MIND THE GAP: OFFLINE POLICY OPTIMIZATION FOR IMPERFECT REWARDS

Anonymous authors
Paper under double-blind review

ABSTRACT

Reward function is essential in reinforcement learning (RL), serving as the guiding signal to incentivize an agent to solve a given task. However, reward function is notoriously difficult to design or even approximate. In many cases, only a sub-par reward function can be obtained, and it is even more challenging when zero reward signal is available, which often inflicts substantial performance loss or stringent restrictive requirements on expert demonstrations. In this study, we propose a unified offline policy optimization approach, *RGM (Reward Gap Minimization)*, which can smartly handle diverse types of imperfect rewards. RGM is formulated as a bi-level optimization problem: the upper layer optimizes a reward correction term that performs state-action visitation distribution matching w.r.t. a small set of expert data; and the lower layer solves a pessimistic RL problem with the corrected rewards. By exploiting the duality of the lower level problem, we derive a tractable algorithm that enables sampled-based learning without any online interactions. Comprehensive experiments demonstrate that RGM achieves superior performance to existing methods under diverse settings of imperfect rewards. Further, RGM can effectively correct wrong or inconsistent rewards against expert preference, as well as retrieving useful information from biased rewards.

1 INTRODUCTION

Defining reward plays an imperative role in every reinforcement learning (RL) problem. It encodes the desired system behaviors, serving as a central signal to incentivize agents to learn and solve a given task (Abel et al., 2021). However, accurately describing a complex task using a numerical reward function is often impossible (Abel et al., 2021; Li et al., 2019). In rare cases, a perfect reward function can be approximated when task is extremely simple, such as rewarding large throttle to guide an autonomous vehicle to achieve a simple "accelerate task". This can be regarded as the *Perfect Reward*. Whereas, perfect rewards are commonly not obtainable due to task complexity or human cognitive biases (Hadfield-Menell et al., 2017). For example, when modeling drift in a car race (Cutler & How, 2016), only limited learning signals of every vehicular action are available, thus merely a roughly estimated reward can be used to represent the acceleration and steering actions of a vehicle. We call this the *Partially Correct Reward*. More often than not, no learning signals are available to aid in solving the given task, leading to an *Incorrect Reward* scenario.

In reality, such imperfect rewards (partially correct or incorrect) are common when unlimited reward tuning is not available due to costly or even dangerous online interactions (e.g., applying premature policies in medical applications). Generally, we can only access pre-collected datasets paired with imperfect rewards in real-world applications (Zhan et al., 2022), where a significant gap exists between the given reward in the offline dataset and the true reward in the intended task. Consequently, it is of great significance and practical value to devise adaptable methods that can perform robust offline policy optimization under imperfect rewards.

Reward shaping (Dorigo & Colombetti, 1994; Ng et al., 1999; Marthi, 2007; Hu et al., 2020; Hadfield-Menell et al., 2017) is the most common approach to addressing imperfect reward signals, which requires tremendous human efforts and numerous online evaluations. Another possible avenue is imitation learning (IL) (Pomerleau, 1988; Kostrikov et al., 2019; Sasaki & Yamashina, 2020; Kim et al., 2021; Ma et al., 2022; Xu et al., 2022b; Zhang et al., 2022) or inverse reinforcement learning methods (IRL) (Jarboui & Perchet, 2021), by directly imitating or deriving new reward functions

from expert behaviors. However, these methods heavily depend on the quantity and quality of expert demonstrations and offline datasets, which is often beyond reach in practice. Another key challenge is how to precisely measure the distance between the given rewards in data and the true rewards of the real task. Particularly, in the offline setting, it is impossible to evaluate the learned policy under a specific reward through environment interactions and verify if the policy produces desired behaviors, let alone revising the reward.

In this paper, we investigate this challenge of imperfect rewards under the offline RL setting, where no environment interactions are available. We first formally define the relative gap between given and perfect rewards based on discounted state-action visitation distribution matching, and formulate the offline policy learning problem as a bi-level optimization. In the upper layer, the imperfect rewards are adjusted by a reward correction term, which is learned by minimizing the reward gap towards expert behaviors. In the lower layer, we solve a pessimistic RL problem to obtain the optimized policy under the corrected rewards. By exploiting Lagrangian duality of the lower level problem , the overall optimization procedure can be tractably solved in a fully-offline manner without any online interactions. We call this approach 'Reward Gap Minimization' (*RGM*). Compared to existing methods, RGM can: 1) evaluate and minimize the reward gap without any online interactions; 2) eliminate the strong dependency on human efforts and expert demonstrations; and 3) handle all three types of rewards (perfect, partially correct, incorrect) in a unified framework for reliable offline policy optimization.

In practice, we implement RGM using stochastic first-order two-timescale optimization. Through extensive experiments on D4RL datasets (Fu et al., 2020) and a discrete-space navigation task, we demonstrate that RGM achieves superior performance to existing methods. Furthermore, we show that RGM effectively corrects wrong/inconsistent rewards against expert preference and successfully retrieves useful information from biased rewards, making it an ideal tool for practical tasks where reward functions are difficult to design.

2 Related work

We here briefly summarize relevant methodological approaches that handle different types of rewards.

Perfect Rewards. Directly applying offline RL algorithms is a natural choice for offline policy optimization when rewards are assumed to be perfect w.r.t the given task (Fujimoto et al., 2019; Kumar et al., 2019; 2020; Xu et al., 2021; Wu et al., 2021; Niu et al., 2022; Fujimoto & Gu, 2021; Kostrikov et al., 2021a;b; Xu et al., 2022a;c; Li et al., 2022; Lee et al., 2021; Bai et al., 2021; An et al., 2021). However, specifying a reward function that aligns well with the given task requires domain knowledge and deep understanding of the task. Even given the perfect rewards, some offline RL methods still need to reshape or shift the rewards to achieve the best policy performance (Kostrikov et al., 2021a; Kumar et al., 2020), the equivalence to engineering the initialization of Q-function estimation that encourages conservative exploitation under offline learning (Sun et al., 2022).

Partially Correct Rewards. Reward shaping is the most common approach to handle partially correct rewards, by modifying the original reward function to incorporate task-specific domain knowledge (Dorigo & Colombetti, 1994; Randløv & Alstrøm, 1998). Potential-based reward shaping (PBRS) is the first reward shaping approach from the perspective of the policy invariance property (Ng et al., 1999). Other reward shaping approaches include belief reward shaping (Marom & Rosman, 2018) and ethics shaping (Wu & Lin, 2018). However, these approaches follow a trial-and-error paradigm and require tremendous human efforts. Recent approaches such as population-based method (Jaderberg et al., 2019), optimal reward framework (Chentanez et al., 2004; Sorg et al., 2010; Zheng et al., 2018) and automatic reward shaping (Hu et al., 2020; Devidze et al., 2021; Marthi, 2007) can automatically shape the reward functions when online interaction is allowed. To the best knowledge of the authors, there is no reward shaping or correction mechanism existing for offline policy optimization. Therefore, when the given rewards are mostly correct but contain errors that lead to undesired agent behavior, people have to discard the given rewards and resort to other stopgaps like offline imitation learning, losing potentially useful information inside the existing rewards.

Incorrect Rewards. When rewards are believed to be totally wrong or missing, existing offline policy learning methods typically adopt offline imitation learning (IL) methods. These methods directly mimic the expert behavior from demonstration data without the presence of reward signal.

Among these approaches, behavior cloning (BC) (Pomerleau, 1988; Florence et al., 2022) is the simplest method, but is vulnerable to covariate shift and compounding errors (Rajaraman et al., 2020). Recent work alleviates this issue via distribution matching, which can be solved by learning a reward function (Jarboui & Perchet, 2021) or by tractable reformulation using Fenchel duality (Kostrikov et al., 2019; Kim et al., 2021; Ma et al., 2022). Some recent works also try to tackle this problem by leveraging a small expert dataset and a large, potentially sub-optimal dataset (Zolna et al., 2020; Xu et al., 2022b; Zhang et al., 2022). They use a discriminator to measure the optimality level of the data and further guide policy learning. The downside of these approaches is that they have strong requirements on the offline dataset to have a full coverage of expert distribution and only try to imitate the expert, rather than performing RL w.r.t the underlying reward of the task to improve beyond the policies in data.

3 PRELIMINARIES

Markov Decision Process under Imperfect Rewards. We consider the typical Markov Decision Process (MDP) setting (Puterman, 2014), which is defined by a tuple $\mathcal{M} := (S, A, r, T, \mu_0, \gamma)$. S and A represent the state and action space, $r : S \times A \to \mathbb{R}$ is the perfect reward function, $T : S \times A \to \Delta(S)$ is the transition dynamics which represents the probability $T(s_{t+1}|s_t, a_t)$ of the transition from state s_t to state s_{t+1} by executing action a_t at timestep t. $\mu_0 \in \Delta(S)$ is the distribution of the initial state s_0, and $\gamma \in (0, 1)$ is the discount factor.

The perfect reward function $r(s, a)$ encodes the desired behaviors of the task. But in most cases, we only have access to an imperfect human-designed reward function $\tilde{r}(s, a)$, which may not align well with the target task. This leads to a biased MDP $\widetilde{\mathcal{M}} := (S, A, \tilde{r}, T, \mu_0, \gamma)$ as compared to the original MDP $\mathcal{M}$. To remedy the adverse effects of imperfect reward signals, existing offline policy learning studies (Zolna et al., 2020; Xu et al., 2022b; Ma et al., 2022; Kim et al., 2021; Jarboui & Perchet, 2021) introduce additional expert demonstrations $\mathcal{D}^E = \left\{ \left(s_0^E, a_0^E, s_1^E, \cdots\right)^{(i)} \right\}_{i=0}^{N^E}$ to provide extra information on the desired policy behaviors. We follow a similar setup, but only consume very limited expert demonstrations. In our offline policy optimization setting, we are given a pre-collected dataset $\mathcal{D}^O = \left\{ (s_0, a_0, \tilde{r}_0, s_1, \cdots)^{(i)} \right\}_{i=0}^{N^O}$ that is generated by an unknown behavior policy π^β and annotated with imperfect rewards $\tilde{r}$. We aim to learn an effective policy $\pi : S \to \Delta(A)$ to capture the optimized agent behavior in $\mathcal{M}$ rather than $\widetilde{\mathcal{M}}$ using both $\mathcal{D}^O$ and a very small expert dataset $\mathcal{D}^E$. For simplicity, we denote $\mathcal{D} = \mathcal{D}^O \cup \mathcal{D}^E$.

Reinforcement Learning. With a given MDP and the reward function $r(s, a)$, the goal of RL is to find an optimized policy π_r^* to maximize the expected cumulative discount reward:

$$\pi_r^* = \arg\max_{\pi_r} (1-\gamma)\mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \,|s_0 \sim \mu_0(\cdot), a_t \sim \pi_r(\cdot|s_t), s_{t+1} \sim T(\cdot|s_t, a_t)\right] \quad (1)$$

This optimization objective can be equivalently written into the following succinct form (Puterman, 2014; Nachum et al., 2019b) by defining the normalized discounted state-action visitation distribution $d^{\pi_r}(s, a)$:

$$\pi_r^* = \arg\max_{\pi_r} \mathbb{E}_{(s,a)\sim d^{\pi_r}}[r(s, a)] \quad (2)$$

$$d^{\pi_r}(s, a) = (1-\gamma)\sum_{t=0}^{\infty} \gamma^t \mathrm{Pr}\left[s_t = s, a_t = a | s_0 \sim \mu_0(\cdot), a_t \sim \pi_r(\cdot|s_t), s_{t+1} \sim T(\cdot|s_t, a_t)\right] \quad (3)$$

This RL objective is not directly applicable to offline setting, as it is no longer possible to sample from d^{π_r} via online interactions, thus serious distributional shift (Kumar et al., 2019) may occur without proper data-related regularization when learning from offline datasets. To tackle these problems, several recent works (Nachum et al., 2019b; Nachum & Dai, 2020; Lee et al., 2021) incorporate a regularizer into Eq. (2) to formulate a pessimistic RL framework that is usable in the offline setting:

$$\pi_r^* = \arg\max_{\pi_r} \mathbb{E}_{(s,a)\sim d^{\pi_r}}[r(s, a)] - \alpha \cdot \mathrm{D}\left(d^{\pi_r} \| d^{\mathcal{D}}\right) \quad (4)$$

where $d^{\mathcal{D}}$ represents the discounted state-action visitation distribution of dataset $\mathcal{D}$, $\mathrm{D}(\cdot\|\cdot)$ represents some statistical discrepancy measures and $\alpha > 0$ controls the strength of the regularization.

