyond: Formal languages for reward function specification in reinforcement learning. In *IJCAI*, volume 19, pp. 6065–6073, 2019.

Maxime Chevalier-Boisvert, Bolun Dai, Mark Towers, Rodrigo de Lazcano, Lucas Willems, Salem Lahlou, Suman Pal, Pablo Samuel Castro, and Jordan Terry. Minigrid. *CoRR*, abs/2306.13831, 2023.

Rati Devidze, Goran Radanovic, Parameswaran Kamalaruban, and Adish Singla. Explicable reward design for reinforcement learning agents. *Advances in Neural Information Processing Systems*, 34: 20118–20131, 2021.

Daniel Dewey. Reinforcement learning and the reward engineering principle. In *2014 AAAI Spring Symposium Series*, 2014.

Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Vlad Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, et al. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. In *International conference on machine learning*, pp. 1407–1416. PMLR, 2018.

Conor F Hayes, Roxana Rădulescu, Eugenio Bargiacchi, Johan Källström, Matthew Macfarlane, Mathieu Reymond, Timothy Verstraeten, Luisa M Zintgraf, Richard Dazeley, Fredrik Heintz, et al. A practical guide to multi-objective reinforcement learning and planning. *Autonomous Agents and Multi-Agent Systems*, 36(1):26, 2022.

Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Mark K Ho, Carlos G. Correa, and Daniel Ritter. Models of Sequential Decision Making (msdm), 5 2021. URL https://github.com/markkho/msdm.

Rodrigo Toro Icarte, Toryn Klassen, Richard Valenzano, and Sheila McIlraith. Using reward machines for high-level task specification and decomposition in reinforcement learning. In *International Conference on Machine Learning*, pp. 2107–2116. PMLR, 2018.

Rodrigo Toro Icarte, Toryn Q Klassen, Richard Valenzano, and Sheila A McIlraith. Reward machines: Exploiting reward function structure in reinforcement learning. *Journal of Artificial Intelligence Research*, 73:173–208, 2022.

Sven Koenig and Reid G Simmons. Complexity analysis of real-time reinforcement learning. In *AAAI*, volume 93, pp. 99–105, 1993.

Siri Leknes and Irene Tracey. Pain and pleasure: masters of mankind. *Pleasures of the brain*, pp. 320–335, 2010.

Xiao Li, Cristian-Ioan Vasile, and Calin Belta. Reinforcement learning with temporal logic rewards. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3834–3839. IEEE, 2017.

Michael L Littman, Ufuk Topcu, Jie Fu, Charles Isbell, Min Wen, and James MacGlashan. Environment-independent task specifications via gltl. *arXiv preprint arXiv:1704.04341*, 2017.

Runze Liu, Fengshuo Bai, Yali Du, and Yaodong Yang. Meta-reward-net: Implicitly differentiable reward learning for preference-based reinforcement learning. *Advances in Neural Information Processing Systems*, 35:22270–22284, 2022.

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

Fiorenzo Mornati. Pareto optimality in the work of pareto. *Revue européenne des sciences sociales. European Journal of Social Sciences*, (51-2):65–82, 2013.

Raghav Nagpal, Achyuthan Unni Krishnan, and Hanshen Yu. Reward engineering for object pick and place training. *arXiv preprint arXiv:2001.03792*, 2020.

Edita Navratilova and Frank Porreca. Reward and motivation in pain and pain relief. *Nature neuroscience*, 17(10):1304–1312, 2014.

Andrew Y Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Icml*, volume 99, pp. 278–287, 1999.

Takato Okudo and Seiji Yamada. Subgoal-based reward shaping to improve efficiency in reinforcement learning. *IEEE Access*, 9:97557–97568, 2021.

Frank Porreca and Edita Navratilova. Reward, motivation and emotion of pain and its relief. *Pain*, 158(Suppl 1):S43, 2017.

Julien Roy, Roger Girgis, Joshua Romoff, Pierre-Luc Bacon, and Christopher Pal. Direct behavior specification via constrained reinforcement learning. *arXiv preprint arXiv:2112.12228*, 2021.

Staurt J Russell and Peter Norvig. Artificial intelligence (a modern approach), 2010.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Joar Skalse, Nikolaus Howe, Dmitrii Krasheninnikov, and David Krueger. Defining and characterizing reward gaming. *Advances in Neural Information Processing Systems*, 35:9460–9471, 2022.

