LEARNING LONG-TERM REWARD REDISTRIBUTION VIA RANDOMIZED RETURN DECOMPOSITION

Zhizhou Ren[1], **Ruihan Guo**[2], **Yuan Zhou**[3], **Jian Peng**[145]
[1]University of Illinois at Urbana-Champaign [2]Shanghai Jiao Tong University [3]BIMSA
[4]AIR, Tsinghua University [5]HeliXon Limited
zhizhour@illinois.edu, guoruihan@sjtu.edu.cn
timzhouyuan@gmail.com, jianpeng@illinois.edu

ABSTRACT

Many practical applications of reinforcement learning require agents to learn from sparse and delayed rewards. It challenges the ability of agents to attribute their actions to future outcomes. In this paper, we consider the problem formulation of episodic reinforcement learning with trajectory feedback. It refers to an extreme delay of reward signals, in which the agent can only obtain one reward signal at the end of each trajectory. A popular paradigm for this problem setting is learning with a designed auxiliary dense reward function, namely proxy reward, instead of sparse environmental signals. Based on this framework, this paper proposes a novel reward redistribution algorithm, randomized return decomposition (RRD), to learn a proxy reward function for episodic reinforcement learning. We establish a surrogate problem by Monte-Carlo sampling that scales up least-squares-based reward redistribution to long-horizon problems. We analyze our surrogate loss function by connection with existing methods in the literature, which illustrates the algorithmic properties of our approach. In experiments, we extensively evaluate our proposed method on a variety of benchmark tasks with episodic rewards and demonstrate substantial improvement over baseline algorithms.

1 INTRODUCTION

Scaling reinforcement learning (RL) algorithms to practical applications has become the focus of numerous recent studies, including resource management (Mao et al., 2016), industrial control (Hein et al., 2017), drug discovery (Popova et al., 2018), and recommendation systems (Chen et al., 2018). One of the challenges in these real-world problems is the sparse and delayed environmental rewards. For example, in the molecular structure design problem, the target molecule property can only be evaluated after completing the whole sequence of modification operations (Zhou et al., 2019b). The sparsity of environmental feedback would complicate the attribution of rewards on agent actions and therefore can hinder the efficiency of learning (Rahmandad et al., 2009). In practice, it is a common choice to formulate the RL objective with a meticulously designed reward function instead of the sparse environmental rewards. The design of such a reward function is crucial to the performance of the learned policies. Most standard RL algorithms, such as temporal difference learning and policy gradient methods, prefer dense reward functions that can provide instant feedback for every step of environment transitions. Designing such dense reward functions is not a simple problem even with domain knowledge and human supervision. It has been widely observed in prior works that handcrafted heuristic reward functions may lead to unexpected and undesired behaviors (Randløv & Alstrøm, 1998; Bottou et al., 2013; Andrychowicz et al., 2017). The agent may find a shortcut solution that formally optimizes the given objective but deviates from the desired policies (Dewey, 2014; Amodei et al., 2016). The reward designer can hardly anticipate all potential side effects of the designed reward function, which highlights the difficulty of reward engineering.

To avoid the unintended behaviors induced by misspecified reward engineering, a common paradigm is considering the reward design as an online problem within the trial-and-error loop of reinforcement learning (Sorg et al., 2010). This algorithmic framework contains two components, namely reward modeling and policy optimization. The agent first learns a proxy reward function from the experience data and then optimizes its policy based on the learned per-step rewards. By iterating this

procedure and interacting with the environment, the agent is able to continuously refine its reward model so that the learned proxy reward function can better approximate the actual objective given by the environmental feedback. More specifically, this paradigm aims to reshape the sparse and delayed environmental rewards to a dense Markovian reward function while trying to avoid misspecifying the goal of given tasks.

In this paper, we propose a novel reward redistribution algorithm based on a classical mechanism called *return decomposition* (Arjona-Medina et al., 2019). Our method is built upon the least-squares-based return decomposition (Efroni et al., 2021) whose basic idea is training a regression model that decomposes the trajectory return to the summation of per-step proxy rewards. This paradigm is a promising approach to redistributing sparse environmental feedback. Our proposed algorithm, *randomized return decomposition* (RRD), establish a surrogate optimization of return decomposition to improve the scalability in long-horizon tasks. In this surrogate problem, the reward model is trained to predict the episodic return from a random subsequence of the agent trajectory, i.e., we conduct a structural constraint that the learned proxy rewards can approximately reconstruct environmental trajectory return from a small subset of state-action pairs. This design enables us to conduct return decomposition effectively by mini-batch training. Our analysis shows that our surrogate loss function is an upper bound of the original loss of deterministic return decomposition, which gives a theoretical interpretation of this randomized implementation. We also present how the surrogate gap can be controlled and draw connections to another method called uniform reward redistribution. In experiments, we demonstrate substantial improvement of our proposed approach over baseline algorithms on a suite of MuJoCo benchmark tasks with episodic rewards.

2 Background

2.1 Episodic Reinforcement Learning with Trajectory Feedback

In standard reinforcement learning settings, the environment model is usually formulated by a *Markov decision process* (MDP; Bellman, 1957), defined as a tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, P, R, \mu \rangle$, where $\mathcal{S}$ and $\mathcal{A}$ denote the spaces of environment states and agent actions. $P(s'|s,a)$ and $R(s,a)$ denote the unknown environment transition and reward functions. μ denotes the initial state distribution. The goal of reinforcement learning is to find a policy $\pi : \mathcal{S} \to \mathcal{A}$ maximizing cumulative rewards. More specifically, a common objective is maximizing infinite-horizon discounted rewards based on a pre-defined discount factor γ as follows:

$$\text{(standard objective)} \qquad J(\pi) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, \pi(s_t)) \;\middle|\; s_0 \sim \mu,\ s_{t+1} \sim P(\cdot \mid s_t, \pi(s_t)) \right]. \quad (1)$$

In this paper, we consider the episodic reinforcement learning setting with trajectory feedback, in which the agent can only obtain one reward feedback at the end of each trajectory. Let τ denote an agent trajectory that contains all experienced states and behaved actions within an episode. We assume all trajectories terminate in finite steps. The episodic reward function $R_{\text{ep}}(\tau)$ is defined on the trajectory space, which represents the overall performance of trajectory τ. The goal of episodic reinforcement learning is to maximize the expected trajectory return:

$$\text{(episodic objective)} \qquad J_{\text{ep}}(\pi) = \mathbb{E}\Big[R_{\text{ep}}(\tau) \;\Big|\; s_0 \sim \mu,\ a_t = \pi(s_t),\ \tau = \langle s_0, a_0, s_1, \cdots, s_T \rangle \Big]. \quad (2)$$

In general, the episodic-reward setting is a particular form of *partially observable Markov decision processes* (POMDPs) where the reward function is non-Markovian. The worst case may require the agent to enumerate the entire exponential-size trajectory space for recovering the episodic reward function. In practical problems, the episodic environmental feedback usually has structured representations. A common structural assumption is the existence of an underlying Markovian reward function $\widehat{R}(s,a)$ that approximates the episodic reward $R_{\text{ep}}(\tau)$ by a sum-form decomposition,

$$\text{(sum-decomposable episodic reward)} \qquad R_{\text{ep}}(\tau) \approx \widehat{R}_{\text{ep}}(\tau) = \sum_{t=0}^{T-1} \widehat{R}(s_t, a_t). \quad (3)$$

This structure is commonly considered by both theoretical (Efroni et al., 2021) and empirical studies (Liu et al., 2019; Raposo et al., 2021) on long-horizon episodic rewards. It models the situations

where the agent objective is measured by some metric with additivity properties, e.g., the distance of robot running, the time cost of navigation, or the number of products produced in a time interval.

2.2 Reward Redistribution

The goal of *reward redistribution* is constructing a proxy reward function $\widehat{R}(s_t, a_t)$ that transforms the episodic-reward problem stated in Eq. (2) to a standard dense-reward setting. By replacing environmental rewards with such a Markovian proxy reward function $\widehat{R}(s_t, a_t)$, the agent can be trained to optimize the discounted objective in Eq. (1) using any standard RL algorithms. Formally, the proxy rewards $\widehat{R}(s_t, a_t)$ form a sum-decomposable reward function $\widehat{R}_{\text{ep}}(\tau) = \sum_{t=0}^{T-1} \widehat{R}(s_t, a_t)$ that is expected to have high correlation to the environmental reward $R_{\text{ep}}(\tau)$. Here, we introduce two branches of existing reward redistribution methods, *return decomposition* and *uniform reward redistribution*, which are the most related to our proposed approach. We defer the discussions of other related work to section 5.

Return Decomposition. The idea of *return decomposition* is training a reward model that predicts the trajectory return with a given state-action sequence (Arjona-Medina et al., 2019). In this paper, without further specification, we focus on the least-squares-based implementation of return decomposition (Efroni et al., 2021). The reward redistribution is given by the learned reward model, i.e., decomposing the environmental episodic reward $R_{\text{ep}}(\tau)$ to a Markovian proxy reward function $\widehat{R}(s, a)$. In practice, the reward modeling is formulated by optimizing the following loss function:

$$\mathcal{L}_{\text{RD}}(\theta) = \mathop{\mathbb{E}}_{\tau\sim\mathcal{D}} \left[\left(R_{\text{ep}}(\tau) - \sum_{t=0}^{T-1} \widehat{R}_\theta(s_t, a_t) \right)^2 \right], \tag{4}$$

where $\widehat{R}_\theta$ denotes the parameterized proxy reward function, θ denotes the parameters of the learned reward model, and $\mathcal{D}$ denotes the experience dataset collected by the agent. Assuming the sum-decomposable structure stated in Eq. (3), $\widehat{R}_\theta(s, a)$ is expected to asymptotically concentrate near the ground-truth underlying rewards $\widehat{R}(s, a)$ when Eq. (4) is properly optimized (Efroni et al., 2021).

One limitation of the least-squares-based return decomposition method specified by Eq. (4) is its scalability in terms of the computation costs. Note that the trajectory-wise episodic reward is the only environmental supervision for reward modeling. Computing the loss function $\mathcal{L}_{\text{RD}}(\theta)$ with a single episodic reward label requires to enumerate all state-action pairs along the whole trajectory. This computation procedure can be expensive in numerous situations, e.g., when the task horizon T is quite long, or the state space $\mathcal{S}$ is high-dimensional. To address this practical barrier, recent works focus on designing reward redistribution mechanisms that can be easily integrated in complex tasks. We will discuss the implementation subtlety of existing methods in section 4.

