A Broader Impacts

Our motivation is to strive to make decisions that are both understood and trusted by humans. By increasing the credibility and transparency of the decision-making process, human users develop a better understanding, validate and enhance the decisions made by algorithms. As a result, it is possible to bridge the gap between humans and AI, fostering a symbiotic relationship that leverages the strengths of both to enable more reliable and responsible decision-making in various filed. For example, in finance, transparent algorithms can enhance risk assessment, investment strategies, and fraud detection, while in transportation, trustworthy decision-making algorithms can contribute to safe and efficient navigation and logistics. To advance this mission, our work on the GRD algorithm represents a significant step forward. GRD explores facilitating policy learning in the presence of delayed rewards. By decomposing the overall return into Markovian rewards, we provide a clearer understanding of the contribution made by each state-action pair. Furthermore, we go beyond simply explaining the rewards and delve into the causal view of reward generation. This approach allows us to provide interpretable explanations of how Markovian rewards are generated, enabling a more transparent decision-making process. This interpretability is vital for building trust with human users who may need to understand and validate the decisions made by the algorithm. Moreover, with an interpretable reward function, we can readily incorporate additional restrictions, such as security constraints, into the decision-making process. This flexibility allows us to tailor the algorithm to specific needs and requirements, further enhancing its trustworthiness. Additionally, since the principles and techniques we are exploring can be applied across a wide range of domains and industries, our collaboration with GRD not only contributes to the field of reinforcement learning but also has broader implications beyond robotics.

B Proofs and Causal Background

B.1 Markov and faithfulness assumptions

A directed acyclic graph (DAG), $\mathcal{G} = (\boldsymbol{V}, \boldsymbol{E})$, can be deployed to represent a graphical criterion carrying out a set of conditions on the paths, where $\boldsymbol{V}$ and $\boldsymbol{E}$ denote the set of nodes and the set of directed edges, separately.

Definition 1. *(d-separation [50]). A set of nodes $\mathbf{Z} \subseteq \boldsymbol{V}$ blocks the path p if and only if (1) p contains a chain $i \to m \to j$ or a fork $i \leftarrow m \to j$ such that the middle node m is in $\mathbf{Z}$, or (2) p contains a collider $i \to m \leftarrow j$ such that the middle node m is not in $\mathbf{Z}$ and such that no descendant of m is in $\mathbf{Z}$. Let $\boldsymbol{X}$, $\boldsymbol{Y}$ and $\mathbf{Z}$ be disjunct sets of nodes. If and only if the set $\mathbf{Z}$ blocks all paths from one node in $\boldsymbol{X}$ to one node in $\boldsymbol{Y}$, $\mathbf{Z}$ is considered to d-separate $\boldsymbol{X}$ from $\boldsymbol{Y}$, denoting as $(\boldsymbol{X} \perp_d \boldsymbol{Y} \mid \mathbf{Z})$.*

Definition 2. *(Global Markov Condition [51, 50]). If, for any partition $(\boldsymbol{X}, \boldsymbol{Y}, \mathbf{Z})$, $\boldsymbol{X}$ is d-separated from $\boldsymbol{Y}$ given $\mathbf{Z}$,* i.e.*, $\boldsymbol{X} \perp_d \boldsymbol{Y} \mid \mathbf{Z}$. Then the distribution P over $\boldsymbol{V}$ satisfies the global Markov condition on graph G, and can be factorizes as, $P(\boldsymbol{X}, \boldsymbol{Y} \mid \mathbf{Z}) = P(\boldsymbol{X} \mid \mathbf{Z})P(\boldsymbol{Y} \mid \mathbf{Z})$. That is, $\boldsymbol{X}$ is conditionally independent of $\boldsymbol{Y}$ given $\mathbf{Z}$, writing as $\boldsymbol{X} \perp\!\!\!\perp \boldsymbol{Y} \mid \mathbf{Z}$.*

Definition 3. *(Faithfulness Assumption [51, 50]). The variables, which are not entailed by the Markov Condition, are not independent of each other.*

Under the above assumptions, we can apply d-separation as a criterion to understand the conditional independencies from a given DAG $\mathcal{G}$. That is, for any disjoint subset of nodes $\boldsymbol{X}, \boldsymbol{Y}, \mathbf{Z} \subseteq \boldsymbol{V}$, $(\boldsymbol{X} \perp\!\!\!\perp \boldsymbol{Y} \mid \mathbf{Z})$ and $\boldsymbol{X} \perp_d \boldsymbol{Y} \mid \mathbf{Z}$ are the necessary and sufficient condition of each other.