4 REWARD GAP MINIMIZATION

To handle diverse reward settings (perfect and imperfect rewards) in real-world tasks under a unified framework, we have to investigate three major questions:

1) *How to measure the gap between the given rewards and the underlying unknown perfect rewards?*
2) *How to unify different reward settings and bridge the reward gap?*
3) *How to perform offline policy optimization using an integrated framework?*

Our answer to these questions is Reward Gap Minimization (RGM). We first formally define the reward gap in the perspective of discounted state-action visitation distribution matching. A reward correction term is introduced to correct the problematic rewards that are inconsistent with expert behaviors, and to enable the preservation of useful information in partially correct rewards. Under this setup, we can model the offline policy learning problem as a bi-level optimization problem, with the upper layer minimizing the reward gap and the lower layer solving a pessimistic RL problem. To derive a tractable algorithm, we leverage Lagrangian duality to eliminate the requirement of online samples. Finally, we present a practical implementation of RGM based on stochastic first-order two-timescale optimization and extract the policy during the course of reward learning.

4.1 DEFINITION OF REWARD GAP

Previous approaches that tackle imperfect rewards typically resort to inverse reward design (Hadfield-Menell et al., 2017) or inverse RL (Ng et al., 2000; Abbeel & Ng, 2004), with the final goal of learning an explicit reward function that is well-aligned with the task. These algorithms, however, heavily rely on online interactions and are expensive to implement. Moreover, as observed in recent literature, there exist some tasks that cannot be captured by a numerical Markov reward function (Abel et al., 2021). Hence, learning an explicit proxy of the perfect reward function and comparing it to the given rewards is unlikely the best option to characterize the reward gap. In this study, we define the reward gap based on the outcome of the learned agent behavior, i.e., from the perspective of discounted state-action visitation distribution matching.

Definition 1. *(Reward gap) Given an arbitrary reward function $\hat{r}(s,a)$ and the discounted state-action visitation distribution d^* of the optimal policy induced from the perfect rewards r, the reward gap between $\hat{r}$ and r is defined as*

$$D_f\left(d^{\pi^*_{\hat{r}}} \| d^*\right) \tag{5}$$

where $D_f\left(p\|q\right) = \mathbb{E}_{z\sim q}\left[f\left(\frac{p(z)}{q(z)}\right)\right]$ is the f-divergence between distributions p and q, and $d^{\pi^*_{\hat{r}}}$ represents the discounted state-action visitation distribution induced by $\pi^*_{\hat{r}}$, which is derived using Eq. (4) with the reward function $\hat{r}$.

Note that d^* is generally unobtainable since the perfect reward function is unknown. We can alternatively use the discounted state-action visitation distribution d^E induced by unknown π^E in expert demonstrations $\mathcal{D}^E$ to approximate d^*. In the next sub-section, we discuss how to adjust $\hat{r}$ to achieve the goal of reward gap minimization.

4.2 BI-LEVEL OPTIMIZATION

Reward Correction. In our study, we consider $\hat{r}(s,a) := \tilde{r}(s,a) + \Delta r(s,a,\tilde{r})$, where $\Delta r(s,a,\tilde{r})$ is a learnable reward correction term that is correlated with the given imperfect rewards $\tilde{r}$ in $\mathcal{D}$. The introduction of $\Delta r(s,a,\tilde{r})$ enables us to exploit useful information within the partially correct rewards, while also correcting the wrong or inconsistent reward signals. We can further use it to construct a bi-level optimization formulation for RGM, where the upper-level problem optimizes the reward correction term to minimize the f-divergence between $d^{\pi^*_{\hat{r}}}$ and d^E, and the lower-level problem solves $\pi^*_{\hat{r}}$ as the optimal policy of a pessimistic RL problem with the corrected rewards:

$$\Delta r^* = \arg\min_{\Delta r} \mathrm{D}_f\left(d^{\pi^*_{\hat{r}}} \| d^E\right) \tag{6}$$

$$\text{s.t.} \quad \pi^*_{\hat{r}} = \arg\max_{\pi} \mathbb{E}_{(s,a)\sim d^{\pi_{\hat{r}}}}\left[\hat{r}(s,a)\right] - \alpha \mathrm{D}_f\left(d^{\pi_{\hat{r}}} \| d^{\mathcal{D}}\right) \tag{7}$$

The above bi-level optimization formulation poses several technical difficulties, stemming from the complexity of deriving $d^{\pi^*_{\hat{r}}}$ from $\pi^*_{\hat{r}}$, as well as the requirement of online samples from $d^{\pi^*_{\hat{r}}}$, which is

impossible under the offline setting. In the following, we present reformulations for both lower and upper level problems, which leads to a tractable form and an easy-to-implement algorithm.

Reformulation of Lower Level Problem. We first focus on reformulating the lower level problem by exploiting the Bellman flow constraint (Puterman, 2014) and the duality of the problem.

Definition 2. *(Bellman flow constraint) Let $\mathcal{T}_\star d(s) = \sum_{\bar{s},\bar{a}} T(s|\bar{s},\bar{a})d(\bar{s},\bar{a})$ denote the transpose (or adjoint) transition operator, then the Bellman flow constraint for the discounted state-action visitation distribution $d(s,a)$ is given as:*

$$\sum_a d(s,a) = (1-\gamma)\mu_0(s) + \gamma\mathcal{T}_\star d(s), \forall s \in \mathcal{S} \tag{8}$$

If $d(s,a) \geq 0$ satisfies the Bellman flow constraint, then $d(s,a)$ is feasible and there is a one-to-one correspondence between d and the related policy π: i.e., d is the only discounted state-action visitation distribution for policy $\pi(a|s) = \frac{d(s,a)}{\sum_{\bar{a}} d(s,\bar{a})}$, while π is the only policy whose discounted state-action visitation distribution is d (for detailed proof see Puterman (2014)). Leveraging the Bellman flow constraint and the one-to-one correspondence property, the lower level problem Eq. (7) can be re-written to a constraint maximization problem w.r.t. d in place of $\pi_{\hat{r}}$:

$$\begin{aligned} d^{\pi^*_{\hat{r}}} = & \arg\max_{d\geq 0} \mathbb{E}_{(s,a)\sim d}[\hat{r}(s,a)] - \alpha \mathrm{D}_f\left(d \| d^{\mathcal{D}}\right) \\ \text{s.t.} \quad & \sum_a d(s,a) = (1-\gamma)\mu_0(s) + \gamma\mathcal{T}_\star d(s), \forall s \in S \end{aligned} \tag{9}$$

The Lagrange dual problem of Eq. (9) is as follow:

$$\min_{V(s)} \max_{d\geq 0} \mathbb{E}_{(s,a)\sim d}[\hat{r}(s,a)] - \alpha \mathrm{D}_f\left(d \| d^{\mathcal{D}}\right) + \sum_s V(s)\left[(1-\gamma)\mu_0(s) + \gamma\mathcal{T}_\star d(s) - \sum_a d(s,a)\right] \tag{10}$$

where $V(s)$ are Lagrange multipliers. Note that the primal problem Eq. (9) is convex w.r.t. d, and under a mild assumption (see Assumption 1 in Appendix A.2), the Slater's condition (Boyd et al., 2004) holds, which means by strong duality, we can solve the original primal problem by solving Eq. (10). After rearranging the terms, Eq. (10) can be equivalently written as the following form (see Lemma 2 in Appendix A.2 for detailed deduction):

$$\min_{V(s)} \max_{d\geq 0} (1-\gamma)\mathbb{E}_{s\sim\mu_0}[V(s)] + \mathbb{E}_{(s,a)\sim d}\left[\hat{r}(s,a) + \gamma\mathcal{T}V(s,a) - V(s)\right] - \alpha \mathrm{D}_f\left(d \| d^{\mathcal{D}}\right) \tag{11}$$

in which $\mathcal{T}V(s,a) = \sum_{s'} T(s'|s,a)V(s')$ denotes the transition operator. Next, by exploiting the Fenchel conjugate, we can further transform the minimax problem Eq. (11) into a tractable single-level unconstrained minimization problem (see Proposition 1 in Appendix A.2 for detailed derivation), which eliminates the requirement of online samples:

$$\min_{V(s)} (1-\gamma)\mathbb{E}_{s\sim\mu_0}[V(s)] + \alpha\, \mathbb{E}_{(s,a)\sim d^{\mathcal{D}}}\left[f_\star\left(\frac{\hat{r}(s,a) + \gamma\mathcal{T}V(s,a) - V(s)}{\alpha}\right)\right] \tag{12}$$

where $f_\star$ is the Fenchel conjugate of f. In the above formulation, the Lagrange multipliers $V(s)$ can be equivalently perceived as some sort of state-value functions, which can be learned and optimized via a parameterized neural network, similar to the treatment used in the DICE-family of RL algorithms (Nachum et al., 2019a; Nachum & Dai, 2020).

Reformulation of Upper Level Problem. Using the property of Fenchel conjugate, we show that the optimal d^* and V^* from the lower level problem satisfy the following ideal relationship (see Proposition 2 in Appendix A.3 for details):

$$\frac{d^{\pi^*_{\hat{r}}}(s,a)}{d^{\mathcal{D}}(s,a)} = f'_\star\left(\frac{\hat{r}(s,a) + \gamma\mathcal{T}V^*(s,a) - V^*(s)}{\alpha}\right) \tag{13}$$

Plugging the above equation into Eq. (7), we can obtain a new objective for the upper level problem:

$$\Delta r^* = \arg\min_{\Delta r} \mathrm{D}_f\left(f'_\star\left(\frac{\hat{r} + \gamma\mathcal{T}V^* - V^*}{\alpha}\right) d^{\mathcal{D}} \| d^E\right) \tag{14}$$

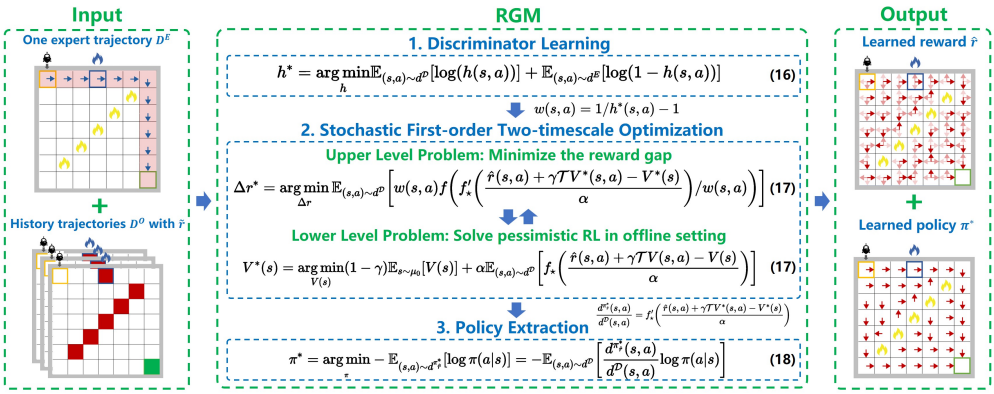


Figure 1: Illustration of the reformulated bi-level optimization problem.

For simplicity, we denote $f_\star'\left(\frac{\hat{r}+\gamma\mathcal{T}V^*-V^*}{\alpha}\right)$ as g. By expanding the f-divergence, we have:

$$\mathrm{D}_f\left(d^{\mathcal{D}}g \| d^E\right) = \mathbb{E}_{(s,a)\sim d^E}\left[f\left(\frac{d^{\mathcal{D}}(s,a)g(s,a)}{d^E(s,a)}\right)\right] = \mathbb{E}_{(s,a)\sim d^{\mathcal{D}}}\left[\frac{d^E(s,a)}{d^{\mathcal{D}}(s,a)} f\left(\frac{d^{\mathcal{D}}(s,a)}{d^E(s,a)} g(s,a)\right)\right] \tag{15}$$

The above objective involves computing the distribution ratio $w(s,a) \triangleq d^E(s,a)/d^{\mathcal{D}}(s,a)$. In the tabular case, we can empirically estimate $w(s,a) = \frac{\sum_{(\bar{s},\bar{a})\in\mathcal{D}^E} \mathbf{1}(\bar{s}=s,\bar{a}=a)/N^E}{\sum_{(\bar{s},\bar{a})\in\mathcal{D}} \mathbf{1}(\bar{s}=s,\bar{a}=a)/N}$. But in the continuous state-action settings, estimating the distribution ratio w using only samples from $d^{\mathcal{D}}$ and d^E becomes a challenge. Inspired by previous studies (Goodfellow et al., 2020; Ma et al., 2022), we instead train a discriminator $h : S \times A \to (0,1)$ to infer if (s,a) samples are from $\mathcal{D}^E$ or not:

$$h^* = \arg\min_h \mathbb{E}_{(s,a)\sim d^{\mathcal{D}}}\left[\log(h(s,a))\right] + \mathbb{E}_{(s,a)\sim d^E}\left[\log(1-h(s,a))\right] \tag{16}$$

where the optimal discriminator is $h^*(s,a) = \frac{d^{\mathcal{D}}(s,a)}{d^{\mathcal{D}}(s,a)+d^E(s,a)}$ (Goodfellow et al., 2020). We can optimize the above objective to obtain the optimal h^*, and further recover $w(s,a) = 1/h^*(s,a) - 1$.