Uniform Reward Redistribution. To pursue a simple but effective reward redistribution mechanism, IRCR (Gangwani et al., 2020) considers *uniform reward redistribution* which assumes all state-action pairs equally contribute to the return value. It is designed to redistribute rewards in the absence of any prior structure or information. More specifically, the proxy reward $\widehat{R}_{\text{IRCR}}(s, a)$ is computed by averaging episodic return values over all experienced trajectories containing (s, a),

$$\widehat{R}_{\text{IRCR}}(s, a) = \mathop{\mathbb{E}}_{\tau\sim\mathcal{D}} \big[R_{\text{ep}}(\tau) \mid (s, a) \in \tau \big]. \tag{5}$$

In this paper, we will introduce a novel reward redistribution mechanism that bridges between return decomposition and uniform reward redistribution.

3 Reward Redistribution via Randomized Return Decomposition

In this section, we introduce our approach, randomized return decomposition (RRD), which sets up a surrogate optimization problem of the least-squares-based return decomposition. The proposed surrogate objective allows us to conduct return decomposition on short subsequences of agent trajectories, which is scalable in long-horizon tasks. We provide analyses to characterize the algorithmic property of our surrogate objective function and discuss connections to existing methods.

3.1 Randomized Return Decomposition with Monte-Carlo Return Estimation

One practical barrier to apply least-squares-based return decomposition methods in long-horizon tasks is the computation costs of the regression loss in Eq. (4), i.e., it requires to enumerate all state-action pairs within the agent trajectory. To resolve this issue, we consider a randomized method that uses a Monte-Carlo estimator to compute the predicted episodic return $\widehat{R}_{\text{ep},\theta}(\tau)$ as follows:

$$\underbrace{\widehat{R}_{\text{ep},\theta}(\tau) = \sum_{t=0}^{T-1} \widehat{R}_\theta(s_t, a_t)}_{\text{Deterministic Computation}} = \mathop{\mathbb{E}}_{\mathcal{I}\sim\rho_T(\cdot)} \left[\frac{T}{|\mathcal{I}|} \sum_{t\in\mathcal{I}} \widehat{R}_\theta(s_t, a_t) \right] \approx \underbrace{\frac{T}{|\mathcal{I}|} \sum_{t\in\mathcal{I}} \widehat{R}_\theta(s_t, a_t)}_{\text{Monte-Carlo Estimation}} , \tag{6}$$

where $\mathcal{I}$ denotes a subset of indices. $\rho_T(\cdot)$ denotes an unbiased sampling distribution where each index t has the same probability to be included in $\mathcal{I}$. In this paper, without further specification, $\rho_T(\cdot)$ is constructed by uniformly sampling K distinct indices.

$$\rho_T(\cdot) = \text{Uniform}\left(\left\{\mathcal{I} \subseteq \mathbb{Z}_T : |\mathcal{I}| = K\right\}\right), \tag{7}$$

where K is a hyper-parameter. In this sampling distribution, each timestep t has the same probability to be covered by the sampled subsequence $\mathcal{I} \sim \rho_T(\cdot)$ so that it gives an unbiased Monte-Carlo estimation of the episodic summation $\widehat{R}_{\text{ep},\theta}(\tau)$.

Randomized Return Decomposition. Based on the idea of using Monte-Carlo estimation shown in Eq. (6), we introduce our approach, *randomized return decomposition* (RRD), to improve the scalability of least-squares-based reward redistribution methods. The objective function of our approach is formulated by the randomized return decomposition loss $\mathcal{L}_{\text{Rand-RD}}(\theta)$ stated in Eq. (8), in which the parameterized proxy reward function $\widehat{R}_\theta$ is trained to predict the episodic return $R_{\text{ep}}(\tau)$ given a random subsequence of the agent trajectory. In other words, we integrate the Monte-Carlo estimator (see Eq. (6)) into the return decomposition loss to obtain the following surrogate loss function:

$$\mathcal{L}_{\text{Rand-RD}}(\theta) = \mathop{\mathbb{E}}_{\tau\sim\mathcal{D}} \left[\mathop{\mathbb{E}}_{\mathcal{I}\sim\rho_T(\cdot)} \left[\left(R_{\text{ep}}(\tau) - \frac{T}{|\mathcal{I}|} \sum_{t\in\mathcal{I}} \widehat{R}_\theta(s_t, a_t) \right)^2 \right] \right]. \tag{8}$$

In practice, the loss function $\mathcal{L}_{\text{Rand-RD}}(\theta)$ can be estimated by sampling a mini-batch of trajectory subsequences instead of computing $\widehat{R}_\theta(s_t, a_t)$ for the whole agent trajectory, and thus the implementation of randomized return decomposition is adaptive and flexible in long-horizon tasks.

3.2 Analysis of Randomized Return Decomposition

The main purpose of our approach is establishing a surrogate loss function to improve the scalability of least-squares-based return decomposition in practice. Our proposed method, randomized return decomposition, is a trade-off between the computation complexity and the estimation error induced by the Monte-Carlo estimator. In this section, we show that our approach is an interpolation between between the return decomposition paradigm and uniform reward redistribution, which can be controlled by the hyper-parameter K used in the sampling distribution (see Eq. (7)). We present Theorem 1 as a formal characterization of our proposed surrogate objective function.

Theorem 1 (Loss Decomposition)**.** *The surrogate loss function $\mathcal{L}_{\textit{Rand-RD}}(\theta)$ can be decomposed to two terms interpolating between return decomposition and uniform reward redistribution.*

$$\mathcal{L}_{\textit{Rand-RD}}(\theta) = \mathcal{L}_{\textit{RD}}(\theta) + \mathop{\mathbb{E}}_{\tau\sim\mathcal{D}} \left[\underbrace{\mathop{\textit{Var}}_{\mathcal{I}\sim\rho_T(\cdot)} \left[\frac{T}{|\mathcal{I}|} \sum_{t\in\mathcal{I}} \widehat{R}_\theta(s_t, a_t) \right]}_{\textit{variance of the Monte-Carlo estimator}} \right] \tag{9}$$

$$= \underbrace{\mathcal{L}_{\textit{RD}}(\theta)}_{\textit{return decomposition}} + \mathop{\mathbb{E}}_{\tau\sim\mathcal{D}} \left[T^2 \cdot \underbrace{\mathop{\textit{Var}}_{(s_t,a_t)\sim\tau} \left[\widehat{R}_\theta(s_t, a_t) \right]}_{\textit{uniform reward redistribution}} \cdot \underbrace{\frac{1}{K} \left(1 - \frac{K-1}{T-1} \right)}_{\textit{interpolation weight}} \right], \tag{10}$$

where K denotes the length of sampled subsequences defined in Eq. (7)*.*

The proof of Theorem 1 is based on the bias-variance decomposition formula of mean squared error (Kohavi & Wolpert, 1996). The detailed proofs are deferred to Appendix A.

Interpretation as Regularization. As presented in Theorem 1, the randomized decomposition loss can be decomposed to two terms, the deterministic return decomposition loss $\mathcal{L}_{\text{RD}}(\theta)$ and a variance penalty term (see the second term of Eq. (9)). The variance penalty term can be regarded as a regularization that is controlled by hyper-parameter K. In practical problems, the objective of return decomposition is ill-posed, since the number of trajectory labels is dramatically less than the number of transition samples. There may exist solutions of reward redistribution that formally optimize the least-squares regression loss but serve little functionality to guide the policy learning. Regarding this issue, the variance penalty in randomized return decomposition is a regularizer for reward modeling. It searches for smooth proxy rewards that has low variance within the trajectory. This regularization effect is similar to the mechanism of uniform reward redistribution (Gangwani et al., 2020), which achieves state-of-the-art performance in the previous literature. In section 4, our experiments demonstrate that the variance penalty is crucial to the empirical performance of randomized return decomposition.

A Closer Look at Loss Decomposition. In addition to the intuitive interpretation of regularization, we will present a detailed characterization of the loss decomposition shown in Theorem 1. We interpret this loss decomposition as below:

1. Note that the Monte-Carlo estimator used by randomized return decomposition is an unbiased estimation of the proxy episodic return $\widehat{R}_{\text{ep},\theta}(\tau)$ (see Eq. (6)). This unbiased property gives the first component of the loss decomposition, i.e, the original return decomposition loss $\mathcal{L}_{\text{RD}}(\theta)$.
2. Although the Monte-Carlo estimator is unbiased, its variance would contribute to an additional loss term induced by the mean-square operator, i.e., the second component of loss decomposition presented in Eq. (10). This additional term penalizes the variance of the learned proxy rewards under random sampling. This penalty expresses the same mechanism as uniform reward redistribution (Gangwani et al., 2020) in which the episodic return is uniformly redistributed to the state-action pairs in the trajectory.

Based on the above discussions, we can analyze the algorithmic properties of randomized return decomposition by connecting with previous studies.

3.2.1 Surrogate Optimization of Return Decomposition

Randomized return decomposition conducts a surrogate optimization of the actual return decomposition. Note that the variance penalty term in Eq. (10) is non-negative, our loss function $\mathcal{L}_{\text{Rand-RD}}(\theta)$ serves an upper bound estimation of the original loss $\mathcal{L}_{\text{RD}}(\theta)$ as the following statement.

Proposition 1 (Surrogate Upper Bound)**.** *The randomized return decomposition loss $\mathcal{L}_{\textit{Rand-RD}}(\theta)$ is an upper bound of the actual return decomposition loss function $\mathcal{L}_{\textit{RD}}(\theta)$, i.e., $\mathcal{L}_{\textit{Rand-RD}}(\theta) \geq \mathcal{L}_{\textit{RD}}(\theta)$.*

Proposition 1 suggests that optimizing our surrogate loss $\mathcal{L}_{\text{Rand-RD}}(\theta)$ guarantees to optimize an upper bound of the actual return decomposition loss $\mathcal{L}_{\text{RD}}(\theta)$. According to Theorem 1, the gap between $\mathcal{L}_{\text{Rand-RD}}(\theta)$ and $\mathcal{L}_{\text{RD}}(\theta)$ refers to the variance of subsequence sampling. The magnitude of this gap can be controlled by the hyper-parameter K that refers to the length of sampled subsequences.