Jonathan Sorg, Richard L Lewis, and Satinder Singh. Reward design via online gradient ascent. *Advances in Neural Information Processing Systems*, 23, 2010.

Henry Sowerby, Zhiyuan Zhou, and Michael L Littman. Designing rewards for fast learning. 2022.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, 1998.

Rodrigo Toro Icarte, Toryn Q Klassen, Richard Valenzano, and Sheila A McIlraith. Teaching multiple tasks to an rl agent using ltl. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 452–461, 2018.

Peter Vamplew, Richard Dazeley, Adam Berry, Rustam Issabekov, and Evan Dekker. Empirical evaluation methods for multiobjective reinforcement learning algorithms. *Machine Learning*, 84 (1):51–80, 2011.

Kristof Van Moffaert and Ann Nowé. Multi-objective reinforcement learning using sets of pareto dominating policies. *The Journal of Machine Learning Research*, 15(1):3483–3512, 2014.

Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3):279–292, 1992.

Christian Wirth, Riad Akrour, Gerhard Neumann, Johannes Fürnkranz, et al. A survey of preference-based reinforcement learning methods. *Journal of Machine Learning Research*, 18 (136):1–46, 2017.

Tichakorn Wongpiromsarn, Ufuk Topcu, and Richard M Murray. Receding horizon control for temporal logic specifications. In *Proceedings of the 13th ACM international conference on Hybrid systems: computation and control*, pp. 101–110, 2010.

Yuexiang Zhai, Christina Baek, Zhengyuan Zhou, Jiantao Jiao, and Yi Ma. Computational benefits of intermediate rewards for goal-reaching policy learning. *Journal of Artificial Intelligence Research*, 73:847–896, 2022.

Haoqi Zhang, David C Parkes, and Yiling Chen. Policy teaching through reward function learning. In *Proceedings of the 10th ACM conference on Electronic commerce*, pp. 295–304, 2009.

Weichao Zhou and Wenchao Li. Programmatic reward design by example. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 9233–9241, 2022.

A Proof of Theorem 5.3

Proof. Let π^* be the optimal policy induced with Tiered Reward $R(s)$. Suppose, for the sake of contradiction, there exists some policy π that dominates π^*. Then, by our definition of Pareto dominance,

$$\sum_{i=0}^{t} o_i \leq \sum_{i=0}^{t} o_i^*, \quad \forall t = 0, 1, 2, ..., \infty,$$

$$\sum_{i=0}^{t} g_i \geq \sum_{i=0}^{t} g_i^*, \quad \forall t = 0, 1, 2, ..., \infty,$$

where o_t and g_t are the probabilities of reaching obstacles and goals in exactly t steps following π, and o_t^* and g_t^* are the same for π^*. We can write the value function (of π being evaluated on $R(s)$) as

$$V = \sum_{t=0}^{\infty} g_t(\gamma^t r_{goal} + \sum_{j=0}^{t-1} \gamma^j r_{back}) + o_t(\gamma^t r_{obs} + \sum_{j=0}^{t-1} \gamma^j r_{back}).$$

The value of π^* (V^*) can be written similarly. Denote

$$f_t^g = \gamma^t r_{goal} + \sum_{j=0}^{t-1} \gamma^j r_{back}, \text{ and}$$

$$f_t^o = \gamma^t r_{obs} + \sum_{j=0}^{t-1} \gamma^j r_{back}.$$

That is, f_g^t is the reward obtained on a trajectory that reaches a goal in t steps and f_o^t is the reward obtained on a trajectory that reaches an obstacle in t steps. With $r_{obs} < \frac{1}{1-\gamma} r_{back} < r_{goal}$, we show below that f_t^g is strictly decreasing and f_t^o strictly increasing with respect to t.

Proof that f_t^g is strictly decreasing:

$$\begin{aligned} f_{t+1}^g - f_t^g &= \gamma^{t+1} r_{goal} + \sum_{j=0}^{t} \gamma^j r_{back} - \gamma^t r_{goal} - \sum_{j=0}^{t-1} \gamma^j r_{back} \\ &= \gamma^t(\gamma - 1) r_{goal} + \gamma^t r_{back} \\ &= \gamma^t(1-\gamma)(\frac{1}{1-\gamma} r_{back} - r_{goal}) \\ &< 0 \end{aligned}$$

because $0 < \gamma < 1$ and $\frac{1}{1-\gamma} r_{back} < r_{goal}$.