Finally, combining all the reformulations, the final tractable form of the original bi-level optimization problem Eq. (6)-(7) is given as follows:

$$\begin{aligned} \Delta r^* &= \arg\min_{\Delta r} \mathbb{E}_{(s,a)\sim d^{\mathcal{D}}}\left[w(s,a) f\left(f_\star'\left(\frac{\hat{r}(s,a)+\gamma\mathcal{T}V^*(s,a)-V^*(s)}{\alpha}\right)/w(s,a)\right)\right] \\ \text{s.t. } V^*(s) &= \arg\min_{V(s)} (1-\gamma)\mathbb{E}_{s\sim\mu_0}[V(s)] + \alpha\, \mathbb{E}_{(s,a)\sim d^{\mathcal{D}}}\left[f_\star\left(\frac{\hat{r}(s,a)+\gamma\mathcal{T}V(s,a)-V(s)}{\alpha}\right)\right] \end{aligned} \tag{17}$$

Policy Extraction. With the learned reward correction term $\Delta r(s,a,\tilde{r})$, we can in principle use existing offline RL algorithms to learn the policy with the corrected rewards. However, this implicates additional policy evaluation and policy improvement steps. A more elegant way is to extract the policy through weighted BC as follows, which is substantially more robust and less expensive:

$$\pi^* = \arg\min_\pi -\mathbb{E}_{(s,a)\sim d^{\pi^*_{\hat{r}}}}[\log \pi(a|s)] = -\mathbb{E}_{(s,a)\sim d^{\mathcal{D}}}\left[\frac{d^{\pi^*_{\hat{r}}}(s,a)}{d^{\mathcal{D}}(s,a)} \log \pi(a|s)\right] \tag{18}$$

where $\frac{d^{\pi^*_{\hat{r}}}(s,a)}{d^{\mathcal{D}}(s,a)}$ can be calculated from Eq. (13).

4.3 Practical Implementation

There exists a few classical algorithms (Colson et al., 2007; Sinha et al., 2017) to solve bi-level optimization problems. However, they typically require the problem to be mathematically well-behaved and these methods do not scale well with the problem size. In our implementation, we use the stochastic first-order two-timescale optimization technique (Borkar, 1997), which has been successfully applied in several RL algorithms (Hong et al., 2020; Cheng et al., 2022). Specifically, we make the gradient update step size of the upper level problem much smaller than the one of

Table 1: Average normalized scores of RGM compared with offline IL and RL baselines on D4RL datasets. The scores are from the final 10 evaluations with 5 seeds. (T) means policy optimization with true rewards. "-r","-m","-m-r", and "-m-e" are short for random, medium, medium-replay, and medium-expert, respectively. (B) means only imperfect rewards are available. We obtain the results via running author-provided open-source code, and some scores are reported from TD3+BC (Fujimoto & Gu, 2021) and IQL (Kostrikov et al., 2021b) papers. The top 2 scores for each dataset under imperfect rewards are marked in blue.

D4RL Dataset	Offline IL			Offline RL			RGM (T/B)
	BC	DWBC	SMODICE	TD3+BC (T/B)	IQL (T/B)	CQL (T/B)	
hopper-r	4.9	23.9 ±2.7	5.9±4.6	8.5±0.6 / 13.3±14.9	7.9±0.4 / 1.3±0.2	8.3±0.2 / 1.7±0.9	29.7±1.6 / 21.2 ±0.4
halfcheetah-r	0.2	2.0±0.9	2.6 ±1.0	11.0±1.1 / -17.1±4.0	11.2±2.9 / 2.2 ±0.0	20.0±0.4 / -0.4±0.4	0.2±0.3 / 0.2±0.0
walker2d-r	1.7	68.3 ±13.2	-0.2±0.1	1.6±1.7 / 0.8±0.9	5.9±0.5 / 0.3±0.1	8.3±0.1 / 0.1±0.2	3.9±5.3 / 7.7 ±3.3
hopper-m	52.9	16.5±3.2	54.5±4.0	59.3±4.2 / 13.7±12.9	66.2±5.7 / 34.0±7.6	58.5±2.1 / 56.4 ±5.8	56.2±1.2 / 55.5 ±1.0
halfcheetah-m	42.6	8.2±4.1	42.9 ±0.9	48.3±0.3 / 35.2±1.4	47.4±0.2 / 42.0±0.5	44.0±5.4 / 43.5 ±1.0	40.4±1.4 / 40.7±1.4
walker2d-m	75.3	18.8±14.9	1.0±1.5	83.7±2.1 / 30.1±24.2	78.3±8.7 / 68.9±4.4	72.5±0.8 / 71.1±3.6	73.3±0.1 / 72.3 ±10.7
hopper-m-r	18.1	21.4±2.3	20.4±5.3	60.9±18.8 / 23.5 ±10.0	94.7±8.6 / 0.7±0.0	95.0±6.4 / 11.5±4.2	60.3±2.3 / 59.1 ±15.3
halfcheetah-m-r	36.6	9.2±3.1	37.1 ±3.3	44.6±0.5 / 31.8±4.0	44.2±1.2 / 18.1±13.4	45.5±0.5 / 16.5±20.0	37.7±2.7 / 37.8 ±2.6
walker2d-m-r	26.0	56.6 ±17.8	41.1±33.5	81.8±5.5 / 7.8±2.3	73.8±7.1 / 4.9±7.3	77.2±5.5 / 17.4±3.9	46.3±9.9 / 48.6 ±3.6
hopper-m-e	52.5	16.5±3.2	75.4 ±2.6	98.0±9.4 / 50.8±40.8	91.5±14.3 / 49.3±22.7	105.4±6.8 / 68.3±13.9	106.1±1.2 / 87.1 ±10.7
halfcheetah-m-e	55.2	0.0±0.8	88.2 ±4.5	90.7±4.3 / 35.3±6.8	86.7±5.3 / 53.4±6.8	91.6±2.8 / 64.8±4.0	85.6±6.5 / 81.5 ±0.9
walker2d-m-e	107.5	54.3±21.5	29.8±33.9	110.1±0.5 / 44.7±35.4	109.6±1.0 / 108.3±0.7	108.8±0.7 / 75.4±7.5	108.3±0.3 / 108.8 ±0.4
Mean Score	39.5	22.6	33.2	58.2 / 24.6	59.8 / 32.0	60.5 / 35.5	54.1 / 52.0

the lower level problem (see Figure 1 for RGM framework. Refer to Appendix B for additional implementation details of RGM with KL-divergence). In the following section, we will demonstrate through empirical experiments the efficacy of the the two-timescale optimization techniques for our bi-level optimization solution.

5 Experiments

In this section, we present empirical evaluations of RGM under imperfect reward settings[1]. We first evaluate RGM against existing methods on D4RL-v2 (Fu et al., 2020) benchmark datasets. We then provide an in-depth property analysis of the learned corrected rewards through both illustrative examples and ablations. Considering expert data collections may be costly in most real-world applications, we use **only 1 expert trajectory** (serving as $\mathcal{D}^E$) in all our experiments for RGM.

5.1 Comparison with Offline RL

We train RGM and SOTA offline RL methods (including TD3+BC (Fujimoto & Gu, 2021), IQL (Florence et al., 2022) and CQL (Kumar et al., 2020)) under imperfect corrupted rewards and report their performances evaluated based on the the perfect rewards[2] in Table 1. Since offline RL methods are not specifically designed to handle imperfect rewards thus suffer from severe performance drop (illustrated in Figure 4), we also report their performance with perfect rewards.

Table 1 shows that RGM surpasses offline RL methods under imperfect rewards by a large margin and achieves similar performance to offline RL policies that are trained on perfect rewards. This shows a remarkable advantage of RGM as it remedies the negative effects of imperfect rewards using only one expert trajectory, which can be particularly useful for a wide range of real-world scenarios. It removes the restrictive requirements on perfect rewards from offline policy optimization and eliminates severe performance degeneration when perfect rewards are unattainable.

5.2 Comparison with Offline IL

We compare RGM with BC and SOTA offline IL methods (DWBC (Xu et al., 2022b) and SMODICE (Ma et al., 2022)) that can learn from mixed-quality data. Only offline IL methods that tackle imperfect rewards are considered as baselines, because other existing methods such as reward shaping can only be applied to online settings (see Section 2 for detailed discussions).

[1] 50% of rewards in the original D4RL datasets are corrupted via flipping the signs.

[2] We assume the reward functions in D4RL datasets are perfect and evaluate based on these perfect rewards.

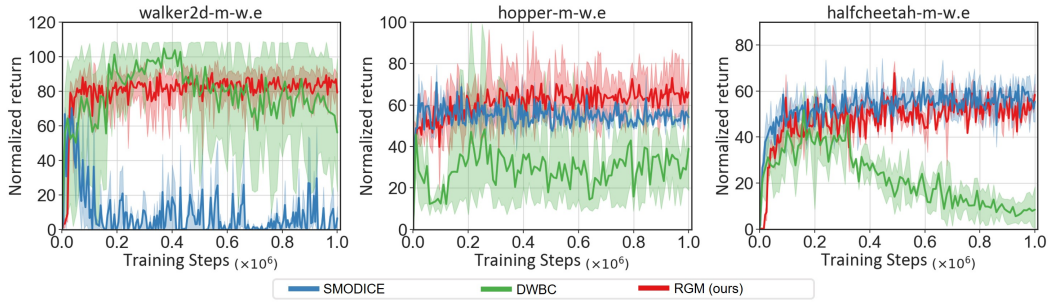
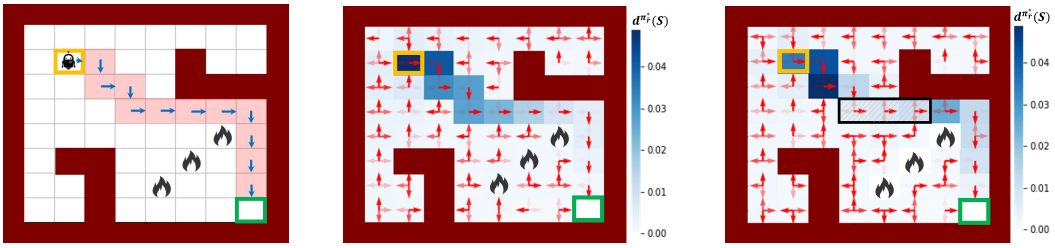


Figure 2: Comparison under offline IL settings, where the non-expert dataset $\mathcal{D}^O$ already contains plenty of expert trajectories. The note "-w.e" stands for the mixed dataset that combines the original D4RL dataset with many expert trajectories (see Table 6 in Appendix E for additional results).



(a) Expert demonstration (b) Results of zero $\tilde{r}$ (c) Results of partially correct $\tilde{r}$

Figure 3: Learned rewards $\hat{r}$ and optimal distribution $d^{\pi^*_{\hat{r}}}$ trained on two types of imperfect rewards $\tilde{r}$. The opacity of each square is determined by the marginal state distribution $d^{\pi^*_{\hat{r}}}(s)$. The opacity of the arrow shows the learned reward $\hat{r}$, where the darkest arrow points to the direction of the highest reward. The expert starts from, follows the path and arrow to reach the goal. $\tilde{r}$ in (b) is +10 at the goal and is zero at other states. $\tilde{r}$ in (c) falsely punishes the agent on and correctly punishes the RL agent on a set of fires.