Proposition 2 (Objective Gap)**.** *Let $\mathcal{L}^{(K)}_{\textit{Rand-RD}}(\theta)$ denote the randomized return decomposition loss that samples subsequences with length K. The gap between $\mathcal{L}^{(K)}_{\textit{Rand-RD}}(\theta)$ and $\mathcal{L}_{\textit{RD}}(\theta)$ can be reduced by using larger values of hyper-parameter K.*

$$\forall\theta, \quad \mathcal{L}^{(1)}_{\textit{Rand-RD}}(\theta) \geq \mathcal{L}^{(2)}_{\textit{Rand-RD}}(\theta) \geq \cdots \geq \mathcal{L}^{(T-1)}_{\textit{Rand-RD}}(\theta) \geq \mathcal{L}^{(T)}_{\textit{Rand-RD}}(\theta) = \mathcal{L}_{\textit{RD}}(\theta). \tag{11}$$

This gap can be eliminated by choosing $K = T$ in the sampling distribution (see Eq. (7)) so that our approach degrades to the original deterministic implementation of return decomposition.

3.2.2 Generalization of Uniform Reward Redistribution

The reward redistribution mechanism of randomized return decomposition is a generalization of uniform reward redistribution. To serve intuitions, we start the discussions with the simplest case using $K = 1$ in subsequence sampling, in which our approach degrades to the uniform reward redistribution as the following statement.

Proposition 3 (Uniform Reward Redistribution)**.** *Assume all trajectories have the same length and the parameterization space of θ serves universal representation capacity. The optimal solution $\theta^\star$ of minimizing $\mathcal{L}^{(1)}_{\textit{Rand-RD}}(\theta)$ is stated as follows:*

$$\widehat{R}_{\theta^\star}(s,a) = \mathop{\mathbb{E}}_{\tau\sim\mathcal{D}}\big[R_{ep}(\tau)/T \mid (s,a)\in\tau\big], \tag{12}$$

where $\mathcal{L}^{(1)}_{\textit{Rand-RD}}(\theta)$ denotes the randomized return decomposition loss with $K=1$.

A minor difference between Eq. (12) and the proxy reward designed by Gangwani et al. (2020) (see Eq. (5)) is a multiplier scalar $1/T$. Gangwani et al. (2020) interprets such a proxy reward mechanism as a trajectory-space smoothing process or a non-committal reward redistribution. Our analysis can give a mathematical characterization to illustrate the objective of uniform reward redistribution. As characterized by Theorem 1, uniform reward redistribution conducts an additional regularizer to penalize the variance of per-step proxy rewards. In the view of randomized return decomposition, the functionality of this regularizer is requiring the reward model to reconstruct episodic return from each single-step transition.

By using larger values of hyper-parameter K, randomized return decomposition is trained to reconstruct the episodic return from a subsequence of agent trajectory instead of the single-step transition used by uniform reward redistribution. This mechanism is a generalization of uniform reward redistribution, in which we equally assign rewards to subsequences generated by uniformly random sampling. It relies on the concentratability of random sampling, i.e., the average of a sequence can be estimated by a small random subset with sub-linear size. The individual contribution of each transition within the subsequence is further attributed by return decomposition.

3.3 Practical Implementation of Randomized Return Decomposition

In Algorithm 1, we integrate randomized return decomposition with policy optimization. It follows an iterative paradigm that iterates between the rewarding modeling and policy optimization modules.

Algorithm 1 Policy Optimization with Randomized Return Decomposition

Initialize $\mathcal{D}\leftarrow\emptyset$
for $\ell = 1,2,\cdots$ **do**
  Collect a rollout trajectory τ using the current policy.
  Store trajectory τ and feedback $R_{\text{ep}}(\tau)$ into the replay buffer $\mathcal{D}\leftarrow\mathcal{D}\cup\{(\tau,R_{\text{ep}}(\tau))\}$.
  for $i = 1,2,\cdots$ **do**
    Sample M trajectories $\{\tau_j\in\mathcal{D}\}_{j=1}^M$ from the replay buffer.
    Sample subsequences $\{\mathcal{I}_j\subseteq\mathbb{Z}_{T_j}\}_{j=1}^M$ for these trajectories.
    Estimate randomized return decomposition loss $\widehat{\mathcal{L}}_{\text{Rand-RD}}(\theta)$,

$$\widehat{\mathcal{L}}_{\text{Rand-RD}}(\theta) = \frac{1}{M}\sum_{j=1}^{M}\left[\left(R_{\text{ep}}(\tau_j) - \frac{T_j}{|\mathcal{I}_j|}\sum_{t\in\mathcal{I}_j}\widehat{R}_\theta(s_{j,t},a_{j,t})\right)^2\right], \tag{13}$$

    where T_j denotes the length of trajectory $\tau_j = \langle s_{j,1},a_{j,1},\cdots,s_{j,T_j}\rangle$.
    Perform a gradient update on the reward model $\widehat{R}_\theta$,

$$\theta\leftarrow\theta-\alpha\nabla_\theta\widehat{\mathcal{L}}_{\text{Rand-RD}}(\theta), \tag{14}$$

    where α denotes the learning rate.
  Perform policy optimization using the learned proxy reward function $\widehat{R}_\theta(s,a)$.

As presented in Eq. (13) and Eq. (14), the optimization of our loss function $\mathcal{L}_{\text{Rand-RD}}(\theta)$ can be easily conducted by mini-batch gradient descent. This surrogate loss function only requires computations on short-length subsequences. It provides a scalable implementation for return decomposition that can be generalized to long-horizon tasks with manageable computation costs. In section 4, we will show that this simple implementation can also achieve state-of-the-art performance in comparison to other existing methods.

4 EXPERIMENTS

In this section, we investigate the empirical performance of our proposed methods by conducting experiments on a suite of MuJoCo benchmark tasks with episodic rewards. We compare our approach with several baseline algorithms in the literature and conduct an ablation study on subsequence sampling that is the core component of our algorithm.

4.1 PERFORMANCE EVALUATION ON MUJOCO BENCHMARK WITH EPISODIC REWARDS

Experiment Setting. We adopt the same experiment setting as Gangwani et al. (2020) to compare the performance of our approach with baseline algorithms. The experiment environments is based on the MuJoCo locomotion benchmark tasks created by OpenAI Gym (Brockman et al., 2016). These tasks are long-horizon with maximum trajectory length $T = 1000$, i.e., the task horizon is definitely longer than the batch size used by the standard implementation of mini-batch gradient estimation. We modify the reward function of these environments to set up an episodic-reward setting. Formally, on non-terminal states, the agent will receive a zero signal instead of the per-step dense rewards. The agent can obtain the episodic feedback $R_{\text{ep}}(\tau)$ at the last step of the rollout trajectory, in which $R_{\text{ep}}(\tau)$ is computed by the summation of per-step instant rewards given by the standard setting. We evaluate the performance of our proposed methods with the same configuration of hyper-parameters in all environments. A detailed description of implementation details is included in Appendix B.

We evaluate two implementations of randomized return decomposition (RRD):

- **RRD (ours)** denotes the default implementation of our approach. We train a reward model using randomized return decomposition loss $\mathcal{L}_{\text{Rand-RD}}$, in which we sample subsequences with length $K = 64$ in comparison to the task horizon $T = 1000$. The reward model $\widehat{R}_\theta$ is parameterized by a two-layer fully connected network. The policy optimization module is implemented by soft actor-critic (SAC; Haarnoja et al., 2018a). [1]
- **RRD-$\mathcal{L}_{\text{RD}}$ (ours)** is an alternative implementation that optimizes $\mathcal{L}_{\text{RD}}$ instead of $\mathcal{L}_{\text{Rand-RD}}$. Note that Theorem 1 gives a closed-form characterization of the gap between $\mathcal{L}_{\text{Rand-RD}}(\theta)$ and $\mathcal{L}_{\text{RD}}(\theta)$, which is represented by the variance of the learned proxy rewards. By subtracting an unbiased variance estimation from loss function $\mathcal{L}_{\text{Rand-RD}}(\theta)$, we can estimate loss function $\mathcal{L}_{\text{RD}}(\theta)$ by sampling short subsequences. It gives a computationally efficient way to optimize $\mathcal{L}_{\text{RD}}(\theta)$. We include this alternative implementation to reveal the functionality of the regularization given by variance penalty. A detailed description is deferred to Appendix B.3.

We compare with several existing methods for episodic or delayed reward settings:

- **IRCR** (Gangwani et al., 2020) is an implementation of uniform reward redistribution. The reward redistribution mechanism of IRCR is non-parametric, in which the proxy reward of a transition is set to be the normalized value of corresponding trajectory return. It is equivalent to use a Monte-Carlo estimator of Eq. (5). Due to the ease of implementation, this method achieves state-of-the-art performance in the literature.
- **RUDDER** (Arjona-Medina et al., 2019) is based on the idea of return decomposition but does not directly optimize $\mathcal{L}_{\text{RD}}(\theta)$. Instead, it trains a return predictor based on trajectory, and the step-wise credit is assigned by the prediction difference between two consecutive states. By using the warm-up technique of LSTM, this transform prevents its training computation costs from depending on the task horizon T so that it is adaptive to long-horizon tasks.
- **GASIL** (Guo et al., 2018), generative adversarial self-imitation learning, is a generalization of GAIL (Ho & Ermon, 2016). It formulates an imitation learning framework by imitating best trajectories in the replay buffer. The proxy rewards are given by a discriminator that is trained to classify the agent and expert trajectories.
- **LIRPG** (Zheng et al., 2018) aims to learn an intrinsic reward function from sparse environment feedback. Its policy is trained to optimized the sum of the extrinsic and intrinsic rewards. The parametric intrinsic reward function is updated by meta-gradients to optimize the actual extrinsic rewards achieved by the policy

[1] The source code of our implementation is available at `https://github.com/Stilwell-Git/Randomized-Return-Decomposition`.

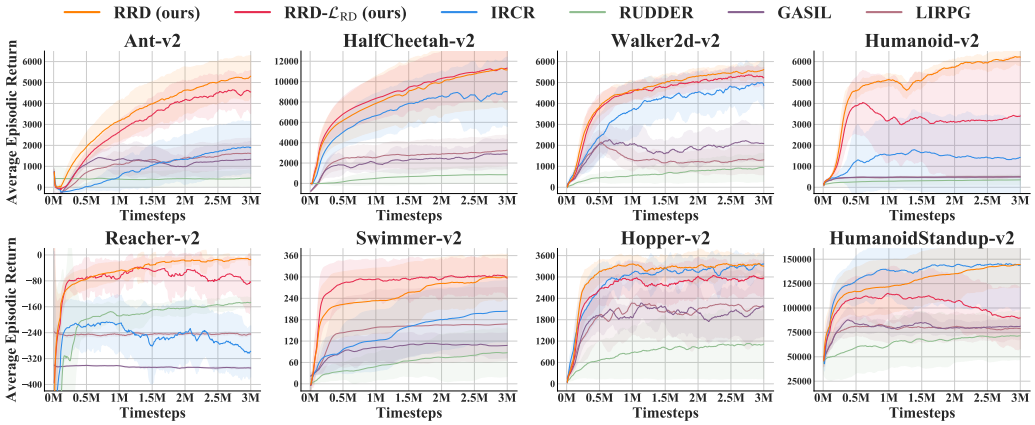

Figure 1: Learning curves on a suite of MuJoCo benchmark tasks with episodic rewards. All curves for MuJoCo benchmark are plotted from 30 runs with random initializations. The shaded region indicates the standard deviation. To make the comparison more clear, the curves are smoothed by averaging 10 most recent evaluation points. We set up an evaluation point every 10^4 timesteps.

