# Recursive Reward Aggregation

**Anonymous authors**
Paper under double-blind review

**Keywords:** Markov decision process, reward aggregation, policy preference, Bellman equation, algebraic data type, dynamic programming, recursion scheme, algebra fusion, bidirectional process

## Summary

In reinforcement learning (RL), agents typically learn desired behaviors by maximizing the (discounted) sum of rewards, making the design of reward functions crucial for aligning the agent behavior with specific objectives. However, since rewards often carry intrinsic meanings tied to the task, modifying them can be challenging and may introduce complex trade-offs in real-world scenarios. In this work, rather than modifying the reward function itself, we propose leveraging different reward aggregation functions to achieve different behaviors. By introducing an algebraic perspective on Markov decision processes, we show that the Bellman equations naturally emerge from the recursive generation and aggregation of rewards. This perspective enables the generalization of the standard discounted sum to other recursive aggregation functions, such as discounted max and variance-regularized mean. We empirically evaluate our approach across diverse environments using value-based, policy-based, and actor-critic algorithms, demonstrating its effectiveness in optimizing a wide range of objectives. Furthermore, we apply our method to a real-world portfolio optimization task, showcasing its potential for practical deployment in decision-making applications where objectives cannot easily be expressed as the discounted sum of rewards.

## Contribution(s)

1. We provide an algebraic perspective on Markov decision process based on algebra fusion and bidirectional process.
   **Context:** The algebra of recursive functions (Meijer et al., 1991; De Moor, 1994; Bird & de Moor, 1997; Hutton, 1999) is a well-studied topic in functional programming. The algebra fusion technique, explored in Hinze et al. (2010), has been applied in dynamic programming. In the context of RL, the recursive structure of the discounted sum of rewards was studied in Hedges & Sakamoto (2022). The diagrammatic representation of bidirectional processes for recursive reward generation and aggregation was inspired by Gavranović (2022).
2. We generalize the Bellman equations and Bellman operators for the standard discounted sum to other recursive aggregation functions, providing greater flexibility in RL optimization.
   **Context:** The problem of alternative reward aggregations is not entirely new. Prior works have explored objectives such as optimizing the maximum (Quah & Quek, 2006; Gottipati et al., 2020; Veviurko et al., 2024), minimum (Cui & Yu, 2023), top-k (Wang et al., 2020), and Sharpe ratio (Nägele et al., 2024) of rewards. Specifically, the method proposed by Cui & Yu (2023) is a special case of our framework, where the recursive structure is on the original reward space, and the update function is order-preserving.
3. We extend existing RL algorithms by incorporating the generalized Bellman operators and empirically demonstrate their effectiveness across various tasks.
   **Context:** While our method modifies the Bellman operators within the base RL algorithms, the fundamental structures of Q-learning (Watkins, 1989; Watkins & Dayan, 1992), PPO (Schulman et al., 2017), and TD3 (Fujimoto et al., 2018) remain unchanged.

# Recursive Reward Aggregation

**Anonymous authors**
Paper under double-blind review

## Abstract

In reinforcement learning (RL), aligning agent behavior with specific objectives typically requires careful design of the reward function, which can be challenging when the desired objectives are complex. In this work, we propose an alternative approach for flexible behavior alignment that eliminates the need to modify the reward function by selecting appropriate reward aggregation functions. By introducing an algebraic perspective on Markov decision processes, we show that the Bellman equations naturally emerge from the recursive generation and aggregation of rewards, allowing for the generalization of the standard discounted sum to other recursive aggregations, such as discounted max and variance-regularized mean. Our approach applies to both deterministic and stochastic settings and integrates seamlessly with value-based and policy-based RL algorithms. Experimental results demonstrate that our approach effectively optimizes diverse objectives, highlighting its versatility and potential for real-world applications.[1]

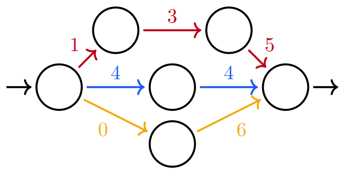
## 1 Introduction

In reinforcement learning (RL), an agent interacts with an environment modeled as a Markov decision process (MDP) to optimize a predefined objective. Traditionally, this objective is formulated as the discounted cumulative reward over an episode (Sutton & Barto, 1998; Kaelbling et al., 1996). This formulation has been widely adopted across various domains, including Atari games (Mnih et al., 2015), stock trading (Wu et al., 2020; Kabbani & Duman, 2022), and autonomous driving (Zhu et al., 2020; Kiran et al., 2021), where cumulative rewards effectively capture long-term performance.

However, in many real-world applications, optimizing solely for cumulative rewards may not fully align with the desired objectives. In some cases, the objective focuses on stability, making the minimization of reward variance more important than simply maximizing expected returns (Tamar et al., 2012; La & Ghavamzadeh, 2013). For instance, in finance, the Sharpe Ratio (Sharpe, 1966) prioritizes reducing return variance to improve risk-adjusted performance, while in process control, robust optimization (Nilim & El Ghaoui, 2005) is used to mitigate uncertainty and ensure system stability. Furthermore, in drug discovery, the goal is often to maximize the peak reward to identify the most effective compounds (Quah & Quek, 2006; Gottipati et al., 2020). Risk-sensitive applications like autonomous driving prioritize minimizing the worst-case outcome to ensure safety and robustness (Wang et al., 2020; Abouelazm et al., 2024). These examples illustrate that different objectives beyond cumulative reward optimization are necessary for effective decision-making.

The traditional approach to tailoring specific objectives in RL is to modify the reward function (Moody et al., 1998; Moody & Saffell, 2001; Nägele et al., 2024). However, this approach has several drawbacks. It often requires expanding the state space (Mannor & Tsitsiklis, 2011; Wang et al., 2020) or altering the underlying MDP structure (Ng et al., 1999), which increases computational complexity. Moreover, manually redesigning the reward function is challenging (Leike et al., 2017; Hadfield-Menell et al., 2017), making practical implementation difficult and potentially causing unintended goal misalignment (Amodei et al., 2016; Christiano et al., 2017).

[1] Code of implication: `https://anonymous.4open.science/status/RRA-534F`.




| | sum | mean | max | min | range |
|---|---|---|---|---|---|
| $\pi_1$ | $\underline{9}$ | 3 | 5 | 1 | 4 |
| $\pi_2$ | 8 | $\underline{4}$ | 4 | $\underline{4}$ | 0 |
| $\pi_3$ | 6 | 3 | $\underline{6}$ | 0 | $\underline{6}$ |

Figure 1: Illustration of three deterministic policies in a simple MDP, shown as color-coded paths, with their rewards on edges. The table on the right shows the aggregated rewards for each policy. We can observe that different aggregation functions lead to different policy preferences.

Given these challenges, a natural alternative to modifying the reward function is to optimize different aggregations of the existing reward signals, rather than relying solely on the standard cumulative reward formulation. As illustrated in Fig. 1, different aggregation functions can lead to distinct policy preferences in a simple MDP. This suggests that by appropriately choosing the reward aggregation method, we can directly influence policy behavior without modifying the reward or MDP structure.

Motivated by this insight, we propose a more general and flexible framework by leveraging algebraic structures that retains the original reward function and state representation while redefining the optimization objective. Instead of modifying the reward function or MDP structure, our approach extends the standard reward aggregation mechanism by decomposing it into a recursively designed statistic and aggregation function. This enables the optimization of various objectives, such as minimizing reward variance or maximizing the minimum reward, while maintaining computational efficiency. We further derive the corresponding Bellman equations, extend our method to policy gradient algorithms, and demonstrate its effectiveness in both discrete and continuous environments. Moreover, we show that our approach can be seamlessly integrated into state-of-the-art RL algorithms and validate its effectiveness through extensive experiments on various non-cumulative objectives.

**Related work** RL traditionally optimizes policies by maximizing cumulative rewards. However, in many cases, achieving the desired objective requires optimizing alternative criteria. A common approach is to either modify the reward function (Moody et al., 1998; Moody & Saffell, 2001; Nägele et al., 2024) or augment the state space (Mannor & Tsitsiklis, 2011; Wang et al., 2020; Veviurko et al., 2024), both of which introduce additional complexity and potential inefficiencies.

Another line of research has focused on modifying the Bellman equation, extending its formulation to optimize objectives beyond cumulative rewards. Quah & Quek (2006) introduced a learning rule for the maximum reward value function, later refined by Gottipati et al. (2020) to correct technical issues related to interchanging expectation and maximum operators. However, their approach is limited to deterministic environment. Cui & Yu (2023) further extended the Bellman update to non-cumulative rewards, yet their approach struggles with stochastic environment. Veviurko et al. (2024) proposed a max-based objective with convergence guarantees for both deterministic and stochastic cases, but their method is restricted to max aggregation and requires additional state augmentation. These limitations underscore the need for a unified framework that generalizes reward structures while ensuring computational efficiency and convergence in both deterministic and stochastic cases.

**Contributions** In this paper, we introduce an *algebraic perspective* on the MDP model, showing that the Bellman equations naturally emerge from the *recursive* generation and aggregation of rewards (Section 2). This perspective allows us to generalize the standard discounted sum to other recursive aggregation functions, such as discounted max and mean-variance (Section 3), while unifying deterministic and stochastic settings within the same framework (Section 4). We provide theoretical justification for our approach, which enables the optimization of various objectives beyond cumulative rewards while maintaining computational efficiency. Finally, we validate the effectiveness of our method in both discrete and continuous environments across various recursive reward aggregation functions, showcasing its flexibility and scalability in handling diverse reward structures (Section 5).

## 2 An algebraic perspective on Bellman equations

In this section, we introduce the standard MDP model (Puterman, 1994) for sequential decision-making problems from an algebraic perspective. Using a technique known as *fusion* in algebra and functional programming (Meijer et al., 1991; Hinze et al., 2010), we show that the *Bellman equations* (Bellman, 1966) naturally arise from the *recursive* generation and aggregation of rewards. This perspective reveals opportunities for generalizing to alternative reward aggregation functions.

In this section, we focus on the standard *discounted sum* of rewards and *deterministic* transitions and policies. We generalize them to other *aggregation functions* in Section 3 and *stochastic* transitions and policies in Section 4.

### 2.1 Preliminaries

**Notation** In this section, $S$ is the set of *states*, $A$ is the set of *actions*, and $R$ is the set of *rewards*, which can be finite or infinite. The dynamics of the environment is given by a (deterministic) *transition function* $\mathrm{p} : S \times A \to S$. An agent interacts with the environment by following a (deterministic) *policy* $\pi : S \to A$ that maps states to actions. A *reward function* $\mathrm{r} : S \times A \to R$ assigns a reward to each state-action pair. Furthermore, we assume that there is an *initial state* $s_0 \in S$ and a subset $S_\omega \subset S$ of *terminal states*, whose indicator function is $\omega$. The *horizon* $\Omega$ of the task can be fixed or varying, depending on the terminal condition $\omega$.

Moreover, $\{*\}$ denotes a *singleton* (any set with a single element $*$). $[R]$ denotes the set of *finite lists* of rewards, defined using the *empty list function* $\mathrm{nil} : \{*\} \to [R]$, which represents the empty list $[\,]$, and the *list constructor function* $\mathrm{cons} : R \times [R] \to [R]$, which prepends an element to a list. We have $\mathrm{cons}(r, [\,]) = [r]$ and $\mathrm{cons}(r_t, [r_{t+1}, \ldots, r_\Omega]) = [r_t, r_{t+1}, \ldots, r_\Omega]$, which we abbreviate as $r_{t:\Omega}$.

**Composite functions** Let us introduce some composite functions that are useful for defining the recursive generation of states, actions, and rewards. Given a policy $\pi : S \to A$, the *paring function* $\langle \mathrm{id}_S, \pi \rangle : S \to S \times A = s \mapsto (s, \pi(s))$ keeps a copy of the current state $s \in S$ and outputs the next action $\pi(s) \in A$.[2] Then, pre-composing this function with the transition function $\mathrm{p} : S \times A \to S$ and the reward function $\mathrm{r} : S \times A \to R$ yields two functions:

- (policy-dependent) *state transition* $\mathrm{p}_\pi : S \to S := \mathrm{p} \circ \langle \mathrm{id}_S, \pi \rangle = s \mapsto \mathrm{p}(s, \pi(s))$ and
- (policy-dependent) *state reward function* $\mathrm{r}_\pi : S \to R := \mathrm{r} \circ \langle \mathrm{id}_S, \pi \rangle = s \mapsto \mathrm{r}(s, \pi(s))$.

We use the subscripts $\pi$ to explicitly indicate the dependence on the policy $\pi$.

### 2.2 Recursive generation of rewards

Using the state transition $\mathrm{p}_\pi$ and reward function $\mathrm{r}_\pi$, we can generate states and rewards step by step:

$$\mathrm{step}_{\pi,\mathrm{p},\mathrm{r},\omega} : S \to \{*\} + R \times S := s \mapsto \begin{cases} * & s \in S_\omega, \\ (\mathrm{r}_\pi(s), \mathrm{p}_\pi(s)) & s \notin S_\omega. \end{cases} \tag{1}$$

Let us take a closer look at this step function. The codomain, $\{*\} + R \times S$, is the *disjoint union* $(+)$ of a *singleton* $\{*\}$, representing termination, and the *Cartesian product* $R \times S$ of rewards and states. At each step, the step function either halts by returning the *termination signal* $*$ if the current state $s$ is terminal or continues by returning a pair of the reward $\mathrm{r}_\pi(s) \in R$ and the next state $\mathrm{p}_\pi(s) \in S$, both determined by the policy $\pi$.

*Remark* 1 (Terminal condition). By incorporating the terminal condition $\omega$ into the step function, we can describe both *episodic* and *continuing* tasks for any reward aggregation, without relying on a special absorbing state and the unit of the aggregation function, e.g., $0$ for the discounted sum function. See also Sutton & Barto (1998, Section 3.4).

[2] For a set $C$, $\mathrm{id}_C : C \to C$ is the *identity function* mapping an element $c \in C$ to itself. For two functions $f : C \to A$ and $g : C \to B$, their *pairing* $\langle f, g \rangle : C \to A \times B$ is the unique function that applies these two functions to the same input, mapping an input $c \in C$ to a pair $(f(c), g(c)) \in A \times B$ of outputs.

Starting from an initial state, by recursively applying this step function and collecting the results, we can obtain a sequence of rewards:

**Definition 2.1** (Recursive generation)**.** Given a policy $\pi$, a transition function $\mathrm{p}$, a reward function $\mathrm{r}$, and a terminal condition $\omega$, a *recursive generation function* $\mathrm{gen}_{\pi,\mathrm{p},\mathrm{r},\omega} : S \to [R]$ of rewards is defined as follows:

$$\mathrm{gen}_{\pi,\mathrm{p},\mathrm{r},\omega} : S \to [R] := s \mapsto \begin{cases} [\,] & s \in S_\omega, \\ \mathrm{cons}(\mathrm{r}_\pi(s), \mathrm{gen}_{\pi,\mathrm{p},\mathrm{r},\omega}(\mathrm{p}_\pi(s))) & s \notin S_\omega. \end{cases} \tag{2}$$

### 2.3 Recursive aggregation of rewards

Given a sequence of rewards, we can aggregate them into a single value using an aggregation function. In the standard MDP setting, the *discounted sum* $\mathrm{sum}_\gamma : [R] \to R = r_{1:\Omega} \mapsto \sum_{t=1}^{\Omega} \gamma^{t-1} r_t$ of rewards is a common choice, where $\gamma \in [0, 1]$ is a *discount factor*.

Note that the discounted sum function can be expressed as a recursive function:

$$\mathrm{sum}_\gamma : [R] \to R := \begin{cases} [\,] & \mapsto & 0, \\ r_{t:\Omega} & \mapsto & r_t + \gamma \cdot \mathrm{sum}_\gamma(r_{t+1:\Omega}). \end{cases} \tag{3}$$

In other words, the discounted sum function is uniquely defined by two functions: the base case $0 \in R$ and the recursive case $r + \gamma \cdot s : R \times R \to R$. In Section 3, we will show that various other aggregation functions can also be defined recursively in this way.

### 2.4 Bellman equation for the state value function

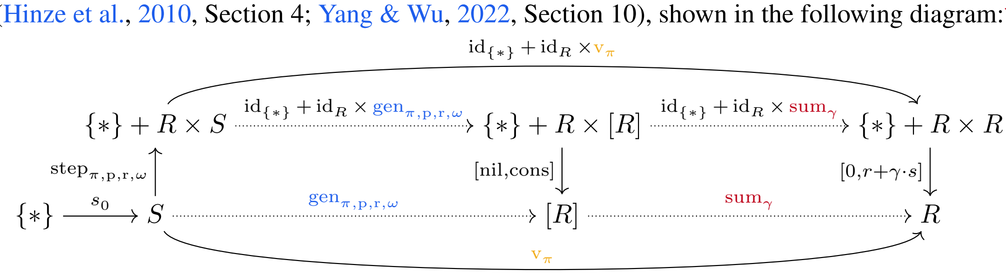
We have introduced the recursive generation and aggregation of rewards in a standard MDP model. The generation function $\mathrm{gen}_{\pi,\mathrm{p},\mathrm{r},\omega} : S \to [R]$ is the *producer* of rewards, and the discounted sum function $\mathrm{sum}_\gamma : [R] \to R$ is the *consumer* of rewards. By composing these two recursive functions, we obtain a *state value function* $\mathrm{v}_\pi : S \to R$, which can also be calculated recursively:

$$\mathrm{v}_\pi : S \to R := \mathrm{sum}_\gamma \circ \mathrm{gen}_{\pi,\mathrm{p},\mathrm{r},\omega} = s \mapsto \begin{cases} 0 & s \in S_\omega, \\ \mathrm{r}_\pi(s) + \gamma \cdot \mathrm{v}_\pi(\mathrm{p}_\pi(s)) & s \notin S_\omega. \end{cases} \tag{4}$$

This recursive calculation of the state value function $\mathrm{v}_\pi : S \to R$ is known as the *Bellman equation* (Bellman, 1966), which expresses the value of a state $s$ under a policy $\pi$ as the sum of the immediate reward $\mathrm{r}_\pi(s)$ and the discounted value of the next state $\mathrm{p}_\pi(s)$.

*Remark* 2 (State-action recursion). We can define the state-action transition/step/generation functions and derive a Bellman equation for the *state-action value function* $\mathrm{q}_\pi : S \times A \to R$ in a similar way, which is omitted here for brevity and discussed in Appendix A.

*Remark* 3 (Algebra fusion). For readers familiar with algebra and functional programming, we point out that the Bellman equation emerges as a consequence of the *fusion law* for recursive coalgebras (Hinze et al., 2010, Section 4; Yang & Wu, 2022, Section 10), shown in the following diagram:[3]



(5)

The left square is the recursive definition of the generation function in Eq. (2), and the right square is the recursive definition of the discounted sum function in Eq. (3). Consequently, the whole rectangle is the Bellman equation for the state value function in Eq. (4). See Appendix B for more details.

[3] For two functions $f : A \to C$ and $g : B \to C$, their *copairing* $[f, g] : A + B \to C$ is the unique function defined by cases, mapping an input $x \in A + B$ to $f(x)$ if $x \in A$, to $g(x)$ if $x \in B$.

Table 1: Recursive aggregation functions

| | definition $[R] \to R$ | initial value $\text{init} \in T$ | update function $\triangleright : R \times T \to T$ | post-processing $\text{post} : T \to R$ |
|---|---|---|---|---|
| discounted sum | $r_1 + \gamma r_2 + \cdots + \gamma^{t-1} r_t$ | $0 \in \mathbb{R}$ | $(r, s) \mapsto r + \gamma \cdot s$ | $\text{id}_{\mathbb{R}}$ |
| discounted max | $\max\{r_1, \gamma r_2, \ldots, \gamma^{t-1} r_t\}$ | $-\infty \in \overline{\mathbb{R}}$ | $(r, m) \mapsto \max(r, \gamma \cdot m)$ | $\text{id}_{\overline{\mathbb{R}}}$ |
| log-sum-exp | $\log(e^{r_1} + e^{r_2} + \cdots + e^{r_t})$ | $-\infty \in \overline{\mathbb{R}}$ | $(r, m) \mapsto \log(e^r + e^m)$ | $\text{id}_{\overline{\mathbb{R}}}$ |
| min | $\min(r_{1:t})$ | $\infty \in \overline{\mathbb{R}}$ | $(r, n) \mapsto \min(r, n)$ | $\text{id}_{\overline{\mathbb{R}}}$ |
| range | $\max(r_{1:t}) - \min(r_{1:t})$ | $\begin{matrix}\text{max}\\ \text{min}\end{matrix} \begin{bmatrix}-\infty \\ \infty\end{bmatrix} \in \overline{\mathbb{R}}^2$ | $\left(r, \begin{bmatrix}m \\ n\end{bmatrix}\right) \mapsto \begin{bmatrix}\max(r, m) \\ \min(r, n)\end{bmatrix}$ | $\begin{bmatrix}m \\ n\end{bmatrix} \mapsto m - n$ |
| mean | $\bar{r} := \frac{1}{t}\sum_{i=1}^{t} r_i$ | $\begin{matrix}\text{length}\\ \text{sum}\end{matrix} \begin{bmatrix}0 \\ 0\end{bmatrix} \in \begin{bmatrix}\mathbb{N} \\ \mathbb{R}\end{bmatrix}$ | $\left(r, \begin{bmatrix}n \\ s\end{bmatrix}\right) \mapsto \begin{bmatrix}n+1 \\ s+r\end{bmatrix}$ | $\begin{bmatrix}n \\ s\end{bmatrix} \mapsto \frac{s}{n}$ |
| | | $\begin{matrix}\text{length}\\ \text{mean}\end{matrix} \begin{bmatrix}0 \\ 0\end{bmatrix} \in \begin{bmatrix}\mathbb{N} \\ \mathbb{R}\end{bmatrix}$ | $\left(r, \begin{bmatrix}n \\ m\end{bmatrix}\right) \mapsto \begin{bmatrix}n+1 \\ \frac{r+n\cdot m}{n+1}\end{bmatrix}$ | $\begin{bmatrix}n \\ m\end{bmatrix} \mapsto m$ |
| variance | $\frac{1}{t}\sum_{i=1}^{t}(r_i - \bar{r})^2$ $= \frac{1}{t}\sum_{i=1}^{t} r_i^2 - \bar{r}^2$ | $\begin{matrix}\text{length}\\ \text{sum}\\ \text{sum square}\end{matrix} \begin{bmatrix}0 \\ 0 \\ 0\end{bmatrix} \in \begin{bmatrix}\mathbb{N} \\ \mathbb{R} \\ \mathbb{R}_{\geq 0}\end{bmatrix}$ | $\left(r, \begin{bmatrix}n \\ s \\ q\end{bmatrix}\right) \mapsto \begin{bmatrix}n+1 \\ s+r \\ q+r^2\end{bmatrix}$ | $\begin{bmatrix}n \\ s \\ q\end{bmatrix} \mapsto \frac{q}{n} - \left(\frac{s}{n}\right)^2$ |

## 3 Recursive reward aggregation functions

In this section, we generalize the discounted sum function in Eq. (3) to other recursive aggregation functions that summarize a sequence of rewards into a single value. Our primary goal is to derive a generalized Bellman equation extending Eq. (4) and provide theoretical insights for efficient policy evaluation and optimization with recursive reward aggregation.