B.2 Proofs

Proposition 1 (Identifiability)**.** *Suppose the state $\boldsymbol{s}_t$, action $\boldsymbol{a}_t$, trajectory-wise long-term return R are observable while Markovian rewards r_t are unobservable, and they form an MDP, as described in Eq. 2. Then under the global Markov condition and faithfulness assumption, the reward function g and the Markovian rewards r_t are identifiable, as well as the causal structure that is characterized by binary masks $\boldsymbol{c}^{\cdot\to\cdot}$ and $\boldsymbol{C}^{\cdot\to\cdot}$ and the transition dynamics f.*

Below is the proof of Proposition 1. We begin by clarifying the assumptions we made and then provide the mathematical proof.

Assumption We assume that, $\epsilon_{s,i,t}$ and $\epsilon_{r,t}$ in Eq. 2 are i.i.d additive noise. From the weight-space view of Gaussian Process [52], equivalently, the causal models for $\boldsymbol{s}_{i,t+1}$ and r_t can be represented as follows, respectively,

$$\boldsymbol{s}_{i,t+1} = f_i(\boldsymbol{s}_t, \boldsymbol{a}_t) + \epsilon_{s,i,t} = W_{i,f}^T \phi_f(\boldsymbol{s}_t, \boldsymbol{a}_t) + \epsilon_{s,i,t}, \tag{A1}$$

$$r_t = g(\boldsymbol{s}_t, \boldsymbol{a}_t) + \epsilon_{r,t} = W_g^T \phi_g(\boldsymbol{s}_t, \boldsymbol{a}_t) + \epsilon_{r,t}, \tag{A2}$$

where $\forall i \in [1, d^{\boldsymbol{s}}]$, and ϕ_f and ϕ_g denote basis function sets.

Then we denote the variable set in the system by $\boldsymbol{V}$, with $\boldsymbol{V} = \{\boldsymbol{s}_{1,t}, \ldots, \boldsymbol{s}_{d^{\boldsymbol{s}},t}, \boldsymbol{a}_{1,t}, \ldots, \boldsymbol{a}_{d^{\boldsymbol{a}},t}, r_t\}_{t=1}^T \cup R$, and the variables form a Bayesian network $\mathcal{G}$. Following AdaRL [23], there are possible edges only from $\boldsymbol{s}_{i,t-1} \in \boldsymbol{s}_{t-1}$ to $\boldsymbol{s}_{i',t} \in \boldsymbol{s}_t$, from $\boldsymbol{a}_{j,t-1} \in \boldsymbol{a}_{t-1}$ to $\boldsymbol{s}_{i',t} \in \boldsymbol{s}_t$, from $\boldsymbol{s}_{i,t} \in \boldsymbol{s}_t$ to r_t, and from $\boldsymbol{a}_{j,t} \in \boldsymbol{a}_t$ to r_t in $\mathcal{G}$. In particular, the r_ts are unobserved, while $R = \sum_{t=1}^T \gamma^{t-1} o_t$ is observed. Thus there are deterministic edges from each r_t to R.

Below we omit the γ for simplicity.

Proof. **Given trajectory-wise long-term return R, the binary masks, $\boldsymbol{c}^{\boldsymbol{s}\to r}$, $\boldsymbol{c}^{\boldsymbol{a}\to r}$ and Markovian reward function g and the rewards r_t are identifiable.** Following the above assumption, we first rewrite the function to calculate trajectory-wise long-term return in Eq. 2 as,

$$\begin{aligned} R &= \sum_{t=1}^T r_t \\ &= \sum_{t=1}^T \left[W_g^T \phi_g(\boldsymbol{s}_t, \boldsymbol{a}_t) + \epsilon_{r,t} \right] \\ &= W_g^T \sum_{t=1}^T \phi_g(\boldsymbol{s}_t, \boldsymbol{a}_t) + \sum_{t=1}^T \epsilon_{r,t}. \end{aligned} \tag{A3}$$

For simplicity, we replace the components in Eq. A3 by,

$$\begin{aligned} \zeta_g(X) &= \sum_{t=1}^T \phi_g(\boldsymbol{s}_t, \boldsymbol{a}_t), \\ E_r &= \sum_{t=1}^T \epsilon_{r,t}, \end{aligned} \tag{A4}$$

where $X := [\boldsymbol{s}_t, \boldsymbol{a}_t]_{t=1}^T$ representing the concatenation of the covariates $\boldsymbol{s}_t$ and $\boldsymbol{a}_t$ from $t = 1$ to T. Consequently, we derive the following equation,

$$R = W_g^T \zeta_g(X) + E_r. \tag{A5}$$

Then we can obtain a closed-form solution of W_g^T in Eq. A5 by modeling the dependencies between the covariates X_τ and response variables R_τ, where both are continuous. One classical approach to finding such a solution involves minimizing the quadratic cost and incorporating a weight-decay regularizer to prevent overfitting. Specifically, we define the cost function as,