Overall Performance Comparison. As presented in Figure 1, randomized return decomposition generally outperforms baseline algorithms. Our approach can achieve higher sample efficiency and produce better policies after convergence. RUDDER is an implementation of return decomposition that represents single-step rewards by the differences between the return predictions of two consecutive states. This implementation maintains high computation efficiency but long-term return prediction is a hard optimization problem and requires on-policy samples. In comparison, RRD is a more scalable and stable implementation which can better integrate with off-policy learning for improving sample efficiency. The uniform reward redistribution considered by IRCR is simple to implement but cannot extract the temporal structure of episodic rewards. Thus the final policy quality produced by RRD is usually better than that of IRCR. GASIL and LIRPG aim to construct auxiliary reward functions that have high correlation to the environmental return. These two methods cannot achieve high sample efficiency since their objectives require on-policy training.

Variance Penalty as Regularization. Figure 1 also compares two implementations of randomized return decomposition. In most testing environments, RRD optimizing $\mathcal{L}_{\text{Rand-RD}}$ outperforms the unbiased implementation RRD-$\mathcal{L}_{\text{RD}}$. We consider RRD using $\mathcal{L}_{\text{Rand-RD}}$ as our default implementation since it performs better and its objective function is simpler to implement. As discussed in section 3.2, the variance penalty conducted by RRD aims to minimize the variance of the Monte-Carlo estimator presented in Eq. (6). It serves as a regularization to restrict the solution space of return decomposition, which gives two potential effects: (1) RRD prefers smooth proxy rewards when the expressiveness capacity of reward network over-parameterizes the dataset. (2) The variance of mini-batch gradient estimation can also be reduced when the variance of Monte-Carlo estimator is small. In practice, this regularization would benefit the training stability. As presented in Figure 1, RRD achieves higher sample efficiency than RRD-$\mathcal{L}_{\text{RD}}$ in most testing environments. The quality of the learned policy of RRD is also better than that of RRD-$\mathcal{L}_{\text{RD}}$. It suggests that the regularized reward redistribution can better approximate the actual environmental objective.

4.2 Ablation Studies

We conduct an ablation study on the hyper-parameter K that represent the length of subsequences used by randomized return decomposition. As discussed in section 3.2, the hyper-parameter K controls the interpolation ratio between return decomposition and uniform reward redistribution. It trades off the accuracy of return reconstruction and variance regularization. In Figure 2, we evaluate RRD with a set of choices of hyper-parameter $K \in \{1, 8, 16, 32, 64, 128\}$. The experiment results show that, although the sensitivity of this hyper-parameter depends on the environment, a relatively large value of K generally achieves better performance, since it can better approximate the environmental objective. In this experiment, we ensure all runs use the same input size for mini-batch training, i.e., using larger value of K leads to less number of subsequences in the mini-batch.

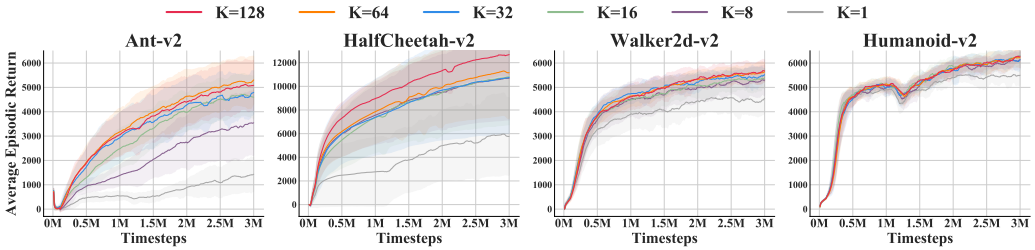

Figure 2: Learning curves of RRD with different choices of hyper-parameter K. The curves with $K = 64$ correspond to the default implementation of RRD presented in Figure 1.

More specifically, in this ablation study, all algorithm instances estimate the loss function $\mathcal{L}_{\text{Rand-RD}}$ using a mini-batch containing 256 transitions. We consider $K = 64$ as our default configuration. As presented in Figure 2, larger values give marginal improvement in most environments. The benefits of larger values of K can only be observed in HalfCheetah-v2. We note that HalfCheetah-v2 does not have early termination and thus has the longest average horizon among these locomotion tasks. It highlights the trade-off between the weight of regularization and the bias of subsequence estimation.

5 Related Work

Reward Design. The ability of RL agents highly depends on the designs of reward functions. It is widely observed that reward shaping can accelerate learning (Mataric, 1994; Ng et al., 1999; Devlin et al., 2011; Wu & Tian, 2017; Song et al., 2019). Many previous works study how to automatically design auxiliary reward functions for efficient reinforcement learning. A famous paradigm, inverse RL (Ng & Russell, 2000; Fu et al., 2018), considers to recover a reward function from expert demonstrations. Another branch of work is learning an intrinsic reward function that guides the agent to maximize extrinsic objective (Sorg et al., 2010; Guo et al., 2016). Such an intrinsic reward function can be learned through meta-gradients (Zheng et al., 2018; 2020) or self-imitation (Guo et al., 2018; Gangwani et al., 2019). A recent work (Abel et al., 2021) studies the expressivity of Markov rewards and proposes algorithms to design Markov rewards for three notions of abstract tasks.

Temporal Credit Assignment. Another methodology for tackling long-horizon sequential decision problems is assigning credits to emphasize the contribution of each single step over the temporal structure. These methods directly consider the specification of the step values instead of manipulating the reward function. The simplest example is studying how the choice of discount factor γ affects the policy learning (Petrik & Scherrer, 2008; Jiang et al., 2015; Fedus et al., 2019). Several previous work consider to extend the λ-return mechanism (Sutton, 1988) to a more generalized credit assignment framework, such as adaptive λ (Xu et al., 2018) and pairwise weights (Zheng et al., 2021). RUDDER (Arjona-Medina et al., 2019) proposes a return-equivalent formulation for the credit assignment problem and establish theoretical analyses (Holzleitner et al., 2021). Aligned-RUDDER (Patil et al., 2020) considers to use expert demonstrations for higher sample efficiency. Harutyunyan et al. (2019) opens up a new family of algorithms, called hindsight credit assignment, that attributes the credits from a backward view. In Appendix F, we cover more topics of related work and discuss the connections to the problem focused by this paper.

6 Conclusion

In this paper, we propose randomized return decomposition (RRD), a novel reward redistribution algorithm, to tackle the episodic reinforcement learning problem with trajectory feedback. RRD uses a Monte-Carlo estimator to establish a surrogate optimization problem of return decomposition. This surrogate objective implicitly conducts a variance reduction penalty as regularization. We analyze its algorithmic properties by connecting with previous studies in reward redistribution. Our experiments demonstrate that RRD outperforms previous methods in terms of both sample efficiency and policy quality. The basic idea of randomized return decomposition can potentially generalize to other related problems with sum-decomposition structure, such as preference-based reward modeling (Lee et al., 2021) and multi-agent value decomposition (Sunehag et al., 2018). It is also promising to consider non-linear decomposition as what is explored in multi-agent value factorization (Rashid et al., 2018). We leave these investigations as our future work.

ACKNOWLEDGMENTS

The authors would like to thank Kefan Dong for insightful discussions. This work is supported by the National Science Foundation under Grant CCF-2006526.

REFERENCES

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

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

Saidhiraj Amuru and R Michael Buehrer. Optimal jamming using delayed learning. In *2014 IEEE Military Communications Conference*, pp. 1528–1533. IEEE, 2014.

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In *Advances in Neural Information Processing Systems*, pp. 5048–5058, 2017.

András Antos, Csaba Szepesvári, and Rémi Munos. Learning near-optimal policies with Bellman-residual minimization based fitted policy iteration and a single sample path. *Machine Learning*, 71(1):89–129, 2008.

Jose A Arjona-Medina, Michael Gillhofer, Michael Widrich, Thomas Unterthiner, Johannes Brandstetter, and Sepp Hochreiter. RUDDER: Return decomposition for delayed rewards. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Richard Bellman. Dynamic programming. *Princeton University Press*, 89:92, 1957.

Wendelin Böhmer, Vitaly Kurin, and Shimon Whiteson. Deep coordination graphs. In *International Conference on Machine Learning*, pp. 980–991. PMLR, 2020.

Léon Bottou, Jonas Peters, Joaquin Quiñonero-Candela, Denis X Charles, D Max Chickering, Elon Portugaly, Dipankar Ray, Patrice Simard, and Ed Snelson. Counterfactual reasoning and learning systems: The example of computational advertising. *Journal of Machine Learning Research*, 14 (11), 2013.

Yann Bouteiller, Simon Ramstedt, Giovanni Beltrame, Christopher Pal, and Jonathan Binas. Reinforcement learning with random delays. In *International Conference on Learning Representations*, 2021.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. OpenAI gym. *arXiv preprint arXiv:1606.01540*, 2016.

Pablo Samuel Castro, Subhodeep Moitra, Carles Gelada, Saurabh Kumar, and Marc G Bellemare. Dopamine: A research framework for deep reinforcement learning. *arXiv preprint arXiv:1812.06110*, 2018.

Niladri Chatterji, Aldo Pacchiano, Peter Bartlett, and Michael Jordan. On the theory of reinforcement learning with once-per-episode feedback. *Advances in Neural Information Processing Systems*, 34, 2021.

Shi-Yong Chen, Yang Yu, Qing Da, Jun Tan, Hai-Kuan Huang, and Hai-Hong Tang. Stabilizing reinforcement learning in dynamic environment with application to online recommendation. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1187–1196, 2018.

Edward H Clarke. Multipart pricing of public goods. *Public choice*, pp. 17–33, 1971.

Sam Devlin, Daniel Kudenko, and Marek Grześ. An empirical study of potential-based reward shaping and advice in complex, multi-agent systems. *Advances in Complex Systems*, 14(02): 251–278, 2011.