To align with our setting, we train offline IL baselines based on the joint dataset $\mathcal{D} = \mathcal{D}^O \cup \mathcal{D}^{E_1}$, where $\mathcal{D}^O$ is the original D4RL dataset and $\mathcal{D}^{E_1}$ contains only one expert trajectory. It is worth mentioning that DWBC and SMODICE both build on the strong assumption that $\mathcal{D}^O$ already covers a large expert dataset $\mathcal{D}^{E_2}$[3], which is a rare case in real scenarios. As a result, we can see from Table 1 that these two methods suffer from inferior performance when the restrictive requirements about full coverage of expert distribution are not satisfied. RGM, however, performs well when nearly no expert trajectories are contained in the non-expert dataset, because RGM is optimizing a RL objective that dictates relatively relaxed requirements on the quality of dataset.

To further illustrate the superiority of RGM, we compare RGM with DWBC and SMODICE under their settings by adding 100∼200 expert trajectories into $\mathcal{D}^O$. Results show that RGM can still outperform SOTA offline IL methods by a large margin (see Figure 2 and Table 6 in Appendix E).

5.3 Investigations on Learned Reward Correction Term

Benefits of Learned Rewards. We investigate the potential benefits of the learned rewards via demonstrative experiments in a 8×8 grid world, and observe that the learned rewards enjoy three desirable properties: 1) *encode long horizon information*; 2) *correct wrong rewards against expert preference*; and 3) *retrieve useful information from existing rewards*, as shown in Figure 3.

Figure 3b shows that the learned rewards not only perceive correct learning signals on expert paths, but also generalize well on non-expert paths. We can successfully navigate to the destination at most positions by only maximizing one-step reward, meaning the learned rewards encode long horizon information. However, the learned rewards may lead the agent to collide the fire. By contrast, Figure 3c shows that the learned rewards can avoid the fire by retrieving useful information in imperfect $\tilde{r}$, meanwhile correcting the wrong signals against expert preference. All three desirable

[3] $\mathcal{D}^{E_2}$ is a much larger expert dataset compared to D^{E_1}.

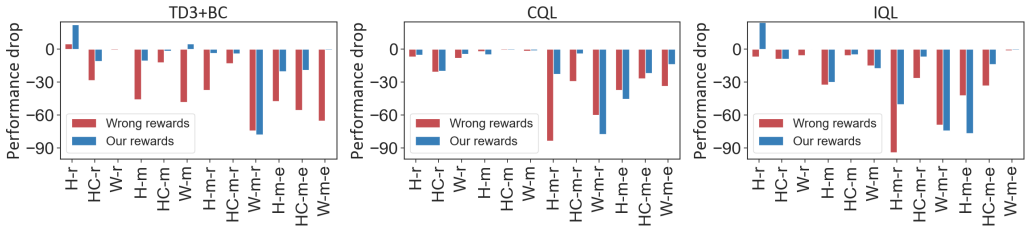
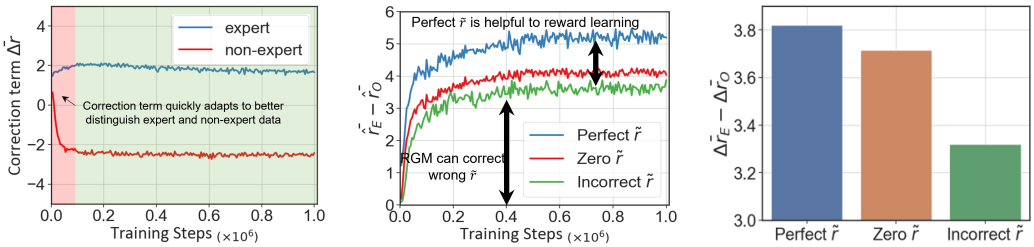


Figure 4: Performance drop of normalized returns of SOTA offline RL methods under imperfect and RGM corrected rewards. *H*: Hopper; *HC*: HalfCheetah; *W*: Walker2d. *r*: random; *m*: medium; *m-r*: medium-replay; *m-e*: medium-expert. Severe performance drop shows that offline RL is largely affected given imperfect rewards. The corrected rewards learned by RGM, however, can largely overcome the negative effect of imperfect rewards.



(a) Learning curve of Δr (b) Effect of $\tilde{r}$ on $\hat{r}$ (c) Effect of $\tilde{r}$ on Δr

Figure 5: Experiments on learned rewards in hopper-m-r task. The superscript "$^-$" denotes the mean value of mini-batch samples. The subscript "E" and "O" denote the value on expert and non-expert data. In (b)(c), large $\hat{\bar{r}}_E - \hat{\bar{r}}_O$ and $\Delta\bar{r}_E - \Delta\bar{r}_O$ indicate that expert and non-expert data are clearly distinguishable according to the learned rewards, and small values mean the opposite. RGM can remedy the negative impacts of incorrect rewards. Perfect rewards are beneficial to reward learning, while incorrect rewards are quite the opposite.

properties are hard to achieve in existing methods, which commonly depend on discriminator-related rewards only[4].

Offline RL with Corrected Rewards. We highlight that the learned corrected rewards $\hat{r}$ obtained by RGM can also be utilized in other offline RL approaches. To be mentioned, the corrected rewards are optimized based on the specific α in Eq. (7), hence may not be optimal to other offline RL methods. Whereas, Figure 4 shows that in D4RL experiments, the corrected rewards largely remedy most of the negative effects of the imperfect rewards and even surpass perfect rewards in some datasets.

Ablations on Learned Rewards. Additionally, we investigate the learned rewards in high-dimensional MDPs with continuous state-action spaces via monitoring the learning dynamics of both the reward correction term Δr and the final learned rewards $\hat{r}$. Figure 5a shows that the reward correction term Δr initially cannot distinguish expert and non-expert data well, but adapts and converges quickly. After a few training steps, Δr can reward expert data and punish non-expert very well. We also perform ablation on the effect of diverse types of imperfect rewards $\tilde{r}$ on Δr and $\hat{r}$. Figure 5b shows that a perfect $\tilde{r}$ is beneficial in obtaining a clear classification, and an incorrect $\tilde{r}$ can be counterproductive. Whereas, RGM can correct the wrong rewards by a large margin and gives reasonable rewards. Similar effects can also be observed on Δr, as Figure 5c shows.

6 Discussion and Conclusion

In this paper, we propose RGM (Reward Gap Minimization), a unified offline policy optimization approach for diverse settings of imperfect rewards. RGM is formulated as a bi-level optimization problem, which achieves reward correction and simultaneous policy learning in a fully offline paradigm. Extensive experiments and illustrative examples show that RGM can perform robust policy optimization under imperfect rewards. Several desirable properties are also identified in the corrected rewards learned by RGM. One limitation of RGM is the need for a small expert dataset, which may not be easily accessible in some applications. However, RGM relaxes the strong dependencies on online reward tuning to obtain high-quality rewards, without relying on diverse expert demonstrations and tedious human efforts, which renders it a powerful tool to solve many real-world problems.

[4]SMODICE and DEMODICE treat $\log \frac{d^E}{d^D}$ as rewards, which can only provide correct signals when $d^E > 0$.

REFERENCES

Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, pp. 1, 2004.

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

Gaon An, Seungyong Moon, Jang-Hyun Kim, and Hyun Oh Song. Uncertainty-based offline reinforcement learning with diversified q-ensemble. *Advances in neural information processing systems*, 34:7436–7447, 2021.

Chenjia Bai, Lingxiao Wang, Zhuoran Yang, Zhi-Hong Deng, Animesh Garg, Peng Liu, and Zhaoran Wang. Pessimistic bootstrapping for uncertainty-driven offline reinforcement learning. In *International Conference on Learning Representations*, 2021.

Vivek S Borkar. Stochastic approximation with two time scales. *Systems & Control Letters*, 29(5): 291–294, 1997.

Stephen Boyd, Stephen P Boyd, and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

Ching-An Cheng, Tengyang Xie, Nan Jiang, and Alekh Agarwal. Adversarially trained actor critic for offline reinforcement learning. In *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pp. 3852–3878. PMLR, 2022.

Nuttapong Chentanez, Andrew Barto, and Satinder Singh. Intrinsically motivated reinforcement learning. *Advances in neural information processing systems*, 17, 2004.

Benoît Colson, Patrice Marcotte, and Gilles Savard. An overview of bilevel optimization. *Annals of operations research*, 153(1):235–256, 2007.

Mark Cutler and Jonathan P How. Autonomous drifting using simulation-aided reinforcement learning. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5442–5448. IEEE, 2016.

Bo Dai, Niao He, Yunpeng Pan, Byron Boots, and Le Song. Learning from conditional distributions via dual embeddings. In *Artificial Intelligence and Statistics*, pp. 1458–1467. PMLR, 2017.

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

Marco Dorigo and Marco Colombetti. Robot shaping: Developing autonomous agents through learning. *Artificial intelligence*, 71(2):321–370, 1994.

Pete Florence, Corey Lynch, Andy Zeng, Oscar A Ramirez, Ayzaan Wahid, Laura Downs, Adrian Wong, Johnny Lee, Igor Mordatch, and Jonathan Tompson. Implicit behavioral cloning. In *Conference on Robot Learning*, pp. 158–168. PMLR, 2022.

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. *Advances in Neural Information Processing Systems*, 34, 2021.

Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *International Conference on Machine Learning*, pp. 2052–2062. PMLR, 2019.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

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, 2018.

Dylan Hadfield-Menell, Smitha Milli, Pieter Abbeel, Stuart J Russell, and Anca Dragan. Inverse reward design. *Advances in neural information processing systems*, 30, 2017.

Mingyi Hong, Hoi-To Wai, Zhaoran Wang, and Zhuoran Yang. A two-timescale framework for bilevel optimization: Complexity analysis and application to actor-critic. *arXiv preprint arXiv:2007.05170*, 2020.

Yujing Hu, Weixun Wang, Hangtian Jia, Yixiang Wang, Yingfeng Chen, Jianye Hao, Feng Wu, and Changjie Fan. Learning to utilize shaping rewards: A new approach of reward shaping. *Advances in Neural Information Processing Systems*, 33:15931–15941, 2020.

Max Jaderberg, Wojciech M Czarnecki, Iain Dunning, Luke Marris, Guy Lever, Antonio Garcia Castaneda, Charles Beattie, Neil C Rabinowitz, Ari S Morcos, Avraham Ruderman, et al. Human-level performance in 3d multiplayer games with population-based reinforcement learning. *Science*, 364(6443):859–865, 2019.

Firas Jarboui and Vianney Perchet. Offline inverse reinforcement learning. *arXiv preprint arXiv:2106.05068*, 2021.

Geon-Hyeong Kim, Seokin Seo, Jongmin Lee, Wonseok Jeon, HyeongJoo Hwang, Hongseok Yang, and Kee-Eung Kim. Demodice: Offline imitation learning with supplementary imperfect demonstrations. In *International Conference on Learning Representations*, 2021.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR (Poster)*, 2015.

Ilya Kostrikov, Ofir Nachum, and Jonathan Tompson. Imitation learning via off-policy distribution matching. In *International Conference on Learning Representations*, 2019.

Ilya Kostrikov, Rob Fergus, Jonathan Tompson, and Ofir Nachum. Offline reinforcement learning with fisher divergence critic regularization. In *International Conference on Machine Learning*, pp. 5774–5783. PMLR, 2021a.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. In *International Conference on Learning Representations*, 2021b.

Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. *Advances in Neural Information Processing Systems*, 32, 2019.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33:1179–1191, 2020.

Jongmin Lee, Wonseok Jeon, Byungjun Lee, Joelle Pineau, and Kee-Eung Kim. Optidice: Offline policy optimization via stationary distribution correction estimation. In *International Conference on Machine Learning*, pp. 6120–6130. PMLR, 2021.

Jianxiong Li, Xianyuan Zhan, Haoran Xu, Xiangyu Zhu, Jingjing Liu, and Ya-Qin Zhang. Distance-sensitive offline reinforcement learning. *arXiv preprint arXiv:2205.11027*, 2022.

Xiao Li, Zachary Serlin, Guang Yang, and Calin Belta. A formal methods approach to interpretable reinforcement learning for robotic planning. *Science Robotics*, 4(37):eaay6276, 2019.

Yecheng Ma, Andrew Shen, Dinesh Jayaraman, and Osbert Bastani. Versatile offline imitation from observations and examples via regularized state-occupancy matching. In *International Conference on Machine Learning*, pp. 14639–14663. PMLR, 2022.