$$C(W_g) = \frac{1}{2} \sum_{X_\tau, R_\tau \sim \mathcal{D}} (R_\tau - W_g^T \zeta_g(X_\tau))^2 + \frac{1}{2} \lambda \|W_g\|^2. \tag{A6}$$

where τ represents trajectories consisting of state-action pairs X_τ and long-term returns R_τ, which are sampled from the replay buffer $\mathcal{D}$. λ is the weight-decay regularization parameter. To find the closed-form solution, we differentiate the cost function with respect to W_g and set the derivative to zero:

$$\frac{\partial C(W_g)}{\partial W_g} = 0. \tag{A7}$$

Solving this equation will yield the closed-form solution for W_g^T, *i.e.*,

$$W_g \quad = (\lambda I_d + \zeta_g \zeta_g^T)^{-1} \zeta_g R = \zeta_g (\zeta_g^T \zeta_g + \lambda I_n)^{-1} R \tag{A8}$$

Therefore, W_g, which indicates the causal structure and strength of the edge, can be identified from the observed data. In summary, given trajectory-wise long-term return R, the binary masks, $\boldsymbol{c}^{s \to r}$, $\boldsymbol{c}^{a \to r}$ and Markovian reward function g and the rewards r_t are identifiable.

The binary masks, $\boldsymbol{C}^{s \to s}$, $\boldsymbol{C}^{a \to s}$ and the transition dynamics f are identifiable In a similar manner, based on the assumption and Eq. 2, we can rewrite Eq. A1 to,

$$\boldsymbol{s}_{t+1} = W_{i,f}^T \phi_f(\boldsymbol{s}_t) + \epsilon_{s,i,t}. \tag{A9}$$

To obtain a closed-form solution for Wi, f^T in Equation A9, we can model the dependencies between the covariates X_t and the response variables $\boldsymbol{s}t + 1$, both of which are continuous. The closed-form solution can be represented as:

$$C(W_{i,f}) = \frac{1}{2} \sum_{\boldsymbol{s}_{i,t}, \boldsymbol{s}_{i,t+1} \sim \mathcal{D}} (\boldsymbol{s}_{i,t+1} - W_{i,f}^T \phi_{i,f}(\boldsymbol{s}_t))^2 + \frac{1}{2} \lambda \|W_{i,f}\|^2. \tag{A10}$$

By taking derivatives of the cost function and setting them to zero, we can obtain the closed-form solution,

$$\begin{aligned} W_{i,f} \quad &= (\lambda I_d + \phi_{i,f} \phi_{i,f}^T)^{-1} \phi_{i,f} \boldsymbol{s}_{i,t+1} \\ &= \phi_{i,f} (\phi_{i,f}^T \phi_{i,f} + \lambda I_n)^{-1} \boldsymbol{s}_{i,t+1}. \end{aligned} \tag{A11}$$

Therefore, $W_{i,f}$ can be identified from the observed data. This conclusion applies to all dimensions of the state. As a result, the f, which indicates the parent nodes of the i-dimension of the state, as well as the strength of the causal edge, are identifiable. In summary, the binary masks, $\boldsymbol{C}^{s \to s}$, $\boldsymbol{C}^{a \to s}$ and the transition dynamics f are identifiable.

Considering the Markov condition and faithfulness assumption, we can conclude that for any pair of variables $V_i, V_j \in \boldsymbol{V}$, V_i and V_j are not adjacent in the causal graph $\mathcal{G}$ if and only if they are conditionally independent given some subset of $\{V_l \mid l \neq i, l \neq j\}$. Additionally, since there are no instantaneous causal relationships and the direction of causality can be determined if an edge exists, the binary structural masks $\boldsymbol{c}^{s \to r}$, $\boldsymbol{c}^{a \to r}$, $\boldsymbol{C}^{s \to s}$, and $\boldsymbol{C}^{a \to s}$ defined over the set $\boldsymbol{V}$ are identifiable with conditional independence relationships [26]. Consequently, the functions f and g in Equation 2 are also identifiable.

□

C Implementation Details

C.1 Baselines

We compare our method against the following baselines,

- RRD (biased). This baseline utilizes a surrogate objective called randomized return decomposition loss for reducing the consumption of estimating the Markovian reward function. It applies Monte-Carlo sampling to get a biased estimation of the Mean Square Error (MSE) between the observed episodic reward and the sum of Markovian reward predictions in a sequence. We keep the same setting and hyper-parameters with its official implementation to reproduce the results, in which the policy module is optimized by soft actor-critic (SAC) algorithm [44].
- RRD (unbiased). This variant of RRD (biased) provides an unbiased estimation of MSE by sampling short sub-sequences. It offers a computationally efficient approach to optimize MSE. According to **(author?)** [14], RRD (biased) and RRD (unbiased) achieve state-of-the-art performance in episodic MuJoCo tasks.
- This baseline performs non-parametric uniform reward redistribution. At each time step, the proxy reward is set to the normalized value of the trajectory return. IRCR is a simple and efficient approach, and except for RRD, it achieves state-of-the-art performance in the literature. The implementation is from RRD [14].