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

Yali Du, Lei Han, Meng Fang, Tianhong Dai, Ji Liu, and Dacheng Tao. LIIR: learning individual intrinsic reward in multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 4403–4414, 2019.

Yonathan Efroni, Nadav Merlis, and Shie Mannor. Reinforcement learning with trajectory feedback. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 7288–7295, 2021.

William Fedus, Carles Gelada, Yoshua Bengio, Marc G Bellemare, and Hugo Larochelle. Hyperbolic discounting and learning over multiple horizons. *arXiv preprint arXiv:1902.06865*, 2019.

Justin Fu, Katie Luo, and Sergey Levine. Learning robust rewards with adverserial inverse reinforcement learning. In *International Conference on Learning Representations*, 2018.

Tanmay Gangwani, Qiang Liu, and Jian Peng. Learning self-imitating diverse policies. In *International Conference on Learning Representations*, 2019.

Tanmay Gangwani, Yuan Zhou, and Jian Peng. Learning guidance rewards with trajectory-space smoothing. In *Advances in Neural Information Processing Systems*, volume 33, pp. 822–832, 2020.

Theodore Groves. Incentives in teams. *Econometrica: Journal of the Econometric Society*, pp. 617–631, 1973.

Xiaoxiao Guo, Satinder P Singh, Richard L Lewis, and Honglak Lee. Deep learning for reward design to improve Monte Carlo tree search in ATARI games. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, pp. 1519–1525, 2016.

Yijie Guo, Junhyuk Oh, Satinder Singh, and Honglak Lee. Generative adversarial self-imitation learning. *arXiv preprint arXiv:1812.00950*, 2018.

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*, pp. 1861–1870. PMLR, 2018a.

Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018b.

Beining Han, Zhizhou Ren, Zuofan Wu, Yuan Zhou, and Jian Peng. Off-policy reinforcement learning with delayed rewards. *arXiv preprint arXiv:2106.11854*, 2021.

Anna Harutyunyan, Will Dabney, Thomas Mesnard, Mohammad Gheshlaghi Azar, Bilal Piot, Nicolas Heess, Hado P van Hasselt, Gregory Wayne, Satinder Singh, Doina Precup, et al. Hindsight credit assignment. In *Advances in neural information processing systems*, volume 32, pp. 12488–12497, 2019.

Daniel Hein, Stefan Depeweg, Michel Tokic, Steffen Udluft, Alexander Hentschel, Thomas A Runkler, and Volkmar Sterzing. A benchmark environment motivated by industrial control problems. In *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 1–8. IEEE, 2017.

Amélie Héliou, Panayotis Mertikopoulos, and Zhengyuan Zhou. Gradient-free online learning in continuous games with delayed rewards. In *International Conference on Machine Learning*, pp. 4172–4181. PMLR, 2020.

Todd Hester and Peter Stone. TEXPLORE: real-time sample-efficient reinforcement learning for robots. *Machine learning*, 90(3):385–429, 2013.

Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In *Advances in neural information processing systems*, volume 29, pp. 4565–4573, 2016.

Markus Holzleitner, Lukas Gruber, José Arjona-Medina, Johannes Brandstetter, and Sepp Hochreiter. Convergence proof for actor-critic methods applied to PPO and RUDDER. In *Transactions on Large-Scale Data-and Knowledge-Centered Systems XLVIII*, pp. 105–130. Springer, 2021.

Nan Jiang, Alex Kulesza, Satinder Singh, and Richard Lewis. The dependence of effective planning horizon on model accuracy. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, pp. 1181–1189. Citeseer, 2015.

Konstantinos V Katsikopoulos and Sascha E Engelbrecht. Markov decision processes with delays and asynchronous cost collection. *IEEE transactions on automatic control*, 48(4):568–574, 2003.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

Ron Kohavi and David Wolpert. Bias plus variance decomposition for zero-one loss functions. In *Proceedings of the Thirteenth International Conference on International Conference on Machine Learning*, pp. 275–283, 1996.

Kimin Lee, Laura Smith, and Pieter Abbeel. PEBBLE: Feedback-efficient interactive reinforcement learning via relabeling experience and unsupervised pre-training. In *International Conference on Machine Learning*, 2021.

Lei Lei, Yue Tan, Kan Zheng, Shiwen Liu, Kuan Zhang, and Xuemin Shen. Deep reinforcement learning for autonomous internet of things: Model, applications and challenges. *IEEE Communications Surveys & Tutorials*, 22(3):1722–1760, 2020.

Yang Liu, Yunan Luo, Yuanyi Zhong, Xi Chen, Qiang Liu, and Jian Peng. Sequence modeling of temporal credit assignment for episodic reinforcement learning. *arXiv preprint arXiv:1905.13420*, 2019.

Horia Mania, Aurelia Guy, and Benjamin Recht. Simple random search provides a competitive approach to reinforcement learning. *arXiv preprint arXiv:1803.07055*, 2018.

Hongzi Mao, Mohammad Alizadeh, Ishai Menache, and Srikanth Kandula. Resource management with deep reinforcement learning. In *Proceedings of the 15th ACM workshop on hot topics in networks*, pp. 50–56, 2016.

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

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.

Roger B Myerson. Optimal auction design. *Mathematics of operations research*, 6(1):58–73, 1981.

Somjit Nath, Mayank Baranwal, and Harshad Khadilkar. Revisiting state augmentation methods for reinforcement learning with stochastic delays. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pp. 1346–1355, 2021.

Andrew Y Ng and Stuart Russell. Algorithms for inverse reinforcement learning. In *Proceedings of the Seventeenth International Conference on Machine Learning*. Citeseer, 2000.

Andrew Y Ng, Daishi Harada, and Stuart J Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of the Sixteenth International Conference on Machine Learning*, pp. 278–287. Morgan Kaufmann Publishers Inc., 1999.

Duc Thien Nguyen, Akshat Kumar, and Hoong Chuin Lau. Credit assignment for collective multiagent RL with global rewards. In *Advances in Neural Information Processing Systems*, volume 31, pp. 8102–8113, 2018.

Johan Nilsson, Bo Bernhardsson, and Björn Wittenmark. Stochastic analysis and control of real-time systems with random time delays. *Automatica*, 34(1):57–64, 1998.

Vihang P Patil, Markus Hofmarcher, Marius-Constantin Dinu, Matthias Dorfer, Patrick M Blies, Johannes Brandstetter, Jose A Arjona-Medina, and Sepp Hochreiter. Align-RUDDER: Learning from few demonstrations by reward redistribution. *arXiv preprint arXiv:2009.14108*, 2020.

Marek Petrik and Bruno Scherrer. Biasing approximate dynamic programming with a lower discount factor. *Advances in neural information processing systems*, 21:1265–1272, 2008.

Mariya Popova, Olexandr Isayev, and Alexander Tropsha. Deep reinforcement learning for de novo drug design. *Science advances*, 4(7):eaap7885, 2018.

Hazhir Rahmandad, Nelson Repenning, and John Sterman. Effects of feedback delay on learning. *System Dynamics Review*, 25(4):309–338, 2009.

Jette Randløv and Preben Alstrøm. Learning to drive a bicycle using reinforcement learning and shaping. In *International Conference on Machine Learning*, pp. 463–471. Citeseer, 1998.

David Raposo, Sam Ritter, Adam Santoro, Greg Wayne, Theophane Weber, Matt Botvinick, Hado van Hasselt, and Francis Song. Synthetic returns for long-term credit assignment. *arXiv preprint arXiv:2102.12425*, 2021.

Tabish Rashid, Mikayel Samvelyan, Christian Schroeder, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *International Conference on Machine Learning*, pp. 4295–4304. PMLR, 2018.

Erik Schuitema, Lucian Buşoniu, Robert Babuška, and Pieter Jonker. Control delay in reinforcement learning for real-time dynamic systems: a memoryless approach. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3226–3231. IEEE, 2010.

Sarjinder Singh. *Advanced Sampling Theory With Applications: How Michael"" Selected"" Amy*, volume 2. Springer Science & Business Media, 2003.

Kyunghwan Son, Daewoo Kim, Wan Ju Kang, David Earl Hostallero, and Yung Yi. QTRAN: Learning to factorize with transformation for cooperative multi-agent reinforcement learning. In *International Conference on Machine Learning*, pp. 5887–5896. PMLR, 2019.

Shihong Song, Jiayi Weng, Hang Su, Dong Yan, Haosheng Zou, and Jun Zhu. Playing FPS games with environment-aware hierarchical reinforcement learning. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, pp. 3475–3482, 2019.

Jonathan Sorg, Richard L Lewis, and Satinder Singh. Reward design via online gradient ascent. In *Advances in Neural Information Processing Systems*, volume 23, pp. 2190–2198, 2010.

Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z Leibo, Karl Tuyls, et al. Value-decomposition networks for cooperative multi-agent learning based on team reward. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 2085–2087, 2018.

Richard S Sutton. Learning to predict by the methods of temporal differences. *Machine learning*, 3 (1):9–44, 1988.

Wei Tang, Chien-Ju Ho, and Yang Liu. Bandit learning with delayed impact of actions. *Advances in Neural Information Processing Systems*, 34, 2021.

William Vickrey. Counterspeculation, auctions, and competitive sealed tenders. *The Journal of finance*, 16(1):8–37, 1961.

Thomas J Walsh, Ali Nouri, Lihong Li, and Michael L Littman. Learning and planning in environments with delayed feedback. *Autonomous Agents and Multi-Agent Systems*, 18(1):83–105, 2009.

Jianhao Wang, Zhizhou Ren, Beining Han, Jianing Ye, and Chongjie Zhang. Towards understanding cooperative multi-agent q-learning with value factorization. In *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021a.

Jianhao Wang, Zhizhou Ren, Terry Liu, Yang Yu, and Chongjie Zhang. QPLEX: Duplex dueling multi-agent q-learning. In *International Conference on Learning Representations*, 2021b.

Jianhong Wang, Yuan Zhang, Tae-Kyun Kim, and Yunjie Gu. Shapley Q-value: a local reward approach to solve global reward games. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 7285–7292, 2020.

Yuxin Wu and Yuandong Tian. Training agent for first-person shooter game with actor-critic curriculum learning. In *International Conference on Learning Representations*, 2017.

Zhongwen Xu, Hado P van Hasselt, and David Silver. Meta-gradient reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 31, pp. 2396–2407, 2018.

Zeyu Zheng, Junhyuk Oh, and Satinder Singh. On learning intrinsic rewards for policy gradient methods. In *Advances in Neural Information Processing Systems*, volume 31, pp. 4644–4654, 2018.