Ofir Marom and Benjamin Rosman. Belief reward shaping in reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Bhaskara Marthi. Automatic shaping and decomposition of reward functions. In *Proceedings of the 24th International Conference on Machine learning*, pp. 601–608, 2007.

Ofir Nachum and Bo Dai. Reinforcement learning via fenchel-rockafellar duality. *arXiv preprint arXiv:2001.01866*, 2020.

Ofir Nachum, Yinlam Chow, Bo Dai, and Lihong Li. Dualdice: Behavior-agnostic estimation of discounted stationary distribution corrections. *Advances in Neural Information Processing Systems*, 32, 2019a.

Ofir Nachum, Bo Dai, Ilya Kostrikov, Yinlam Chow, Lihong Li, and Dale Schuurmans. Algaedice: Policy gradient from arbitrary experience. *arXiv preprint arXiv:1912.02074*, 2019b.

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

Andrew Y Ng, Stuart Russell, et al. Algorithms for inverse reinforcement learning. In *Icml*, volume 1, pp. 2, 2000.

Haoyi Niu, Sharma Shubham, Yiwen Qiu, Ming Li, Guyue Zhou, Jianming Hu, and Xianyuan Zhan. When to trust your simulator: Dynamics-aware hybrid offline-and-online reinforcement learning. In *Advances in Neural Information Processing Systems*, 2022.

Art B. Owen. *Monte Carlo theory, methods and examples*. 2013.

Dean A Pomerleau. Alvinn: An autonomous land vehicle in a neural network. *Advances in neural information processing systems*, 1, 1988.

Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

Nived Rajaraman, Lin Yang, Jiantao Jiao, and Kannan Ramchandran. Toward the fundamental limits of imitation learning. *Advances in Neural Information Processing Systems*, 33:2914–2924, 2020.

Jette Randløv and Preben Alstrøm. Learning to drive a bicycle using reinforcement learning and shaping. In *ICML*, volume 98, pp. 463–471. Citeseer, 1998.

R Tyrrell Rockafellar and Roger J-B Wets. *Variational analysis*, volume 317. Springer Science & Business Media, 2009.

Fumihiro Sasaki and Ryota Yamashina. Behavioral cloning from noisy demonstrations. In *International Conference on Learning Representations*, 2020.

Ankur Sinha, Pekka Malo, and Kalyanmoy Deb. A review on bilevel optimization: from classical to evolutionary approaches and applications. *IEEE Transactions on Evolutionary Computation*, 22 (2):276–295, 2017.

Yang Song and Diederik P Kingma. How to train your energy-based models. *arXiv preprint arXiv:2101.03288*, 2021.

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

Hao Sun, Lei Han, Rui Yang, Xiaoteng Ma, Jian Guo, and Bolei Zhou. Exploiting reward shifting in value-based deep rl. In *Advances in Neural Information Processing Systems*, 2022.

Yue Wu, Shuangfei Zhai, Nitish Srivastava, Joshua M Susskind, Jian Zhang, Ruslan Salakhutdinov, and Hanlin Goh. Uncertainty weighted actor-critic for offline reinforcement learning. In *International Conference on Machine Learning*, pp. 11319–11328. PMLR, 2021.

Yueh-Hua Wu and Shou-De Lin. A low-cost ethics shaping approach for designing reinforcement learning agents. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Haoran Xu, Xianyuan Zhan, Jianxiong Li, and Honglei Yin. Offline reinforcement learning with soft behavior regularization. *arXiv preprint arXiv:2110.07395*, 2021.

Haoran Xu, Jiang Li, Jianxiong Li, and Xianyuan Zhan. A policy-guided imitation approach for offline reinforcement learning. In *Advances in Neural Information Processing Systems*, 2022a.

Haoran Xu, Xianyuan Zhan, Honglei Yin, and Huiling Qin. Discriminator-weighted offline imitation learning from suboptimal demonstrations. In *International Conference on Machine Learning*, pp. 24725–24742. PMLR, 2022b.

Haoran Xu, Xianyuan Zhan, and Xiangyu Zhu. Constraints penalized q-learning for safe offline reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2022c.

Xianyuan Zhan, Haoran Xu, Yue Zhang, Xiangyu Zhu, Honglei Yin, and Yu Zheng. Deepthermal: Combustion optimization for thermal power generating units using offline reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2022.

Wenjia Zhang, Haoran Xu, Haoyi Niu, Peng Cheng, Ming Li, Heming Zhang, Guyue Zhou, and Xianyuan Zhan. Discriminator-guided model-based offline imitation learning. In *Conference on Robot Learning*, 2022.

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

Konrad Zolna, Alexander Novikov, Ksenia Konyushkova, Caglar Gulcehre, Ziyu Wang, Yusuf Aytar, Misha Denil, Nando de Freitas, and Scott Reed. Offline learning from demonstrations and unlabeled experience. *arXiv preprint arXiv:2011.13885*, 2020.

A PROOFS

A.1 BACKGROUND

We begin by briefly introducing the Fenchel conjugate (also known as convex conjugate or Legendre–Fenchel transformation):

Definition 3. *(Fenchel conjugate) In a real Hilbert space $\mathcal{X}$, if a function $f(x)$ is proper, then the Fenchel conjugate $f_\star$ of f at y is:*

$$f_\star(y) = \sup_{x \in \mathcal{X}} (y^T x - f(x)) \tag{19}$$

where the domain of the $f_\star(y)$ is given by:

$$\operatorname{dom} f_\star = \left\{ y : \sup_{x \in \operatorname{dom} f} \left(y^T x - f(x) \right) < \infty \right\} \tag{20}$$

If f is convex and lower semi-continuous as well, we have the duality $f_{\star\star}(x) = f(x)$. Furthermore, if f is also differentiable, then the maximizer x^* of $f_\star(y)$ satisfies:

$$x^* = f'_\star(y) \tag{21}$$

Next we present the interchangeability principle, which plays a key role in Proposition 1.

Lemma 1. *(Interchangeability principle) Let ξ be a random variable on Ξ and assume for any $\xi \in \Xi$, function $g(\cdot, \xi)$ is a proper and upper semi-continuous concave function. Then*

$$\mathbb{E}_\xi \left[\max_{u \in \mathbb{R}} g(u, \xi) \right] = \max_{u(\cdot) \in \mathcal{G}(\Xi)} \mathbb{E}_\xi [g(u(\xi), \xi)] \tag{22}$$

where $\mathcal{G}(\Xi) = \{u(\cdot) : \Xi \to \mathbb{R}\}$ is the entire space of functions defined on support Ξ.

Proof. Please refer to (Dai et al., 2017; Rockafellar & Wets, 2009). □

A.2 PROOF OF TRACTABLE TRANSFORMATION OF LOWER LEVEL PROBLEM

We start our proof from the original bi-level optimization problem Eq. (6) and Eq. (7). Using the Bellman flow constraint for Eq. (7) yields:

$$\begin{aligned} \Delta r^* = \arg\min_{\Delta r} \; & \mathrm{D}_f \left(d^{\pi^*_{\hat{r}}} \| d^E \right) \\ \text{s.t. } d^{\pi^*_{\hat{r}}} = \; & \arg\max_{d \geq 0} \; \mathbb{E}_{(s,a)\sim d}[\hat{r}(s,a)] - \alpha \mathrm{D}_f \left(d \| d^{\mathcal{D}} \right) \\ & \text{s.t. } \sum_a d(s,a) = (1-\gamma)\mu_0(s) + \gamma \mathcal{T}_\star d(s), \forall s \in S \end{aligned} \tag{23}$$

Assumption 1. *There exists at least one d such that:*

$$\sum_a d(s,a) = (1-\gamma)\mu_0(s) + \gamma \mathcal{T}_\star d(s), \; d(s) > 0, \; \forall s \in S \tag{24}$$

We note that this assumption is mild since when every state is reachable from the initial state distribution, the assumption is satisfied, which is common in practice.

The Slater's theorem (Boyd et al., 2004) states that strong duality holds, if the optimization problem is strictly feasible (Slater's condition holds) and the problem is convex. So under Assumption 1 with the fact that the lower level problem is convex w.r.t. d, the strong duality holds, which means that the above lower level problem can be re-written as the following form:

$$\min_{V(s)} \max_{d \geq 0} \; \mathbb{E}_{(s,a)\sim d}[\hat{r}(s,a)] - \alpha \mathrm{D}_f \left(d \| d^{\mathcal{D}} \right) + \sum_s V(s) \left[(1-\gamma)\mu_0(s) + \gamma \mathcal{T}_\star d(s) - \sum_a d(s,a) \right] \tag{25}$$

Lemma 2. *The minimax problem:*

$$\min_{V(s)} \max_{d \geq 0} \ \mathbb{E}_{(s,a)\sim d}[\hat{r}(s,a)] - \alpha D_f\left(d \| d^{\mathcal{D}}\right) + \sum_s V(s)\left[(1-\gamma)\mu_0(s) + \gamma \mathcal{T}_\star d(s) - \sum_a d(s,a)\right] \tag{26}$$

can be equivalently written as:

$$\min_{V(s)} \max_{d \geq 0} \ (1-\gamma)\mathbb{E}_{s\sim\mu_0}[V(s)] + \mathbb{E}_{(s,a)\sim d}\left[\hat{r}(s,a) + \gamma \mathcal{T} V(s,a) - V(s))\right] - \alpha D_f\left(d \| d^{\mathcal{D}}\right) \tag{27}$$

Proof.

$$\begin{aligned}
&\mathbb{E}_{(s,a)\sim d}[\hat{r}(s,a)] - \alpha \mathrm{D}_f\left(d \| d^{\mathcal{D}}\right) + \sum_s V(s)\left[(1-\gamma)\mu_0(s) + \gamma \mathcal{T}_\star d(s) - \sum_a d(s,a)\right] \\
&= \mathbb{E}_{(s,a)\sim d}[\hat{r}(s,a)] - \alpha \mathrm{D}_f\left(d \| d^{\mathcal{D}}\right) + \sum_s V(s)\left[(1-\gamma)\mu_0(s) + \gamma \sum_{\bar{s},\bar{a}} T(s|\bar{s},\bar{a}) d(\bar{s},\bar{a}) - \sum_a d(s,a)\right] \\
&= \sum_{s,a} d(s,a)\hat{r}(s,a) - \alpha \mathrm{D}_f\left(d \| d^{\mathcal{D}}\right) + (1-\gamma)\sum_s \mu_0(s)V(s) + \gamma \sum_{\bar{s},\bar{a}} d(\bar{s},\bar{a}) \sum_s T(s|\bar{s},\bar{a})V(s) - \sum_{s,a} d(s,a)V(s) \\
&= \sum_{s,a} d(s,a)\hat{r}(s,a) - \alpha \mathrm{D}_f\left(d \| d^{\mathcal{D}}\right) + (1-\gamma)\sum_s \mu_0(s)V(s) + \gamma \sum_{s,a} d(s,a) \sum_{s'} T(s'|s,a)V(s') - \sum_{s,a} d(s,a)V(s) \\
&= (1-\gamma)\sum_s \mu_0(s)V(s) + \sum_{s,a} d(s,a)\left[\hat{r}(s,a) + \gamma \sum_{s'} T(s'|s,a)V(s') - V(s)\right] - \alpha \mathrm{D}_f\left(d \| d^{\mathcal{D}}\right) \\
&= (1-\gamma)\mathbb{E}_{s\sim\mu_0}[V(s)] + \mathbb{E}_{(s,a)\sim d}\left[\hat{r}(s,a) + \gamma \mathcal{T} V(s,a) - V(s))\right] - \alpha \mathrm{D}_f\left(d \| d^{\mathcal{D}}\right)
\end{aligned} \tag{28}$$

□

Proposition 1. *The minimax problem:*

$$\min_{V(s)} \max_{d \geq 0} \ \mathbb{E}_{(s,a)\sim d}[\hat{r}(s,a)] - \alpha D_f\left(d \| d^{\mathcal{D}}\right) + \sum_s V(s)\left[(1-\gamma)\mu_0(s) + \gamma \mathcal{T}_\star d(s) - \sum_a d(s,a)\right] \tag{29}$$

shares the same optimal value as the following minimization problem:

$$\min_{V(s)} (1-\gamma)\mathbb{E}_{s\sim\mu_0}[V(s)] + \alpha\, \mathbb{E}_{(s,a)\sim d^{\mathcal{D}}}\left[f_\star\left(\frac{\hat{r}(s,a) + \gamma \mathcal{T} V(s,a) - V(s)}{\alpha}\right)\right] \tag{30}$$

where $f_\star$ is the Fenchel conjugate function of f with $\operatorname{dom} f = \{u : u \geq 0\}$