Algorithm 1 Learning the generative process and policy jointly.

Initialize: Environment $\mathcal{E}$, trajectory $\tau \leftarrow \emptyset$, buffer $\mathcal{D} \leftarrow \emptyset$
Initialize: Generative Model $\Phi_{\mathrm{m}} := [\phi_{\mathrm{cau}}, \phi_{\mathrm{dyn}}, \phi_{\mathrm{rew}}]$; Policy Model Φ_π
for $i = 1, 2, \ldots, 3 \times 10^4$ **do**
  $\tau \leftarrow \emptyset$, reset $\mathcal{E}$
  for $n_{\mathrm{step}} = 1, 2, \ldots, 100$ **do**
    sample data $\langle \boldsymbol{s}_t, \boldsymbol{a}_t, o_t \rangle$ from $\mathcal{E}$, and store them to trajectory τ
    if $\mathcal{E}$ done **then**
      store trajectory $\tau = \{\boldsymbol{s}_{1:T}, \boldsymbol{a}_{1:T}, R\}$ to buffer $\mathcal{D}$, where $R = \sum_{i=1}^{T} \gamma^{t-1} o_t$
      $\tau \leftarrow \emptyset$, reset $\mathcal{E}$
    end if
    for $n_{\mathrm{batch}} = 1, 2, \ldots,$ train_batches **do**
      Sample D_1 consisting of M trajectories from $\mathcal{D}$: $D_1 = \{\langle \boldsymbol{s}_t^m, \boldsymbol{a}_t^m \rangle \mid_{t=1}^{T}, R^m\} \mid_{m=1}^{M}$
      Sample D_2 consisting of N samples from $\mathcal{D}$: $D_2 = \{\boldsymbol{s}_{t_n}, \boldsymbol{a}_{t_n}, \boldsymbol{s}_{t_n+1}\} \mid_{n=1}^{N}$
      Sample binary masks by Gumbel-Softmax from ϕ_{cau}: $\hat{\boldsymbol{c}}^{\boldsymbol{s}\to r}$, $\hat{\boldsymbol{c}}^{\boldsymbol{a}\to r}$, $\hat{\boldsymbol{C}}^{\boldsymbol{s}\to \boldsymbol{s}}$ and $\hat{\boldsymbol{C}}^{\boldsymbol{a}\to \boldsymbol{s}}$
      Sample binary masks deterministically from ϕ_{cau}: $\tilde{\boldsymbol{c}}^{\boldsymbol{s}\to r}$, $\tilde{\boldsymbol{c}}^{\boldsymbol{a}\to r}$, and $\tilde{\boldsymbol{C}}^{\boldsymbol{s}\to \boldsymbol{s}}$
      Calculate $\tilde{\boldsymbol{c}}^{\boldsymbol{s}\to \pi}$ based on $\tilde{\boldsymbol{c}}^{\boldsymbol{s}\to r}$ and $\tilde{\boldsymbol{C}}^{\boldsymbol{s}\to \boldsymbol{s}}$
      Update D_2: $D_2 \leftarrow \{\tilde{\boldsymbol{c}}^{\boldsymbol{s}\to \pi} \odot \boldsymbol{s}_{t_n}, \boldsymbol{a}_{t_n}, \boldsymbol{s}_{t_n+1}, \phi_{\mathrm{rew}}(\boldsymbol{s}_{t_n}, \boldsymbol{a}_{t_n}, \tilde{\boldsymbol{c}}^{\boldsymbol{s}\to r}, \tilde{\boldsymbol{c}}^{\boldsymbol{a}\to r})\} \mid_{n=1}^{N}$
      Optimize ϕ_{rew} with D_1 (Using $\hat{\boldsymbol{c}}^{\boldsymbol{s}\to r}$ and $\hat{\boldsymbol{c}}^{\boldsymbol{s}\to r}$): $\phi_{\mathrm{rew}} \leftarrow \phi_{rew} - \alpha \nabla_{\phi_{\mathrm{rew}}} L_{\mathrm{rew}}$ (Eq. 4)
      Optimize ϕ_{dyn} with D_2 (Using $\hat{\boldsymbol{C}}^{\boldsymbol{s}\to \boldsymbol{s}}$ and $\hat{\boldsymbol{C}}^{\boldsymbol{s}\to \boldsymbol{s}}$): $\phi_{\mathrm{dyn}} \leftarrow \phi_{dyn} - \alpha \nabla_{\phi_{dyn}} L_{dyn}$ (Eq. 5)
      Optimize ϕ_{cau}: $\phi_{\mathrm{cau}} \leftarrow \phi_{\mathrm{cau}} - \alpha \nabla_{\phi_{\mathrm{cau}}} (L_{\mathrm{sp}} + L_{\mathrm{rew}} + L_{\mathrm{dyn}})$ (Eq. 6)
      Optimize Φ_π: $\Phi_\pi \leftarrow \Phi_\pi - \alpha \nabla_{\Phi_\pi} J_\pi$ (Eq. 8)
    end for
  end for
end for