### 3.1 Bellman equation for the state statistic function

First, we observe that many aggregation functions are inherently recursive; however, the recursive structure does not always operate directly within the original space. For instance, we can calculate the arithmetic mean by calculating both the sum and the length recursively and then dividing the sum by the length. Based on this observation, we propose the following definition:

**Definition 3.1** (Recursive aggregation). Let $T$ be a set of *statistics*. Given an *initial value* $\text{init} \in T$, an *update function* $\triangleright : R \times T \to T$, and a *post-processing function* $\text{post} : T \to R$, a *recursive aggregation function* $\text{agg}_{\text{init},\triangleright} : [R] \to T$ of statistics is defined as follows:

$$\text{agg}_{\text{init},\triangleright} : [R] \to T := \begin{cases} [\,] & \mapsto \quad \text{init}, \\ r_{t:\Omega} & \mapsto \quad r_t \triangleright \text{agg}_{\text{init},\triangleright}(r_{t+1:\Omega}), \end{cases} \tag{6}$$

and a *recursive aggregation function* $\text{post} \circ \text{agg}_{\text{init},\triangleright} : [R] \to R$ of rewards is the composition of this function with the post-processing function $\text{post} : T \to R$, shown in the following diagram:

$$\begin{array}{ccccc} \{*\} + R \times [R] & \xrightarrow{\text{id}_{\{*\}} + \text{id}_R \times \text{agg}_{\text{init},\triangleright}} & \{*\} + R \times T & & \\ {\scriptstyle [\text{nil},\text{cons}]}\downarrow & & \downarrow{\scriptstyle [\text{init},\triangleright]} & & \\ [R] & \xrightarrow{\text{agg}_{\text{init},\triangleright}} & T & \xrightarrow{\text{post}} & R \end{array} \tag{7}$$

Examples of recursive reward aggregation functions are provided in Table 1. By substituting the discounted sum function with a general recursive reward aggregation function, we can generalize the Bellman equation in Eq. (4) as follows:

**Theorem 3.2** (Bellman equation for the state statistic function). *Given a recursive generation function* $\text{gen}_{\pi,\text{p},\text{r},\omega}$ *(Definition 2.1) and a recursive statistic aggregation function* $\text{agg}_{\text{init},\triangleright}$ *(Definition 3.1), their composition, called the state statistic function* $\tau_\pi : S \to T$*, satisfies the following equation:*

$$\tau_\pi : S \to T := \text{agg}_{\text{init},\triangleright} \circ \text{gen}_{\pi,\text{p},\text{r},\omega} = s \mapsto \begin{cases} \text{init} & s \in S_\omega, \\ \text{r}_\pi(s) \triangleright \tau_\pi(\text{p}_\pi(s)) & s \notin S_\omega. \end{cases} \tag{8}$$

The state value function $\text{v}_\pi : S \to R := \text{post} \circ \tau_\pi$ is the composition of the state statistic function $\tau_\pi : S \to T$ with the post-processing function $\text{post} : T \to R$.

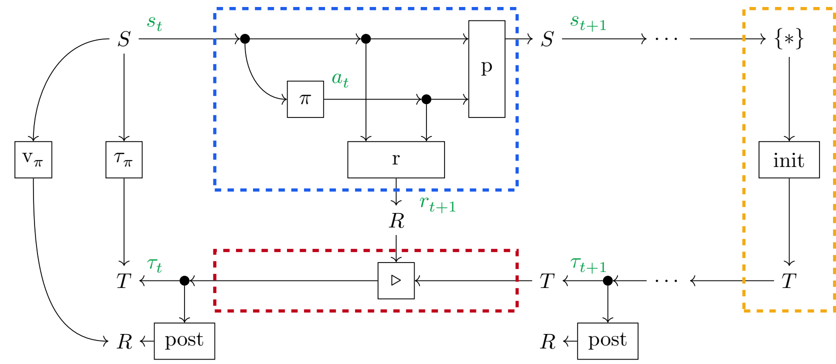



Figure 2: State statistic bidirectional process $\tau_\pi : S \to T$ and state value function $\mathrm{v}_\pi : S \to R$, showing the forward process, backward process, and termination.

*Remark* 4 (Bidirectional process). By combining the recursive generation and aggregation processes, we can express the state statistic function $\tau_\pi : S \to T$ as a composition of *bidirectional processes*, as illustrated in Fig. 2. The *forward process* $S \to R \times S$, parameterized by a policy $\pi$, takes a state $s_t \in S$ and generates a reward $r_{t+1} \in R$ and the next state $s_{t+1} \in S$. The *backward process* $R \times T \to T$ takes a statistic $\tau_{t+1} \in T$ from the future and updates it with the previously generated reward $r_{t+1} \in R$ to produce the current statistic $\tau_t \in T$. These bidirectional processes continue until a terminal state is reached, at which point its statistic is assigned the initial value $\mathrm{init} \in T$. Such bidirectional processes (Riley, 2018) have been applied to study supervised learning (Fong & Johnson, 2019), Bayesian inference (Smithe, 2020), gradient-based learning (Cruttwell et al., 2022), and reinforcement learning (Hedges & Sakamoto, 2022). See Appendix B for more details.

### 3.2 Policy evaluation: Iterative statistic function estimation

Next, we consider how to estimate the state statistic function $\tau_\pi : S \to T$ for an arbitrary policy $\pi$, known as the *policy evaluation* problem (Sutton & Barto, 1998, Sections 4.1 and 11.4). We introduce a generalized *Bellman operator* and prove the uniqueness of its fixed points under certain conditions. This result enables iterative statistic/value function estimation used in *policy iteration* and modern *actor-critic* methods (Barto et al., 1983; Mnih et al., 2016; Haarnoja et al., 2018; Fujimoto et al., 2018). Concretely, the Bellman operator is defined as follows:

**Definition 3.3** (Bellman operator)**.** Given a policy $\pi$, a transition function $\mathrm{p}$, a reward function $\mathrm{r}$, a terminal condition $\omega$, and a recursive statistic aggregation function $\mathrm{agg}_{\mathrm{init},\triangleright}$ (Definition 3.1), the *Bellman operator* $\mathcal{B}_\pi : [S, T] \to [S, T]$ for a function $\tau : S \to T$ is defined by

$$\mathcal{B}_\pi \tau : S \to T := s \mapsto \begin{cases} \mathrm{init} & s \in S_\omega, \\ \mathrm{r}_\pi(s) \triangleright \tau(\mathrm{p}_\pi(s)) & s \notin S_\omega. \end{cases} \tag{9}$$

According to the Bellman equation in Theorem 3.2, we have $\mathcal{B}_\pi \tau_\pi = \tau_\pi$, which means that the state statistic function $\tau_\pi$ is a fixed point of the Bellman operator. Then, we can generalize the classical fixed point theorem under the following condition:

**Definition 3.4** (Contractive update function)**.** An update function $\triangleright : R \times T \to T$ is *contractive* with respect to a premetric $d_T$ on statistics $T$ if $\forall r \in R.\ \forall t_1, t_2 \in T.\ d_T(r \triangleright t_1, r \triangleright t_2) \leq k \cdot d_T(t_1, t_2)$, where $k \in [0, 1)$ is a constant. In other words, $r \triangleright (-) : T \to T$ is a contraction for all $r \in R$.

**Theorem 3.5** (Uniqueness of fixed points of Bellman operator)**.** *Let $\tau_1, \tau_2 : S \to T$ be fixed points of the Bellman operator $\mathcal{B}_\pi$ (Definition 3.3). If the update function $\triangleright$ is contractive with respect to a premetric $d_T$ on statistics $T$ (Definition 3.4), then $d_T(\tau_1(s), \tau_2(s)) = 0$ for all states $s \in S$. If $d_T$ is a strict premetric, then $\tau_1 = \tau_2 = \tau_\pi$.*

This result applies to a broad class of recursive aggregation functions beyond the discounted sum. See Appendix C for further discussion on the premetric $d_T$ and the Bellman operator $\mathcal{B}_\pi$.

### 3.3 Policy optimization: Optimal policies and optimal value functions

Finally, we consider how to find an *optimal policy* and compute its statistic/value functions recursively based on the Bellman equation in Theorem 3.2:

**Definition 3.6** (Optimal policy). Given a preorder $\leq_T$ on statistics $T$, a policy $\pi_*$ is an *optimal policy* if $\forall \pi.\ \forall s \in S.\ \tau_\pi(s) \leq_T \tau_{\pi_*}(s)$, which has the *optimal state statistic function* $\tau_* : S \to T := \tau_{\pi_*}$ and the *optimal state value function* $\mathrm{v}_* : S \to R := \mathrm{post} \circ \tau_*$.

**Theorem 3.7** (Bellman optimality equation for the state statistic function). *Given a preorder $\leq_T$ on statistics $T$, the optimal state statistic function $\tau_*$ (Definition 3.6) satisfies the following equation:*

$$\tau_* : S \to T := s \mapsto \begin{cases} \mathrm{init} & s \in S_\omega, \\ \sup\limits_{a \in A}(\mathrm{r}(s,a) \triangleright \tau_*(\mathrm{p}(s,a))) & s \notin S_\omega. \end{cases} \tag{10}$$

Definition 3.6 and Theorem 3.7 are analogous to their classical counterparts (Sutton & Barto, 1998, Section 3.6), but they extend to arbitrary recursive aggregation functions and allow comparisons using a preorder $\leq_T$ on statistics. A *Bellman optimality operator* $\mathcal{B}_*$ can be defined similarly to the Bellman operator in Definition 3.3, and we can prove the uniqueness of its fixed points under certain conditions. This result enables the *value iteration* algorithm (Sutton & Barto, 1998, Section 4.4), *temporal difference* methods such as *Q-learning* (Watkins, 1989), and deep Q-network (DQN) based methods (Mnih et al., 2013; Bellemare et al., 2017) to find the optimal policy $\pi_*$. See Appendix D for further discussion on the preorder $\leq_T$ and the Bellman optimality operator $\mathcal{B}_*$.

## 4 From deterministic to stochastic Markov decision processes

In this section, we briefly discuss the extension of our framework to the stochastic setting. We show that the deterministic and stochastic settings share a fundamental similarity: all *recursive structures* remain unchanged, except that (deterministic) functions are replaced by *stochastic functions*, and function composition is replaced by marginalization over the intermediate variable, as described by the *Chapman–Kolmogorov equation* (Giry, 1982; Puterman, 1994). The main difference is that the stochastic setting allows for a richer class of aggregation functions (Bellemare et al., 2023), where the non-commutativity and non-distributivity of certain operations can lead to more complex behaviors.

**Notation** Slightly abusing notation, we use the same symbols to denote the *measurable spaces* of states $S$, actions $A$, rewards $R$, and statistics $T$. For a measurable space $C$, we write $\mathbb{P}C$ for the measurable space of all *probability measures* on $C$, and we denote by $\delta_c \in \mathbb{P}C$ the *Dirac measure* concentrated at $c \in C$. An identity stochastic function $\mathrm{id}_C : C \to \mathbb{P}C : c \mapsto \delta_c$ maps an element $c \in C$ to the Dirac measure $\delta_c \in \mathbb{P}C$. We consider stochastic transition $\mathrm{p} : S \times A \to \mathbb{P}S$ and policy $\pi : S \to \mathbb{P}A$, while other functions can be deterministic. We also use the usual conditional distribution notation such as $\mathrm{p}(s'|s,a)$ and $\pi(a|s)$.

**Stochastic composite functions** In the stochastic setting, we can compose two stochastic functions by marginalizing over the intermediate variable. Additionally, we can compose a stochastic function with a deterministic one using the *pushforward* operation, which is equivalent to treating deterministic functions as stochastic functions to Dirac measures. For example, we can define

- *stochastic state transition* $\mathrm{p}_\pi : S \to \mathbb{P}S := \mathrm{p} \circ \langle \mathrm{id}_S, \pi \rangle = s \mapsto s' \sim \int_A \mathrm{p}(s'|s,a)\pi(a|s)\,\mathrm{d}a$ and
- *stochastic state reward function* $\mathrm{r}_\pi : S \to \mathbb{P}R := \mathrm{r} \circ \langle \mathrm{id}_S, \pi \rangle = s \mapsto r \sim \int_A \delta_{\mathrm{r}(s,a)}(r)\pi(a|s)\,\mathrm{d}a$.

**Stochastic recursive functions** Analogous to Theorem 3.2, we can derive the recursive calculation of the *stochastic state statistic function* $\tau_\pi : S \to \mathbb{P}T$, known as the *distributional Bellman equation* (Morimura et al., 2010a;b; Bellemare et al., 2017), for any recursive aggregation function $\mathrm{agg}_{\mathrm{init},\triangleright}$:

$$\tau_\pi : S \to \mathbb{P}T = s \mapsto \tau \sim \begin{cases} \delta_{\mathrm{init}} & s \in S_\omega, \\ r \triangleright \tau' \;\Big|\; r \sim \mathrm{r}_\pi(r|s), \tau' \sim \displaystyle\int_S \tau_\pi(\tau'|s')\mathrm{p}_\pi(s'|s)\,\mathrm{d}s' & s \notin S_\omega. \end{cases} \tag{11}$$

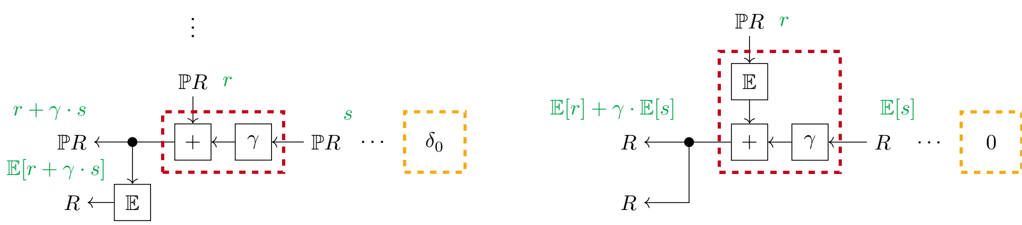



Figure 3: The recursive structures of the expected discounted sum of rewards $\mathbb{E}[r + \gamma \cdot s]$ and the discounted sum of expected rewards $\mathbb{E}[r] + \gamma \cdot \mathbb{E}[s]$ showing the update function and initial value.

**Stochastic aggregation functions** Note that this framework also accommodates the traditional *expected discounted sum of rewards* $\mathbb{E}\left[\sum_{t=1}^{\Omega} \gamma^{t-1} r_t\right]$ learning objective, by selecting $\delta_0$ as init, the (pushforward through) discounted addition function $r + \gamma \cdot s : R \times R \to R$ as the update function $\triangleright$, and the expectation operator $\mathbb{E} : \mathbb{P}R \to R$ as post. The stochastic statistic function $\tau_\pi : S \to \mathbb{P}R$ in Eq. (11), refered to as the *value distribution* in Bellemare et al. (2017), outputs the distribution of the discounted sum of rewards, while the value function outputs its expectation. Since the expectation distributes over the discounted addition, by changing the update function and initial value, we can recursively calculate the *discounted sum of expected rewards* $\sum_{t=1}^{\Omega} \gamma^{t-1} \mathbb{E}[r_t]$ instead (see Fig. 3), which is the traditional approach in RL (Sutton & Barto, 1998). In this case, the statistic function and the value function coincide, as no post-processing is required. However, Bellemare et al. (2017) have shown that even in the discounted sum setting, the Bellman operator may be a contraction in some metrics but not in others, while the Bellman optimality operator is a contraction only in expectation and not in any distributional metric, leading to different convergence behaviors. These challenges persist and may become unavoidable when using alternative aggregation functions due to the inconsistency between expected aggregated rewards and aggregated expected rewards. We discuss this further in Appendix E and leave a full investigation for future work.

## 5 Experiments

In this section, we empirically evaluate the proposed *recursive reward aggregation* technique across a variety of environments and optimization objectives to support the following claims:

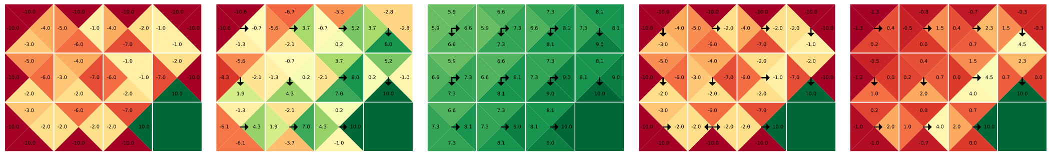
- Different aggregation functions significantly influence policy preferences. Selecting an appropriate aggregation function is an alternative approach to optimizing policies for specific objectives and aligning agent behaviors with task-specific goals without modifying rewards (Sections 5.1 to 5.3).
- In challenging real-world applications such as portfolio optimization, our method can directly optimize desired evaluation criteria, demonstrating superior performance compared to existing approaches and showcasing its practical effectiveness (Section 5.4).

### 5.1 Grid-world: Value-based methods for discrete planning

First, we present illustrative experiments in a simple grid-world environment to demonstrate the fundamental impact of different recursive reward aggregation functions on learned policies.

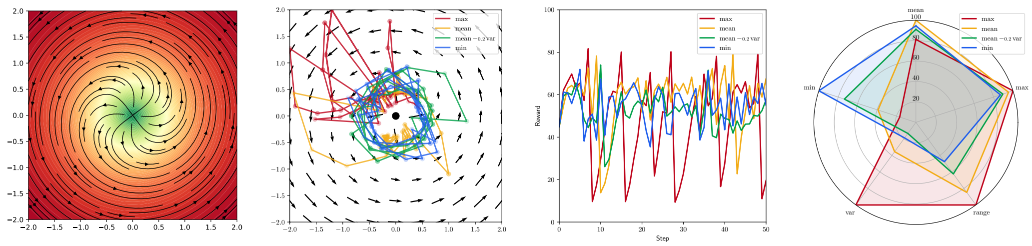
**Environment** Fig. 4a shows the results for a $3 \times 4$ grid environment, where an agent navigates from the top-left corner to a fixed goal at the bottom-right corner. As shown in Fig. 4a, the agent receives a small negative reward at each step, which varies across states, and a positive reward upon reaching the terminal state.

**Method** For this discrete environment, we modified the Q-learning algorithm (Watkins, 1989; Watkins & Dayan, 1992) using the Bellman optimality operator introduced in Section 3.3 (more specifically, the one for the state-action statistic function in Definition D.9). We used four recursive aggregation functions: discounted sum, discounted max, min, and mean, as detailed in Table 1. The detailed algorithm is provided in Algorithm 1 in Appendix G.



(a) Environment (b) $\text{sum}_{0.9}$ (c) $\max_{0.9}$ (d) min (e) mean

Figure 4: **Grid-world**: Fig. 4a shows the discrete environment and the reward function $\mathrm{r}(s, a)$, where the agent starts from the top-left corner and needs to reach the goal at the bottom-right corner. Figs. 4b to 4e show the optimal state-action value functions $\mathrm{q}_*(s, a)$ under different aggregation functions.



(a) Environment (b) Trajectory (c) Rewards (d) Metrics

Figure 5: **Wind-world**: Fig. 5a shows the continuous environment, where the agent encounters wind disturbances (visualized with streamlines) and receives higher rewards near the center (depicted with colored contours). Fig. 5b illustrates the trajectories of agents trained using different aggregation functions, while Fig. 5c compares the rewards obtained by each agent. Fig. 5d presents the evaluation metrics, highlighting the impact of aggregation functions on performance.

**Results** Compared to the standard discounted sum aggregation (Fig. 4b), optimizing for discounted max reward (Fig. 4c) makes the agent indifferent to intermediate costs, favoring shorter paths to the goal. In contrast, minimum aggregation (Fig. 4d) encourages risk-averse behavior, while mean aggregation (Fig. 4e) promotes efficiency by maximizing average reward per step. Further results and discussions are provided in Appendix H.1. Overall, these results demonstrate how each aggregation function uniquely impacts reward evaluation and policy preferences.

### 5.2 Wind-world: Policy improvement methods for trajectory optimization

Next, we show that the recursive reward aggregation technique can also be seamlessly integrated into methods for continuous state and action spaces to optimize trajectories in complex environments.

**Environment** Inspired by Dorfman et al. (2021); Ackermann et al. (2024), we designed a two-dimensional continuous environment where an agent navigates to a fixed goal amidst varying wind disturbances, as shown in Fig. 5a. This setup allows us to evaluate the impact of different aggregation functions on trajectory optimization.

**Method** For this continuous environment, we utilized the Proximal Policy Optimization (PPO) algorithm (Schulman et al., 2017), which is a widely used policy improvement method. We estimated the value function using the Bellman operator for the state statistic function in Definition 3.3. The detailed algorithm is provided in Algorithm 2 in Appendix G.

**Results** The results in Figs. 5b to 5d show that different aggregation functions lead to distinct trade-offs in trajectory optimization. Specifically, the max aggregation function prioritizes high-reward paths, while the min function ensures more conservative and consistent behavior. The variance-regularized mean aggregation provide balanced strategies, demonstrating the flexibility of the recursive reward aggregation technique in optimizing diverse objectives.