Zeyu Zheng, Junhyuk Oh, Matteo Hessel, Zhongwen Xu, Manuel Kroiss, Hado Van Hasselt, David Silver, and Satinder Singh. What can learned intrinsic rewards capture? In *International Conference on Machine Learning*, pp. 11436–11446. PMLR, 2020.

Zeyu Zheng, Risto Vuorio, Richard Lewis, and Satinder Singh. Pairwise weights for temporal credit assignment. *arXiv preprint arXiv:2102.04999*, 2021.

Zhengyuan Zhou, Panayotis Mertikopoulos, Nicholas Bambos, Peter Glynn, Yinyu Ye, Li-Jia Li, and Li Fei-Fei. Distributed asynchronous optimization with unbounded delays: How slow can you go? In *International Conference on Machine Learning*, pp. 5970–5979. PMLR, 2018.

Zhengyuan Zhou, Renyuan Xu, and Jose Blanchet. Learning in generalized linear contextual bandits with stochastic delays. In *Advances in Neural Information Processing Systems*, volume 32, pp. 5197–5208, 2019a.

Zhenpeng Zhou, Steven Kearnes, Li Li, Richard N Zare, and Patrick Riley. Optimization of molecules via deep reinforcement learning. *Scientific reports*, 9(1):1–10, 2019b.

A OMITTED PROOFS

Theorem 1 (Loss Decomposition)**.** *The surrogate loss function $\mathcal{L}_{\textit{Rand-RD}}(\theta)$ can be decomposed to two terms interpolating between return decomposition and uniform reward redistribution.*

$$\mathcal{L}_{\textit{Rand-RD}}(\theta) = \mathcal{L}_{\textit{RD}}(\theta) + \mathop{\mathbb{E}}_{\tau\sim\mathcal{D}}\Bigg[\underbrace{\mathop{\textit{Var}}_{\mathcal{I}\sim\rho_T(\cdot)}\left[\frac{T}{|\mathcal{I}|}\sum_{t\in\mathcal{I}}\widehat{R}_\theta(s_t,a_t)\right]}_{\textit{variance of the Monte-Carlo estimator}}\Bigg] \tag{9}$$

$$= \underbrace{\mathcal{L}_{\textit{RD}}(\theta)}_{\textit{return decomposition}} + \mathop{\mathbb{E}}_{\tau\sim\mathcal{D}}\Bigg[T^2\cdot\underbrace{\mathop{\textit{Var}}_{(s_t,a_t)\sim\tau}\left[\widehat{R}_\theta(s_t,a_t)\right]}_{\textit{uniform reward redistribution}}\cdot\underbrace{\frac{1}{K}\left(1-\frac{K-1}{T-1}\right)}_{\textit{interpolation weight}}\Bigg], \tag{10}$$

where K denotes the length of sampled subsequences defined in Eq. (7)*.*

Proof. First, we note that random sampling serves an unbiased estimation. i.e.,

$$\mathop{\mathbb{E}}_{\mathcal{I}\sim\rho_T(\cdot)}\left[\frac{T}{|\mathcal{I}|}\sum_{t\in\mathcal{I}}\widehat{R}_\theta(s_t,a_t)\right] = \sum_{t=0}^{T-1}\widehat{R}_\theta(s_t,a_t) = \widehat{R}_{\text{ep},\theta}(\tau). \tag{15}$$

We can decompose our loss function $\mathcal{L}_{\text{Rand-RD}}(\theta)$ as follows:

$$\begin{aligned}
\mathcal{L}_{\text{Rand-RD}}(\theta) &= \mathop{\mathbb{E}}_{\tau\sim\mathcal{D}}\left[\mathop{\mathbb{E}}_{\mathcal{I}\sim\rho_T(\cdot)}\left[\left(R_{\text{ep}}(\tau)-\frac{T}{|\mathcal{I}|}\sum_{t\in\mathcal{I}}\widehat{R}_\theta(s_t,a_t)\right)^2\right]\right]\\
&= \mathop{\mathbb{E}}_{\tau\sim\mathcal{D}}\left[\mathop{\mathbb{E}}_{\mathcal{I}\sim\rho_T(\cdot)}\left[\left(R_{\text{ep}}(\tau)-\widehat{R}_{\text{ep},\theta}(\tau)+\widehat{R}_{\text{ep},\theta}(\tau)-\frac{T}{|\mathcal{I}|}\sum_{t\in\mathcal{I}}\widehat{R}_\theta(s_t,a_t)\right)^2\right]\right]\\
&= \underbrace{\mathop{\mathbb{E}}_{\tau\sim\mathcal{D}}\left[\mathop{\mathbb{E}}_{\mathcal{I}\sim\rho_T(\cdot)}\left[\left(R_{\text{ep}}(\tau)-\widehat{R}_{\text{ep},\theta}(\tau)\right)^2\right]\right]}_{=\mathcal{L}_{\text{RD}}(\theta)}\\
&\quad+\mathop{\mathbb{E}}_{\tau\sim\mathcal{D}}\Bigg[2\big(R_{\text{ep}}(\tau)-\widehat{R}_{\text{ep},\theta}(\tau)\big)\cdot\underbrace{\mathop{\mathbb{E}}_{\mathcal{I}\sim\rho_T(\cdot)}\left[\left(\widehat{R}_{\text{ep},\theta}(\tau)-\frac{T}{|\mathcal{I}|}\sum_{t\in\mathcal{I}}\widehat{R}_\theta(s_t,a_t)\right)\right]}_{=0}\Bigg]\\
&\quad+\mathop{\mathbb{E}}_{\tau\sim\mathcal{D}}\Bigg[\underbrace{\mathop{\mathbb{E}}_{\mathcal{I}\sim\rho_T(\cdot)}\left[\left(\widehat{R}_{\text{ep},\theta}(\tau)-\frac{T}{|\mathcal{I}|}\sum_{t\in\mathcal{I}}\widehat{R}_\theta(s_t,a_t)\right)^2\right]}_{=\text{Var}\left[(T/|\mathcal{I}|)\cdot\sum_{t\in\mathcal{I}}\widehat{R}_\theta(s_t,a_t)\right]}\Bigg]\\
&= \mathcal{L}_{\text{RD}}(\theta)+\mathop{\mathbb{E}}_{\tau\sim\mathcal{D}}\left[\mathop{\text{Var}}_{\mathcal{I}\sim\rho_T(\cdot)}\left[\frac{T}{|\mathcal{I}|}\sum_{t\in\mathcal{I}}\widehat{R}_\theta(s_t,a_t)\right]\right].
\end{aligned}$$

Note our sampling distribution defined in Eq. (7) refers to "*sampling without replacement*" (Singh, 2003) whose variance can be further decomposed as follows:

$$\begin{aligned}
\mathcal{L}_{\text{Rand-RD}}(\theta) &= \mathcal{L}_{\text{RD}}(\theta)+\mathop{\mathbb{E}}_{\tau\sim\mathcal{D}}\left[\mathop{\text{Var}}_{\mathcal{I}\sim\rho_T(\cdot)}\left[\frac{T}{|\mathcal{I}|}\sum_{t\in\mathcal{I}}\widehat{R}_\theta(s_t,a_t)\right]\right]\\
&= \mathcal{L}_{\text{RD}}(\theta)+\mathop{\mathbb{E}}_{\tau\sim\mathcal{D}}\left[T^2\cdot\mathop{\text{Var}}_{(s_t,a_t)\sim\tau}\left[\widehat{R}_\theta(s_t,a_t)\right]\cdot\frac{1}{K}\left(1-\frac{K-1}{T-1}\right)\right].
\end{aligned}$$

□

The proof of Theorem 1 follows a particular form of bias-variance decomposition formula (Kohavi & Wolpert, 1996). Similar decomposition form can also be found in other works in the literature of reinforcement learning (Antos et al., 2008).

Proposition 1 (Surrogate Upper Bound)**.** *The randomized return decomposition loss $\mathcal{L}_{\text{Rand-RD}}(\theta)$ is an upper bound of the actual return decomposition loss function $\mathcal{L}_{\text{RD}}(\theta)$, i.e., $\mathcal{L}_{\text{Rand-RD}}(\theta) \geq \mathcal{L}_{\text{RD}}(\theta)$.*

Proof. Note that the second term of Eq. (9) in Theorem 1 expresses the variance of a Monte-Carlo estimator which is clearly non-negative. It directly gives $\mathcal{L}_{\text{Rand-RD}}(\theta) \geq \mathcal{L}_{\text{RD}}(\theta)$. □

An alternative proof of Proposition 1 can be directly given by Jensen's inequality.

Proposition 2 (Objective Gap)**.** *Let $\mathcal{L}^{(K)}_{\text{Rand-RD}}(\theta)$ denote the randomized return decomposition loss that samples subsequences with length K. The gap between $\mathcal{L}^{(K)}_{\text{Rand-RD}}(\theta)$ and $\mathcal{L}_{\text{RD}}(\theta)$ can be reduced by using larger values of hyper-parameter K.*

$$\forall\theta, \quad \mathcal{L}^{(1)}_{\text{Rand-RD}}(\theta) \geq \mathcal{L}^{(2)}_{\text{Rand-RD}}(\theta) \geq \cdots \geq \mathcal{L}^{(T-1)}_{\text{Rand-RD}}(\theta) \geq \mathcal{L}^{(T)}_{\text{Rand-RD}}(\theta) = \mathcal{L}_{\text{RD}}(\theta). \tag{11}$$

Proof. In Eq. (10) of Theorem 1, the last term $\frac{1}{K}\left(1 - \frac{K-1}{T-1}\right)$ monotonically decreases as the hyper-parameter K increases. When $K = T$, this coefficient is equal to zero. It derives Eq. (11) in the given statement. □

Proposition 3 (Uniform Reward Redistribution)**.** *Assume all trajectories have the same length and the parameterization space of θ serves universal representation capacity. The optimal solution $\theta^\star$ of minimizing $\mathcal{L}^{(1)}_{\text{Rand-RD}}(\theta)$ is stated as follows:*

$$\widehat{R}_{\theta^\star}(s,a) = \mathop{\mathbb{E}}_{\tau\sim\mathcal{D}}\big[R_{ep}(\tau)/T \mid (s,a)\in\tau\big], \tag{12}$$

where $\mathcal{L}^{(1)}_{\text{Rand-RD}}(\theta)$ denotes the randomized return decomposition loss with $K = 1$.