C.2 Detailed Generative Model

The parametric generative model Φ_{m} used in the MDP environment consists of three components: ϕ_{cau}, ϕ_{rew}, and ϕ_{dyn}. We provide a detailed description of their model structures below.

ϕ_{cau} for predicting the causal structure ϕ_{cau} comprises a set of free parameters without input. We divide ϕ_{cau} into four parts, each corresponding to the binary masks in Equation 2. Specifically, we have

- $\phi_{\mathrm{cau}}^{\boldsymbol{s}\to \boldsymbol{s}} \in \mathbb{R}^{d^{\boldsymbol{s}} \times d^{\boldsymbol{s}} \times 2}$ for $\boldsymbol{C}^{\boldsymbol{s}\to \boldsymbol{s}} \in \{0,1\}^{d^{\boldsymbol{s}} \times d^{\boldsymbol{s}}}$,
- $\phi_{\mathrm{cau}}^{\boldsymbol{a}\to \boldsymbol{s}} \in \mathbb{R}^{d^{\boldsymbol{a}} \times d^{\boldsymbol{s}} \times 2}$ for $\boldsymbol{C}^{\boldsymbol{a}\to \boldsymbol{s}} \in \mathbb{R}^{d^{\boldsymbol{a}} \times d^{\boldsymbol{s}}}$,
- $\phi_{\mathrm{cau}}^{\boldsymbol{s}\to r} \in \mathbb{R}^{d^{\boldsymbol{s}} \times 2}$ for $\boldsymbol{c}^{\boldsymbol{s}\to r} \in \mathbb{R}^{d^{\boldsymbol{s}}}$,
- $\phi_{\mathrm{cau}}^{\boldsymbol{a}\to r} \in \mathbb{R}^{d^{\boldsymbol{a}} \times 2}$ for $\boldsymbol{c}^{\boldsymbol{a}\to r} \in \mathbb{R}^{d^{\boldsymbol{a}}}$.

Below we explain the shared workflows in ϕ_{cau} using the example of predicting the causal edge from the i-th dimension of state $\boldsymbol{s}_{i,t}$ to the j-th dimension of the next state $\boldsymbol{s}_{j,t+1}$, by part of the free parameters, $\phi_{\mathrm{cau},i,j}^{\boldsymbol{s}\to \boldsymbol{s}}$.

For simplicity, we denote $\phi_{\mathrm{cau},i,j}^{\boldsymbol{s}\to \boldsymbol{s}}$ as ψ. The shape of ψ is now easy to be determined. That is $\psi \in \mathbb{R}^2$ and we write it as $\psi = [\psi_0, \psi_1]$. With this 2-element vector, we can characterize a Bernoulli distribution, where each element corresponds to the unnormalized probability of classifying the edge as existing (ψ_0) or not existing (ψ_1), respectively. Therefore, the probability of the causal edge existing from the i-th dimension of state $\boldsymbol{s}_{i,t}$ to the j-th dimension of the next state $\boldsymbol{s}_{j,t+1}$ can be calculated as:

$$P(\boldsymbol{C}_{i,j}^{\boldsymbol{s}\to \boldsymbol{s}}) = \frac{\exp(\psi_0)}{\exp(\psi_0) + \exp(\psi_1)} \tag{A12}$$

Obtain $\hat{\boldsymbol{C}}_{i,j}^{\boldsymbol{s}\to \boldsymbol{s}}$ through Gumbel-Softmax sampling in the training phases. During training, it is crucial to maintain the gradient flow for backpropagation. To achieve this, we sample the binary

Layer#	1	2	3
ϕ_{rew}	FC256	FC256	FC1
ϕ_{dyn}	FC256	FC256	FC9
ϕ_π	FC256	FC256	FC2$d^{\boldsymbol{a}}$
ϕ_v	FC256	FC256	FC1

Table A1: The network structures of ϕ_{rew}, ϕ_{dyn}, ϕ_π and ϕ_v. $FC256$ denotes a fully-connected layer with an output size of 256. Each hidden layer is followed by an activation function, ReLU. $d^{\boldsymbol{a}}$ is the number of dimensions of the action in a specific task.

values of $\hat{\boldsymbol{C}}_{i,j}^{\boldsymbol{s}\to\boldsymbol{s}}$ by applying Gumbel-Softmax [41],

$$\hat{\boldsymbol{C}}_{i,j}^{\boldsymbol{s}\to\boldsymbol{s}} = \text{GS}(\psi) \tag{A13}$$

where GS denotes the Gumbel-Softmax sampling, which allows us to obtain binary discrete samples from the Bernoulli distribution. By applying Gumbel Softmax sampling allows us to randomly sample from the Bernoulli distribution in a stochastic manner, rather than simply selecting the class with the highest probability. This introduces some randomness, enabling the model to explore the balance and uncertainty between different classifications more flexibly.