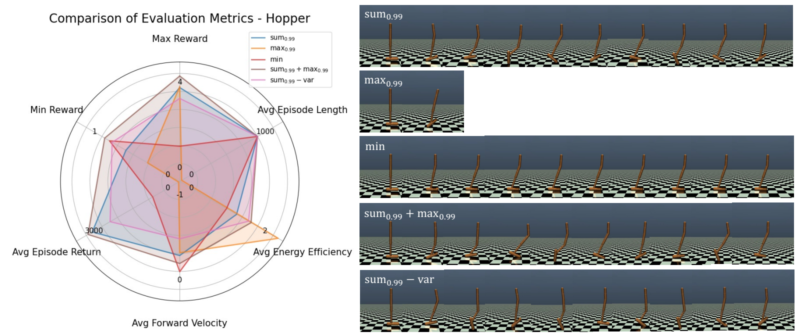



Figure 6: Comparison of evaluation metrics for different reward aggregation methods in the Hopper environment. The radar chart on the left visualizes the performance of different reward aggregation functions across multiple evaluation metrics over four random seeds. The images on the right illustrate the learned behavior of the agent for each reward aggregation method.

### 5.3 Physics simulation: Actor-critic methods for continuous control

Then, we extend our evaluation to more complex physics simulation environments.

**Environment** We conducted experiments on three continuous control environments: Hopper and Ant belong to the MuJoCo environment suite (Todorov et al., 2012), while Lunar Lander Continuous (Brockman et al., 2016) is from Box2D environment. A detailed description of these environments can be found in Appendix H.3.

**Method** In these experiments, we employed the Twin Delayed Deep Deterministic Policy Gradient (TD3) algorithm (Fujimoto et al., 2018), with a modified recursive version detailed in Algorithm 3 in Appendix G. To evaluate policy performance, we considered five different reward aggregation functions: discounted sum ($\mathrm{sum}_{0.99}$), discounted max ($\mathrm{max}_{0.99}$), min ($\mathrm{min}$), discounted sum plus max ($\mathrm{sum}_{0.99} + \mathrm{max}_{0.99}$), and discounted sum minus variance ($\mathrm{sum}_{0.99} - \mathrm{var}$).

**Results** The result for Hopper are provided in Fig. 6, with results for other environments in Appendix H.3. We present the mean values of various metrics across four random seeds using radar charts, and visualize agent trajectories to illustrate the impact of aggregation functions on the learned policy. The $\mathrm{sum}_{0.99}$ aggregation, serving as the baseline method, demonstrates strong overall performance across multiple metrics, as reflected in both the radar chart and motion sequences. In contrast, the $\mathrm{max}_{0.99}$ aggregation focuses solely on optimizing max reward, leading to strong performance in this specific metric but suboptimal outcomes in others. The corresponding images show the agent taking overly aggressive actions to maximize max reward, which causes it to lose balance quickly as the torso angle exceeds the allowed range. The $\mathrm{min}$ aggregation encourages the agent to maximize the minimum reward, which leads to a conservative strategy where the agent remains completely still to avoid negative rewards. The $\mathrm{sum}_{0.99} + \mathrm{max}_{0.99}$ aggregation encourages the agent to optimize both the total reward and the maximum reward within an episode, leading to more aggressive movements and higher overall rewards. While the $\mathrm{sum}_{0.99} - \mathrm{var}$ aggregation prioritizes stability by minimizing the difference between the maximum and minimum rewards, resulting in more controlled and consistent behavior at the cost of slightly lower rewards. These results highlight how different reward aggregation strategies shape the behavior of the agent and its learning outcomes. Demonstration videos are provided in our anonymized code link.

### 5.4 Real-world application: Sharpe ratio in portfolio optimization

Lastly, we evaluated the practical applicability of our method in a real-world application. Portfolio optimization is a fundamental real-world application where an agent (or investor) determines the

Table 2: Performance comparison of different methods for portfolio optimization using the Sharpe ratio. The table reports the mean and standard deviation of the Sharpe ratio across five random seeds during the test period, where a higher value indicates better risk-adjusted returns.

| | DiffSharpe | NCMDP | Ours |
|---|---|---|---|
| Sharpe Ratio (Test) | $0.29 \pm 1.22$ | $0.48 \pm 0.79$ | $1.12 \pm 0.92$ |

optimal allocation of assets across different investment options. It can be framed as a sequential decision-making problem as the agent continuously adjusts the portfolio in response to evolving market conditions, fluctuating asset prices, and shifting risk preferences, rather than setting a static allocation. Each decision not only influences immediate returns but also conditions future decisions.

A key metric for evaluating the performance of an investment strategy is the Sharpe ratio (Sharpe, 1966), which measures the trade-off between return and risk. It is defined as the ratio of the mean return to the standard deviation of returns:

$$\mathrm{SharpeRatio}(r_{1:t}) := \frac{\mathrm{mean}(r_{1:t})}{\mathrm{std}(r_{1:t})}, \tag{12}$$

where $r_t := (P_{t+1} - P_t)/P_t$ represents the simple returns, and $P_t$ is the portfolio value at time $t$. Since the Sharpe ratio is non-cumulative, previous RL approaches have relied on the approximate differential Sharpe ratio (Moody et al., 1998; Moody & Saffell, 2001) as a reward signal to facilitate learning. However, this approach introduces an inconsistency between the learning objective and the actual Sharpe ratio, potentially leading to suboptimal policy learning.

**Environment** This experiment was conducted in a financial market simulation, where an agent learned to optimize portfolio allocations across 11 different S&P 500 sector indices from 2006 to 2021. The environment is the same as that described by Sood et al. (2023); Nägele et al. (2024), with further details provided in Appendix H.4.

**Baselines** We considered two baseline methods: (i) DiffSharpe (Moody et al., 1998; Moody & Saffell, 2001), which optimizes an approximate differential Sharpe ratio, and (ii) a non-cumulative Markov decision process (NCMDP) method proposed by Nägele et al. (2024), which maps NCMDPs to standard MDPs and defines per-step rewards based on consecutive differences.

**Method** As demonstrated in Table 1, since both mean and variance admit recursive computation, the Sharpe ratio can also be expressed and updated in a recursive manner. This property allows our method to address the aforementioned inconsistency, aligning the learning objective with the true Sharpe ratio. Our method is built upon the PPO (Schulman et al., 2017) algorithm, with specific modifications on Bellman equation detailed in Algorithm 2 in Appendix G.

**Results** We conducted experiments across five random seeds, reporting the mean and standard deviation of test set performance. Since a higher Sharpe ratio reflects superior risk-adjusted returns, the results in Table 2 confirm that our method consistently outperforms the baselines by effectively balancing risk and reward. These results illustrate that modifying either the local reward signal or the global performance measure can create misalignment, leading to inconsistencies in policy training and suboptimal learning outcomes. In contrast to the baseline methods, our method maintains the original per-step reward structure while estimating and optimizing the exact Sharpe ratio over the entire trajectory. This ensures consistency between training and evaluation, allows the agent to capture long-term dependencies, and reduces sensitivity to local noise. As a result, our approach achieves superior risk-adjusted returns with improved stability and robustness in portfolio management. Moreover, its ability to maintain alignment between learning objectives and evaluation metrics suggests strong potential for broader applications in various real-world decision-making domains.

## 6 Conclusion

In this paper, we revealed that the recursive structures in the standard MDP can be generalized to a broader class of recursive reward aggregation functions, resulting in generalized Bellman equations and operators. Our theoretical analysis on the existence and uniqueness of fixed points of the generalized Bellman operators provided a solid foundation for designing RL algorithms based on recursive reward aggregation and understanding their convergence properties. Empirical evaluations across discrete and continuous environments confirmed that different aggregation functions significantly influence policy preferences, and we can align the agent behavior with the task requirements by selecting appropriate aggregation functions. These findings highlight the flexibility of recursive reward aggregation, paving the way for more versatile RL algorithms that can be tailored to complex task requirements.

Future research could explore several extensions of the proposed recursive reward aggregation framework. First, since the framework does not require the outputs of the generation function and the inputs of the aggregation function (i.e., the internal states of the bidirectional processes in Fig. 2, see also Appendix B) to be real values, one promising direction is to investigate the use of *multi-dimensional signals*, enhancing the flexibility and expressiveness of policy preferences, particularly in complex environments with intricate reward structures (Abouelazm et al., 2024). Second, exploring the theoretical properties of the generalized Bellman operators in the *stochastic setting*, especially their contraction behavior under different distributional metrics (see also Appendix E), is an important area of study (Bellemare et al., 2023). Additionally, applying recursive reward aggregation to real-world applications, such as risk-sensitive decision-making, risk-adjusted returns and portfolio diversification in finance, and safe, robust, and multi-objective control in robotics, presents promising directions (Kober et al., 2013; Kiran et al., 2021; Liu et al., 2024).

## References


Ahmed Abouelazm, Jonas Michel, and J Marius Zoellner. A review of reward functions for reinforcement learning in the context of autonomous driving. *arXiv preprint*, 2024. URL https://arxiv.org/abs/2405.01440.

Johannes Ackermann, Takayuki Osa, and Masashi Sugiyama. Offline reinforcement learning from datasets with structured non-stationarity. In *Reinforcement Learning Conference*, 2024. URL https://openreview.net/forum?id=qowNlhKcPw.

Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in AI safety. *arXiv preprint*, 2016. URL https://arxiv.org/abs/1606.06565.

Saurabh Arora and Prashant Doshi. A survey of inverse reinforcement learning: Challenges, methods and progress. *Artificial Intelligence*, 297:103500, 2021. URL https://doi.org/10.1016/j.artint.2021.103500.

Leemon C. Baird. Reinforcement learning in continuous time: Advantage updating. In *IEEE International Conference on Neural Networks*, volume 4, pp. 2448–2453. IEEE, 1994. URL https://doi.org/10.1109/ICNN.1994.374604.

Andrew G. Barto, Richard S. Sutton, and Charles W. Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-13(5):834–846, 1983. URL https://doi.org/10.1109/TSMC.1983.6313077.

Marc G Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In *International Conference on Machine Learning*, 2017. URL https://proceedings.mlr.press/v70/bellemare17a.html.

Marc G Bellemare, Will Dabney, and Mark Rowland. *Distributional Reinforcement Learning*. MIT Press, 2023. URL https://doi.org/10.7551/mitpress/14207.001.0001.

Richard Bellman. Dynamic programming. *Science*, 153(3731):34–37, 1966. URL https://doi.org/10.1126/science.153.3731.34.

Richard Bird and Oege de Moor. *Algebra of Programming*. Prentice Hall, 1997.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/jax-ml/jax.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. OpenAI Gym. *arXiv preprint*, 2016. URL https://arxiv.org/abs/1606.01540.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. In *Neural Information Processing Systems*, volume 30, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/d5e2c0adad503c91f91df240d0cd4e49-Abstract.html.

Geoffrey SH Cruttwell, Bruno Gavranović, Neil Ghani, Paul Wilson, and Fabio Zanasi. Categorical foundations of gradient-based learning. In *European Symposium on Programming*, pp. 1–28, 2022. URL https://doi.org/10.1007/978-3-030-99336-8_1.

Wei Cui and Wei Yu. Reinforcement learning with non-cumulative objective. *IEEE Transactions on Machine Learning in Communications and Networking*, 1:124–137, 2023. URL https://doi.org/10.1109/TMLCN.2023.3285543.

Oege De Moor. Categories, relations and dynamic programming. *Mathematical Structures in Computer Science*, 4(1):33–69, 1994. URL https://doi.org/10.1017/S0960129500000360.

Ron Dorfman, Idan Shenfeld, and Aviv Tamar. Offline meta reinforcement learning – identifiability challenges and effective data collection strategies. In *Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=IBdEfhLveS.

Brendan Fong and Michael Johnson. Lenses and learners. In *International Workshop on Bidirectional Transformations*, 2019. URL https://arxiv.org/abs/1903.03671.

Tobias Fritz. A synthetic approach to Markov kernels, conditional independence and theorems on sufficient statistics. *Advances in Mathematics*, 370:107239, 2020. URL https://doi.org/10.1016/j.aim.2020.107239. https://arxiv.org/abs/1908.07021.

Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, 2018. URL https://proceedings.mlr.press/v80/fujimoto18a.html.

Bruno Gavranović. Space-time tradeoffs of lenses and optics via higher category theory. *arXiv preprint*, 2022. URL https://arxiv.org/abs/2209.09351.

Michèle Giry. A categorical approach to probability theory. *Categorical Aspects of Topology and Analysis*, pp. 68–85, 1982. URL https://doi.org/10.1007/BFb0092872.

Sai Krishna Gottipati, Yashaswi Pathak, Rohan Nuttall, Raviteja Chunduru, Ahmed Touati, Sriram Ganapathi Subramanian, Matthew E Taylor, Sarath Chandar, et al. Maximum reward formulation in reinforcement learning. *arXiv preprint*, 2020. URL https://arxiv.org/abs/2010.03744.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft Actor-Critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, 2018. URL https://proceedings.mlr.press/v80/haarnoja18b.html.

Dylan Hadfield-Menell, Anca Dragan, Pieter Abbeel, and Stuart Russell. The off-switch game. In *Workshops at the Thirty-First AAAI Conference on Artificial Intelligence*, 2017. URL https://arxiv.org/abs/1611.08219.

Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, 2020. URL https://doi.org/10.1038/s41586-020-2649-2. https://numpy.org.

Jules Hedges and Riu Rodríguez Sakamoto. Value iteration is optic composition. In *International Conference on Applied Category Theory*, 2022. URL https://arxiv.org/abs/2206.04547.

Ralf Hinze, Thomas Harper, and Daniel W. H. James. Theory and practice of fusion. In *Symposium on Implementation and Application of Functional Languages*, pp. 19–37, 2010. URL https://doi.org/10.1007/978-3-642-24276-2_2.

Ahmed Hussein, Mohamed Medhat Gaber, Eyad Elyan, and Chrisina Jayne. Imitation learning: A survey of learning methods. *ACM Computing Surveys (CSUR)*, 50(2):1–35, 2017. URL https://doi.org/10.1145/3054912.

Graham Hutton. A tutorial on the universality and expressiveness of fold. *Journal of Functional Programming*, 9(4):355–372, 1999. URL https://doi.org/10.1017/S0956796899003500.

Taylan Kabbani and Ekrem Duman. Deep reinforcement learning approach for trading automation in the stock market. *IEEE Access*, 10:93564–93574, 2022. URL https://doi.org/10.1109/ACCESS.2022.3203697.

Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996. URL https://doi.org/10.1613/jair.301.

B Ravi Kiran, Ibrahim Sobh, Victor Talpaert, Patrick Mannion, Ahmad A Al Sallab, Senthil Yogamani, and Patrick Pérez. Deep reinforcement learning for autonomous driving: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 23(6):4909–4926, 2021. URL https://doi.org/10.1109/TITS.2021.3054625.

Jens Kober, J Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, 2013. URL https://doi.org/10.1177/0278364913495721.

Prashanth La and Mohammad Ghavamzadeh. Actor-critic algorithms for risk-sensitive MDPs. In *Neural Information Processing Systems*, 2013. URL https://proceedings.neurips.cc/paper/2013/hash/eb163727917cbba1eea208541a643e74-Abstract.html.

Robert Tjarko Lange. gymnax: A JAX-based reinforcement learning environment library, 2022. URL http://github.com/RobertTLange/gymnax.

Jan Leike, Miljan Martic, Victoria Krakovna, Pedro A Ortega, Tom Everitt, Andrew Lefrancq, Laurent Orseau, and Shane Legg. AI safety gridworlds. *arXiv preprint*, 2017. URL https://arxiv.org/abs/1711.09883.

Xiao-Yang Liu, Ziyi Xia, Hongyang Yang, Jiechao Gao, Daochen Zha, Ming Zhu, Christina Dan Wang, Zhaoran Wang, and Jian Guo. Dynamic datasets and market environments for financial reinforcement learning. *Machine Learning*, 113(5):2795–2839, 2024. URL https://doi.org/10.1007/s10994-023-06511-w. https://arxiv.org/abs/2304.13174.

Shie Mannor and John Tsitsiklis. Mean-variance optimization in Markov decision processes. In *International Conference on Machine Learning*, 2011. URL https://dl.acm.org/doi/abs/10.5555/3104482.3104505. https://icml.cc/2011/papers/156_icmlpaper.pdf.

Erik Meijer, Maarten Fokkinga, and Ross Paterson. Functional programming with bananas, lenses, envelopes and barbed wire. In *Conference on Functional Programming Languages and Computer Architecture*, pp. 124–144, 1991. URL https://doi.org/10.1007/3540543961_7.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing Atari with deep reinforcement learning. *arXiv preprint*, 2013. URL https://arxiv.org/abs/1312.5602.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015. URL https://doi.org/10.1038/nature14236.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, 2016. URL https://proceedings.mlr.press/v48/mniha16.html.

John Moody and Matthew Saffell. Learning to trade via direct reinforcement. *IEEE transactions on neural Networks*, 12(4):875–889, 2001. URL https://doi.org/10.1109/72.935097.

John Moody, Lizhong Wu, Yuansong Liao, and Matthew Saffell. Performance functions and reinforcement learning for trading systems and portfolios. *Journal of forecasting*, 17(5-6):441–470, 1998. URL https://doi.org/10.1002/(SICI)1099-131X(1998090)17:5/6%3C441::AID-FOR707%3E3.0.CO;2-%23.

Tetsuro Morimura, Masashi Sugiyama, Hisashi Kashima, Hirotaka Hachiya, and Toshiyuki Tanaka. Nonparametric return distribution approximation for reinforcement learning. In *International Conference on Machine Learning*, 2010a. URL https://dblp.org/rec/conf/icml/MorimuraSKHT10.html. https://icml.cc/2010/papers/652.pdf.

Tetsuro Morimura, Masashi Sugiyama, Hisashi Kashima, Hirotaka Hachiya, and Toshiyuki Tanaka. Parametric return density estimation for reinforcement learning. In *Conference on Uncertainty in Artificial Intelligence*, 2010b. URL https://dblp.org/rec/conf/uai/MorimuraSKHT10.html. https://event.cwi.nl/uai2010/papers/UAI2010_0115.pdf.

Kevin Murphy. Reinforcement learning: An overview. *arXiv preprint*, 2024. URL https://arxiv.org/abs/2412.05265.

Maximilian Nägele, Jan Olle, Thomas Fösel, Remmy Zen, and Florian Marquardt. Tackling decision processes with non-cumulative objectives using reinforcement learning. *arXiv preprint*, 2024. URL https://arxiv.org/abs/2405.13609.

Andrew Y. Ng, Daishi Harada, and Stuart J. Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *International Conference on Machine Learning*, 1999. URL https://dl.acm.org/doi/10.5555/645528.657613.

Arnab Nilim and Laurent El Ghaoui. Robust control of Markov decision processes with uncertain transition matrices. *Operations Research*, 53(5):780–798, 2005. URL https://doi.org/10.1287/opre.1050.0216.

Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., 1994. URL https://doi.org/10.1002/9780470316887.

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

Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-Baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021. URL http://jmlr.org/papers/v22/20-1364.html.

Mitchell Riley. Categories of optics. *arXiv preprint*, 2018. URL https://arxiv.org/abs/1809.00738.

John Schulman, Philipp Moritz, Sergey Levine, Michael I. Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. In *International Conference on Learning Representations*, 2016. URL https://arxiv.org/abs/1506.02438.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint*, 2017. URL https://arxiv.org/abs/1707.06347.

William F. Sharpe. Mutual fund performance. *The Journal of Business*, 39(1):119–138, 1966. URL http://www.jstor.org/stable/2351741.

Toby St. Clere Smithe. Bayesian updates compose optically. *arXiv preprint*, 2020. URL https://arxiv.org/abs/2006.01631.

Srijan Sood, Kassiani Papasotiriou, Marius Vaiciulis, and Tucker Balch. Deep reinforcement learning for optimal portfolio allocation: A comparative study with mean-variance optimization. *FinPlan*, pp. 21, 2023. URL https://icaps23.icaps-conference.org/papers/finplan/FinPlan23_paper_4.pdf.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, 1998. URL http://incompleteideas.net/book/the-book.html.

Aviv Tamar, Dotan Di Castro, and Shie Mannor. Policy gradients with variance related risk criteria. In *International Conference on Machine Learning*, 2012. URL https://dl.acm.org/doi/10.5555/3042573.3042784. https://icml.cc/2012/papers/489.pdf.

Emanuel Todorov, Tom Erez, and Yuval Tassa. MuJoCo: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012. URL https://doi.org/10.1109/IROS.2012.6386109.

Grigorii Veviurko, Wendelin Böhmer, and Mathijs de Weerdt. To the max: Reinventing reward in reinforcement learning. In *International Conference on Machine Learning*, 2024. URL https://proceedings.mlr.press/v235/veviurko24a.html.

Ruosong Wang, Peilin Zhong, Simon S Du, Russ R Salakhutdinov, and Lin Yang. Planning with general objective functions: Going beyond total rewards. In *Neural Information Processing Systems*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/a6a767bbb2e3513233f942e0ff24272c-Abstract.html.

Christopher J. C. H. Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8:279–292, 1992. URL https://doi.org/10.1007/BF00992698.

Christopher John Cornish Hellaby Watkins. *Learning from delayed rewards*. PhD thesis, King's College, Cambridge United Kingdom, 1989. URL http://www.cs.rhul.ac.uk/~chrisw/new_thesis.pdf.

Xing Wu, Haolei Chen, Jianjia Wang, Luigi Troiano, Vincenzo Loia, and Hamido Fujita. Adaptive stock trading strategies with deep reinforcement learning methods. *Information Sciences*, 538: 142–158, 2020. URL https://doi.org/10.1016/j.ins.2020.05.066.

Zhixuan Yang and Nicolas Wu. Fantastic morphisms and where to find them: A guide to recursion schemes. In *International Conference on Mathematics of Program Construction*, pp. 222–267, 2022. URL https://doi.org/10.1007/978-3-031-16912-0_9.