Proof. Note that we assume all trajectories have the same length. The optimal solution of this least-squares problem is given by

$$\begin{aligned}
\widehat{R}_{\theta^\star}(s,a) &= \min_{r\in\mathbb{R}} \mathop{\mathbb{E}}_{\tau\sim\mathcal{D}}\left[\left(R_{\text{ep}}(\tau) - T\cdot r\right)^2 \middle| (s,a)\in\tau\right] \\
&= \min_{r\in\mathbb{R}} \mathop{\mathbb{E}}_{\tau\sim\mathcal{D}}\left[\frac{1}{T^2}\left(R_{\text{ep}}(\tau)/T - r\right)^2 \middle| (s,a)\in\tau\right] \\
&= \min_{r\in\mathbb{R}} \frac{1}{T^2}\mathop{\mathbb{E}}_{\tau\sim\mathcal{D}}\left[\left(R_{\text{ep}}(\tau)/T - r\right)^2 \middle| (s,a)\in\tau\right] \\
&= \min_{r\in\mathbb{R}} \mathop{\mathbb{E}}_{\tau\sim\mathcal{D}}\left[\left(R_{\text{ep}}(\tau)/T - r\right)^2 \middle| (s,a)\in\tau\right] \\
&= \mathop{\mathbb{E}}_{\tau\sim\mathcal{D}}\big[R_{\text{ep}}(\tau)/T \mid (s,a)\in\tau\big],
\end{aligned}$$

which depends on the trajectory distribution in dataset $\mathcal{D}$. □

If we relax the assumption that all trajectories have the same length, the solution of the above least-squares problem would be a weighted expectation as follows:

$$\begin{aligned}
\widehat{R}_{\theta^\star}(s,a) &= \min_{r\in\mathbb{R}} \mathop{\mathbb{E}}_{\tau\sim\mathcal{D}}\left[\left(R_{\text{ep}}(\tau) - T_\tau\cdot r\right)^2 \middle| (s,a)\in\tau\right] \\
&= \min_{r\in\mathbb{R}} \mathop{\mathbb{E}}_{\tau\sim\mathcal{D}}\left[T_\tau^2\cdot\left(R_{\text{ep}}(\tau)/T_\tau - r\right)^2 \middle| (s,a)\in\tau\right] \\
&= \frac{\sum_{\tau\in\mathcal{D}:(s,a)\in\tau} T_\tau\cdot R_{\text{ep}}(\tau)}{\sum_{\tau\in\mathcal{D}:(s,a)\in\tau} T_\tau^2},
\end{aligned}$$

where T_τ denotes the length of trajectory τ. This solution can still be interpreted as a uniform reward redistribution, in which the dataset distribution is prioritized by the trajectory length.

B Experiment Settings and Implementation Details

B.1 MuJoCo Benchmark with Episodic Rewards

MuJoCo Benchmark with Episodic Rewards. We adopt the same experiment setting as Gangwani et al. (2020) and compare our approach with baselines in a suite of MuJoCo locomotion benchmark tasks with episodic rewards. This experiment setting is commonly used in the literature (Mania et al., 2018; Guo et al., 2018; Liu et al., 2019; Arjona-Medina et al., 2019; Gangwani et al., 2019; 2020). The environment simulator is based on OpenAI Gym (Brockman et al., 2016). These tasks are long-horizon with maximum trajectory length $T = 1000$. We modify the reward function of these environments to set up an episodic-reward setting. Formally, on non-terminal states, the agent will receive a zero signal instead of the per-step dense rewards. The agent can obtain the episodic feedback $R_{\text{ep}}(\tau)$ at the last step of the rollout trajectory, in which $R_{\text{ep}}(\tau)$ is computed by the summation of per-step instant rewards given by the standard setting.

Hyper-Parameter Configuration For MuJoCo Experiments. In MuJoCo experiments, the policy optimization module of RRD is implemented based on soft actor-critic (SAC; Haarnoja et al., 2018a). We evaluate the performance of our proposed methods with the same configuration of hyper-parameters in all environments. The hyper-parameters of the back-end SAC follow the official technical report (Haarnoja et al., 2018b). We summarize our default configuration of hyper-parameters as the following table:

Hyper-Parameter	Default Configuration
discount factor γ	0.99
# hidden layers (all networks)	2
# neurons per layer	256
activation	ReLU
optimizer (all losses)	Adam (Kingma & Ba, 2015)
learning rate	$3 \cdot 10^{-4}$
initial temperature α_{init}	1.0
target entropy	$-\dim(\mathcal{A})$
Polyak-averaging coefficient	0.005
# gradient steps per environment step	1
# gradient steps per target update	1
# transitions in replay buffer	10^6
# transitions in mini-batch for training SAC	256
# transitions in mini-batch for training $\widehat{R}_\theta$	256
# transitions per subsequence (K)	64
# subsequences in mini-batch for training $\widehat{R}_\theta$	4

Table 1: The hyper-parameter configuration of RRD in MuJoCo experiments.

B.2 Atari Benchmark with Episodic Rewards

Atari Benchmark with Episodic Rewards. In addition, we conduct experiments in a suite of Atari games with episodic rewards. The environment simulator is based on OpenAI Gym (Brockman et al., 2016). Following the standard Atari pre-processing proposed by Mnih et al. (2015), we rescale each RGB frame to an 84×84 luminance map, and the observation is constructed as a stack of 4 recent luminance maps. We modify the reward function of these environments to set up an episodic-reward setting. Formally, on non-terminal states, the agent will receive a zero signal instead of the per-step dense rewards. The agent can obtain the episodic feedback $R_{\text{ep}}(\tau)$ at the last step of the rollout trajectory, in which $R_{\text{ep}}(\tau)$ is computed by the summation of per-step instant rewards given by the standard setting.

Hyper-Parameter Configuration For Atari Experiments. In Atari experiments, the policy optimization module of RRD is implemented based on deep Q-network (DQN; Mnih et al., 2015). We

evaluate the performance of our proposed methods with the same configuration of hyper-parameters in all environments. The hyper-parameters of the back-end DQN follow the technical report (Castro et al., 2018). We summarize our default configuration of hyper-parameters as the following table:

Hyper-Parameter	Default Configuration
discount factor γ	0.99
# stacked frames in agent observation	4
# `noop` actions while starting a new episode	30
network architecture	DQN (Mnih et al., 2015)
optimizer for Q-values	Adam (Kingma & Ba, 2015)
learning rate for Q-values	$6.25 \cdot 10^{-5}$
optimizer for $\widehat{R}_\theta$	Adam (Kingma & Ba, 2015)
learning rate for $\widehat{R}_\theta$	$3 \cdot 10^{-4}$
exploration strategy	ϵ-greedy
ϵ decaying range - start value	1.0
ϵ decaying range - end value	0.01
# timesteps for ϵ decaying schedule	250000
# gradient steps per environment step	0.25
# gradient steps per target update	8000
# transitions in replay buffer	10^6
# transitions in mini-batch for training DQN	32
# transitions in mini-batch for training $\widehat{R}_\theta$	32
# transitions per subsequence (K)	32
# subsequences in mini-batch for training $\widehat{R}_\theta$	1

Table 2: The hyper-parameter configuration of RRD in Atari experiments.

B.3 An Alternative Implementation of Randomized Return Decomposition

Recall that the major practical barrier of the least-squares-based return decomposition method specified by $\mathcal{L}_{\text{RD}}(\theta)$ is its scalability in terms of the computation costs. The trajectory-wise episodic reward is the only environmental supervision for reward modeling. Computing the loss function $\mathcal{L}_{\text{RD}}(\theta)$ with a single episodic reward label requires to enumerate all state-action pairs along the whole trajectory.

Theorem 1 motivates an unbiased implementation of randomized return decomposition that optimizes $\mathcal{L}_{\text{RD}}(\theta)$ instead of $\mathcal{L}_{\text{Rand-RD}}(\theta)$. By rearranging the terms of Eq. (10), we can obtain the difference between $\mathcal{L}_{\text{Rand-RD}}(\theta)$ and $\mathcal{L}_{\text{RD}}(\theta)$ as follows:

$$\mathcal{L}_{\text{Rand-RD}}(\theta) - \mathcal{L}_{\text{RD}}(\theta) = \operatorname*{Var}_{\mathcal{I}\sim\rho_T(\cdot)}\left[\frac{T}{|\mathcal{I}|}\sum_{t\in\mathcal{I}}\widehat{R}_\theta(s_t,a_t)\right].$$

Note our sampling distribution $\rho_T(\cdot)$ defined in Eq. (7) refers to "*sampling without replacement*" (Singh, 2003) whose variance can be estimated as follows:

$$\operatorname*{Var}_{\mathcal{I}\sim\rho_T(\cdot)}\left[\frac{T}{|\mathcal{I}|}\sum_{t\in\mathcal{I}}\widehat{R}_\theta(s_t,a_t)\right] = T^2 \cdot \operatorname*{\mathbb{E}}_{\mathcal{I}\sim\rho_T(\cdot)}\left[\frac{T-K}{T}\cdot\frac{\sum_{t\in\mathcal{I}}\left(\widehat{R}_\theta(s_t,a_t)-\bar{R}_\theta(\mathcal{I};\tau)\right)^2}{K(K-1)}\right], \quad (16)$$

where $\bar{R}_\theta(\mathcal{I};\tau) = \frac{1}{|\mathcal{I}|}\sum_{t\in\mathcal{I}}\widehat{R}_\theta(s_t,a_t)$. Thus we can obtain an unbiased estimation of this variance penalty by sampling a subsequence $\mathcal{I}$. By subtracting this estimation from $\widehat{\mathcal{L}}_{\text{Rand-RD}}(\theta)$, we can obtain an unbiased estimation of $\mathcal{L}_{\text{RD}}(\theta)$. More specifically, we can use the following sample-based

loss function to substitute Eq. (13) in implementation:

$$\widehat{\mathcal{L}}_{\text{RD}}(\theta) = \underbrace{\frac{1}{M}\sum_{j=1}^{M}\left[\left(R_{\text{ep}}(\tau_j) - \frac{T_j}{|\mathcal{I}_j|}\sum_{t\in\mathcal{I}_j}\widehat{R}_\theta(s_{j,t}, a_{j,t})\right)^2\right]}_{\widehat{\mathcal{L}}_{\text{Rand-RD}}(\theta)} - \frac{1}{M}\sum_{j=1}^{M}\left[\frac{T_j(T_j-K)}{|\mathcal{I}_j|(|\mathcal{I}_j|-1)}\cdot\sum_{t\in\mathcal{I}_j}\left(\widehat{R}_\theta(s_{j,t}, a_{j,t}) - \frac{1}{|\mathcal{I}_j|}\sum_{t\in\mathcal{I}_j}\widehat{R}_\theta(s_{j,t}, a_{j,t})\right)^2\right].$$

The above loss function can be optimized through the same mini-batch training paradigm as what is presented in Algorithm 1.