Proof that f_t^o is strictly increasing:

$$\begin{aligned} f_{t+1}^o - f_t^o &= \gamma^{t+1} r_{obs} + \sum_{j=0}^{t} \gamma^j r_{back} - \gamma^t r_{obs} - \sum_{j=0}^{t-1} \gamma^j r_{back} \\ &= \gamma^t(\gamma - 1) r_{obs} + \gamma^t r_{back} \\ &= \gamma^t(1-\gamma)(\frac{1}{1-\gamma} r_{back} - r_{obs}) \\ &> 0 \end{aligned}$$

because $0 < \gamma < 1$ and $r_{obs} < \frac{1}{1-\gamma} r_{back}$.

Then,

$$\begin{aligned} V - V^* &= \sum_{t=0}^{\infty}(g_t - g_t^*)f_t^g + \sum_{t=0}^{\infty}(o_t - o_t^*)f_t^o \\ &= \sum_{t=0}^{\infty}(\sum_{j=0}^{t} g_j - g_j^*)(f_t^g - f_{t+1}^g) + \sum_{t=0}^{\infty}(\sum_{j=0}^{t} o_j - o_j^*)(f_t^o - f_{t+1}^o) \quad (*) \\ &> 0 + 0 \\ &= 0. \end{aligned}$$

The pass from the first equality to the second (*) is justified as follows:

$$\begin{aligned} \sum_{t=0}^{\infty}(g_t - g_t^*)f_t^g &= \sum_{t=0}^{\infty}\sum_{j=0}^{t}(g_j - g_j^*)f_t^g - \sum_{t=0}^{\infty}\sum_{j=0}^{t-1}(g_j - g_j^*)f_t^g \\ &= \sum_{t=0}^{\infty}\sum_{j=0}^{t}(g_j - g_j^*)f_t^g - \sum_{t=1}^{\infty}\sum_{j=0}^{t-1}(g_j - g_j^*)f_t^g \\ &= \sum_{t=0}^{\infty}\sum_{j=0}^{t}(g_j - g_j^*)f_t^g - \sum_{t'=0}^{\infty}\sum_{j=0}^{t'}(g_j - g_j^*)f_{t'+1}^g \\ &= \sum_{t=0}^{\infty}(\sum_{j=0}^{t} g_j - g_j^*)(f_t^g - f_{t+1}^g) \end{aligned}$$

Similarly,

$$\sum_{t=0}^{\infty}(o_t - o_t^*)f_t^o = \sum_{t=0}^{\infty}(\sum_{j=0}^{t} o_j - o_j^*)(f_t^o - f_{t+1}^o)$$

We have shown, through the value function, that π is strictly better than π^* with respect to the reward function R. But π^* was chosen to optimize R, so that's a contradiction. Since no such π can exist, that means π^* is not dominated by any policy, and is therefore Pareto-optimal. □

B Proof of Theorem 5.5

Proof. The proof is similar to that of Theorem 5.3. Suppose, for the sake of contradiction, that there exists some such policy π. We can express the value functions as

$$V = \sum_{t=0}^{\infty} \gamma^t \sum_{m=1}^{k} r_m \cdot p_t^m, \text{ and}$$

$$V^* = \sum_{t=0}^{\infty} \gamma^t \sum_{m=1}^{k} r_m \cdot p_t^{*m}.$$

Denote $f_t^m = \gamma^t r_m$. Then, $f_t^m - f_{t+1}^m = r_m \gamma^t (1-\gamma) \leq 0, \forall m$. It's easy to see $f_t^m - f_{t+1}^m$ is strictly increasing in m, so