Obtain $\tilde{\boldsymbol{C}}_{i,j}^{\boldsymbol{s}\to\boldsymbol{s}}$ by deterministic sampling in the inference phases. During inference, including data sampling and policy learning, we get the prediction of $\boldsymbol{C}_{i,j}^{\boldsymbol{s}\to\boldsymbol{s}}$ through a deterministic sampling,

$$\tilde{\boldsymbol{C}}_{i,j}^{\boldsymbol{s}\to\boldsymbol{s}} = \begin{cases} 1, \psi_0 \geq \psi_1 \\ 0, \psi_0 < \psi_1. \end{cases} \tag{A14}$$

This is a greedy sampling to avoid introducing randomness during the Gumble-Softmax sampling.

The above explanation of the workflow in ϕ_{cau} for predicting a single causal edge provides insight into the overall implementation of the entire module ϕ_{cau} and can be applicable for all the causal edges. Therefore, we can obtain $\hat{\boldsymbol{C}}^{\boldsymbol{a}\to\boldsymbol{s}}$, $\hat{\boldsymbol{c}}^{\boldsymbol{s}\to r}$, $\hat{\boldsymbol{c}}^{\boldsymbol{a}\to r}$, $\tilde{\boldsymbol{c}}^{\boldsymbol{s}\to r}$ and $\tilde{\boldsymbol{c}}^{\boldsymbol{a}\to r}$, using similar procedures.

ϕ_{rew} for predicting the Markovian rewards ϕ_{rew} is a stacked fully-connected network, and the details of the network structure are provided in Table A1.

During training, the prediction of Markovian reward can be written as,

$$\hat{r} = \phi_{\text{rew}}(\boldsymbol{s}_t, \boldsymbol{a}_t, \hat{\boldsymbol{c}}^{\boldsymbol{s}\to r}, \hat{\boldsymbol{c}}^{\boldsymbol{a}\to r}) = \text{FCs}([\hat{\boldsymbol{c}}^{\boldsymbol{s}\to r} \odot \boldsymbol{s}_t, \hat{\boldsymbol{c}}^{\boldsymbol{a}\to r} \odot \boldsymbol{a}_t]), \tag{A15}$$

where $[\cdot,\cdot]$, $\odot$ denotes concatenation and element-wise multiply operations, respectively. FCs denotes the stacked fully-connected network. $\hat{\boldsymbol{c}}^{\boldsymbol{s}\to r}$ and $\hat{\boldsymbol{c}}^{\boldsymbol{s}\to r}$ are derived from ϕ_{cau} by Gumbel-Softmax.

During inference, including policy learning and data sampling, the predicted Markovian reward is

$$\tilde{r} = \phi_{\text{rew}}(\boldsymbol{s}_t, \boldsymbol{a})_t, \tilde{\boldsymbol{c}}^{\boldsymbol{s}\to r}, \tilde{\boldsymbol{c}}^{\boldsymbol{a}\to r}) = \text{FCs}([\tilde{\boldsymbol{c}}^{\boldsymbol{s}\to r} \odot \boldsymbol{s}_t, \tilde{\boldsymbol{c}}^{\boldsymbol{a}\to r} \odot \boldsymbol{a}_t]), \tag{A16}$$

where $\hat{\boldsymbol{c}}^{\boldsymbol{s}\to r}$ and $\hat{\boldsymbol{c}}^{\boldsymbol{s}\to r}$ are derived from ϕ_{cau} greedily by deterministic sampling.

ϕ_{dyn} for modeling the environment dynamics In our experiment, we do not directly utilize ϕ_{dyn} in policy learning. Instead, this module serves as a bridge to optimize $\phi_{\text{cau}}^{\boldsymbol{s}\to\boldsymbol{s}}$ and $\phi_{\text{cau}}^{\boldsymbol{a}\to\boldsymbol{s}}$. Subsequently, $\phi_{\text{cau}}^{\boldsymbol{s}\to\boldsymbol{s}}$ can be utilized in the calculation of $\tilde{\boldsymbol{c}}^{\boldsymbol{s}\to\pi}$.