Meixin Zhu, Yinhai Wang, Ziyuan Pu, Jingyun Hu, Xuesong Wang, and Ruimin Ke. Safe, efficient, and comfortable velocity control based on reinforcement learning for autonomous driving. *Transportation Research Part C: Emerging Technologies*, 117:102662, 2020. URL https://doi.org/10.1016/j.trc.2020.102662.

# Supplementary Materials

*The following content was not necessarily subject to peer review.*

## Contents

## List of Figures



## List of Tables



## List of Algorithms

# A State-action recursion

In Section 2, we introduced the recursive generation of rewards by iterating over *states* $S$. In this section, we extend this framework to iterate over *state-action pairs* $S \times A$, which is crucial for defining the *state-action value function* $\mathrm{q}_\pi : S \times A \to R$.

## A.1 State-action transition

First, note that both *pre-composing* and *post-composing* the pairing function $\langle \mathrm{id}_S, \pi \rangle : S \to S \times A$ with the transition function $\mathrm{p} : S \times A \to S$ yield transition functions:

- *state transition* $\mathrm{p}_\pi^S : S \to S := \mathrm{p} \circ \langle \mathrm{id}_S, \pi \rangle = s \mapsto \mathrm{p}(s, \pi(s))$ and
- *state-action transition* $\mathrm{p}_\pi^{S\times A} : S \times A \to S \times A := \langle \mathrm{id}_S, \pi \rangle \circ \mathrm{p} = (s, a) \mapsto (\mathrm{p}(s, a), \pi(\mathrm{p}(s, a)))$.

We use the superscripts $S$ and $S \times A$ to indicate the domains/codomains of these transition functions.

## A.2 State-action step function and generation function

Then, following the definitions of the state step function $\mathrm{step}^S_{\pi,\mathrm{p},\mathrm{r},\omega} : S \to \{*\} + R \times S$ in Eq. (1) and generation function $\mathrm{gen}^S_{\pi,\mathrm{p},\mathrm{r},\omega} : S \to [R]$ in Eq. (2), we can define the *state-action step/generation functions* using the state-action transition $\mathrm{p}_\pi^{S\times A}$ and the reward function r:

$$\mathrm{step}^{S\times A}_{\pi,\mathrm{p},\mathrm{r},\omega} : S \times A \to \{*\} + R \times (S \times A) := (s, a) \mapsto \begin{cases} * & s \in S_\omega, \\ (\mathrm{r}(s, a), \mathrm{p}_\pi^{S\times A}(s, a)) & s \notin S_\omega. \end{cases} \tag{13}$$

$$\mathrm{gen}^{S\times A}_{\pi,\mathrm{p},\mathrm{r},\omega} : S \times A \to [R] := (s, a) \mapsto \begin{cases} [\,] & s \in S_\omega, \\ \mathrm{cons}(\mathrm{r}(s, a), \mathrm{gen}^{S\times A}_{\pi,\mathrm{p},\mathrm{r},\omega}(\mathrm{p}_\pi^{S\times A}(s, a))) & s \notin S_\omega. \end{cases} \tag{14}$$

## A.3 State-action statistic function and value function

Applying the same algebraic fusion technique (Hinze et al., 2010) used for the state statistic function $\tau_\pi^S : S \to T$ in Theorem 3.2, we can define the *state-action statistic function* $\tau_\pi^{S\times A} : S \times A \to T$ and derive its corresponding Bellman equation as follows:

**Theorem A.1** (Bellman equation for the state-action statistic function)**.** *Given a recursive generation function* $\mathrm{gen}^{S\times A}_{\pi,\mathrm{p},\mathrm{r},\omega}$ *and a recursive statistic aggregation function* $\mathrm{agg}_{\mathrm{init},\triangleright}$ *(Definition 3.1), their composition, called the state-action statistic function* $\tau_\pi^{S\times A} : S \to T$*, satisfies the following equation:*

$$\begin{aligned} \tau_\pi^{S\times A} : S \times A \to T &:= \mathrm{agg}_{\mathrm{init},\triangleright} \circ \mathrm{gen}^{S\times A}_{\pi,\mathrm{p},\mathrm{r},\omega} \\ &= (s, a) \mapsto \begin{cases} \mathrm{init} & s \in S_\omega, \\ (\mathrm{r}(s, a) \triangleright \tau_\pi^{S\times A}(\mathrm{p}_\pi^{S\times A}(s, a))) & s \notin S_\omega. \end{cases} \end{aligned} \tag{15}$$

Similarly, the *state-action value function* $\mathrm{q}_\pi : S \times A \to R := \mathrm{post} \circ \tau_\pi^{S\times A}$ is the composition of the state-action statistic function $\tau_\pi^{S\times A} : S \times A \to T$ with the post-processing function $\mathrm{post} : T \to R$.

## A.4 Relationship between state and state-action statistic functions

We can now state the theorem that relates the state and state-action statistic functions:

**Theorem A.2** (Relationship between state and state-action statistic functions)**.** *Given a recursive generation function* $\mathrm{gen}_{\pi,\mathrm{p},\mathrm{r},\omega}$ *(Definition 2.1) and a recursive statistic aggregation function* $\mathrm{agg}_{\mathrm{init},\triangleright}$ *(Definition 3.1), the state statistic function* $\tau_\pi^S : S \to T$ *in Eq. (8) and the state-action statistic function* $\tau_\pi^{S\times A} : S \times A \to T$ *in Eq. (15) satisfy the following equations:*

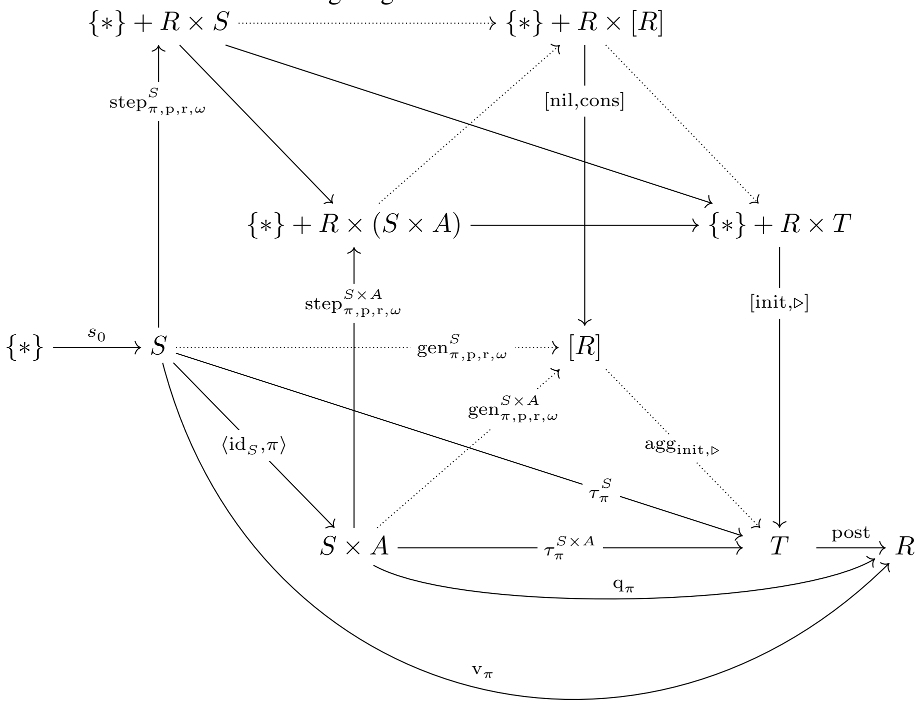
$$\tau_\pi^S = \tau_\pi^{S\times A} \circ \langle \mathrm{id}_S, \pi \rangle : S \to T \qquad \textit{(for all states)}, \tag{16}$$

$$\tau_\pi^{S\times A} = \mathrm{r} \triangleright (\tau_\pi^S \circ \mathrm{p}) : S \times A \to T \qquad \textit{(for all non-terminal states)}. \tag{17}$$

**Corollary A.3** (Relationship between state and state-action value functions)**.** *The state value function* $\mathrm{v}_\pi : S \to R$ *and the state-action value function* $\mathrm{q}_\pi : S \times A \to R$ *satisfy the following equation:*

$$\mathrm{v}_\pi = \mathrm{q}_\pi \circ \langle \mathrm{id}_S, \pi \rangle : S \to R. \tag{18}$$

In summary, the relationships between the state/state-action step, generation, statistic, and value functions are shown in the following diagram:




(19)

## A.5 Advantage function

The *advantage function* (Baird, 1994; Schulman et al., 2016),

$$\alpha_\pi : S \times A \to R := \mathrm{q}_\pi - \mathrm{v}_\pi \circ p_1 = (s, a) \mapsto \mathrm{q}_\pi(s, a) - \mathrm{v}_\pi(s), \tag{20}$$

is defined as the difference between the state-action value function $\mathrm{q}_\pi : S \times A \to R$ and the state value function $\mathrm{v}_\pi : S \to R$, where $p_1 : S \times A \to S$ is the projection function that extracts the state from a state-action pair. The advantage function measures the advantage of taking an action $a$ in a state $s$ over the average value of all actions in that state following the policy $\pi$, which is used widely in RL algorithms such as Asynchronous Advantage Actor-Critic (A3C) (Mnih et al., 2016) and Proximal Policy Optimization (PPO) (Schulman et al., 2017).

For a general recursive statistic aggregation function $\mathrm{agg}_{\mathrm{init},\triangleright}$ and a post-processing function $\mathrm{post}$, the advantage function can be expressed using the state-action statistic function $\tau^{S\times A}_\pi : S \times A \to T$ and the state statistic function $\tau^S_\pi : S \to T$ as follows:

$$\alpha_\pi : S \times A \to R = (s, a) \mapsto \mathrm{post}(\tau^{S\times A}_\pi(s, a)) - \mathrm{post}(\tau^S_\pi(s)) \tag{21}$$

$$= (s, a) \mapsto \begin{cases} 0 & s \in S_\omega, \\ \mathrm{post}(\mathrm{r}(s, a) \triangleright \tau^S_\pi(\mathrm{p}(s, a))) - \mathrm{post}(\tau^S_\pi(s)) & s \notin S_\omega. \end{cases} \tag{22}$$

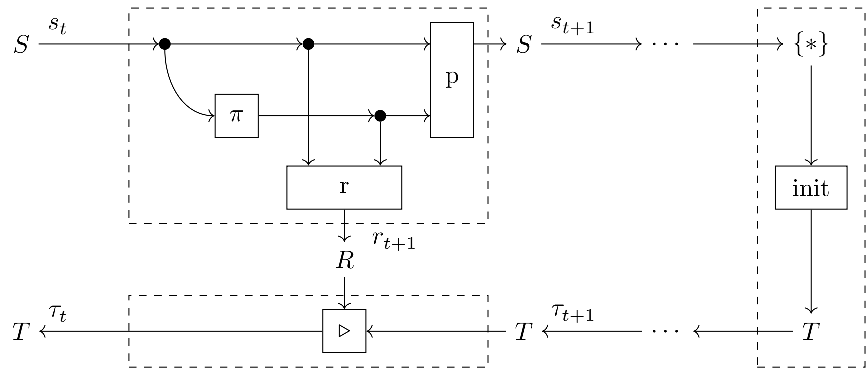



Figure 7: State statistic bidirectional process $\tau_\pi^S : S \to T$

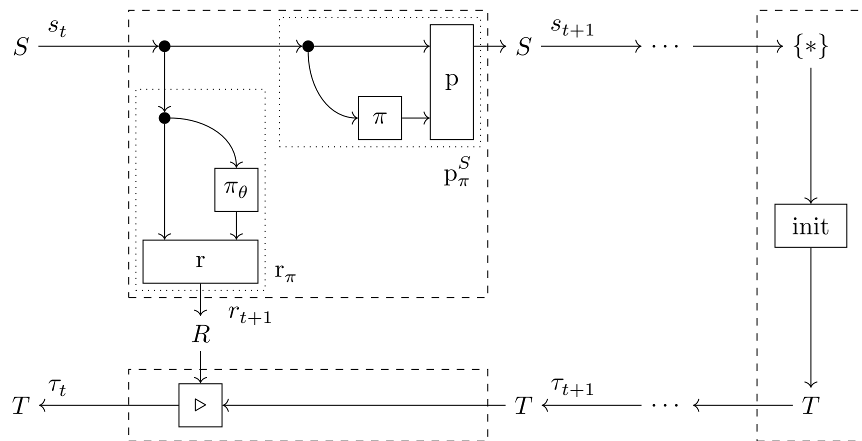



Figure 8: State statistic bidirectional process (with different behavior and target policies)

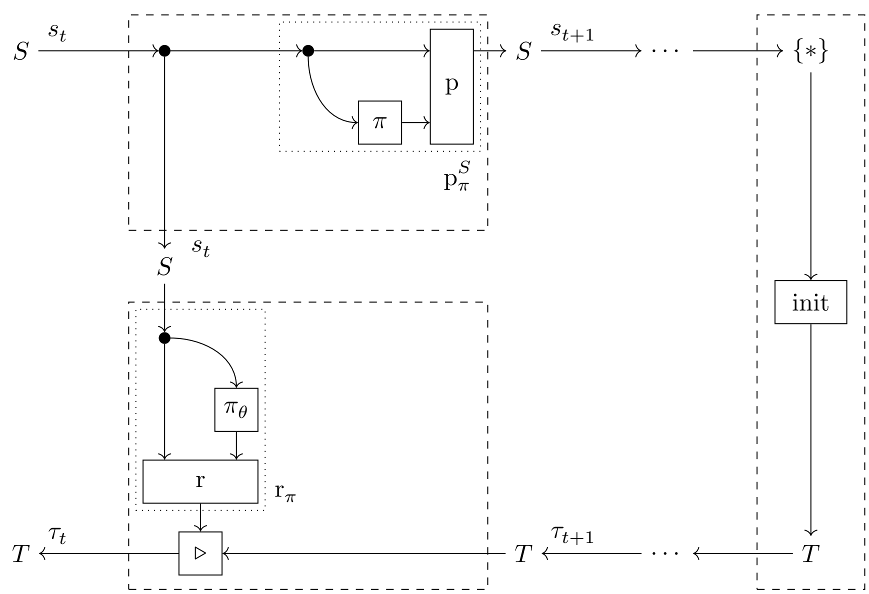



Figure 9: State statistic bidirectional process (with state as the residual)

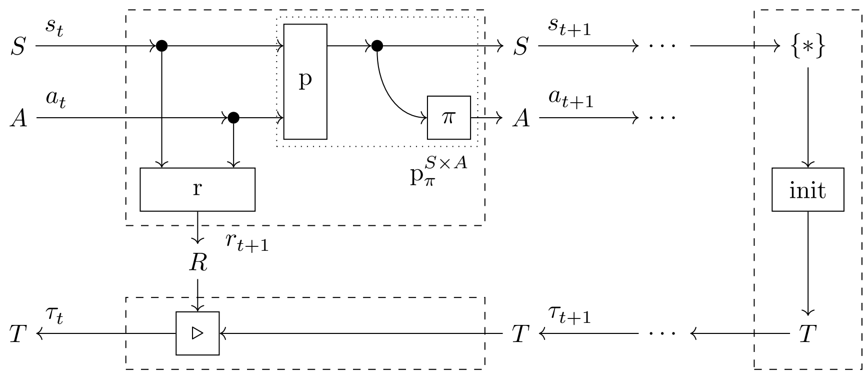



Figure 10: State-action statistic bidirectional process $\tau_\pi^{S\times A} : S \times A \to T$

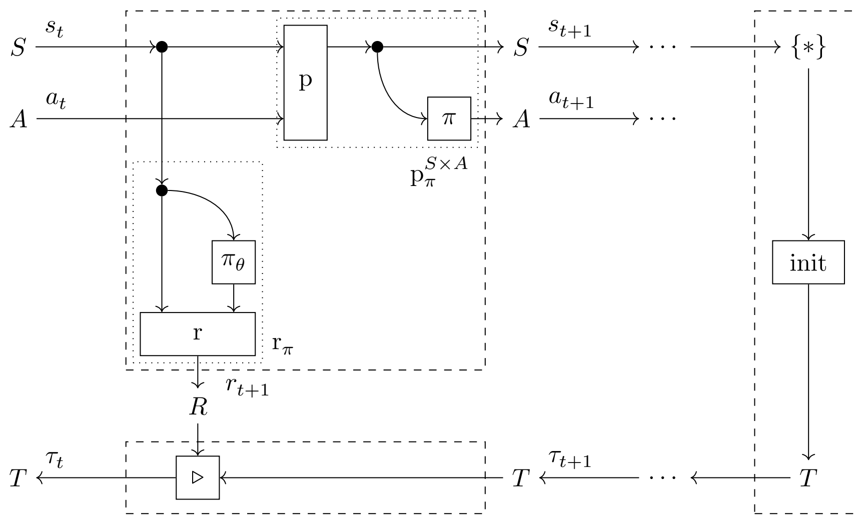



Figure 11: State-action statistic bidirectional process (with different behavior and target policies)

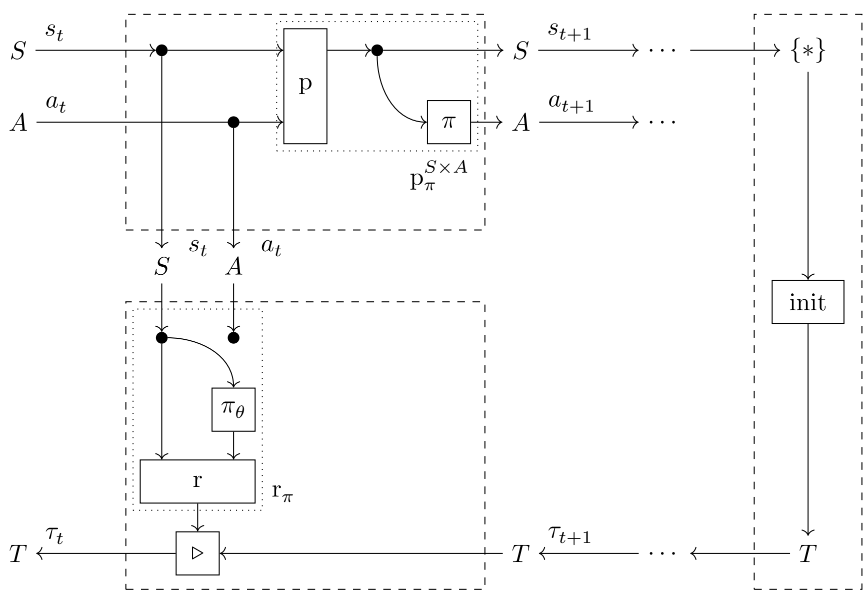



Figure 12: State-action statistic bidirectional process (with state-action as the residual)

# B Algebraic structures in Markov decision process

In this section, we briefly discuss the algebraic structures used in this work. For a tutorial on algebraic programming, we refer the reader to Hutton (1999). For a theoretical treatment of algebra fusion, see Hinze et al. (2010). For an accessible and illustrative introduction to bidirectional processes, we recommend Gavranović (2022).

## B.1 Algebra fusion

In this work, we mainly considered algebras and coalgebras of signature $\{*\} + R \times (-)$, i.e., lists of rewards. An *algebra* is a pair $(A, f)$ consisting of a carrier set $A$ and a function $f : \{*\}+R\times A \to A$. A *coalgebra* is a pair $(C, g)$ consisting of a carrier set $C$ and a function $g : C \to \{*\} + R \times C$. For example, the list construction $[\mathrm{nil}, \mathrm{cons}] : \{*\} + R \times [R] \to [R]$ is an algebra on the set $[R]$ of lists of rewards, while the step function $\mathrm{step}^S_{\pi,\mathrm{p},\mathrm{r},\omega} : S \to \{*\} + R \times S$ is a coalgebra on the set $S$ of states.

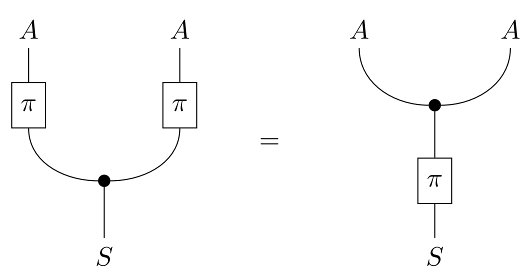
Note that the list construction $[\mathrm{nil}, \mathrm{cons}]$ is the *initial algebra*, the discounted sum function $\mathrm{sum}_\gamma$ is defined as the *catamorphism* (algebra homomorphism) from the initial algebra to the algebra $[0, r + \gamma \cdot s]$, while the recursive generation function $\mathrm{gen}_{\pi,\mathrm{p},\mathrm{r},\omega}$ is defined as the *hylomorphism* (coalgebra homomorphism) from the coalgebra $\mathrm{step}_{\pi,\mathrm{p},\mathrm{r},\omega}$ to the initial algebra. In the field of functional programming, such operations are also known as `fold` and `unfold` (Meijer et al., 1991; Bird & de Moor, 1997; Hutton, 1999; Yang & Wu, 2022).

Due to the recursive nature of the generation and aggregation functions, we can derive the recursive structure of their composition using the algebra fusion technique (Hinze et al., 2010), which leads to the Bellman equations for the state statistic function $\tau^S_\pi : S \to T$ in Theorem 3.2 and the state-action statistic function $\tau^{S\times A}_\pi : S \times A \to T$ in Theorem A.1.

## B.2 Bidirectional process

In Fig. 2, we illustrate the bidirectional processes for the state statistic function and state value function. In algebra, such bidirectional processes are called *lenses* and *optics* (Riley, 2018).

Note that there is a slight difference between the definitions of step/generation/statistic functions in Eqs. (1), (2) and (8) and the bidirectional process in Fig. 2 (reproduced in Fig. 7). In Eq. (1), a state $s$ is duplicated and passed separately to the transition function $\mathrm{p}_\pi$ and the reward function $\mathrm{r}_\pi$, requiring the policy $\pi$ to compute the action $a$ twice. In contrast, in Fig. 7, the state $s$ is passed to the policy $\pi$ only once, and the action $a$ is computed only once and then copied to the transition function $\mathrm{p}$ and the reward function $\mathrm{r}$. These two approaches are equivalent only when the following equation holds:




(23)

For functions, copying an input and then passing the copies to two identical functions is equivalent to passing the input to the function once and then copying the output. However, for stochastic functions, these two approaches are not equivalent, which requires additional care when defining bidirectional processes for stochastic functions (see also Fritz, 2020, Definition 10.1).