$$\begin{aligned}
V - V^* &= \sum_{t=0}^{\infty} \gamma^t \sum_{m=1}^{k} r_m (p_t^m - p_t^{*m}) \\
&= \sum_{m=1}^{k} \sum_{t=0}^{\infty} f_t^m (p_t^m - p_t^{*m}) \\
&= \sum_{m=1}^{k} \sum_{t=0}^{\infty} (f_t^m - f_{t+1}^m)(\sum_{j=0}^{t} p_j^m - p_j^{*m}) \\
&> \sum_{m=1}^{k} \sum_{t=0}^{\infty} (f_t^1 - f_{t+1}^1)(\sum_{j=0}^{t} p_j^m - p_j^{*m}) \quad (**) \\
&= \sum_{t=0}^{\infty} (f_t^1 - f_{t+1}^1) \sum_{m=1}^{k} (\sum_{j=0}^{t} p_j^m - p_j^{*m}) \\
&= \sum_{t=0}^{\infty} (f_t^1 - f_{t+1}^1) \cdot \sum_{j=0}^{t} (\sum_{m=1}^{k} p_j^m - \sum_{m=1}^{k} p_j^{*m}) \\
&= \sum_{t=0}^{\infty} (f_t^1 - f_{t+1}^1) \cdot \sum_{j=0}^{t} (1 - 1) \\
&= 0
\end{aligned}$$

Note that the $(**)$ step is justified only because $\sum_{j=0}^{t} p_j^m - p_j^{*m} \geq 0, \forall m = [2..k], \forall t$. The inequalities show that π achieves higher reward than the optimal policy, which is a contradiction. No such π exists. □

C Visualization of Pareto-optimal Policies

Going back to the example of the Russell/Norvig grid, we can visualize how the probability of reaching the goal (g_t) and reaching lava (o_t) changes over time for different policies. Consider two simple policies on the Russell/Norvig grid—(1) going left from all states ("always left") and (2) going right from all states ("always right").

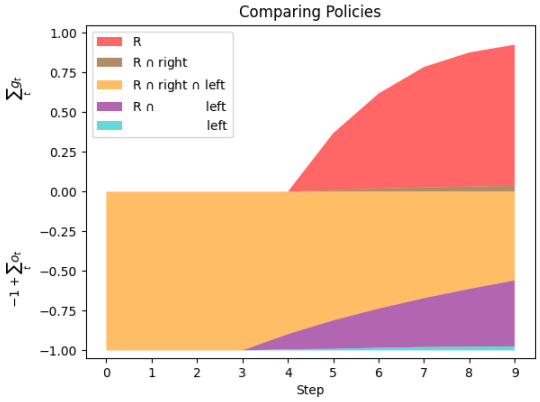


Figure 7: Visualization of the policies of always going left and always going right in the (stochastic) Russell/Norvig grid. The policy R is the same as R in Figure 8. To avoid color overlapping, we separated each policy into disjoint regions visualized by distinct colors. Each colored region in the figure represent the probability-region of one or more policies, joined by $\cap$. For example, the policy "always right" covers the areas in brown and orange.

We visualize each policy as a shaded area upper bounded by $\sum_t g_t$ and lower bounded by $-1+\sum_t o_t$ in Figure 7. This visualization can be understood as separating the probability space into two, with the goal-reaching probability on the top half of the y-axis in $[0, 1]$ and obstacle-hitting probability in the bottom half of the y-axis in $[-1, 0]$. With this visualization, a Pareto-dominated policy will cover an area that is entirely enclosed by that of a dominating policy because of lower goal-reaching probabilities on the top half and higher obstacle-hitting probabilities on the bottom half. As Figure 7 shows, "always right" and "always left" do not cover each other, so they are incomparable. Specifically, "always right" has a slightly higher probability of reaching the goal (brown), but "always left' has a lower probability of reaching the lava (purple and teal).

For comparison, we plot another policy, which we call R, that is state-dependent and moves in the direction of the goal. For policy R, the probability of reaching the target increases with time because each step has a 20% slip probability; agents could slip early on and take longer to reach the goal. Note that area covered by R (red, brown, orange, and purple) completely subsumes that of "always right" (brown and orange), demonstrating that "always right" is dominated by R. "Always left", on the other hand, is not dominated by R because it has a lower probability of reaching lava (teal). However, "always left" is not Pareto-optimal of course, because it is dominated by policy G in Table 1.

In Figure 8, we visualize the three policies R, G, and B in Table 1. Both R and G are Pareto-optimal, while B is Pareto-dominated because B's areas are entirely enclosed by that of R and of G.