During training, we initially sample $\hat{\boldsymbol{C}}^{\boldsymbol{s}\to\boldsymbol{s}}$ and $\hat{\boldsymbol{C}}^{\boldsymbol{a}\to\boldsymbol{s}}$ using Gumbel-Softmax. The prediction for the i-th dimension of the next state can be represented as follows,

$$\hat{\boldsymbol{s}}_{i,t} = \text{MDN}([\hat{\boldsymbol{C}}_{\cdot,i}^{\boldsymbol{s}\to\boldsymbol{s}} \odot \boldsymbol{s}_t, \hat{\boldsymbol{C}}_{\cdot,i}^{\boldsymbol{a}\to\boldsymbol{s}} \odot \boldsymbol{a}_t]), \tag{A17}$$

where $[\cdot,\cdot]$ denotes concatenation and MDN denotes the Mixture Density Network which outputs the means, variances, and probabilities for N_{Gau} Gaussian cores. The parameters of MDN are shared across the predictions of different dimensions of the next state. We set $N_{cau} = 3$ in our experiments. More details about ϕ_{dyn} can be found in Table A1.

C.3 Detailed Policy Model

Considering the specific requirements of the employed RL algorithm, Soft Actor-Critic (SAC), our Policy Model Φ_π comprises two components, the actor ϕ_π and the critic ϕ_v. Detailed network structures for both components can be found in Table A1.

Table A2: The table of the hyper-parameters used in the experiments for GRD.

Envs	λ_1	λ_2	λ_3	λ_4	λ_5
Ant	10^{-5}	0	10^{-7}	10^{-8}	10^{-8}
HalfCheetah	10^{-5}	10^{-5}	10^{-5}	10^{-6}	10^{-5}
Walker2d	10^{-5}	10^{-5}	10^{-6}	10^{-6}	10^{-7}
Humanoid	10^{-5}	10^{-8}	10^{-5}	10^{-7}	10^{-8}
Reacher	5×10^{-7}	10^{-8}	10^{-8}	10^{-8}	10^{-8}
Swimmer	10^{-7}	10^{-9}	10^{-9}	0	10^{-9}
Hopper	10^{-6}	10^{-6}	10^{-6}	10^{-7}	10^{-6}
HumanStandup	10^{-5}	10^{-4}	10^{-6}	10^{-7}	10^{-7}

Table A3: The hyper-parameters.

hyperparameters	value	hyperparameters	value
epochs	3	optimizer	Adam
cycles	100	learning rate	3×10^{-4}
iteration	100	N	256
train_batches	100	M	4
replay buffer size	10^6	γ	1.00
evaluation episodes	10	Polyak-averaging coefficient	0.0005

C.4 Training Process.

We follow the line of joint learning in **(author?)** [14], which avoids learning a return decomposition model in advance using data sampled by optimal or sub-optimal policies [13]. During each mini-batch training iteration, we sample two sets of data separately from the replay buffer $\mathcal{D}$:

- $D_1 = \{\langle \boldsymbol{s}_t^m, \boldsymbol{a}_t^m \rangle \mid_{t=1}^{T}, R^m\} \mid_{m=1}^{M}$ consists of M trajectories. Provided with the trajectory-wise long-term returns $R^m \mid_{m=1}^{M}$, D_1 is utilized to optimize $\phi_{\text{cau}}^{\boldsymbol{s}\to r}$, $\phi_{\text{cau}}^{\boldsymbol{a}\to r}$ and ϕ_{rew}, with L_{rew}.

- $D_2 = \{\boldsymbol{s}_{t_n}, \boldsymbol{a}_{t_n}, \boldsymbol{s}_{t_n+1}\} \mid_{n=1}^{N}$ consists of N state-action pairs. D_2 are used for policy optimization and optimize the parts of causal structure, $\phi_{\text{cau}}^{\boldsymbol{s}\to\boldsymbol{s}}$ and $\phi_{\text{cau}}^{\boldsymbol{a}\to\boldsymbol{s}}$, ϕ_{dyn}. With such a D_2, GRD breaks the temporal cues in the training data to learn the policy and dynamics function.

Please refer to Algorithm 1 for a detailed training process.

C.5 Hyper-Parameters.