Proof. Using Lemma 2, this minimax problem can be re-written as:

$$\min_{V(s)} \max_{d \geq 0} (1-\gamma)\mathbb{E}_{s\sim\mu_0}[V(s)] + \mathbb{E}_{(s,a)\sim d}\left[\hat{r}(s,a) + \gamma \mathcal{T} V(s,a) - V(s)\right] - \alpha \mathrm{D}_f\left(d \| d^{\mathcal{D}}\right) \tag{31}$$

Next,

$$\begin{aligned}
&\min_{V(s)} \max_{d \geq 0} (1-\gamma)\mathbb{E}_{s\sim\mu_0}[V(s)] + \mathbb{E}_{(s,a)\sim d}\left[\hat{r}(s,a) + \gamma \mathcal{T} V(s,a) - V(s)\right] - \alpha \mathrm{D}_f\left(d \| d^{\mathcal{D}}\right) \\
&= \min_{V(s)} (1-\gamma)\mathbb{E}_{s\sim\mu_0}[V(s)] + \max_{d \geq 0} \mathbb{E}_{(s,a)\sim d}\left[\hat{r}(s,a) + \gamma \mathcal{T} V(s,a) - V(s)\right] - \alpha \mathrm{D}_f\left(d \| d^{\mathcal{D}}\right) \\
&= \min_{V(s)} (1-\gamma)\mathbb{E}_{s\sim\mu_0}[V(s)] + \underbrace{\alpha\left[\max_{d \geq 0} \mathbb{E}_{(s,a)\sim d}\left[\frac{\hat{r}(s,a) + \gamma \mathcal{T} V(s,a) - V(s)}{\alpha}\right] - \mathrm{D}_f\left(d \| d^{\mathcal{D}}\right)\right]}_{L}
\end{aligned} \tag{32}$$

L in the last step reduces to:

$$
\begin{aligned}
&\alpha\left[\max_{d\geq 0}\mathbb{E}_{(s,a)\sim d}\left[\frac{\hat{r}(s,a)+\gamma\mathcal{T}V(s,a)-V(s)}{\alpha}\right]-\mathrm{D}_f\left(d\|d^{\mathcal{D}}\right)\right]\\
&=\alpha\left[\max_{d\geq 0}\mathbb{E}_{(s,a)\sim d^{\mathcal{D}}}\left[\frac{d(s,a)}{d^{\mathcal{D}}(s,a)}\frac{(\hat{r}(s,a)+\gamma\mathcal{T}V(s,a)-V(s))}{\alpha}\right]-\mathbb{E}_{(s,a)\sim d^{\mathcal{D}}}\left[f\left(\frac{d(s,a)}{d^{\mathcal{D}}(s,a)}\right)\right]\right]\\
&=\alpha\left[\max_{d\geq 0}\mathbb{E}_{(s,a)\sim d^{\mathcal{D}}}\left[\frac{d(s,a)}{d^{\mathcal{D}}(s,a)}\frac{(\hat{r}(s,a)+\gamma\mathcal{T}V(s,a)-V(s))}{\alpha}-f\left(\frac{d(s,a)}{d^{\mathcal{D}}(s,a)}\right)\right]\right]\\
&=\alpha\,\mathbb{E}_{(s,a)\sim d^{\mathcal{D}}}\left[\max_{d(s,a)\geq 0}\frac{d(s,a)}{d^{\mathcal{D}}(s,a)}\frac{(\hat{r}(s,a)+\gamma\mathcal{T}V(s,a)-V(s))}{\alpha}-f\left(\frac{d(s,a)}{d^{\mathcal{D}}(s,a)}\right)\right]\\
&=\alpha\,\mathbb{E}_{(s,a)\sim d^{\mathcal{D}}}\left[\max_{\frac{d(s,a)}{d^{\mathcal{D}}(s,a)}\geq 0}\frac{d(s,a)}{d^{\mathcal{D}}(s,a)}y(s,a)-f\left(\frac{d(s,a)}{d^{\mathcal{D}}(s,a)}\right)\right]\\
&=\alpha\,\mathbb{E}_{(s,a)\sim d^{\mathcal{D}}}\left[f_\star(y(s,a))\right]
\end{aligned}
\tag{33}
$$

where $y(s,a)=\frac{\hat{r}(s,a)+\gamma\mathcal{T}V(s,a)-V(s)}{\alpha}$, the third step follows the interchangeability principle (Lemma 1) and the last step comes from the Fenchel conjugate of convex function f [5]. □

Using this result, we finally yield the tractable lower level problem Eq. (12).

A.3 Proof of tractable transformation of upper level problem

Proposition 2. *The original upper level problem*

$$\min_{\Delta r} D_f\left(d^{\pi^*_{\hat{r}}}\|d^E\right) \tag{34}$$

can be equivalently written as:

$$\min_{\Delta r} D_f\left(f'_\star\left(\frac{\hat{r}+\gamma\mathcal{T}V^*-V^*}{\alpha}\right)d^{\mathcal{D}}\|d^E\right) \tag{35}$$

where $d^{\pi^*_{\hat{r}}}$ is the optimal state-action visitation distribution of Eq. (9)

Proof. By the property Eq. (21), the maximizer $\left(\frac{d(s,a)}{d^{\mathcal{D}}(s,a)}\right)^*$ of $f_\star(y(s,a))$ in Eq. (33) satisfies

$$\left(\frac{d(s,a)}{d^{\mathcal{D}}(s,a)}\right)^*=f'_\star\left(\frac{\hat{r}(s,a)+\gamma\mathcal{T}V(s,a)-V(s)}{\alpha}\right) \tag{36}$$

Given V^*, we have:

$$\frac{d^{\pi^*_{\hat{r}}}(s,a)}{d^{\mathcal{D}}(s,a)}=f'_\star\left(\frac{\hat{r}(s,a)+\gamma\mathcal{T}V^*(s,a)-V^*(s)}{\alpha}\right) \tag{37}$$

Substituting this result into the original upper level problem completes the proof. □

Next, we denote $f'_\star\left(\frac{\hat{r}+\gamma\mathcal{T}V^*-V^*}{\alpha}\right)$ as g. By expanding the f-divergence, we have the upper level objective:

$$\mathrm{D}_f\left(d^{\mathcal{D}}g\|d^E\right)=\mathbb{E}_{(s,a)\sim d^E}\left[f\left(\frac{d^{\mathcal{D}}(s,a)g(s,a)}{d^E(s,a)}\right)\right] \tag{38}$$

$$=\mathbb{E}_{(s,a)\sim d^{\mathcal{D}}}\left[\frac{d^E(s,a)}{d^{\mathcal{D}}(s,a)}f\left(\frac{d^{\mathcal{D}}(s,a)}{d^E(s,a)}g(s,a)\right)\right] \tag{39}$$

$$=\mathbb{E}_{(s,a)\sim d^{\mathcal{D}}}\left[w(s,a)f\left(\frac{g(s,a)}{w(s,a)}\right)\right] \tag{40}$$

[5] $\mathrm{dom}\, f=\{u:u\geq 0\}$ and f is convex, so $f_\star(y)=-f(0)$ when $y\leq f'(0)$.

where the distribution ratio $w(s,a) \triangleq d^E(s,a)/d^{\mathcal{D}}(s,a)$.

Finally, by combining proposition 1 and proposition 2, the original bi-level optimization problem Eq. (6)-(7) is rewritten equivalently as follows:

$$\begin{aligned} \Delta r^* &= \arg\min_{\Delta r} \mathbb{E}_{(s,a)\sim d^{\mathcal{D}}}\left[w(s,a) f\left(f'_\star\left(\frac{\hat{r}(s,a)+\gamma\mathcal{T}V^*(s,a)-V^*(s)}{\alpha}\right)/w(s,a)\right)\right] \\ \text{s.t. } V^*(s) &= \arg\min_{V(s)} (1-\gamma)\mathbb{E}_{s\sim\mu_0}[V(s)] + \alpha\,\mathbb{E}_{(s,a)\sim d^{\mathcal{D}}}\left[f_\star\left(\frac{\hat{r}(s,a)+\gamma\mathcal{T}V(s,a)-V(s)}{\alpha}\right)\right] \end{aligned} \tag{41}$$

B Implementation details of RGM

B.1 RGM with KL-divergence

In this section, we introduce the implementation details of RGM. Specifically, we implement RGM using KL-divergence for all tasks. For KL-divergence, we have $f(x) = x\log x$ and its Fenchel conjugate is $f_\star(x) = e^{x-1}$. However, this exponential form is numerically unstable and prone to explosion in practice. We address this issue by using the fact that the conjugate of the negative entropy function, restricted to the probability simplex, is the log-sum-exp function (Boyd et al., 2004), i.e., $\mathrm{D}_{\star,f}(y) = \log \mathbb{E}_{x\sim q}[\exp y(x)]$. Then, the optimization problem of RGM with KL divergence is

$$\begin{aligned} &\min_{\Delta r} \mathbb{E}_{(s,a)\sim d^{\mathcal{D}}}\left[\mathrm{Softmax}\left(\frac{\mathrm{Adv}(\Delta r, V^*)}{\alpha}\right)\left(\log\frac{d^{\mathcal{D}}(s,a)}{d^E(s,a)} + \log\left(\mathrm{Softmax}\left(\frac{\mathrm{Adv}(\Delta r, V^*)}{\alpha}\right)\right)\right)\right] \\ &\text{s.t.} V^* = \arg\min_V (1-\gamma)\mathbb{E}_{s\sim\mu_0}[V(s)] + \alpha\log\mathbb{E}_{(s,a)\sim d^{\mathcal{D}}}\left[\exp\left(\frac{\mathrm{Adv}(\Delta r, V)}{\alpha}\right)\right] \end{aligned} \tag{42}$$

where, $\mathrm{Adv}(\Delta r, V) := \hat{r}(s,a)+\gamma\mathcal{T}V(s,a)-V(s) = \tilde{r}(s,a)+\Delta r(s,a,\tilde{r})+\gamma\mathcal{T}V(s,a)-V(s)$ and $\log\frac{d^{\mathcal{D}(s,a)}}{d^{E(s,a)}}$ can be obtained by training a discriminator $\log\frac{d^{\mathcal{D}(s,a)}}{d^{E(s,a)}} = -\log\left(\frac{1}{h^*}-1\right)$ using Eq. (16) in continuous MDPs. The importance ratio used to extract the policy is

$$\psi^*(s,a) = \frac{d^{\pi^*_{\tilde{r}}}(s,a)}{d^{\mathcal{D}}(s,a)} = \mathrm{Softmax}\left[\frac{\tilde{r}+\Delta r+\gamma\mathcal{T}V^*(s,a)-V^*(s)}{\alpha}\right] \tag{43}$$

B.1.1 Optimize without Sum-exp

Note that in the upper level objective of Eq. (42), we need to calculate a log-sum-exp value in the denominator of the log(Softmax) term, where $\log\left(\mathrm{Softmax}(\mathrm{Adv}(\Delta r, V^*)/\alpha)\right) = \mathrm{Adv}(\Delta r, V^*)/\alpha - \log\sum_{s,a\in\mathcal{S}\times\mathcal{A}}\exp(\mathrm{Adv}(\Delta r, V^*)/\alpha)$. In low-dimensional discrete state-action space, we can easily get this value via summing over the overall space. In high-dimensional continuous MDPs, however, it is pretty difficult to retrieve the value because it requires integration over the entire space. CQL (Kumar et al., 2020) approximates this value via importance sampling but requires additional samples from the entire state-action space. There are some other methods like Markov Chain Monte Carlo (MCMC) or Score Match (SM) (Song & Kingma, 2021) that can approximate the update gradient but bring additional computation cost and suffer from some technical issues.