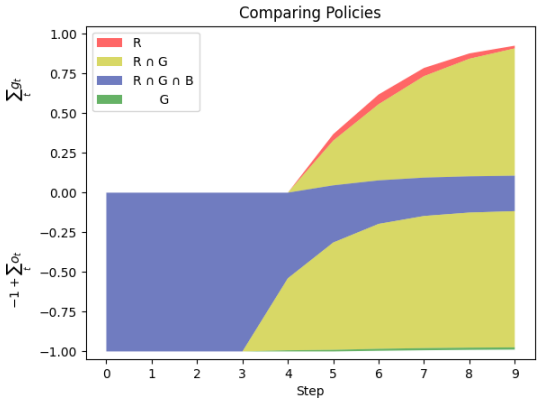


Figure 8: Visualization of three different policies (R, G, B) on Russell/Norvig grid. Visualization scheme is the same as described in Figure 7.

D Tabular Grid Worlds Experiment Details

The implementation of many tabular environments and algorithms is based on the MSDM library by Ho et al. (2021). All MSDM experiments are run on an Ubuntu 18.04 system with an Intel Core i7-9700K CPU and 32G of RAM.

Following Koenig & Simmons (1993), we use a greedy policy for action selection and initialize the Q-values optimistically to exploit directed exploration. Since all reward values are non-positive, it is sufficient to initialize the Q-values to 0. We use a learning rate of $\alpha = 0.90$ (tuned from $\alpha \in \{0.95, 0.90, 0.85\}$, all of which performed similarly). We set the small constant $\delta = 0.1$.

E RMAX results

We set maximal reward $r_{max} = 10^5$, use the first $m = 3$ transition samples to model the MDP (tuned from $m \in \{2, 3, 5, 7, 10\}$ and selected based on good resulting policy while taking similar learning time to Q-Learning), and did 200 iterations of value iteration during each update.

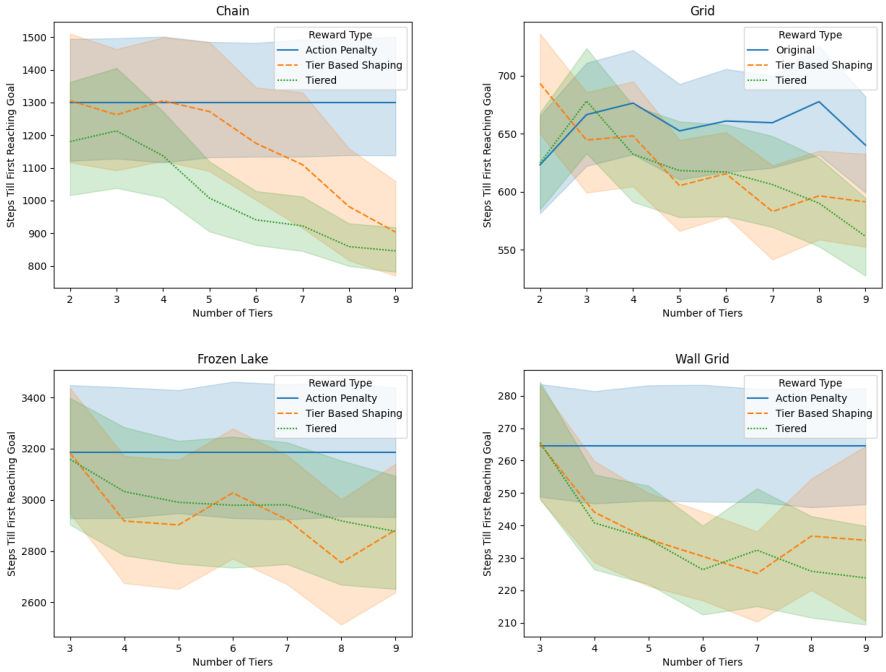


Figure 9: RMAX results with $\gamma = 0.9$. Each experiment was run with 300 seeds.

F MiniGrid Experiment Details

The DeepRL experiments on MiniGrid environment were run on an Ubuntu 18.04 system using a single 6GB GPU (NVIDIA GeForce GTX 980 Ti) and took between 40-60min per seed, depending on the environment.

A Proximal Policy Optimization (PPO) agent was trained on the three MiniGrid environments using the following hyperparameters:

Hyperparameter	Paramter Value	Note
Epochs	4	
Batch Size	256	
Learning Rate (α)	1×10^{-3}	
Total Env Steps	200000	Total training steps in each environment
Discount Factor (γ)	0.5	See Appendix H
Small Constant (δ)	5	Used to satisfy the strict inequality in Tiered Reward
GAE λ	0.95	λ coefficient in the GAE formula
Entropy Coefficient	0.01	
Value Loss Coefficient	0.5	
Max Grad Norm	0.5	Maximum norm of gradient
Clipping ϵ	0.2	

Table 2: PPO hyperparameters on MiniGrid experiments.