Strictly speaking, the definitions in Eqs. (1), (2) and (8) correspond to a bidirectional process illustrated in Fig. 8, where different behavior and target policies can be considered. In this setting, the target policy $\pi_\theta$, parameterized by $\theta$, is used to compute the reward and is optimized, while the potentially unknown behavior policy $\pi$ is passed to the transition function. Further, the *internal state* between the forward and backward processes — also known as the *residual* (Gavranović, 2022) — can be the state itself rather than the reward, as shown in Fig. 9. Similar considerations extend to the state-action statistic function, as illustrated in Figs. 10 to 12.

We believe that such bidirectional processes offer a clearer framework for reinforcement learning, including offline reinforcement learning, inverse reinforcement learning, and imitation learning (Hussein et al., 2017; Arora & Doshi, 2021; Hedges & Sakamoto, 2022; Murphy, 2024). Further research is needed to explore the full potential of bidirectional processes in reinforcement learning.

### B.3 Non-uniqueness of update function and post-processing function

It is important to note that for a given aggregation function, the corresponding update function $\triangleright : R \times T \to T$ and post-processing function $\mathrm{post} : T \to R$ are not necessarily unique. For example, as shown in Table 1, the mean function can be computed recursively in different ways: one approach updates the sum and the length, while another updates the mean and the length. Each approach has its own advantages and disadvantages. Updating the sum allows for a straightforward implementation, but when both the sum and the length are large, numerical instability may arise. In contrast, updating the mean may require additional computation, but if the rewards are bounded, the mean remains bounded as well, which can improve numerical stability.

Table 3: Properties of metrics

| | Premetric | Strict premetric | Metric |
|---|---|---|---|
| Indiscernibility of identities $(a_1 = a_2) \to (d_A(a_1, a_2) = 0)$ | ✓ | ✓ | ✓ |
| Identity of indiscernibles $(d_A(a_1, a_2) = 0) \to (a_1 = a_2)$ | | ✓ | ✓ |
| Symmetry $d_A(a_1, a_2) = d_A(a_2, a_1)$ | | | ✓ |
| Triangle inequality $d_A(a_1, a_3) \leq d_A(a_1, a_2) + d_A(a_2, a_3)$ | | | ✓ |

# C Metrics and Bellman operators

In this section, we discuss the *metrics* on the statistics $T$ and rewards $R$ and the *Bellman operators* for the state/state-action statistic functions.

## C.1 Preliminaries

Recall the definitions of metrics, as summarized in Table 3:

**Definition C.1** (Premetric)**.** A *premetric* on a set $A$ is a function $d_A : A \times A \to [0, \infty]$ such that $\forall a \in A.\ d_A(a, a) = 0$.

**Definition C.2** (Strict premetric)**.** A *strict premetric* on a set $A$ is a function $d_A : A \times A \to [0, \infty]$ such that $\forall a_1, a_2 \in A.\ (d_A(a_1, a_2) = 0) \leftrightarrow (a_1 = a_2)$.

Given a function to a premetric space, we can define a premetric on the domain by pullback:

**Lemma C.3** (Pullback premetric)**.** *Let $d_B : B \times B \to [0, \infty]$ be a premetric on a set $B$, and let $f : A \to B$ be a function. The pullback premetric $d_A : A \times A \to [0, \infty]$ is defined by*

$$\forall a_1, a_2 \in A.\ d_A(a_1, a_2) := d_B(f(a_1), f(a_2)). \tag{24}$$

*If $d_B$ is a strict premetric, then $d_A$ is also a strict premetric if and only if the function $f$ is injective.*

## C.2 Metrics on statistics and rewards

By Lemma C.3, we can define a premetric $d_T$ on statistics $T$ by pulling back a premetric $d_R$ on rewards $R$ through a post-processing function $\mathrm{post} : T \to R$:

$$\forall t_1, t_2 \in T.\ d_T(t_1, t_2) := d_R(\mathrm{post}(t_1), \mathrm{post}(t_2)). \tag{25}$$

However, when rewards $R$ are real-valued while statistics $T$ are multi-dimensional, the pullback premetric $d_T$ may not be a strict premetric, as different statistics may map to the same reward value.

For example, consider the range of rewards, where the statistics $T = \mathbb{R}^2$ are the maximum and minimum of rewards. We can directly define a metric on statistics by

$$d_T\left(\begin{bmatrix} m_1 \\ n_1 \end{bmatrix}, \begin{bmatrix} m_2 \\ n_2 \end{bmatrix}\right) := \sqrt{(m_1 - m_2)^2 + (n_1 - n_2)^2}. \tag{26}$$

If we use the pullback premetric, we have

$$d_T\left(\begin{bmatrix} m_1 \\ n_1 \end{bmatrix}, \begin{bmatrix} m_2 \\ n_2 \end{bmatrix}\right) := d_R\left(\mathrm{post}\left(\begin{bmatrix} m_1 \\ n_1 \end{bmatrix}\right), \mathrm{post}\left(\begin{bmatrix} m_2 \\ n_2 \end{bmatrix}\right)\right) \tag{27}$$

$$= d_R(m_1 - n_1, m_2 - n_2) = |(m_1 - n_1) - (m_2 - n_2)|. \tag{28}$$

### C.3 Bellman operators

Recall the definition of the Bellman operator for a state statistic function $\tau^S : S \to T$:

**Definition 3.3** (Bellman operator)**.** Given a policy $\pi$, a transition function $\mathrm{p}$, a reward function $\mathrm{r}$, a terminal condition $\omega$, and a recursive statistic aggregation function $\mathrm{agg}_{\mathrm{init},\triangleright}$ (Definition 3.1), the *Bellman operator* $\mathcal{B}_\pi : [S, T] \to [S, T]$ for a function $\tau : S \to T$ is defined by

$$\mathcal{B}_\pi \tau : S \to T := s \mapsto \begin{cases} \mathrm{init} & s \in S_\omega, \\ \mathrm{r}_\pi(s) \triangleright \tau(\mathrm{p}_\pi(s)) & s \notin S_\omega. \end{cases} \tag{9}$$

We can define a Bellman operator for a state-action statistic function $\tau^{S\times A} : S \times A \to T$ similarly:

**Definition C.4** (Bellman operator)**.** Given a policy $\pi$, a transition function $\mathrm{p}$, a reward function $\mathrm{r}$, a terminal condition $\omega$, and a recursive statistic aggregation function $\mathrm{agg}_{\mathrm{init},\triangleright}$ (Definition 3.1), the *Bellman operator* $\mathcal{B}_\pi^{S\times A} : [S \times A, T] \to [S \times A, T]$ for a function $\tau^{S\times A} : S \times A \to T$ is defined by

$$\mathcal{B}_\pi^{S\times A} \tau^{S\times A} : S \times A \to T := (s, a) \mapsto \begin{cases} \mathrm{init} & s \in S_\omega, \\ \mathrm{r}(s, a) \triangleright \tau^{S\times A}(\mathrm{p}_\pi^{S\times A}(s, a)) & s \notin S_\omega. \end{cases} \tag{29}$$

### C.4 Existence of fixed points of Bellman operators

The existence of fixed points of the Bellman operators $\mathcal{B}_\pi^S$ and $\mathcal{B}_\pi^{S\times A}$ is established by the Bellman equations for the state statistic function $\tau_\pi^S : S \to T$ in Theorem 3.2 and the state-action statistic function $\tau_\pi^{S\times A} : S \times A \to T$ in Theorem A.1.

*Remark* 5 (Banach fixed point theorem). Note that the classical fixed point theorem for Bellman operators typically relies on the *Banach fixed point theorem*, which requires the underlying space to be a *complete metric space*. This is not an issue in the standard discounted sum setting, as the space $\mathbb{R}$ of real numbers has a complete metric structure. However, in our setting, the space $T$ of statistics may lack such a complete metric structure, posing potential challenges for establishing fixed point guarantees. That said, the triangle inequality of the metric and the completeness of the space are only necessary for ensuring the *existence* of fixed points: the triangle inequality guarantees that the iterative sequence is a Cauchy sequence, while completeness ensures that the sequence has a limit within the space. Since the existence of fixed points follows directly from the Bellman equations, our focus shifts to the *uniqueness* of fixed points, which only requires the space to be a premetric space.

### C.5 Uniqueness of fixed points of Bellman operators

Recall that Theorem 3.5 establishes the uniqueness of fixed points of the Bellman operator $\mathcal{B}_\pi^S$ for state statistic functions $\tau^S : S \to T$:

**Theorem 3.5** (Uniqueness of fixed points of Bellman operator)**.** *Let $\tau_1, \tau_2 : S \to T$ be fixed points of the Bellman operator $\mathcal{B}_\pi$ (Definition 3.3). If the update function $\triangleright$ is contractive with respect to a premetric $d_T$ on statistics $T$ (Definition 3.4), then $d_T(\tau_1(s), \tau_2(s)) = 0$ for all states $s \in S$. If $d_T$ is a strict premetric, then $\tau_1 = \tau_2 = \tau_\pi$.*

Similarly, we can extend this result to the Bellman operator $\mathcal{B}_\pi^{S\times A}$ for state-action statistic functions $\tau^{S\times A} : S \times A \to T$:

**Theorem C.5** (Uniqueness of fixed points of the Bellman operator)**.** *Let $\tau_1^{S\times A}, \tau_2^{S\times A} : S \times A \to T$ be fixed points of the Bellman operator $\mathcal{B}_\pi^{S\times A}$ (Definition C.4). If the update function $\triangleright$ is contractive with respect to a premetric $d_T$ on statistics $T$ (Definition 3.4), then $d_T(\tau_1^{S\times A}(s, a), \tau_2^{S\times A}(s, a)) = 0$ for all states $s \in S$ and actions $a \in A$. If $d_T$ is a strict premetric, then $\tau_1^{S\times A} = \tau_2^{S\times A} = \tau_\pi^{S\times A}$.*

Table 4: Properties of orders

| | Preorder | Partial order | Total preorder | Total order |
|---|---|---|---|---|
| Reflexivity<br>$a \leq_A a$ | ✓ | ✓ | ✓ | ✓ |
| Transitivity<br>$(a_1 \leq_A a_2) \wedge (a_2 \leq_A a_3) \rightarrow (a_1 \leq_A a_3)$ | ✓ | ✓ | ✓ | ✓ |
| Antisymmetry<br>$(a_1 \leq_A a_2) \wedge (a_2 \leq_A a_1) \rightarrow (a_1 = a_2)$ | | ✓ | | ✓ |
| Totality<br>$(a_1 \leq_A a_2) \vee (a_2 \leq_A a_1)$ | | | ✓ | ✓ |

# D Orders and Bellman optimality operators

In this section, we discuss the *orders* on the statistics $T$ and rewards $R$ and the *Bellman optimality operators* for the state/state-action statistic functions.

## D.1 Preliminaries

Recall the definitions of orders, as summarized in Table 4:

**Definition D.1** (Preorder)**.** A *preorder* on a set $A$ is a relation $\leq_A$ that is reflexive $\forall a \in A.\ a \leq_A a$ and transitive $\forall a_1, a_2, a_3 \in A.\ (a_1 \leq_A a_2) \wedge (a_2 \leq_A a_3) \rightarrow (a_1 \leq_A a_3)$.

**Definition D.2** (Partial order)**.** A *partial order* on a set $A$ is a relation $\leq_A$ that is reflexive, transitive, and antisymmetric $\forall a_1, a_2 \in A.\ (a_1 \leq_A a_2) \wedge (a_2 \leq_A a_1) \rightarrow (a_1 = a_2)$.

**Definition D.3** (Total preorder)**.** A *total preorder* on a set $A$ is a relation $\leq_A$ that is reflexive, transitive, and total $\forall a_1, a_2 \in A.\ (a_1 \leq_A a_2) \vee (a_2 \leq_A a_1)$.

**Definition D.4** (Total order)**.** A *total order* on a set $A$ is a relation $\leq_A$ that is reflexive, transitive, antisymmetric, and total.

Given a function to a preorder space, we can define a preorder on the domain by pullback:

**Lemma D.5** (Pullback preorder)**.** *Let $\leq_B$ be a preorder on a set $B$, and let $f : A \rightarrow B$ be a function. The* pullback preorder *$\leq_A$ on a set $A$ is defined by*

$$\forall a_1, a_2 \in A.\ (a_1 \leq_A a_2) := (f(a_1) \leq_B f(a_2)). \tag{30}$$

*If $\leq_B$ is total, then $\leq_A$ is also total. If $\leq_B$ is antisymmetric, then $\leq_A$ is also antisymmetric if and only if $f$ is injective.*

Given a preorder and a premetric, wen can consider how the premetric preserves the preorder:

**Definition D.6** (Preorder-preserving premetric)**.** A *premetric* $d_B : B \times B \rightarrow [0, \infty]$ on a set $B$ *preserves a preorder* $\leq_B$ on the set $B$ if

$$\forall b_1, b_2, b_3 \in B.\ (b_1 \leq_B b_2 \leq_B b_3) \rightarrow (d_B(b_1, b_2) \leq d_B(b_1, b_3)) \wedge (d_B(b_3, b_2) \leq d_B(b_3, b_1)). \tag{31}$$

Note that since a premetric is not required to be symmetric, there are in total eight possible inequalities that we can consider for the preorder preservation of a premetric, which are omitted here for brevity.

Given a preorder-preserving premetric, we can consider an inequality for the supremum of functions:

**Lemma D.7** (Preorder-preserving premetric's supremum inequality)**.** *Let $d_B : B \times B \rightarrow [0, \infty]$ be a premetric that preserves a premetric $\leq_B$ on a set $B$. Then, for functions $f_1, f_2 : A \rightarrow B$ whose suprema are attained in $B$, we have*

$$d_B(\sup_{a \in A} f_1(a), \sup_{a \in A} f_2(a)) \leq \sup_{a \in A} d_B(f_1(a), f_2(a)). \tag{32}$$

This lemma is useful for proving the contraction property of the Bellman optimality operator, as we will see later.

### D.2 Orders on statistics and rewards

By Lemma D.5, we can define a preorder $\leq_T$ on statistics $T$ by pulling back a preorder $\leq_R$ on rewards $R$ through a post-processing function $\mathrm{post} : T \to R$:

$$\forall t_1, t_2 \in T.\ (t_1 \leq_T t_2) := (\mathrm{post}(t_1) \leq_R \mathrm{post}(t_2)). \tag{33}$$

Since the (pre)order $\leq_R$ on rewards $R$ is usually the total order of real numbers, we can guarantee that the preorder $\leq_T$ on statistics $T$ is also total.

For example, consider the arithmetic mean of rewards, where the statistics $T = \mathbb{N} \times \mathbb{R}$ are the length and the sum of rewards. We can compare two statistics $(n_1, s_1)$ and $(n_2, s_2)$ by comparing the means $\frac{s_1}{n_1}$ and $\frac{s_2}{n_2}$. This is a total preorder on the statistics $T$.

### D.3 Bellman optimality operators

We can define the Bellman optimality operators as follows:

**Definition D.8** (Bellman optimality operator)**.** Given a policy $\pi$, a transition function $\mathrm{p}$, a reward function $\mathrm{r}$, a terminal condition $\omega$, a recursive statistic aggregation function $\mathrm{agg}_{\mathrm{init},\triangleright}$ (Definition 3.1), and a preorder $\leq_T$ on statistics $T$, the *Bellman optimality operator* $\mathcal{B}_*^S : [S, T] \to [S, T]$ for a function $\tau^S : S \to T$ is defined by

$$\mathcal{B}_*^S \tau^S : S \to T := s \mapsto \begin{cases} \mathrm{init} & s \in S_\omega, \\ \sup\limits_{a \in A} \Big( \mathrm{r}(s, a) \triangleright \tau^S(\mathrm{p}(s, a)) \Big) & s \notin S_\omega. \end{cases} \tag{34}$$

**Definition D.9** (Bellman optimality operator)**.** Given a policy $\pi$, a transition function $\mathrm{p}$, a reward function $\mathrm{r}$, a terminal condition $\omega$, a recursive statistic aggregation function $\mathrm{agg}_{\mathrm{init},\triangleright}$ (Definition 3.1), and a preorder $\leq_T$ on statistics $T$, the *Bellman optimality operator* $\mathcal{B}_*^{S\times A} : [S \times A, T] \to [S \times A, T]$ for a function $\tau^{S\times A} : S \times A \to T$ is defined by

$$\mathcal{B}_*^{S\times A} \tau^{S\times A} : S \times A \to T := (s, a) \mapsto \begin{cases} \mathrm{init} & s \in S_\omega, \\ \sup\limits_{a' \in A} \Big( \mathrm{r}(s, a) \triangleright \tau^{S\times A}(\mathrm{p}(s, a), a') \Big) & s \notin S_\omega. \end{cases} \tag{35}$$

### D.4 Existence of fixed points of Bellman optimality operators

Recall that Theorem 3.7 establishes the existence of a fixed point of the Bellman optimality operator $\mathcal{B}_*^S$ for state statistic functions $\tau^S : S \to T$:

**Theorem 3.7** (Bellman optimality equation for the state statistic function)**.** *Given a preorder $\leq_T$ on statistics $T$, the optimal state statistic function $\tau_*$ (Definition 3.6) satisfies the following equation:*

$$\tau_* : S \to T := s \mapsto \begin{cases} \mathrm{init} & s \in S_\omega, \\ \sup\limits_{a \in A} (\mathrm{r}(s, a) \triangleright \tau_*(\mathrm{p}(s, a))) & s \notin S_\omega. \end{cases} \tag{10}$$

We can similarly establish the existence of a fixed point of the Bellman optimality operator $\mathcal{B}_*^{S\times A}$ for state-action statistic functions $\tau^{S\times A} : S \times A \to T$:

**Theorem D.10** (Bellman optimality equation for the state-action statistic function)**.** *Given a preorder $\leq_T$ on statistics $T$, the optimal state-action statistic function $\tau_*^{S\times A}$ satisfies the following equation:*

$$\tau_*^{S\times A} : S \times A \to T := (s, a) \mapsto \begin{cases} \mathrm{init} & s \in S_\omega, \\ \sup\limits_{a' \in A} \Big( \mathrm{r}(s, a) \triangleright \tau_*^{S\times A}(\mathrm{p}(s, a), a') \Big) & s \notin S_\omega. \end{cases} \tag{36}$$

### D.5 Uniqueness of fixed points of Bellman optimality operators

Similarly to Theorem 3.5, we can guarantee the uniqueness of fixed points of the Bellman optimality operators $\mathcal{B}_*^S$ and $\mathcal{B}_*^{S\times A}$ under certain conditions:

Table 5: Fixed points of the Bellman operators and the Bellman optimality operators.

| | | Definition | Existence | Uniqueness |
|---|---|---|---|---|
| Bellman operator | $\mathcal{B}_\pi^S$ | Definition 3.3 | Theorem 3.2 | Theorem 3.5 |
| | $\mathcal{B}_\pi^{S\times A}$ | Definition C.4 | Theorem A.1 | Theorem C.5 |
| Bellman optimality operator | $\mathcal{B}_*^S$ | Definition D.8 | Theorem 3.7 | Theorem D.11 |
| | $\mathcal{B}_*^{S\times A}$ | Definition D.9 | Theorem D.10 | Theorem D.12 |

**Theorem D.11** (Uniqueness of fixed points of Bellman optimality operator)**.** *Let $\tau_1^S, \tau_2^S : S \to T$ be fixed points of the Bellman optimality operator $\mathcal{B}_*^S$ (Definition D.8). If the update function $\triangleright$ is contractive with respect to a premetric $d_T$ on statistics $T$ (Definition 3.4), and the premetric $d_T$ preserves the preorder $\leq_T$ on statistics $T$ (Definition D.6), then $d_T(\tau_1^S(s), \tau_2^S(s)) = 0$ for all states $s \in S$. If $d_T$ is a strict premetric, then $\tau_1^S = \tau_2^S = \tau_*^S$.*

**Theorem D.12** (Uniqueness of fixed points of Bellman optimality operator)**.** *Let $\tau_1^{S\times A}, \tau_2^{S\times A} : S \times A \to T$ be fixed points of the Bellman optimality operator $\mathcal{B}_*^{S\times A}$ (Definition D.9). If the update function $\triangleright$ is contractive with respect to a premetric $d_T$ on statistics $T$ (Definition 3.4), and the premetric $d_T$ preserves the preorder $\leq_T$ on statistics $T$ (Definition D.6), then $d_T(\tau_1^{S\times A}(s,a), \tau_2^{S\times A}(s,a)) = 0$ for all states $s \in S$ and actions $a \in A$. If $d_T$ is a strict premetric, then $\tau_1^{S\times A} = \tau_2^{S\times A} = \tau_*^{S\times A}$.*

In summary, the definitions and results on the fixed points of the Bellman operators and the Bellman optimality operators are summarized in Table 5.

# E Stochastic Markov decision process

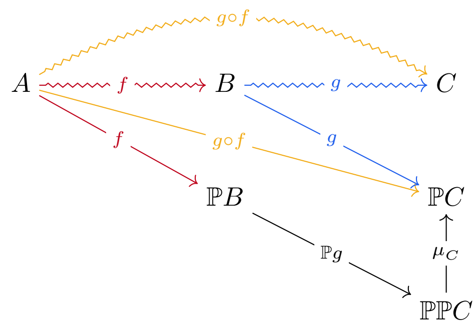
In this section, we discuss the stochastic extension of the deterministic Markov decision processes introduced in Sections 2 and 3.