Fortunately, we can subtly circumvent the log-sum-exp term by optimizing the upper bound of the original upper-level problem using the following inequality (Boyd et al., 2004):

$$\max_{x_i\in\mathcal{B}}\{x_1,...,x_n\} \le \max\{x_1,...,x_n\} \le \log\sum_i^n \exp(x_i) \tag{44}$$

where $\max_{x_i\in\mathcal{B}}\{x_1,...,x_n\}$ is the max value in a mini-batch $\mathcal{B}$ which is sampled from $\{x_1,...,x_n\}$. For simplicity, we denote $\max_{x_i\in\mathcal{B}}\{x_1,...,x_n\}$ as $\max_{x_i\in\mathcal{B}}\{\boldsymbol{x}\}$. Substituting Eq. (44) into the upper level problem of Eq. (42), we get the upper bound of the original upper level optimization objective:

$$
\begin{aligned}
\mathrm{Upper}(42) &= \mathbb{E}_{(s,a)\sim d^{\mathcal{D}}}\left[\mathrm{Softmax}\left(\frac{\mathrm{Adv}(\Delta r, V^*)}{\alpha}\right)\left(\log\frac{d^{\mathcal{D}}(s,a)}{d^E(s,a)} + \frac{\mathrm{Adv}(\Delta r, V^*)}{\alpha} - \log\sum_{s,a\in\mathcal{S}\times\mathcal{A}}\exp\left(\frac{\mathrm{Adv}(\Delta r, V^*)}{\alpha}\right)\right)\right] \\
&\leq \mathbb{E}_{(s,a)\sim d^{\mathcal{D}}}\left[\mathrm{Softmax}\left(\frac{\mathrm{Adv}(\Delta r, V^*)}{\alpha}\right)\left(\log\frac{d^{\mathcal{D}}(s,a)}{d^E(s,a)} + \frac{\mathrm{Adv}(\Delta r, V^*)}{\alpha} - \max_{\mathcal{B}}\left\{\frac{\mathbf{Adv(\Delta r, V^*)}}{\alpha}\right\}\right)\right] \\
&\propto \mathbb{E}_{(s,a)\sim d^{\mathcal{D}}}\left[\exp\left(\frac{\mathrm{Adv}(\Delta r, V^*)}{\alpha}\right)\left(\log\frac{d^{\mathcal{D}}(s,a)}{d^E(s,a)} + \frac{\mathrm{Adv}(\Delta r, V^*)}{\alpha} - \max_{\mathcal{B}}\left\{\frac{\mathbf{Adv(\Delta r, V^*)}}{\alpha}\right\}\right)\right]
\end{aligned}
\tag{45}
$$

where $\mathrm{Upper}(42)$ denotes the upper level objective in Eq. (42).

Replacing Eq. (45) to the upper level objective in Eq. (42), we obtain the final optimization problem:

$$
\begin{aligned}
&\min_{\Delta r}\mathbb{E}_{(s,a)\sim d^{\mathcal{D}}}\left[\exp\left(\frac{\mathrm{Adv}(\Delta r, V^*)}{\alpha}\right)\left(\log\frac{d^{\mathcal{D}}(s,a)}{d^E(s,a)} + \frac{\mathrm{Adv}(\Delta r, V^*)}{\alpha} - \max_{\mathcal{B}}\left\{\frac{\mathbf{Adv(\Delta r, V^*)}}{\alpha}\right\}\right)\right] \\
&\quad \mathrm{s.t.} V^* = \arg\min_V (1-\gamma)\mathbb{E}_{s\sim\mu_0}[V(s)] + \alpha\log\mathbb{E}_{(s,a)\sim d^{\mathcal{D}}}\left[\exp\left(\frac{\mathrm{Adv}(\Delta r, V^*)}{\alpha}\right)\right]
\end{aligned}
\tag{46}
$$

We practically utilize the same mini-batch $\mathcal{B}$ as that of SGD gradient update step to calculate $\max_{\mathcal{B}}\left\{\frac{\mathbf{Adv(\Delta r, V^*)}}{\alpha}\right\}$. Note that the exp term in the upper level problem is prone to explosion in practice, we clip the exp value to $(-\infty, 100]$ like IQL (Kostrikov et al., 2021b) does to improve training stability.

When extracting the policy , we can ignore the annoying sum-exp term in the denominator of Softmax and get the following ratio, because it does not influence the direction of gradient to update policy.

$$
\psi^*(s,a) = \frac{d^{\pi^*_{\tilde{r}}}(s,a)}{d^{\mathcal{D}}(s,a)} \propto \exp\left[\frac{\tilde{r} + \Delta r + \gamma\mathcal{T}V^*(s,a) - V^*(s)}{\alpha}\right] := \tilde{\psi}^*(s,a) \tag{47}
$$

However, using Eq.(47), we can only get an unnormalized distribution ratio instead of an exact one. We resort to self-normalized importance sampling (Owen, 2013) to obtain a normalized ratio:

$$
\psi^*(s,a) = \frac{\tilde{\psi}^*(s,a)}{\mathbb{E}_{(s,a)\sim d^{\mathcal{D}}}[\tilde{\psi}^*(s,a)]} \tag{48}
$$

B.2 RGM WITH OTHER f-DIVERGENCE

Additionally, we can also implement RGM using $\mathcal{X}^2$-divergence. For $\mathcal{X}^2$-divergence, we have $f(x) = (x-1)^2$ with $\mathrm{dom}\, f = \{x : x \geq 0\}$[6] and its Fenchel conjugate is $f_\star(x) = \max\left(-1, \frac{1}{4}x^2 + x\right)$ and $f'_\star(x) = \max\left(0, \frac{1}{2}x + 1\right)$. Then, the optimization objective of RGM with $\mathcal{X}^2$ divergence is

$$
\begin{aligned}
&\min_{\Delta r}\ \mathbb{E}_{(s,a)\sim d^{\mathcal{D}}}\left[\frac{d^E(s,a)}{d^{\mathcal{D}}(s,a)}\left(\max\left(0, \frac{\mathrm{Adv}(\Delta r, V^*)}{2\alpha} + 1\right)\frac{d^{\mathcal{D}}(s,a)}{d^E(s,a)} - 1\right)^2\right] \\
&\mathrm{s.t}\ V^* = \arg\min_V (1-\gamma)\mathbb{E}_{s\sim\mu_0}[V(s)] + \alpha\mathbb{E}_{(s,a)\sim d^{\mathcal{D}}}\left[\max\left\{-1, \frac{1}{4}\left(\frac{\mathrm{Adv}(\Delta r, V)}{\alpha}\right)^2 + \frac{\mathrm{Adv}(\Delta r, V)}{\alpha}\right\}\right]
\end{aligned}
\tag{49}
$$

The importance ratio used to extract the policy is:

$$
\psi^*(s,a) = \frac{d^{\pi^*_{\tilde{r}}}(s,a)}{d^{\mathcal{D}}(s,a)} = \max\left(0, \frac{\tilde{r} + \Delta r + \gamma\mathcal{T}V^*(s,a) - V^*(s)}{2\alpha} + 1\right) \tag{50}
$$

B.3 RGM HYPERPARAMETERS AND PSEUDOCODE

For continuous MDPs with high dimensional state-action spaces, we implement RGM by parameterizing $h_\tau, \Delta r_\phi, V_\theta$ and π_w using deep neural networks with parameter τ, ϕ, θ and w, respectively. We

[6] On account of the state-action visitation distribution $d \geq 0$

implement RGM based on a two-time scale first-order stochastic gradient update, where the reward correction term is updated much slower than the Lagrangian multiplier V. We choose the cosine annealing learning rate schedule of the reward correction term and policy network to stabilize the training process. To make the reward correction term comparable w.r.t the original imperfect rewards, we normalize the imperfect rewards to standard Gaussian distribution $\mathcal{N}(0, 1)$ and strict the output range of Δr_ϕ to $[-3, 3]$ by Tanh function. The conclusive hyperparameters can be found in Table 2.

Table 2: The hyperparameters of RGM with deep neural networks

	Hyperparameter	Value
Architecture	Reward correction hidden dim	256
	Reward correction layers	2
	Reward correction activation function	ReLU
	Discriminator hidden dim	512
	Discriminator layers	3
	Discriminator activation function	Tanh
	V hidden dim	256
	V hidden layers	2
	V activation function	ReLU
	Policy hidden dim	256
	Policy hidden layers	2
	Policy activation function	ReLU
RGM Hyperparameters	Optimizer	Adam (Kingma & Ba, 2015)
	Reward correction learning rate l_ϕ	3e-7
	Reward correction learning rate schedule	cosine annealing
	Discriminator learning rate l_τ	1e-3
	V_θ learning rate l_θ	3e-4
	Policy learning rate l_w	3e-4
	Policy learning rate schedule	cosine annealing
	V_θ gradient L2-regularization	1e-4
	Discount factor	0.99
	α	4 for walker2d-medium-replay 0.5 for other tasks

The pseudocode of RGM with deep neural networks can be found in Algorithm 1. We run RGM on one RTX 3080Ti GPU with about 1h30min training time to apply 1M gradient steps.

Algorithm 1 RGM (KL-divergence) with Deep Neural Networks

Require: One Expert demonstration $\mathcal{D}^E$, offline Dataset $\mathcal{D}^O$, set $\mathcal{D}^O \leftarrow \mathcal{D}^E \cup \mathcal{D}^O$.

```
Initialize τ, ϕ, θ, w.
// Discriminator learning
Train h_τ using D^E and D^O using Eq. (16).
for t = 0, 1, 2, ..., N do
    Sample mini-batch transitions (s, a, r̃, s') ~ D^O
    // Reward Gap Minimization Bi-level optimization
    Update V_θ, Δr_ϕ using Eq. (46) with l_ϕ ≪ l_θ
    // Policy extraction
    Update π_w based on Eq. (18) and Eq. (48)
end for
```

We report the wall-clock training time of RGM compared with SOTA offline RL methods as well as SOTA offline IL methods that can learn from mixed quality data in Table 3. RGM is as efficient as most baselines but has an additional ability to combat the negative impacts of imperfect rewards.

Table 3: Wall-clock run time comparison of RGM and other baselines

BC	DWBC	SMODICE	TD3+BC	CQL	IQL	RGM(ours)
30min	2h40min	2h20min	45min	4h30min	1h30min	2h30min

C Experimental Details

In this section, we introduce the detailed experimental setups in our paper.

C.1 D4RL Experiments

Dataset composition. The D4RL Fu et al. (2020) datasets that we used in this paper contains 5 types of datasets: random: roll out a random policy for 1M steps. expert: roll out an expert policy that trained with SAC (Haarnoja et al., 2018) for 1M steps. medium: roll out a medium policy that achieves 1/3 the performance of the expert for 1M steps. medium-replay: replay buffer of a SAC agent that is trained to the performance of the medium policy. medium-expert: equally mixed dataset combines medium and expert data. We sample **only one trajectory from the expert dataset** to serve as the expert demonstration $\mathcal{D}^E$. The other datasets are treated as non-expert datasets $\mathcal{D}^O$.

Table 4: Dataset compositions for D4RL Experiments

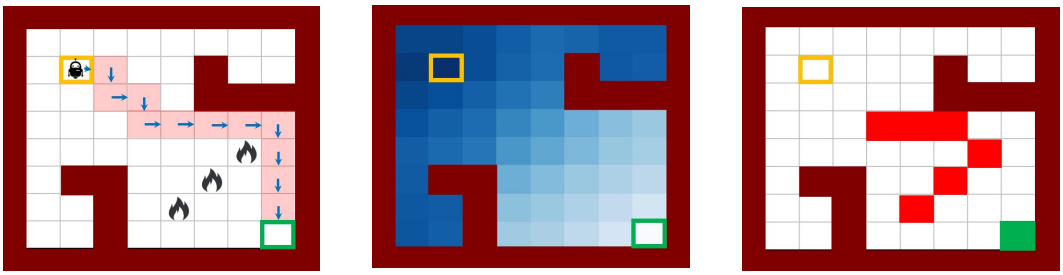
Task	State Dim	Expert Dataset	Number of Trajectories	Expert Data Size
Hopper	11	hopper-expert-v2	1	1000
Halfcheetah	17	halfcheetah-expert-v2	1	1000
Walker2d	17	walker2d-expert-v2	1	1000

Imperfect Rewards. We assume the original rewards in D4RL datasets are perfect, and hence we evaluate the policy performance based on the perfect reward function in original gym environment during evaluation. We randomly flip the sign of 50% original rewards to construct imperfect rewards, where half rewards can provide correct learning signals while the other half cannot.