The network is trained from scratch using the Adam optimizer, without any pre-training. The initial learning rate for both model estimation and policy learning is set to 3×10^{-4}. The hyperparameters for policy learning are shared across all tasks, with a discount factor of 1.00 and a Polyak-averaging coefficient of 5×10^{-4}. The target entropy is set to the negative value of the dimension of the robot action. To facilitate training, we utilize a replay buffer with a size of 1×10^6 time steps. The warmup size of the buffer for training is set to 1×10^4. The model is trained for 3 epochs, with each epoch consisting of 100 training cycles. In each cycle, we repeat the process of data collection and model training for 100 iterations. During each iteration, we collect data from 100 time steps of interaction with the MuJoCo simulation, which is then stored in the replay buffer. For training the ϕ_{rew}, we sample 4 episodes, each containing 5×10^3 steps. As for policy learning and the optimization of ϕ_{dyn}, we use data from 256 time steps. ϕ_{cau} is trained together with ϕ_{rew} and ϕ_{dyn}. Validation is performed after every cycle, and the average metric is computed based on 10 test rollouts. The hyperparameters for learning the GRD model can be found in Table A2. All experiments were conducted on an HPC system equipped with 128 Intel Xeon processors operating at a clock speed of 2.2 GHz and 5 terabytes of memory.

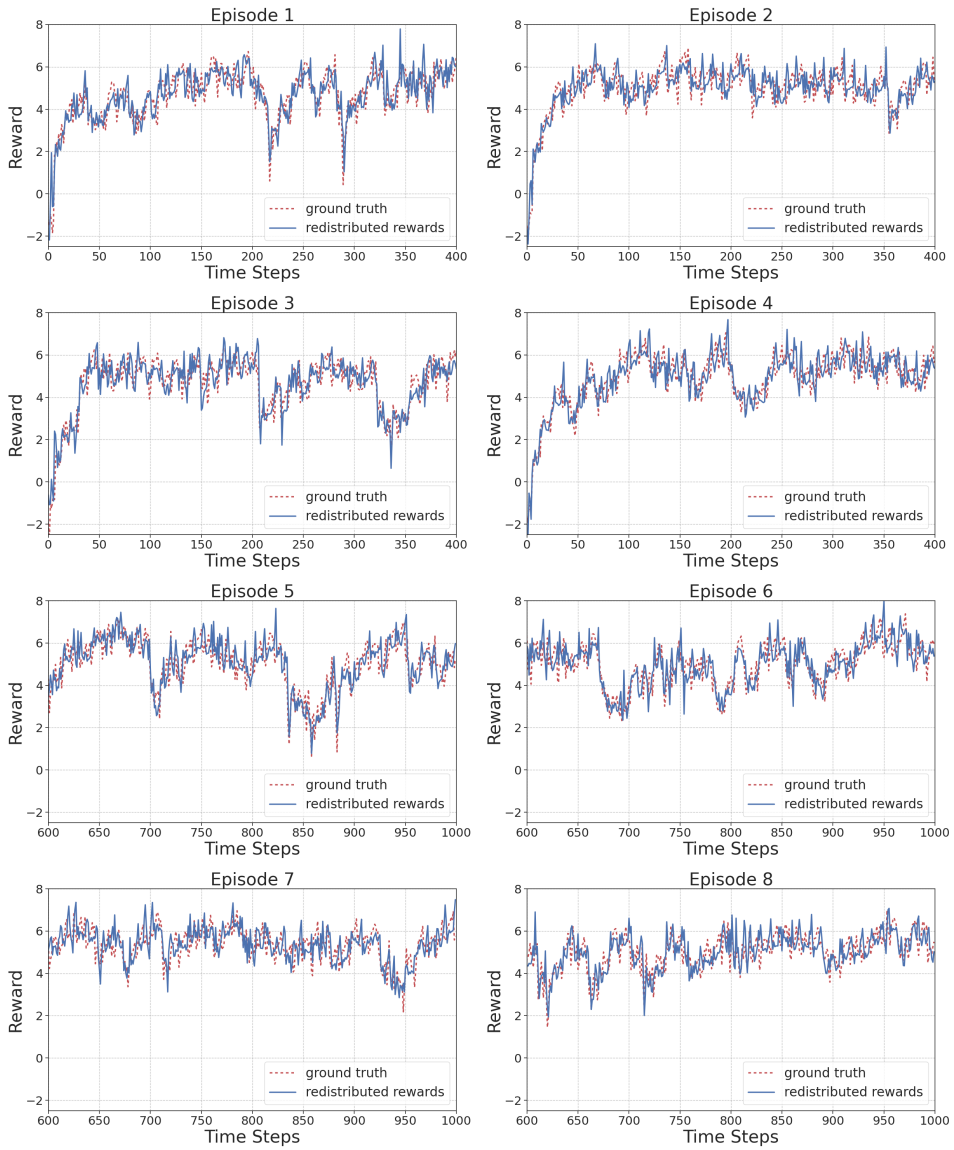

Figure A1: The visualization of redistributed rewards and grounded rewards in *Ant*. The results are produced by the GRD model trained after 1×10^6 steps. The redistributed rewards are shown in red, and the grounded rewards are shown in blue.

D Visualization

As shown in Figure A1, we visualize the redistributed rewards in *Ant* by GRD, as well as the grounded rewards provided by the environment.