## E.1 Composition of stochastic functions

The composition rules of stochastic functions and deterministic functions are defined as follows:

- Composition of two stochastic functions $f : A \to \mathbb{P}B$ and $g : B \to \mathbb{P}C$ by marginalizing over the intermediate variable, as described by the *Chapman–Kolmogorov equation* (Giry, 1982):

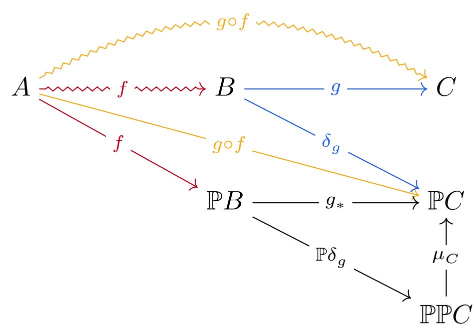
$$(g \circ f)(c|a) := \int_B g(c|b) f(b|a) \, \mathrm{d}b. \tag{37}$$




(38)

- Composition of a stochastic function $f : A \to \mathbb{P}B$ with a deterministic function $g : B \to C$:

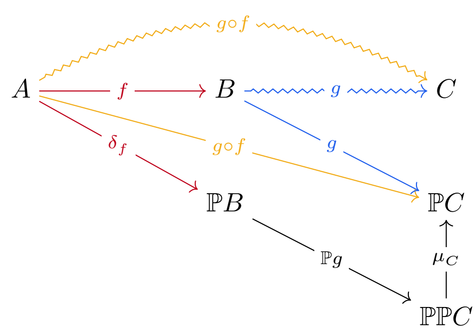
$$(g \circ f)(c|a) := g_* f(b|a) = \int_B \delta_g(b) f(b|a) \, \mathrm{d}b. \tag{39}$$




(40)

- Composition of a deterministic function $f : A \to B$ with a stochastic function $g : B \to \mathbb{P}C$:

$$(g \circ f)(c|a) := g(c|f(a)). \tag{41}$$




(42)

Table 6: Expected aggregated rewards vs. aggregated expected rewards: maximum as an example

| | expected maximum rewards | maximum expected rewards |
|---|---|---|
| definition | $\mathbb{E}_\pi[\max(r_1, r_2, \ldots, r_\Omega)]$ | $\max(\mathbb{E}_\pi[r_1], \mathbb{E}_\pi[r_2], \ldots, \mathbb{E}_\pi[r_\Omega])$ |
| statistic $T$ | max reward distribution $\in \mathbb{P}\overline{\mathbb{R}}$ | max reward expectation $\in \overline{\mathbb{R}}$ |
| initial value | Dirac delta measure $\delta_{-\infty} \in \mathbb{P}\overline{\mathbb{R}}$ | reward value $-\infty \in \overline{\mathbb{R}}$ |
| update function | pushforward measure update | expected value update |
| | $\mathbb{P}\overline{\mathbb{R}} \times \mathbb{P}\overline{\mathbb{R}} \to P(\overline{\mathbb{R}} \times \overline{\mathbb{R}}) \xrightarrow{\max_*} \mathbb{P}\overline{\mathbb{R}}$ | $\mathbb{P}\overline{\mathbb{R}} \times \overline{\mathbb{R}} \xrightarrow{\mathbb{E}_{\overline{\mathbb{R}}} \times \mathrm{id}_{\overline{\mathbb{R}}}} \overline{\mathbb{R}} \times \overline{\mathbb{R}} \xrightarrow{\max} \overline{\mathbb{R}}$ |
| post-processing | expectation $\mathbb{E}_{\overline{\mathbb{R}}} : \mathbb{P}\overline{\mathbb{R}} \to \overline{\mathbb{R}}$ | identity $\mathrm{id}_{\overline{\mathbb{R}}} : \overline{\mathbb{R}} \to \overline{\mathbb{R}}$ |

### E.2 Stochastic recursion

In Section 4, we introduced the stochastic state transition and statistic functions. Similarly, we can define the stochastic state-action transition $\mathrm{p}_\pi^{S \times A}$ as follows:

$$\begin{aligned}\mathrm{p}_\pi^{S \times A} &: S \times A \to \mathbb{P}(S \times A) := \langle \mathrm{id}_S, \pi \rangle \circ \mathrm{p} \\ &= (s, a) \mapsto \left( s' \sim \mathrm{p}(s'|s, a), a' \sim \int_S \pi(a'|s')\mathrm{p}(s'|s, a)\, \mathrm{d}s' \right). \end{aligned} \tag{43}$$

The stochastic state-action statistic function $\tau_\pi^{S \times A}$ satisfies the following recursive equation:

$$\begin{aligned}&\tau_\pi^{S \times A} : S \times A \to \mathbb{P}T \\ &= (s, a) \mapsto \tau \sim \begin{cases} \delta_{\text{init}} & s \in S_\omega, \\ \mathrm{r}(s, a) \triangleright \tau' \;\Big|\; \tau' \sim \int_{S \times A} \tau_\pi^{S \times A}(\tau'|s', a')\mathrm{p}_\pi^{S \times A}(s', a'|s, a)\, \mathrm{d}s'\, \mathrm{d}a' & s \notin S_\omega. \end{cases}\end{aligned} \tag{44}$$

Further characterizations of stochastic state/state-action statistic functions, including the (pre)metrics and (pre)orders on statistics, as well as the contractivity of stochastic Bellman (optimality) operators, are left for future work.

### E.3 Relationship between stochastic state and state-action statistic functions

In the stochastic setting, the state/state-action statistic functions are related by the following equations, which are analogous to Theorem A.2:

$$\tau_\pi^S(\tau|s) = \int_A \tau_\pi^{S \times A}(\tau|s, a)\pi(a|s)\, \mathrm{d}a \qquad \text{(for all states)}, \tag{45}$$

$$\tau_\pi^{S \times A}(\tau|s, a) = \mathrm{r}(s, a) \triangleright \int_S \tau_\pi^S(\tau|s')\mathrm{p}(s'|s, a)\, \mathrm{d}s' \qquad \text{(for all non-terminal states)}. \tag{46}$$

### E.4 Expected aggregated rewards vs. aggregated expected rewards

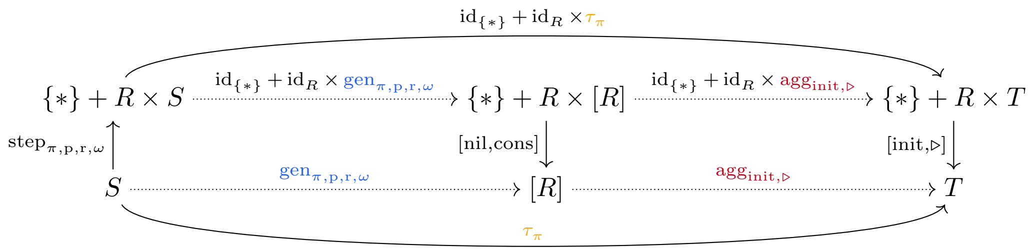
As discussed in Section 4, the expected discounted sum of rewards equals the discounted sum of expected rewards. However, the expected aggregated rewards and the aggregated expected rewards are not equal in general. For example, the expected maximum reward is not equal to the maximum expected reward because the expectation operator does not distribute over the maximum operator, as shown in Table 6. This issue was also raised by Cui & Yu (2023); Veviurko et al. (2024). However, we argue that even though the expected aggregated rewards and the aggregated expected rewards are not equal, they are both valid and useful learning objectives for different purposes, and the choice between them depends on the specific application. If we want to optimize the expected aggregated rewards, a more straightforward approach is to estimate the distributions of the aggregated rewards, using *distributional reinforcement learning* (Morimura et al., 2010a;a; Bellemare et al., 2017; 2023). Further theoretical and empirical investigations are left for future work.

# F Proofs

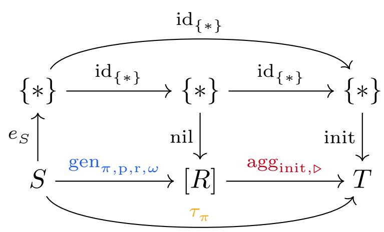
**Theorem 3.2** (Bellman equation for the state statistic function)**.** *Given a recursive generation function* $\mathrm{gen}_{\pi,\mathrm{p},\mathrm{r},\omega}$ *(Definition 2.1) and a recursive statistic aggregation function* $\mathrm{agg}_{\mathrm{init},\triangleright}$ *(Definition 3.1), their composition, called the state statistic function* $\tau_\pi : S \to T$*, satisfies the following equation:*

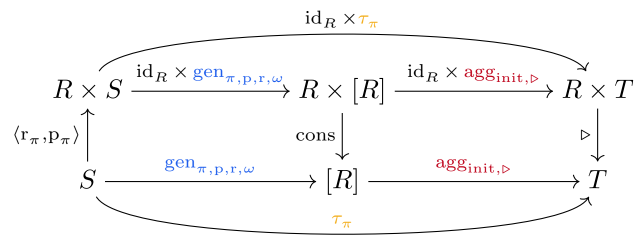
$$\tau_\pi : S \to T := \mathrm{agg}_{\mathrm{init},\triangleright} \circ \mathrm{gen}_{\pi,\mathrm{p},\mathrm{r},\omega} = s \mapsto \begin{cases} \mathrm{init} & s \in S_\omega, \\ \mathrm{r}_\pi(s) \triangleright \tau_\pi(\mathrm{p}_\pi(s)) & s \notin S_\omega. \end{cases} \tag{8}$$

*Proof.* Similarly to the diagram in Eq. (5), the state statistic function $\tau_\pi : S \to T$ can be represented using the following diagram:




which can be non-rigorously interpreted as a "*combination*" of the following two diagrams:




where $e_S : S \to \{*\}$ is the unique function from states to the singleton set, and $\langle \mathrm{r}_\pi, \mathrm{p}_\pi \rangle : S \to R \times S$ is the pairing of the reward and transition functions, which constitute the step function $\mathrm{step}_{\pi,\mathrm{p},\mathrm{r},\omega}$.

The left diagram shows that when a state $s \in S_\omega$ is terminal,

$$\tau_\pi(s) = \mathrm{agg}_{\mathrm{init},\triangleright}(\underbrace{\mathrm{gen}_{\pi,\mathrm{p},\mathrm{r},\omega}(s)}) \qquad \text{(by definition of } \tau_\pi) \tag{47}$$

$$= \underbrace{\mathrm{agg}_{\mathrm{init},\triangleright}}(\mathrm{nil}) \qquad \text{(by terminal condition of } \mathrm{gen}_{\pi,\mathrm{p},\mathrm{r},\omega}) \tag{48}$$

$$= \mathrm{init}\,. \qquad \text{(by initial condition of } \mathrm{agg}_{\mathrm{init},\triangleright}) \tag{49}$$

The right diagram shows that when a state $s \notin S_\omega$ is non-terminal,

$$\tau_\pi(s) = \mathrm{agg}_{\mathrm{init},\triangleright}(\underbrace{\mathrm{gen}_{\pi,\mathrm{p},\mathrm{r},\omega}(s)}) \qquad \text{(by definition of } \tau_\pi) \tag{50}$$

$$= \underbrace{\mathrm{agg}_{\mathrm{init},\triangleright}(\mathrm{cons}}(\mathrm{r}_\pi(s), \mathrm{gen}_{\pi,\mathrm{p},\mathrm{r},\omega}(\mathrm{p}_\pi(s)))) \qquad \text{(by recursive definition of } \mathrm{gen}_{\pi,\mathrm{p},\mathrm{r},\omega}) \tag{51}$$

$$= \mathrm{r}_\pi(s) \triangleright \underbrace{\mathrm{agg}_{\mathrm{init},\triangleright}(\mathrm{gen}_{\pi,\mathrm{p},\mathrm{r},\omega}}(\mathrm{p}_\pi(s))) \qquad \text{(by recursive definition of } \mathrm{agg}_{\mathrm{init},\triangleright}) \tag{52}$$

$$= \mathrm{r}_\pi(s) \triangleright \tau_\pi(\mathrm{p}_\pi(s)). \qquad \text{(by definition of } \tau_\pi) \tag{53}$$

By combining Eq. (49) and Eq. (53), we obtain the desired result in Eq. (8). □

We omit the proof for Theorem A.1 as the derivation is similar to that of Theorem 3.2.

**Lemma C.3** (Pullback premetric)**.** *Let $d_B : B \times B \to [0, \infty]$ be a premetric on a set $B$, and let $f : A \to B$ be a function. The pullback premetric $d_A : A \times A \to [0, \infty]$ is defined by*

$$\forall a_1, a_2 \in A.\ d_A(a_1, a_2) := d_B(f(a_1), f(a_2)). \tag{24}$$

*If $d_B$ is a strict premetric, then $d_A$ is also a strict premetric if and only if the function $f$ is injective.*

*Proof.* The pullback premetric $d_A$ is a premetric because

$$\forall a \in A.\ d_A(a, a) := d_B(f(a), f(a)) = 0. \tag{54}$$

If $d_B$ is a strict premetric, we have

$$\forall a_1, a_2 \in A.\ (d_A(a_1, a_2) := d_B(f(a_1), f(a_2)) = 0) \to (f(a_1) = f(a_2)). \tag{55}$$

For the pullback premetric $d_A$ to be a strict premetric, we require that

$$\forall a_1, a_2 \in A.\ (f(a_1) = f(a_2)) \to (a_1 = a_2), \tag{56}$$

which is equivalent to the injectivity of the function $f$. □

**Lemma D.5** (Pullback preorder)**.** *Let $\leq_B$ be a preorder on a set $B$, and let $f : A \to B$ be a function. The* pullback preorder *$\leq_A$ on a set $A$ is defined by*

$$\forall a_1, a_2 \in A.\ (a_1 \leq_A a_2) := (f(a_1) \leq_B f(a_2)). \tag{30}$$

*If $\leq_B$ is total, then $\leq_A$ is also total. If $\leq_B$ is antisymmetric, then $\leq_A$ is also antisymmetric if and only if $f$ is injective.*

*Proof.* The pullback preorder $\leq_A$ is reflexive because

$$\forall a \in A.\ (a \leq_A a) := (f(a) \leq_B f(a)). \tag{57}$$

The pullback preorder $\leq_A$ is transitive because

$$\forall a_1, a_2, a_3 \in A.\ (a_1 \leq_A a_2) \wedge (a_2 \leq_A a_3) := (f(a_1) \leq_B f(a_2)) \wedge (f(a_2) \leq_B f(a_3)) \tag{58}$$

$$\to (f(a_1) \leq_B f(a_3)) =: (a_1 \leq_A a_3). \tag{59}$$

If $\leq_B$ is total, then $\leq_A$ is also total because

$$\forall a_1, a_2 \in A.\ (a_1 \leq_A a_2) \vee (a_2 \leq_A a_1) := (f(a_1) \leq_B f(a_2)) \vee (f(a_2) \leq_B f(a_1)). \tag{60}$$

If $\leq_B$ is antisymmetric, we have

$$\forall a_1, a_2 \in A.\ (a_1 \leq_A a_2) \wedge (a_2 \leq_A a_1) := (f(a_1) \leq_B f(a_2)) \wedge (f(a_2) \leq_B f(a_1)) \tag{61}$$

$$\to (f(a_1) = f(a_2)). \tag{62}$$

For the pullback preorder $\leq_A$ to be antisymmetric, we require that

$$\forall a_1, a_2 \in A.\ (f(a_1) = f(a_2)) \to (a_1 = a_2), \tag{63}$$

which is equivalent to the injectivity of the function $f$. □

**Lemma D.7** (Preorder-preserving premetric's supremum inequality)**.** *Let $d_B : B \times B \to [0, \infty]$ be a premetric that preserves a premetric $\leq_B$ on a set $B$. Then, for functions $f_1, f_2 : A \to B$ whose suprema are attained in $B$, we have*

$$d_B(\sup_{a \in A} f_1(a), \sup_{a \in A} f_2(a)) \leq \sup_{a \in A} d_B(f_1(a), f_2(a)). \tag{32}$$

*Proof.* By assumption, the functions $f_1$ and $f_2$ have suprema in $B$. We denote $a_1 = \arg\sup_{a \in A} f_1(a)$ and $a_2 = \arg\sup_{a \in A} f_2(a)$. Then, $f_1(a_1) = \sup_{a \in A} f_1(a)$ and $f_2(a_2) = \sup_{a \in A} f_2(a)$.

If $f_1(a_1) \leq_B f_2(a_2)$, we have $f_1(a_2) \leq_B f_1(a_1) \leq_B f_2(a_2)$. By the preorder preservation of the premetric $d_B$, we have

$$d_B(f_1(a_1), f_2(a_2)) \leq d_B(f_1(a_2), f_2(a_2)) \leq \sup_{a \in A} d_B(f_1(a), f_2(a)). \tag{64}$$

Similarly, if $f_2(a_2) \leq_B f_1(a_1)$, we have $f_2(a_1) \leq_B f_2(a_2) \leq_B f_1(a_1)$. By the preorder preservation of the premetric $d_B$, we have

$$d_B(f_1(a_1), f_2(a_2)) \leq d_B(f_1(a_1), f_2(a_1)) \leq \sup_{a \in A} d_B(f_1(a), f_2(a)). \tag{65}$$

Therefore, we have $d_B(\sup_{a \in A} f_1(a), \sup_{a \in A} f_2(a)) \leq \sup_{a \in A} d_B(f_1(a), f_2(a))$. □

We use the following lemmas to prove Theorem 3.5.

**Lemma F.1** (Induced premetric on a set of functions). *Let $d_B : B \times B \to [0, \infty]$ be a premetric on a set $B$. For functions $f, f' : A \to B$, define $d_{[A,B]} : [A, B] \times [A, B] \to [0, \infty]$ as follows:*

$$d_{[A,B]}(f, f') := \sup_{a \in A} d_B(f(a), f'(a)). \tag{66}$$

*Then, $d_{[A,B]}$ is also a premetric. Moreover, if $d_B$ is a strict premetric, $d_{[A,B]}$ is also a strict premetric.*

*Proof.* $d_{[A,B]}$ is a premetric because $d_{[A,B]}(f, f) = \sup_{a \in A} d_B(f(a), f(a)) = 0$. For two functions $f, f' : A \to B$, $d_{[A,B]}(f, f') = \sup_{a \in A} d_B(f(a), f'(a)) = 0$ implies that $d_B(f(a), f'(a)) = 0$ for all $a \in A$. If $d_B$ is a strict premetric, then $d_B(f(a), f'(a)) = 0$ implies $f(a) = f'(a)$ for all $a \in A$, which means that $f = f'$, hence if $d_B$ is a strict premetric, $d_{[A,B]}$ is also a strict premetric. □

**Lemma F.2** (Data processing inequality). *Let $d_{[A,B]}$ be the induced premetric defined in Lemma F.1. For functions $f, f' : A \to B$ and $g : A \to A$, we have*

$$d_{[A,B]}(f \circ g, f' \circ g) \le d_{[A,B]}(f, f'). \tag{67}$$

*Proof.* $d_{[A,B]}(f \circ g, f' \circ g) := \sup_{a \in A} d_B(f(g(a)), f'(g(a))) = \sup_{a' \in g(A)} d_B(f(a'), f'(a'))$ $\le \sup_{a' \in A} d_B(f(a'), f'(a')) =: d_{[A,B]}(f, f')$. □

**Lemma F.3** (Uniqueness of fixed points of a premetric contraction). *Let $a_1$ and $a_2$ be fixed points of a function $f : A \to A$. If the function $f$ is contractive with respect to a premetric $d_A$ on the set $A$, then $d_A(a_1, a_2) = 0$. Moreover, if $d_A$ is a strict premetric, then $a_1 = a_2$.*

*Proof.* Because $a_1$ and $a_2$ are fixed points of $f$, and $f$ is contractive with respect to $d_A$, there exists a constant $k \in [0, 1)$ such that

$$d_A(a_1, a_2) = d_A(f(a_1), f(a_2)) \le k \cdot d_A(a_1, a_2). \tag{68}$$

Given that $d_A(a_1, a_2) \ge 0$, the only possible solution is $d_A(a_1, a_2) = 0$. If $d_A$ is a strict premetric, then $d_A(a_1, a_2) = 0$ implies $a_1 = a_2$. In other words, a premetric contraction has unique fixed points *up to premetric indiscernibility*, while a strict premetric contraction has a unique fixed point. □

**Lemma F.4** (Contraction of Bellman operator). *If the update function $\triangleright$ is contractive with respect to a premetric $d_T$ on statistics $T$ (Definition 3.4), then the Bellman operator $\mathcal{B}^S_\pi$ (Definition 3.3) is contractive with respect to the induced premetric $d_{[S,T]}$ defined in Lemma F.1.*

*Proof.* For any functions $\tau^S_1, \tau^S_2 : S \to T$, we have

$$d_{[S,T]}(\mathcal{B}^S_\pi \tau^S_1, \mathcal{B}^S_\pi \tau^S_2) = \sup_{s \in S} d_T((\mathcal{B}^S_\pi \tau^S_1)(s), (\mathcal{B}^S_\pi \tau^S_2)(s)). \tag{69}$$

When a state $s \in S_\omega$ is terminal, for any $k \in [0, 1)$, we have

$$d_T((\mathcal{B}^S_\pi \tau^S_1)(s), (\mathcal{B}^S_\pi \tau^S_2)(s)) \tag{70}$$
$$= d_T(\mathrm{init}, \mathrm{init}) \qquad \text{(by definition of } \mathcal{B}_\pi) \tag{71}$$
$$= 0 \le k \cdot d_T(\tau^S_1(\mathrm{p}^S_\pi(s)), \tau^S_2(\mathrm{p}^S_\pi(s))) \qquad (d_T \text{ is a premetric}) \tag{72}$$

When a state $s \notin S_\omega$ is non-terminal, there exists a constant $k \in [0, 1)$ such that

$$d_T((\mathcal{B}^S_\pi \tau^S_1)(s), (\mathcal{B}^S_\pi \tau^S_2)(s)) \tag{73}$$
$$= d_T(\mathrm{r}_\pi(s) \triangleright \tau^S_1(\mathrm{p}^S_\pi(s)), \mathrm{r}_\pi(s) \triangleright \tau^S_2(\mathrm{p}^S_\pi(s))) \qquad \text{(by definition of } \mathcal{B}^S_\pi) \tag{74}$$
$$\le k \cdot d_T(\tau^S_1(\mathrm{p}^S_\pi(s)), \tau^S_2(\mathrm{p}^S_\pi(s))) \qquad \text{(by contractivity of } \triangleright) \tag{75}$$

Then, we have

$$d_{[S,T]}(\mathcal{B}^S_\pi \tau^S_1, \mathcal{B}^S_\pi \tau^S_2) \tag{76}$$
$$\le k \cdot \sup_{s \in S} d_T(\tau^S_1(\mathrm{p}^S_\pi(s)), \tau^S_2(\mathrm{p}^S_\pi(s))) \qquad \text{(by monotonicity and homogeneity of sup)} \tag{77}$$
$$= k \cdot d_{[S,T]}(\tau^S_1 \circ \mathrm{p}^S_\pi, \tau^S_2 \circ \mathrm{p}^S_\pi) \qquad \text{(by definition of } d_{[S,T]}) \tag{78}$$
$$\le k \cdot d_{[S,T]}(\tau^S_1, \tau^S_2) \qquad \text{(Lemma F.2)} \tag{79}$$

Therefore, the Bellman operator $\mathcal{B}^S_\pi$ is contractive with respect to the premetric $d_{[S,T]}$. □

**Theorem 3.5** (Uniqueness of fixed points of Bellman operator)**.** *Let $\tau_1, \tau_2 : S \to T$ be fixed points of the Bellman operator $\mathcal{B}_\pi$ (Definition 3.3). If the update function $\triangleright$ is contractive with respect to a premetric $d_T$ on statistics $T$ (Definition 3.4), then $d_T(\tau_1(s), \tau_2(s)) = 0$ for all states $s \in S$. If $d_T$ is a strict premetric, then $\tau_1 = \tau_2 = \tau_\pi$.*

*Proof.* Let $d_{[S,T]}$ be the induced premetric defined in Lemma F.1. By Lemmas F.3 and F.4, we have

$$d_{[S,T]}(\tau_1, \tau_2) = \sup_{s \in S} d_T(\tau_1(s), \tau_2(s)) = 0, \tag{80}$$

which means that $d_T(\tau_1(s), \tau_2(s)) = 0$ for all states $s \in S$. When $d_T$ is a strict premetric, we have $\tau_1 = \tau_2$, which means that $\tau_\pi$ is the unique fixed point of the Bellman operator $\mathcal{B}_\pi$. □

We omit the proof for Theorem C.5 as the derivation is similar to that of Theorem 3.5.