The PPO agent used an ImpalaCNN architecture (Espeholt et al., 2018) for the policy and value network. The architecture of the policy network is as follows:

Layer Name	Layer Type	Output Dimension
Conv Sequence 1	Convolutional Layers	16
Conv Sequence 2	Convolutional Layers	32
Conv Sequence 3	Convolutional Layers	32
Hidden Layer	Linear Layer	256
Logit Layer	Linear Layer	256
Value Layer	Linear Layer	256

Table 3: Architecture of the policy network of the Proximal Policy Optimization agent used in MiniGrid experiments.

Each Conv Sequence in Table 4 is a sequence of convolutional and pooling layers with residual connections. Each convolutional and pooling layer in the conv sequence has the same number of output channels, as specified in the output dimension in Table 4. The architecture of a Conv Sequence is as follows:

There are two residual connections: one between layers 3 and 6, and another between layers 7 and 10.

Layer #	**Layer Type**	**Kernel Size**	**Stride**	**Padding**
1	Conv	3	1	1
2	Max Pool	3	2	1
3	Relu			
4	Conv	3	1	1
5	Relu			
6	Conv	3	1	1
7	Relu			
8	Conv	3	1	1
9	Relu			
10	Conv	3	1	1

Table 4: Architecture of a Conv Sequence.

G Visualizing Additional Tiers on Empty Grid

To aid the understanding of Tiered Reward, we provide below visualizations of Tiered Reward on a grid world with different number of Tiers. In this grid world, the agent starts at the bottom right corner, and the goal is the top left corner. We assign the tiers based on a state's L_1 distance to the goal.

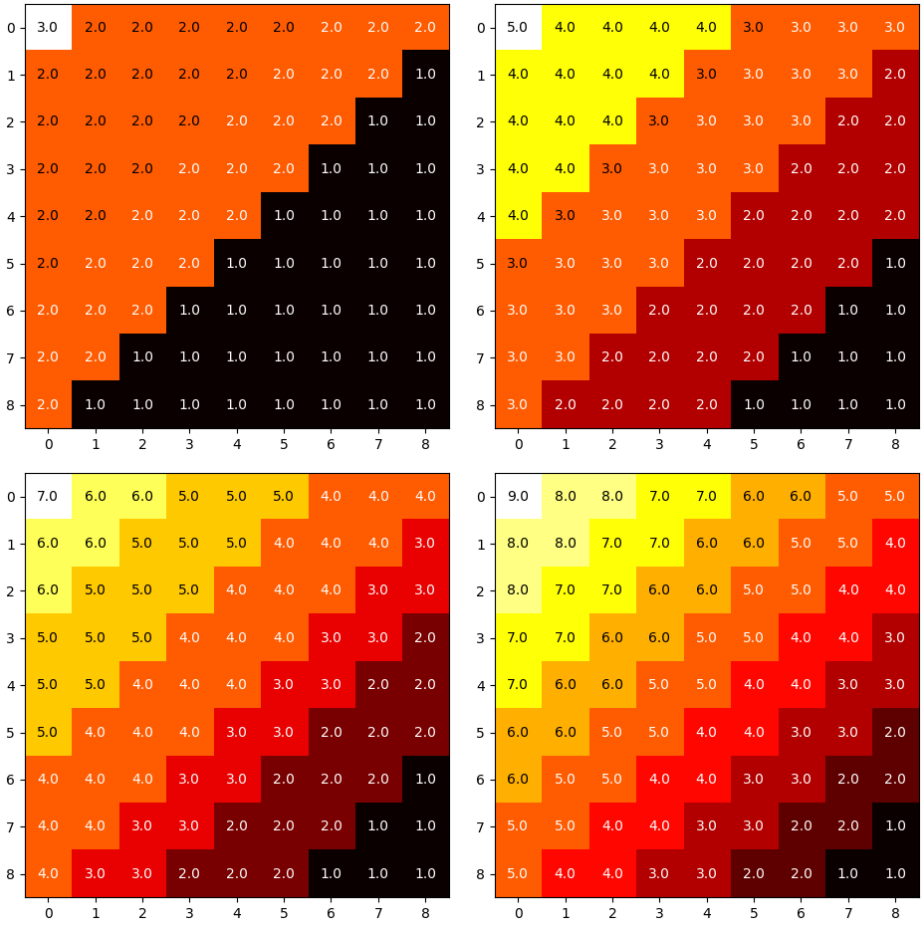


Figure 10: A visualization of Tiered Reward for 3 (top left), 5 (top right), 7 (bottom left), and 9 (bottom right) tiers on a grid world. Each color in the plot represents one single tiers, and the numbers represent the tier number.