C Experiments on Atari Benchmark with Episodic Rewards

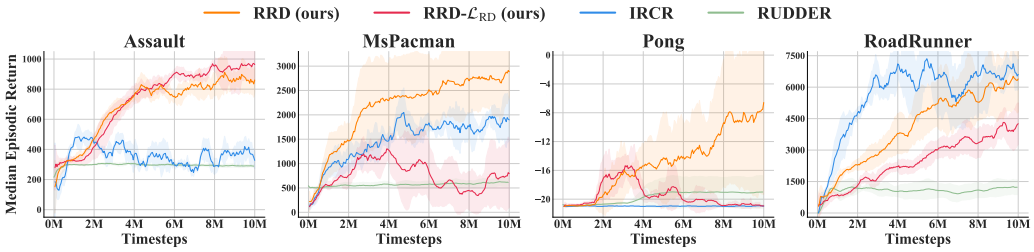

Figure 3: Learning curves on a suite of Atari benchmark tasks with episodic rewards. These curves are plotted from 5 runs with random initializations. The shaded region indicates the standard deviation. To make the comparison more clear, the curves are smoothed by averaging 10 most recent evaluation points. We set up an evaluation point every $5 \cdot 10^4$ timesteps.

Note that our proposed method does not restricts its usage to continuous control problems. It can also be integrated in DQN-based algorithms to solve problems with discrete-action space. We evaluate the performance of our method built upon DQN in several famous Atari games. The reward redistribution problem in these tasks is more challenging than that in MuJoCo locomotion benchmark since the task horizon of Atari is much longer. For example, the maximum task horizon in Pong can exceed 20000 steps in a single trajectory. This setting highlights the scalability advantage of our method, i.e., the objective of RRD can be optimized by sampling short subsequences whose computation cost is manageable. The experiment results are presented in Figure 3. Our method outperforms all baselines in 3 out of 4 tasks. We note that IRCR outperforms RRD in RoadRunner. It may be because IRCR is non-parametric and thus does not suffer from the difficulty of processing visual observations.

D Visualizing the Proxy Rewards of RRD

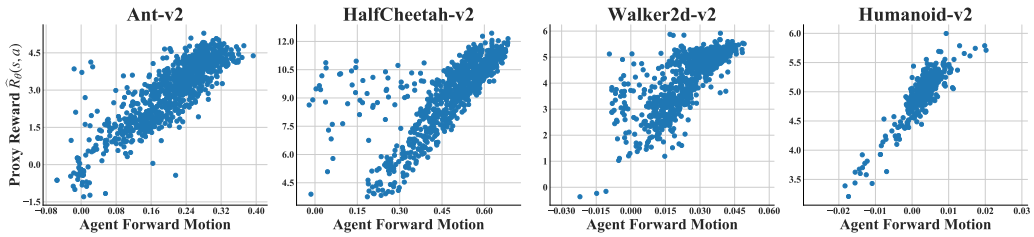

Figure 4: Visualization of the proxy rewards learned by RRD in MuJoCo locomotion tasks.

In MuJoCo locomotion tasks, the goal of agents is running towards a fixed direction. In Figure 4, we visualize the correlation between per-step forward distance and the assigned proxy reward. We uniformly collected 10^3 samples during the first 10^6 training steps. "Agent Forward Motion" denotes the forward distance at a single step. "Proxy Reward $\widehat{R}_\theta(s, a)$" denotes the immediate proxy reward assigned at that step. It shows that the learned proxy reward has high correlation to the forward distance at that step.

E An Ablation Study on the Hyper-Parameters of IRCR

We note that Gangwani et al. (2020) uses a different hyper-parameter configuration from the standard SAC implementation (Haarnoja et al., 2018b). The differences exist in two hyper-parameters:

Hyper-Parameter	Default Configuration
Polyak-averaging coefficient	0.001
# transitions in replay buffer	$3 \cdot 10^5$
# transitions in mini-batch for training SAC	512

Table 3: The hyper-parameters used by Gangwani et al. (2020) in MuJoCo experiments.

To establish a rigorous comparison, we evaluate the performance of IRCR with the hyper-parameter configuration proposed by Haarnoja et al. (2018b), so that IRCR and RRD use the same hyper-parameters in their back-end SAC agents. The experiment results are presented in Figure 5.

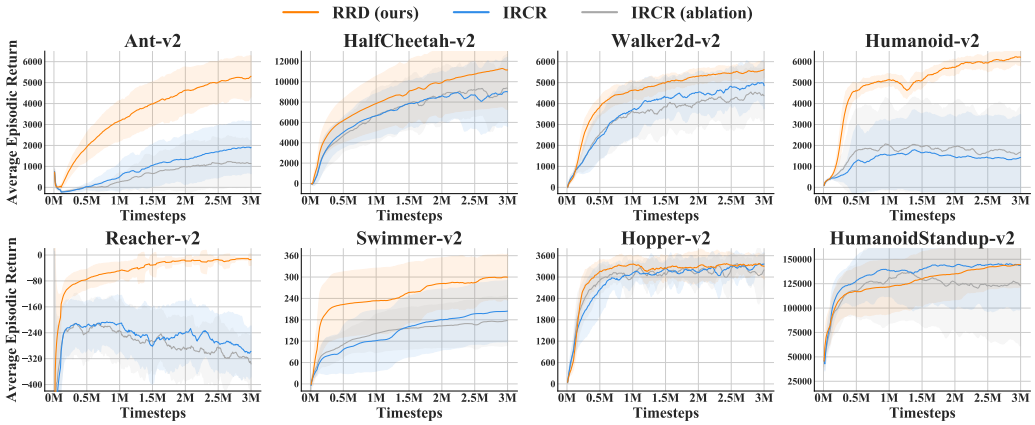

Figure 5: An ablation study on the hyper-parameter configuration of IRCR. The curves of "IRCR" refer to the performance of IRCR using the hyper-parameter setting proposed by Gangwani et al. (2020). The curves of "IRCR (ablation)" refer to the performance of IRCR using the hyper-parameters stated in Table 1. All curves are plotted from 30 runs with random initializations.

As shown in Figure 5, the hyper-parameters tuned by Gangwani et al. (2020) is more stable in most environments. Although using hyper-parameters stated in Table 1 can improve the performance in some cases, the overall performance cannot outperform RRD.

F Related Work

Delayed Feedback. Tackling environmental delays is a long-lasting problem in reinforcement learning and control theory (Nilsson et al., 1998; Walsh et al., 2009; Zhou et al., 2018; 2019a; Héliou et al., 2020; Nath et al., 2021; Tang et al., 2021). In real-world applications, almost all environmental signals have random delays (Schuitema et al., 2010; Hester & Stone, 2013; Amuru & Buehrer, 2014; Lei et al., 2020), which is a fundamental challenge for the designs of RL algorithms. A classical method to handle delayed signals is stacking recent observations within a small sliding window as

the input for decision-making (Katsikopoulos & Engelbrecht, 2003). This simple transformation can establish a Markovian environment formulation, which is widely used to deal with short-term environmental delays (Mnih et al., 2015). Many recent works focus on establishing sample-efficient off-policy RL algorithm that is adaptive to delayed environmental signals (Bouteiller et al., 2021; Han et al., 2021). In this paper, we consider an extreme delay of reward signals, which is a harder problem setting than short-term random delays.

Reward Design. The ability of reinforcement learning agents highly depends on the designs of reward functions. It is widely observed that reward shaping can accelerate learning (Mataric, 1994; Ng et al., 1999; Devlin et al., 2011; Wu & Tian, 2017; Song et al., 2019). Many previous works study how to automatically design auxiliary reward functions for efficient reinforcement learning. A famous paradigm, inverse reinforcement learning (Ng & Russell, 2000; Fu et al., 2018), considers to recover a reward function from expert demonstrations. Another branch of work is learning an intrinsic reward function from experience that guides the agent to maximize extrinsic objective (Sorg et al., 2010; Guo et al., 2016). Such an intrinsic reward function can be learned through meta-gradients (Zheng et al., 2018; 2020) or self-imitation (Guo et al., 2018; Gangwani et al., 2019). A recent work (Abel et al., 2021) studies the expressivity of Markov rewards and proposes algorithms to design Markov rewards for three notions of abstract tasks.

Temporal Credit Assignment. Another methodology for tackling long-horizon sequential decision problems is assigning credits to emphasize the contribution of each single step over the temporal structure. These methods directly consider the specification of the step values instead of manipulating the reward function. The simplest example is studying how the choice of discount factor γ affects the policy learning (Petrik & Scherrer, 2008; Jiang et al., 2015; Fedus et al., 2019). Several previous work consider to extend the λ-return mechanism (Sutton, 1988) to a more generalized credit assignment framework, such as adaptive λ (Xu et al., 2018) and pairwise weights (Zheng et al., 2021). RUDDER (Arjona-Medina et al., 2019) proposes a return-equivalent formulation for the credit assignment problem and establish theoretical analyses (Holzleitner et al., 2021). Aligned-RUDDER (Patil et al., 2020) considers to use expert demonstrations for higher sample efficiency. Harutyunyan et al. (2019) opens up a new family of algorithms, called hindsight credit assignment, that attributes the credits from a backward view.

Value Decomposition. This paper follows the paradigm of reward redistribution that aims to decompose the return value to step-wise reward signals. The simplest mechanism in the literature is the uniform reward redistribution considered by Gangwani et al. (2020). It can be effectively integrated with off-policy reinforcement learning and thus achieves state-of-the-art performance in practice. Least-squares-based reward redistribution is investigated by Efroni et al. (2021) from a theoretical point of view. Chatterji et al. (2021) extends the theoretic results to the logistic reward model. In game theory and multi-agent reinforcement learning, a related problem is how to attribute a global team reward to individual rewards (Nguyen et al., 2018; Du et al., 2019; Wang et al., 2020), which provide agents incentives to optimize the global social welfare (Vickrey, 1961; Clarke, 1971; Groves, 1973; Myerson, 1981). A promising paradigm for multi-agent credit assignment is using structural value representation (Sunehag et al., 2018; Rashid et al., 2018; Son et al., 2019; Böhmer et al., 2020; Wang et al., 2021a;b), which is integrated with end-to-end temporal difference learning.