**Theorem 3.7** (Bellman optimality equation for the state statistic function)**.** *Given a preorder $\leq_T$ on statistics $T$, the optimal state statistic function $\tau_*$ (Definition 3.6) satisfies the following equation:*

$$\tau_* : S \to T := s \mapsto \begin{cases} \mathrm{init} & s \in S_\omega, \\ \sup\limits_{a \in A}(\mathrm{r}(s,a) \triangleright \tau_*(\mathrm{p}(s,a))) & s \notin S_\omega. \end{cases} \tag{10}$$

*Proof.* When a state $s \in S_\omega$ is terminal, we have $\tau_*(s) = \mathrm{init}$. When a state $s \notin S_\omega$ is non-terminal, we have

$$\begin{aligned}
\tau_*(s) &:= \tau_{\pi_*}(s) && \text{(by definition of } \tau_*\text{)} \quad (81)\\
&= \mathrm{r}_{\pi_*}(s) \triangleright \tau_*(\mathrm{p}_{\pi_*}(s)) && \text{(by recursive definition of } \tau_{\pi_*}\text{)} \quad (82)\\
&= \mathrm{r}(s, \pi_*(s)) \triangleright \tau_*(\mathrm{p}(s, \pi_*(s))) && \text{(by definitions of } \mathrm{r}_{\pi_*} \text{ and } \mathrm{p}_{\pi_*}\text{)} \quad (83)\\
&= \sup_{a \in A}(\mathrm{r}(s,a) \triangleright \tau_*(\mathrm{p}(s,a))). && \text{(pointwise maximization)} \quad (84)
\end{aligned}$$

□

**Theorem D.10** (Bellman optimality equation for the state-action statistic function)**.** *Given a preorder $\leq_T$ on statistics $T$, the optimal state-action statistic function $\tau_*^{S\times A}$ satisfies the following equation:*

$$\tau_*^{S\times A} : S \times A \to T := (s,a) \mapsto \begin{cases} \mathrm{init} & s \in S_\omega, \\ \sup\limits_{a' \in A}\left(\mathrm{r}(s,a) \triangleright \tau_*^{S\times A}(\mathrm{p}(s,a), a')\right) & s \notin S_\omega. \end{cases} \tag{36}$$

*Proof.* When a state $s \in S_\omega$ is terminal, we have $\tau_*^{S\times A}(s,a) = \mathrm{init}$ for all actions $a \in A$. When a state $s \notin S_\omega$ is non-terminal, we have

$$\begin{aligned}
\tau_*^{S\times A}(s,a) &:= \tau_{\pi_*}^{S\times A}(s,a) && \text{(by definition of } \tau_*^{S\times A}\text{)} \quad (85)\\
&= \mathrm{r}(s,a) \triangleright \tau_*^{S\times A}(\mathrm{p}_{\pi_*}^{S\times A}(s,a)) && \text{(by recursive definition of } \tau_{\pi_*}^{S\times A}\text{)} \quad (86)\\
&= \mathrm{r}(s,a) \triangleright \tau_*^{S\times A}(\mathrm{p}(s,a), \pi_*(\mathrm{p}(s,a))) && \text{(by definition of } \mathrm{p}_{\pi_*}^{S\times A}\text{)} \quad (87)\\
&= \sup_{a' \in A}\left(\mathrm{r}(s,a) \triangleright \tau_*^{S\times A}(\mathrm{p}(s,a), a')\right). && \text{(pointwise maximization)} \quad (88)
\end{aligned}$$

□

Similarly to Lemma F.4 and Theorem 3.5, we use the following lemma to prove Theorem D.11.

**Lemma F.5** (Contraction of Bellman optimality operator). *If the update function $\triangleright$ is contractive with respect to a premetric $d_T$ on statistics $T$ (Definition 3.4), and the premetric $d_T$ preserves the preorder $\leq_T$ on statistics $T$ (Definition D.6), then the Bellman optimality operator $\mathcal{B}_*^S$ (Definition D.8) is contractive with respect to the induced premetric $d_{[S,T]}$ defined in Lemma F.1.*

*Proof.* For any functions $\tau_1^S, \tau_2^S : S \to T$, we have

$$d_{[S,T]}(\mathcal{B}_*^S \tau_1^S, \mathcal{B}_*^S \tau_2^S) = \sup_{s \in S} d_T((\mathcal{B}_*^S \tau_1^S)(s), (\mathcal{B}_*^S \tau_2^S)(s)). \tag{89}$$

When a state $s \in S_\omega$ is terminal, for any $k \in [0, 1)$, we have

$$\begin{aligned}
& d_T((\mathcal{B}_*^S \tau_1^S)(s), (\mathcal{B}_*^S \tau_2^S)(s)) && (90)\\
&= d_T(\mathrm{init}, \mathrm{init}) && \text{(by definition of } \mathcal{B}_*^S\text{)} \quad (91)\\
&= 0 \leq k \cdot \sup_{a \in A} d_T(\tau_1^S(\mathrm{p}(s,a)), \tau_2^S(\mathrm{p}(s,a))) && (d_T \text{ is a premetric}) \quad (92)
\end{aligned}$$

When a state $s \notin S_\omega$ is non-terminal, there exists a constant $k \in [0, 1)$ such that

$$\begin{aligned}
& d_T((\mathcal{B}_*^S \tau_1^S)(s), (\mathcal{B}_*^S \tau_2^S)(s)) && (93)\\
&= d_T(\sup_{a \in A}(\mathrm{r}(s,a) \triangleright \tau_1^S(\mathrm{p}(s,a))), \sup_{a \in A}(\mathrm{r}(s,a) \triangleright \tau_2^S(\mathrm{p}(s,a)))) && \text{(by definition of } \mathcal{B}_*\text{)} \quad (94)\\
&\leq \sup_{a \in A} d_T(\mathrm{r}(s,a) \triangleright \tau_1^S(\mathrm{p}(s,a)), \mathrm{r}(s,a) \triangleright \tau_2^S(\mathrm{p}(s,a))) && \text{(by monotonicity of } d_T\text{)} \quad (95)\\
&\leq \sup_{a \in A} k \cdot d_T(\tau_1^S(\mathrm{p}(s,a)), \tau_2^S(\mathrm{p}(s,a))) && \text{(by contractivity of } \triangleright\text{)} \quad (96)\\
&= k \cdot \sup_{a \in A} d_T(\tau_1^S(\mathrm{p}(s,a)), \tau_2^S(\mathrm{p}(s,a))) && \text{(by homogeneity of sup)} \quad (97)
\end{aligned}$$

Then, we have

$$\begin{aligned}
& d_{[S,T]}(\mathcal{B}_*^S \tau_1, \mathcal{B}_*^S \tau_2) && (98)\\
&\leq k \cdot \sup_{s \in S} \sup_{a \in A} d_T(\tau_1^S(\mathrm{p}(s,a)), \tau_2^S(\mathrm{p}(s,a))) && \text{(by monotonicity and homogeneity of sup)} \quad (99)\\
&= k \cdot \sup_{a \in A} \sup_{s \in S} d_T(\tau_1^S(\mathrm{p}(s,a)), \tau_2^S(\mathrm{p}(s,a))) && \text{(by commutativity of sup)} \quad (100)\\
&= k \cdot \sup_{a \in A} d_{[S,T]}(\tau_1^S \circ \mathrm{p}(-,a), \tau_2^S \circ \mathrm{p}(-,a)) && \text{(by definition of } d_{[S,T]}\text{)} \quad (101)\\
&\leq k \cdot d_{[S,T]}(\tau_1^S, \tau_2^S) && \text{(Lemma F.2)} \quad (102)
\end{aligned}$$

Therefore, the Bellman optimality operator $\mathcal{B}_*^S$ is contractive with respect to the premetric $d_{[S,T]}$. □

**Theorem D.11** (Uniqueness of fixed points of Bellman optimality operator). *Let $\tau_1^S, \tau_2^S : S \to T$ be fixed points of the Bellman optimality operator $\mathcal{B}_*^S$ (Definition D.8). If the update function $\triangleright$ is contractive with respect to a premetric $d_T$ on statistics $T$ (Definition 3.4), and the premetric $d_T$ preserves the preorder $\leq_T$ on statistics $T$ (Definition D.6), then $d_T(\tau_1^S(s), \tau_2^S(s)) = 0$ for all states $s \in S$. If $d_T$ is a strict premetric, then $\tau_1^S = \tau_2^S = \tau_*^S$.*

*Proof.* Let $d_{[S,T]}$ be the induced premetric defined in Lemma F.1. By Lemmas F.3 and F.5, we have

$$d_{[S,T]}(\tau_1, \tau_2) = \sup_{s \in S} d_T(\tau_1^S(s), \tau_2^S(s)) = 0, \tag{103}$$

which means that $d_T(\tau_1^S(s), \tau_2^S(s)) = 0$ for all states $s \in S$. When $d_T$ is a strict premetric, we have $\tau_1^S = \tau_2^S$, which means that $\tau_*^S$ is the unique fixed point of the Bellman optimality operator $\mathcal{B}_*^S$. □

We omit the proof for Theorem D.12 as the derivation is similar to that of Theorem D.11.

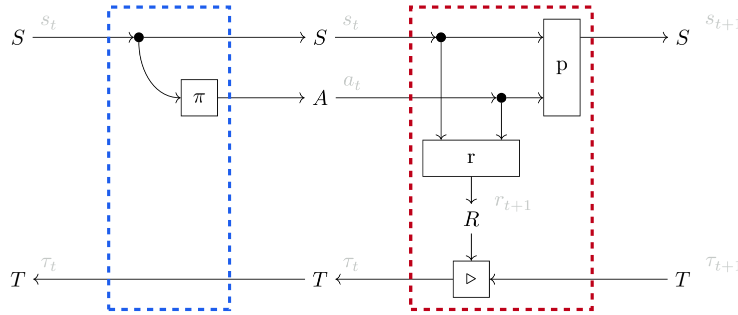



Figure 13: $\tau_\pi^S = \tau_\pi^{S\times A} \circ \langle \mathrm{id}_S, \pi\rangle$ and $\tau_\pi^{S\times A} = \mathrm{r} \triangleright (\tau_\pi^S \circ \mathrm{p})$

**Theorem A.2** (Relationship between state and state-action statistic functions)**.** *Given a recursive generation function* $\mathrm{gen}_{\pi,\mathrm{p},\mathrm{r},\omega}$ *(Definition 2.1) and a recursive statistic aggregation function* $\mathrm{agg}_{\mathrm{init},\triangleright}$ *(Definition 3.1), the state statistic function* $\tau_\pi^S : S \to T$ *in Eq. (8) and the state-action statistic function* $\tau_\pi^{S\times A} : S \times A \to T$ *in Eq. (15) satisfy the following equations:*

$$\tau_\pi^S = \tau_\pi^{S\times A} \circ \langle \mathrm{id}_S, \pi\rangle : S \to T \qquad \textit{(for all states),} \qquad (16)$$

$$\tau_\pi^{S\times A} = \mathrm{r} \triangleright (\tau_\pi^S \circ \mathrm{p}) : S \times A \to T \qquad \textit{(for all non-terminal states).} \qquad (17)$$

*Proof.* Notice the following relation:

$$\mathrm{p}_\pi^{S\times A} \circ \langle \mathrm{id}_S, \pi\rangle = \langle \mathrm{id}_S, \pi\rangle \circ \mathrm{p} \circ \langle \mathrm{id}_S, \pi\rangle = \langle \mathrm{id}_S, \pi\rangle \circ \mathrm{p}_\pi^S : S \to S \times A. \qquad (104)$$

We can show that when a state $s \in S_\omega$ is terminal,

$$\left(\mathrm{gen}_{\pi,\mathrm{p},\mathrm{r},\omega}^{S\times A} \circ \langle \mathrm{id}_S, \pi\rangle\right)(s) = \mathrm{gen}_{\pi,\mathrm{p},\mathrm{r},\omega}^S(s) = [\,], \qquad (105)$$

and when a state $s \notin S_\omega$ is non-terminal,

$$\left(\mathrm{gen}_{\pi,\mathrm{p},\mathrm{r},\omega}^{S\times A} \circ \langle \mathrm{id}_S, \pi\rangle\right)(s) = \left(\mathrm{cons} \circ \langle \mathrm{r}, \mathrm{gen}_{\pi,\mathrm{p},\mathrm{r},\omega}^{S\times A} \circ \mathrm{p}_\pi^{S\times A}\rangle \circ \langle \mathrm{id}_S, \pi\rangle\right)(s) \qquad (106)$$

$$= \left(\mathrm{cons} \circ \langle \mathrm{r} \circ \langle \mathrm{id}_S, \pi\rangle, \mathrm{gen}_{\pi,\mathrm{p},\mathrm{r},\omega}^{S\times A} \circ \mathrm{p}_\pi^{S\times A} \circ \langle \mathrm{id}_S, \pi\rangle\rangle\right)(s) \qquad (107)$$

$$= \left(\mathrm{cons} \circ \langle \mathrm{r}_\pi, \mathrm{gen}_{\pi,\mathrm{p},\mathrm{r},\omega}^{S\times A} \circ \langle \mathrm{id}_S, \pi\rangle \circ \mathrm{p}_\pi^S\rangle\right)(s), \qquad (108)$$

which shows that $\mathrm{gen}_{\pi,\mathrm{p},\mathrm{r},\omega}^{S\times A} \circ \langle \mathrm{id}_S, \pi\rangle$ satisfies the same recursive equation as $\mathrm{gen}_{\pi,\mathrm{p},\mathrm{r},\omega}^S$ in Eq. (2). Due to the uniqueness of the recursive coalgebra (Hinze et al., 2010, Eq. (5)), we can conclude that

$$\mathrm{gen}_{\pi,\mathrm{p},\mathrm{r},\omega}^S = \mathrm{gen}_{\pi,\mathrm{p},\mathrm{r},\omega}^{S\times A} \circ \langle \mathrm{id}_S, \pi\rangle : S \to [R]. \qquad (109)$$

Given Eq. (109), we have

$$\tau_\pi^S := \mathrm{agg}_{\mathrm{init},\triangleright} \circ \mathrm{gen}_{\pi,\mathrm{p},\mathrm{r},\omega}^S = \mathrm{agg}_{\mathrm{init},\triangleright} \circ \mathrm{gen}_{\pi,\mathrm{p},\mathrm{r},\omega}^{S\times A} \circ \langle \mathrm{id}_S, \pi\rangle = \tau_\pi^{S\times A} \circ \langle \mathrm{id}_S, \pi\rangle : S \to T. \qquad (110)$$

Next, for a non-terminal state $s \notin S_\omega$ and an action $a \in A$, we have

$$\tau_\pi^{S\times A}(s, a) = \left(\mathrm{r} \triangleright \left(\tau_\pi^{S\times A} \circ \mathrm{p}_\pi^{S\times A}\right)\right)(s, a) \qquad (111)$$

$$= \left(\mathrm{r} \triangleright \left(\tau_\pi^{S\times A} \circ \langle \mathrm{id}_S, \pi\rangle \circ \mathrm{p}\right)\right)(s, a) \qquad (112)$$

$$= \left(\mathrm{r} \triangleright \left(\tau_\pi^S \circ \mathrm{p}\right)\right)(s, a). \qquad (113)$$

However, for a terminal state $s \in S_\omega$ and an action $a \in A$, the equation $\tau_\pi^{S\times A} = \mathrm{r} \triangleright (\tau_\pi^S \circ \mathrm{p})$ may not always hold and could require additional conditions on the transition function $\mathrm{p}$, the reward function $\mathrm{r}$, the initial value $\mathrm{init}$, and the update function $\triangleright$.

Intuitively, Eqs. (16) and (17) arise from the decomposition of the bidirectional process, as illustrated in Fig. 13. □

*Remark* 6. In fact, we can derive Eq. (109) directly from the relation between the state step function $\mathrm{step}^{S}_{\pi,\mathrm{p},\mathrm{r},\omega}$ and the state-action step function $\mathrm{step}^{S\times A}_{\pi,\mathrm{p},\mathrm{r},\omega}$.

When a state $s \in S_\omega$ is terminal,

$$\left(\mathrm{step}^{S\times A}_{\pi,\mathrm{p},\mathrm{r},\omega} \circ \langle \mathrm{id}_S, \pi\rangle\right)(s) = \left(\mathrm{id}_{\{*\}} \circ \mathrm{step}^{S}_{\pi,\mathrm{p},\mathrm{r},\omega}\right)(s) = *, \tag{114}$$

and when a state $s \notin S_\omega$ is non-terminal,

$$\begin{aligned}
\left(\mathrm{step}^{S\times A}_{\pi,\mathrm{p},\mathrm{r},\omega} \circ \langle \mathrm{id}_S, \pi\rangle\right)(s) &= \left(\langle \mathrm{r}, \mathrm{p}^{S\times A}_{\pi}\rangle \circ \langle \mathrm{id}_S, \pi\rangle\right)(s) && (115)\\
&= \left(\langle \mathrm{r} \circ \langle \mathrm{id}_S, \pi\rangle, \mathrm{p}^{S\times A}_{\pi} \circ \langle \mathrm{id}_S, \pi\rangle\rangle\right)(s) && (116)\\
&= \left(\langle \mathrm{r}_\pi, \langle \mathrm{id}_S, \pi\rangle \circ \mathrm{p}^{S}_{\pi}\rangle\right)(s) && (117)\\
&= \left((\mathrm{id}_R \times \langle \mathrm{id}_S, \pi\rangle) \circ \langle \mathrm{r}_\pi, \mathrm{p}^{S}_{\pi}\rangle\right)(s) && (118)\\
&= \left((\mathrm{id}_R \times \langle \mathrm{id}_S, \pi\rangle) \circ \mathrm{step}^{S}_{\pi,\mathrm{p},\mathrm{r},\omega}\right)(s). && (119)
\end{aligned}$$

We can conclude that

$$\mathrm{step}^{S\times A}_{\pi,\mathrm{p},\mathrm{r},\omega} \circ \langle \mathrm{id}_S, \pi\rangle = \left(\mathrm{id}_{\{*\}} + \mathrm{id}_R \times \langle \mathrm{id}_S, \pi\rangle\right) \circ \mathrm{step}^{S}_{\pi,\mathrm{p},\mathrm{r},\omega} : S \to \{*\} + R \times (S \times A), \tag{120}$$

which means that $\langle \mathrm{id}_S, \pi\rangle$ is a *coalgebra homomorphism* from the state step function $\mathrm{step}^{S}_{\pi,\mathrm{p},\mathrm{r},\omega}$ to the state-action step function $\mathrm{step}^{S\times A}_{\pi,\mathrm{p},\mathrm{r},\omega}$. Then, by the *coalgebra fusion law* (Hinze et al., 2010, Eq. (7)), we can get the result in Eq. (109).

# G Learning algorithms with recursive reward aggregation

In this section, we list the RL algorithms with recursive reward aggregation used in our experiments. The colored lines indicate modifications compared to the standard discounted sum version.