H Scaled Tiered Reward for Different Tiers

We provide here the numerical values of the Tiered Reward after scaling it to be between $[-1, 0]$. This provides intuition on how using smaller discount values can reduce the numerical problems during reward scaling because big values of γ (for example, $\gamma = 0.99$ or $\gamma = 0.9$) will make scaled reward values numerically equal for different tiers, especially the higher tiers.

	Tier	$\gamma = 0.99$	$\gamma = 0.5$
Tiered Reward	5	0	0
	4	-5	-5
	3	-505	-15
	2	-50505	-35
	1	-5050505	-75
Scaled Tiered Reward	5	0	0
	4	-9.9×10^{-7}	-0.06
	3	-9.9×10^{-5}	-0.2
	2	-9×10^{-3}	-0.4667
	1	-1	-1

Table 5: Comparing scaled and unscaled reward values using a total of 5 tiers and $delta = 5$. Reward values for two discount factors (γ) are provided.

	tier	$\gamma = 0.99$	$\gamma = 0.5$
Tiered Reward	9	0	0
	8	-5	-5
	7	-505	-15
	6	-50505	-35
	5	-5050505	-75
	4	-505050505	-155
	3	-50505050505	-315
	2	-5050505050505	-635
	1	-505050505050505	-1275
Scaled Tiered Reward	9	0	0
	8	-9×10^{-15}	-0.0039
	7	-9×10^{-13}	-0.0118
	6	-9×10^{-11}	-0.2258
	5	-9×10^{-9}	-0.0588
	4	-1×10^{-6}	-0.1216
	3	-9×10^{-5}	-0.2471
	2	-0.01	-0.4980
	1	-1	-1

Table 6: Comparing scaled and unscaled reward values using a total of 9 tiers and $delta = 5$. Reward values for two discount factors (γ) are provided.

I Additional Figure: Pareto-optimal Rewards

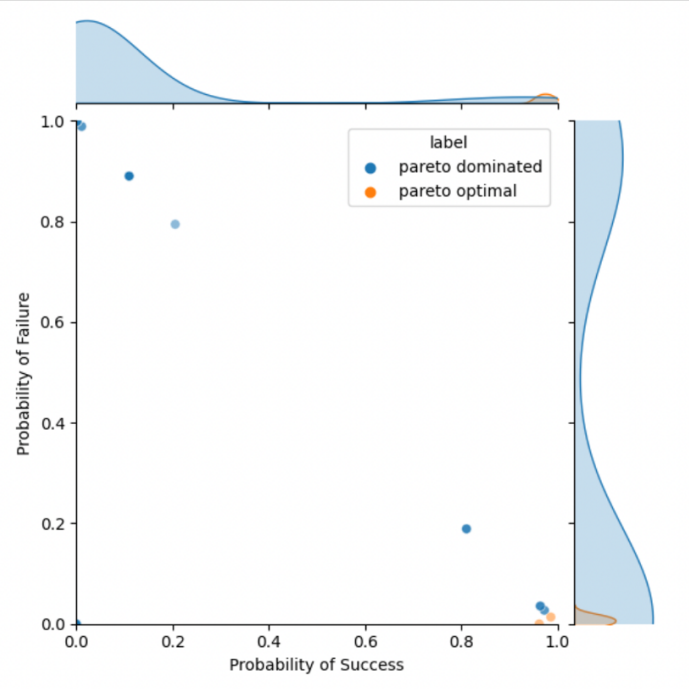


Figure 11: 1000 policies that are induced by random reward functions that satisfy $r_{lava} < r_{back} < r_{goal}$ from the Russell/Norvig grid. Each point in the scatter plot represents one policy's probability of success (reaching the goal) and failure (reaching the lava), showing that the majority (90.5%) of policies in the policy space are Pareto-dominated.