## G.1 Q-learning

**Algorithm 1** Q-learning (Watkins & Dayan, 1992) with recursive reward aggregation

**Input:** transition function $\mathrm{p} : S \times A \to S$, reward function $\mathrm{r} : S \times A \to R$, terminal condition $\omega$, recursive reward aggregation function $\mathrm{post} \circ \mathrm{agg}_{\mathrm{init},\triangleright} : [R] \to R$
**Parameters:** learning rate $\alpha \in (0, 1]$, exploration parameter $\epsilon \in (0, 1)$
**Initialize** state-action statistic function $\tau : S \times A \to T$ with initial value $\mathrm{init} \in T$
**for** each episode **do**
  **Initialize** state $s$
  **while** $s$ is not terminal **do**
    **Compute** state-action value function $\mathrm{q}(s, a) = \mathrm{post}(\tau(s, a))$ for state $s$ and all actions $a$
    Select action $a$ using $\epsilon$-greedy policy based on value function $\mathrm{q}(s, a)$
    Execute action $a$, observe next state $s'$ and reward $r$ according to $\mathrm{p}$ and $\mathrm{r}$
    **Update** statistic function $\tau$:
    $\tau(s, a) \leftarrow \tau(s, a) + \alpha\left(\max_{a' \in A}\left(r \triangleright \tau(s', a')\right) - \tau(s, a)\right),$
    where $\max_{a' \in A}\left(r \triangleright \tau(s', a')\right) = r \triangleright \tau(s', a^*)$ and $a^* = \arg\max_{a' \in A} \mathrm{post}\left(r \triangleright \tau(s', a')\right)$
    **Update** state $s \leftarrow s'$
  **end while**
**end for**
**Output:** estimated optimal statistic function $\tau$, optimal value function $\mathrm{q}(s, a) = \mathrm{post}(\tau(s, a))$, and optimal policy $\pi(s) = \arg\max_{a \in A} \mathrm{q}(s, a)$

### G.2 PPO

**Algorithm 2** PPO (Schulman et al., 2017) with recursive reward aggregation

**Input:** transition function $\mathrm{p} : S \times A \to S$, reward function $\mathrm{r} : S \times A \to R$, terminal condition $\omega$, recursive reward aggregation function $\mathrm{post} \circ \mathrm{agg}_{\mathrm{init}, \triangleright} : [R] \to R$

**Parameters:** bias-variance trade-off parameter $\lambda \in [0, 1]$, clipping parameter $\epsilon$, critic loss coefficient $c_1$, entropy regularization coefficient $c_2$

**Initialize** parameterized policy function (actor) $\pi_\theta : S \to A$

**Initialize** parameterized state statistic function (critic) $\tau_\phi : S \to T$

**for** each iteration **do**

  **Initialize** state $s$

  Collect trajectories of states and rewards following policy $\pi_\theta$ till the end of the horizon $\Omega$

  **Compute** statistics $\hat{\tau}_t^{(i)} = r_t \triangleright r_{t+1} \triangleright \cdots \triangleright r_{t+i-1} \triangleright \tau_\phi(s_{t+i})$ for $i = 1, \ldots, \Omega - t$

  **Compute** state value function $\mathrm{v}_\phi(s_t) = \mathrm{post}(\tau_\phi(s_t))$

  **Compute** advantage estimates $\hat{\alpha}_t^{(i)} = \mathrm{post}(\hat{\tau}_t^{(i)}) - \mathrm{v}_\phi(s_t)$ for $i = 1, \ldots, \Omega - t$

  Use one of the following as advantage $\hat{\alpha}_t$:

  - $\hat{\alpha}_t^{(1)} = \mathrm{post}(r_t \triangleright \tau_\phi(s_{t+1})) - \mathrm{v}_\phi(s_t)$
  - $\hat{\alpha}_t^{(\Omega-t)} = \mathrm{post}(r_t \triangleright r_{t+1} \triangleright \cdots \triangleright \tau_\phi(s_\Omega)) - \mathrm{v}_\phi(s_t)$
  - generalized advantage estimates (GAE) (Schulman et al., 2016) $(1 - \lambda) \sum_{i=1}^{\Omega-t} \lambda^{i-1} \hat{\alpha}_t^{(i)}$

  **Compute** critic loss: $L_c(\phi) = \sum_{t=1}^{\Omega} \left( \mathrm{v}_\phi(s_t) - \mathrm{post}(\hat{\tau}_t^{(\Omega-t)}) \right)^2$

  **Compute** actor loss $L_a(\theta)$ with clipping or penalty using advantage $\hat{\alpha}_t$ (Schulman et al., 2017)

  **Compute** entropy regularization $H(\theta)$

  **Optimize** $L_a(\theta) - c_1 L_c(\phi) + c_2 H(\theta)$

**end for**

**Output:** estimated optimal statistic function $\tau_\phi$ and optimal policy $\pi_\theta$

### G.3 TD3

**Algorithm 3** TD3 (Fujimoto et al., 2018) with recursive reward aggregation

**Input:** transition function $\mathrm{p} : S \times A \to S$, reward function $\mathrm{r} : S \times A \to R$, terminal condition $\omega$, recursive reward aggregation function $\mathrm{post} \circ \mathrm{agg}_{\mathrm{init},\triangleright} : [R] \to R$
**Parameters:** exploration noise parameter $\sigma$, target noise parameter $\tilde{\sigma}$, target clipping parameter $c$, soft target update rate $\lambda \in (0, 1)$, maximum action limit $a_{\max}$
**Initialize** parameterized policy function (actor) $\pi_\theta : S \to A$
**Initialize** two parameterized state-action statistic functions (critics) $\tau_{\phi_1}, \tau_{\phi_2} : S \times A \to T$
**Initialize** targets $\pi_{\theta'} \leftarrow \pi_\theta$, $\tau_{\phi'_1} \leftarrow \tau_{\phi_1}$, $\tau_{\phi'_2} \leftarrow \tau_{\phi_2}$, and replay buffer $\mathcal{D}$
**for** each iteration **do**
  Observe state $s$ and select action $a = \pi_\theta(s) + \epsilon$ with exploration noise $\epsilon \sim \mathcal{N}(0, \sigma)$
  Observe next state $s'$, reward $r$, and done signal $d$ (whether $s'$ is terminal)
  Store transition tuple $(s, a, r, s', d)$ in buffer $\mathcal{D}$
  **if** $s'$ is terminal **then**
    Reset environment state
  **end if**
  **if** update critics **then**
    Randomly sample a batch of transitions $B = \{(s, a, r, s', d)\}$ from $\mathcal{D}$
    **Compute** target actions $\tilde{a} = \mathrm{clip}(\pi_{\theta'}(s') + \epsilon, -a_{\max}, a_{\max})$, $\epsilon \sim \mathrm{clip}(\mathcal{N}(0, \tilde{\sigma}), -c, c)$
    **Update** target critic $\tau_{\text{target}}$:

$$\tau_{\text{target}} \leftarrow \begin{cases} \mathrm{init} & d = 1, \\ r_i \triangleright \tau_{\phi'_1}(s', \tilde{a}) & \mathrm{post}(r_i \triangleright \tau_{\phi'_1}(s', \tilde{a})) \leq \mathrm{post}(r_i \triangleright \tau_{\phi'_2}(s', \tilde{a})), \\ r_i \triangleright \tau_{\phi'_2}(s', \tilde{a}) & \mathrm{post}(r_i \triangleright \tau_{\phi'_2}(s', \tilde{a})) \leq \mathrm{post}(r_i \triangleright \tau_{\phi'_1}(s', \tilde{a})). \end{cases}$$

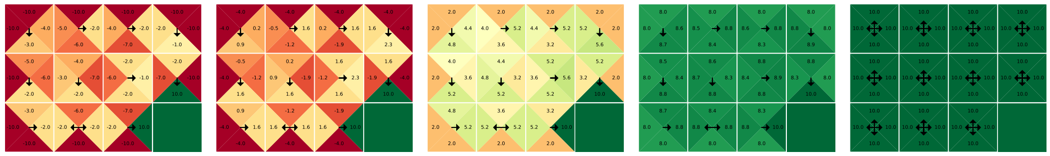
    **Update** critics $\tau_{\phi_i}$ by one step of gradient descent:

$$\nabla_{\phi_i} \frac{1}{|B|} \sum_{(s,a,r,s',d) \in B} \left(\mathrm{post}(\tau_{\phi_i}(s, a)) - \mathrm{post}(\tau_{\text{target}})\right)^2 \quad \text{for } i = 1, 2$$

  **end if**
  **if** update actor **then**
    **Update** actor by one step of gradient ascent using

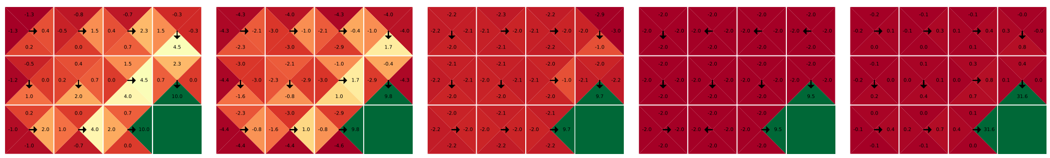
$$\nabla_\theta \frac{1}{|B|} \sum_{(s,a,r,s',d) \in B} \mathrm{post}\left(\tau_{\phi_1}(s, \pi_\theta(s))\right)$$

    **Update** targets with
    $\tau_{\phi'_i} \leftarrow \lambda \tau_{\phi_i} + (1 - \lambda) \tau_{\phi'_i}$ for $i = 1, 2$
    $\pi_{\theta'} \leftarrow \lambda \pi_\theta + (1 - \lambda) \pi_{\theta'}$
  **end if**
**end for**
**Output:** estimated optimal statistic functions $\tau_{\phi_1}$ and $\tau_{\phi_2}$, and optimal policy $\pi_\theta$



(a) min (b) $0.7 \min + 0.3 \max$ (c) $0.4 \min + 0.6 \max$ (d) $0.1 \min + 0.9 \max$ (e) max

Figure 14: $\max - \alpha \, \mathrm{range} = \alpha \min + (1 - \alpha) \max$.



(a) mean (b) $\mathrm{mean} - 0.5 \, \mathrm{std}$ (c) $\mathrm{mean} - 1.0 \, \mathrm{std}$ (d) $\mathrm{mean} - 1.5 \, \mathrm{std}$ (e) $\mathrm{mean}/\mathrm{std}$

Figure 15: $\mathrm{mean} - \alpha \, \mathrm{std}$ and Sharpe ratio $\mathrm{mean}/\mathrm{std}$.

# H Experiments

In this section, we provide detailed descriptions of the environments used in our experiments and the specific configurations and hyperparameters employed for each task. We also present additional results for the grid-world and continuous control environments.

## H.1 Grid-world environment

**Implementation** We implemented the environment and the Q-learning (Watkins & Dayan, 1992) algorithm using NumPy (Harris et al., 2020).

**Hyperparameters** We used a fixed exploration parameter of 0.3. We trained agents for total training timesteps of 10 000.

**Additional results** Similarly to Fig. 4, which showed the policy preferences of discounted sum, discounted max, min, and mean, Fig. 14 shows the policy preferences range-regularized max, which is an interpolation between min and max. Meanwhile, Fig. 15 shows the policy preferences of standard-deviation-regularized mean and Sharpe ratio.

## H.2 Wind-world environment

**Implementation** We implemented the environment and the PPO (Schulman et al., 2017) algorithm using JAX (Bradbury et al., 2018) and gymnax (Lange, 2022).

**Hyperparameters** The PPO clipping parameter was set to 0.2. We used a critic loss coefficient of 0.5 and an entropy regularization coefficient of 0.01. We trained agents using 64 parallel environments for total training timesteps of 500 000.

## H.3 Continuous control environments

The **Hopper** environment is a classic continuous control task from the MuJoCo physics simulation suite (Todorov et al., 2012), where a 2D one-legged robot must learn to balance and move forward efficiently. The agent controls three joints (thigh, knee, and foot) to generate locomotion while maintaining stability. The reward function in Hopper consists of three key components: (i) *healthy reward*, which incentivizes the agent to remain upright; (ii) *forward reward*, which encourages the agent to move forward; and (iii) *control cost*, which penalizes excessive energy use. Then, the total reward function is given by:

$$\text{reward} = \text{healthy reward} + \text{forward reward} - \text{control cost}. \tag{121}$$

The Hopper environment terminates when the agent is deemed unhealthy or reaches the predefined episode length limit. The agent is considered unhealthy if its state variables exceed the allowed range, its height falls below a certain threshold, or its torso angle deviates beyond a specified limit, indicating a loss of stability. If none of these conditions occur, the episode continues until the maximum duration is reached.

The **Ant** environment is also from the MuJoCo physics simulation suite (Todorov et al., 2012), where the four-legged quadrupedal robot must learn to efficiently balance and move forward. The agent controls eight joints (two per leg) to generate stable locomotion while adapting to dynamic interactions with the environment. The reward function in the Ant environment is designed to encourage forward movement while maintaining stability and efficiency. It consists of four key components: (i) a *healthy reward*, which provides a fixed bonus as long as the agent remains upright; (ii) a *forward reward*, which encourages movement in the positive x-direction; (iii) a *control cost*, which penalizes excessive actions to promote energy efficiency; and (iv) a *contact cost*, which discourages large external contact forces. The total reward is calculated by summing the healthy and forward rewards while subtracting the penalties for control effort and contact forces:

$$\text{reward} = \text{healthy reward} + \text{forward reward} - \text{control cost} - \text{contact cost}. \tag{122}$$

In some versions of the environment, the contact cost may be excluded from the reward calculation. The Ant environment terminates when the agent is deemed unhealthy or when the episode reaches its maximum duration of 1000 timesteps. The agent is considered unhealthy if any of its state space values become non-finite or if its torso height falls outside a predefined range, indicating a loss of stability. If neither of these conditions occur, the episode continues until it reaches the time limit.

The **Lunar Lander Continuous** environment, part of the Box2D physics simulation suite (Brockman et al., 2016), involves controlling a lunar lander to safely land on a designated landing pad. The agent has continuous thrust control over the main engine and two side thrusters, which it must use efficiently to achieve a stable landing while minimizing fuel consumption. The reward function is designed to encourage precise and efficient landings. The agent receives positive rewards for (i) moving closer to the landing pad, (ii) achieving a soft landing, and (iii) staying upright. Conversely, penalties are applied for (i) excessive fuel usage, (ii) high-impact landings, and (iii) drifting too far from the target. The episode terminates if the lander successfully lands within the designated zone, crashes, or drifts out of bounds. If none of these conditions occur, the episode continues until reaching the time limit.

**Implementation** We conducted experiments using a modified version of the TD3 (Fujimoto et al., 2018) implementation from Stable-Baselines3 (Raffin et al., 2021).

**Hyperparameters** Our agent performed 100 gradient updates per training episode and used a learning rate of $3 \times 10^{-4}$ to ensure stable learning. Apart from these, our training setup adheres to the default hyperparameters and network architecture of Stable-Baselines3.

**Computational resource** Training a single agent takes approximately 1 hour on an NVIDIA RTX 2080 GPU, with a single CPU core used for environment simulation.

**Additional results: Ant** We provide additional results for the Ant environment, with corresponding animations available at `https://anonymous.4open.science/status/RRA-534F`.

The experimental results in the Ant environment demonstrate the impact of different reward aggregation strategies on agent behavior and performance. The discounted sum ($\text{sum}_{0.99}$) aggregation, serving as the baseline, achieves balanced performance across multiple metrics, effectively promoting stable and efficient locomotion. In contrast, the discounted max ($\text{max}_{0.99}$) aggregation prioritizes obtaining the highest possible reward at an individual time step, leading to highly aggressive movements. As a result, the agent exhibits excessive speed, which ultimately causes instability and

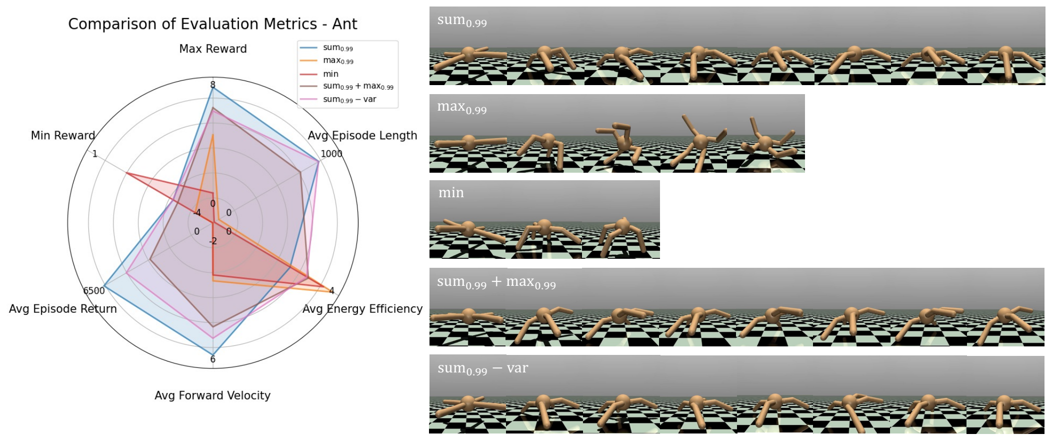



Figure 16: Comparison of evaluation metrics for different reward aggregation methods in the Ant environment. The radar chart on the left visualizes the performance of different reward aggregation functions across multiple evaluation metrics over four random seeds. The images on the right illustrate the learned behavior of the agent for each reward aggregation method.

results in the agent losing control and rolling over. The min ($\min$) aggregation prioritizes minimizing the risk of low rewards, leading to an overly conservative strategy. Instead of efficient locomotion, the agent adopts passive or static behavior, often staying close to the ground to avoid unfavorable rewards. This lack of exploration and controlled movement results in instability, ultimately causing the agent to collapse and terminate early due to height constraints. Moreover, the discounted sum plus max ($\text{sum}_{0.99} + \text{max}_{0.99}$) aggregation drives the agent to optimize both cumulative and peak rewards, resulting in highly aggressive movements. As seen in the motion sequence, the agent exhibits rapid and unstable locomotion, frequently pushing its limits for immediate gains. While this reduces stability, it does not significantly hinder performance, as shown in the radar chart, where reward-related metrics remain high. This suggests that despite instability and occasional failures, the agent achieves strong overall performance at the cost of higher energy consumption and inconsistency. Finally, the discounted sum minus variance ($\text{sum}_{0.99} - \text{var}$) aggregation prioritizes stability by penalizing reward fluctuations, leading to more controlled and consistent locomotion. As seen in the motion sequence, the agent maintains a steady gait and avoids overly aggressive movements, unlike the $\text{sum}_{0.99} + \text{max}_{0.99}$ aggregation. This leads to longer episode durations, as reflected in the radar chart. However, while reducing variance enhances stability, it also limits the ability of agent to explore high-reward strategies, leading to robust locomotion at the cost of suboptimal overall performance.

**Additional results: Lunar Lander Continuous** We provide additional results for the Lunar Lander Continuous environment, with corresponding animations available at https://anonymous.4open.science/status/RRA-534F.

The experimental results in Lunar Lander Continuous, a goal-reaching environment, demonstrate the impact of different reward aggregation strategies on the agent's landing behavior and overall performance in this specific task. With the $\text{sum}_{0.99}$ aggregation, which serves as the baseline, the agent learns a balanced landing strategy, effectively managing thrust control to achieve a smooth descent while minimizing fuel consumption. The $\text{max}_{0.99}$ aggregation encourages the agent to seek high instantaneous rewards, leading to aggressive thrusting behaviors. As a consequence, the lander may exhibit erratic flight patterns, either applying excessive thrust to maximize immediate reward or failing to decelerate properly, which increases the likelihood of hard landings, instability, or even complete mission failure. This outcome underscores the risk of optimizing for short-term reward spikes at the expense of long-term stability and control. The $\min$ aggregation demonstrates its effectiveness in risk-averse tasks, as it prioritizes maximizing the worst-case outcomes rather

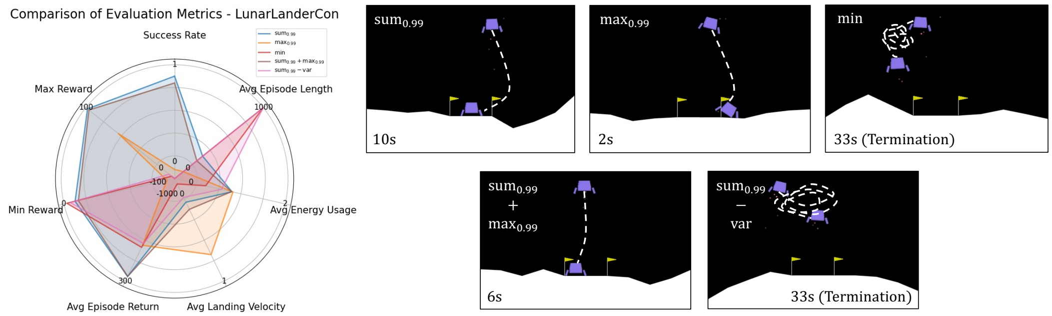



Figure 17: Comparison of evaluation metrics for different reward aggregation methods in the Lunar Lander Continuous environment. The radar chart on the left visualizes the performance of different reward aggregation functions across multiple evaluation metrics over four random seeds. The images on the right illustrate the learned behavior of the agent for each reward aggregation method.

than accumulate reward. As shown in the motion sequence, the lander exhibits a cautious descent, avoiding high-impact crashes by limiting drastic thrust adjustments. Furthermore, since goal-reaching tasks inherently align cumulative and peak rewards, the $\mathrm{sum}_{0.99} + \mathrm{max}_{0.99}$ aggregation performs similarly to $\mathrm{sum}_{0.99}$, as both encourage stable and efficient landings without introducing significant behavioral differences. Finally, in the $\mathrm{sum}_{0.99} - \mathrm{var}$ aggregation, the lander remains airborne, ultimately leading to mission termination. This occurs because both successful and failed landings yield large positive or negative rewards, the agent attempts to avoid these extremes, increasing variance and leading to hesitant and inefficient control. This failure underscores the mismatch between variance minimization and goal-reaching tasks. In environments like Lunar Lander, where success requires decisive control and strategic thrusting, minimizing reward variance conflicts with the primary objective, as it discourages the high-reward actions necessary for effective landings. These results highlight the importance of selecting an appropriate aggregation strategy based on task-specific objectives.

### H.4 Portfolio environment

In our experiment, we trained agents using five different random seeds over a rolling 5-year window, with a total of 10 training periods. Specifically, for each training period, training begins on January 1 of a given year and continues for five years, ending on December 31 of the fifth year. Each training period starts one year after the previous one, resulting in overlapping but not identical training datasets. Following the training phase, we evaluate the performance of agents in the subsequent year, immediately following the training period. Finally, we assess their generalization performance in the test phase, which takes place in the year after the evaluation period. This design allows us to systematically analyze the agents' performance across different temporal contexts while leveraging historical data in a structured and overlapping manner.

**Implementation** We conducted experiments using a modified version of the PPO (Schulman et al., 2017) implementation from Stable-Baselines3 (Raffin et al., 2021).

**Computational resource** Training a single agent takes approximately 1.5 hours on an NVIDIA RTX 2080 GPU, with the environment running in parallel on 10 CPU cores to accelerate data collection.