C.2 Grid World Experiments

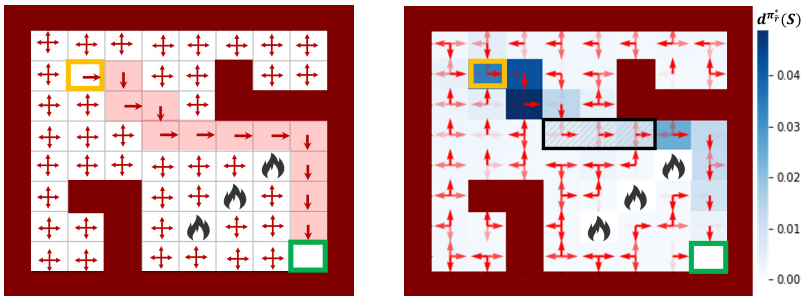
Dataset composition. The offline dataset $\mathcal{D}^O$ we use in grid world experiments consists of 1000 trajectories generated by a completely random policy (Figure 6 (b)). There are two settings of imperfect rewards $\tilde{r}$: (i) $\tilde{r} = +10$ when reaching the goal while $\tilde{r} = 0$ anywhere else. (ii) (Figure 6 (c)) $\tilde{r} = +10$ when reaching the goal, $\tilde{r} = -10$ when encountering the fire (true fire or fake fire), $\tilde{r} = 0$ everywhere else. The expert demonstration dataset $\mathcal{D}^E$ consists of **only one expert demonstration** (Figure 6 (a)).

In Figure 7, we give an illustrative explanation of the differences between RGM and SMODICE/DemoDICE through this grid world example. Figure 7 shows that the rewards used for SMODICE and DemoDICE ($\log d^E/d^{\mathcal{D}}$) give equal values for all actions in regions not covered by expert data, which provide limited learning signal and may even lead the RL agents to collide with the fire. Notably, the rewards corrected by RGM, however, can provide substantially more informative learning signal in no-expert-covered regions. Based on these corrected rewards, the RL agent can successfully navigate to the destination at most locations even by simply maximizing the per-step rewards and avoid the fire.



(a) Expert demonstration (b) Empirical distribution of $\mathcal{D}^O$ (c) Imperfect rewards

Figure 6: (a) The only one expert demonstration path, which starts from □, follows the path ▒ and arrow → to reach the goal □. (b) The empirical distribution heatmap of offline dataset $\mathcal{D}^O$, which consists of trajectories generated by random policy starting from □. The darker the color is, the more frequently the agent passes. (c) Illustration of imperfect rewards. Agent gets $\tilde{r} = -10$ when reaching ■, $\tilde{r} = +10$ when reaching ■, $\tilde{r} = 0$ everywhere else.



(a) SMODICE/DemoDICE rewards (b) RGM rewards

Figure 7: Illustration of the reward differences between RGM and SMODICE/DemoDICE. The opacity of the arrow represents the magnitude of reward value, and the darkest arrow points to the direction of the highest reward at this location. For SMODICE and DemoDICE, the rewards used for policy optimization are not distinguishable in the four action directions in no-expert-covered regions. In contrast, we can navigate to the destination by maximizing per-step RGM corrected rewards and avoid the fire. This is because that RGM rewards can encode long-horizon information and extract useful information from human-designed imperfect rewards.

D Learning Curves

In this section, we present the learning curves of RGM results that are presented in Table 1.

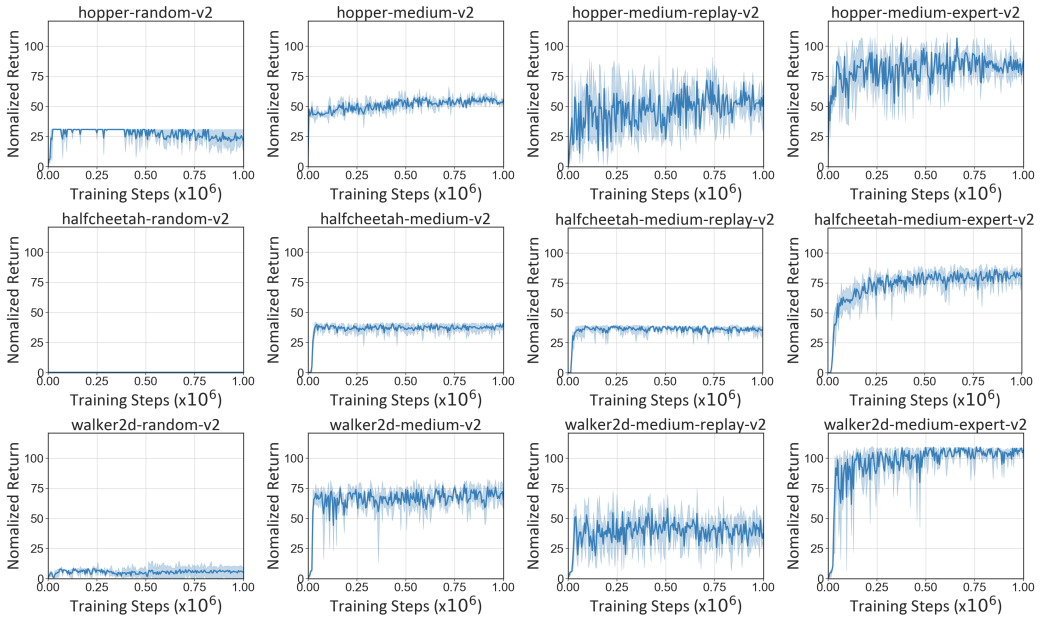


Figure 8: Learning curves of RGM trained on D4RL datasets under imperfect rewards.

E Additional Results

In this section, we provide additional comparative and ablation results of RGM against baseline methods.

E.1 Additional comparison to offline IL

Recall that DWBC (Xu et al., 2022b) and SMODICE (Ma et al., 2022) all assume the offline dataset already covers a lot of expert trajectories, which is more restrictive compared to the requirement of RGM. Therefore, we further demonstrate the superiority of RGM compared to these offline IL methods by evaluating RGM under the same settings of DWBC and SMODICE. We combine the original D4RL dataset with 200 or 100 expert trajectories as the offline dataset $\mathcal{D}$, see Table 5 for descriptions of the expert trajectories. The comparisons under these dataset configurations can be found in Table 6. We can observe from Table 6 that RGM still outperforms existing SOTA offline IL methods under their settings.

Table 5: The details about the expert data that are used to construct the non-expert dataset in offline IL settings.

Task	State Dim	Expert Dataset	Number of Trajectories	Expert Data Size
Hopper	11	hopper-expert-v2	200	193430
Halfcheetah	17	halfcheetah-expert-v2	200	199800
Walker2d	17	walker2d-expert-v2	100	99900

E.2 Experiments on sampling from discounted distributions

We also implemented the discounted visitation distribution sampling in RGM. This is done by augmenting the D4RL datasets that adds the timestep of each (s, a) pair in an episode. When performing sampling in Eq.(16-18) and calculating the gradient, we sample (s, a, t) in the D4RL datasets and then multiply the gradient by γ^t. Empirically, we found that the performance of the discounted visitation distribution version is not better than the sampling distribution version of RGM. Figure 9 and Table 7 show that RGM (sampling distribution) surpasses RGM (discounted visitation

Table 6: Average normalized scores of RGM compared with SOTA offline IL methods that can learn from mixed quality data under their settings. The notation "-w.e" stands for the mixed dataset that combines the original D4RL dataset with some expert trajectories. The scores are taken over the final 10 evaluations with 5 random seeds. We obtain the results via ruining author-provided open-source codes. RGM achieves 7 highest scores in 12 tasks.

Dataset	BC	DWBC	SMODICE	RGM (Ours)
hopper-r-w.e	2.8	59.5±30.3	108.7±5.4	110.4 ±1.2
halfcheetah-r-w.e	0.2	3.3±1.7	89.3 ±1.5	57.6±6.4
walker2d-r-w.e	1.2	81.8±0.6	102.0±9.9	109.2 ±0.2
hopper-m-w.e	54.9	39.0±22.5	54.5±4.3	66.1 ±9.7
halfcheetah-m-w.e	41.2	8.5±9.2	55.4 ±10.9	50.5±7.9
walker2d-m-w.e	62.1	56.1±39.2	6.5±9.7	79.2 ±12.4
hopper-m-r-w.e	23.4	23.1±17.1	53.0±27.5	58.6 ±27.0
halfcheetah-m-r-w.e	24.2	1.1±1.2	84.9 ±7.2	65.4±15.1
walker2d-m-r-w.e	21.8	85.5 ±33.6	8.9±12.3	71.7±32.6
hopper-m-e-w.e	51.2	40.0±22.5	71.4±15.5	89.1 ±13.5
halfcheetah-m-e-w.e	61.7	1.2±0.5	87.2 ±1.9	76.8±8.8
walker2d-m-e-w.e	103.2	76.8±31.7	14.1±2.0	105.8 ±8.6
Mean Score	37.8	39.7±17.5	61.3±9.0	78.3 ±12.0

distribution) in most cases with lower variance, while the latter wins by a slight margin in only a few cases.

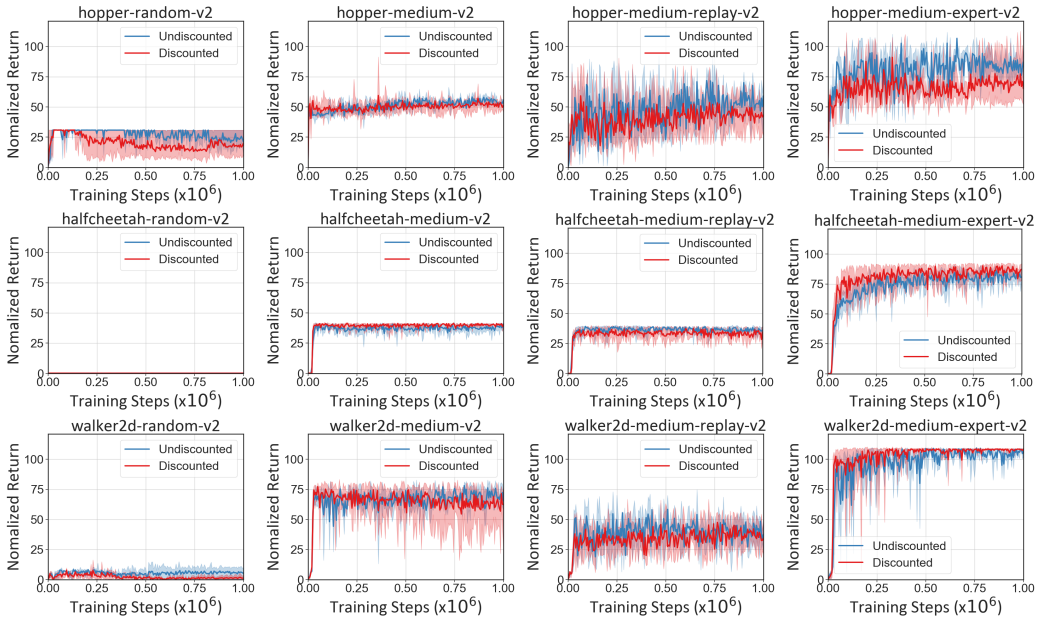


Figure 9: Experiments on sampling from discounted and undiscounted distributions

Table 7: Normalized scores of RGM sampling from discounted distribution and undiscounted distribution

Dataset	RGM (Discounted)	RGM (Undiscounted)
hopper-r	19.8±0.2	21.2 ±0.4
halfcheetah-r	0.2 ±0.0	0.2 ±0.0
walker2d-r	1.2±1.7	7.7 ±3.3
hopper-m	51.1±4.9	55.5 ±1.0
halfcheetah-m	40.3±1.6	40.7 ±1.4
walker2d-m	62.2±22.5	72.3 ±10.7
hopper-m-r	43.3±11.6	59.1 ±15.3
halfcheetah-m-r	34.5±4.5	37.8 ±2.6
walker2d-m-r	34.3±11.0	48.6 ±3.6
hopper-m-e	65.3±19.5	87.1 ±10.7
halfcheetah-m-e	87.3 ±7.8	81.5±0.8
walker2d-m-e	108.4±0.6	108.8 ±0.4
Mean score	45.7±7.2	52.0 ±4.2

E.3 Comparisons under completely incorrect rewards

To validate whether RGM can handle completely incorrect rewards, we construct experiments by flipping all the signs of the original rewards. Such rewards are catastrophic for offline RL methods because they provide completely reversed learning signals for policy optimization. Table 8 shows that RGM can ameliorate the negative impact of completely incorrect rewards, while the learned policies of SOTA offline RL fail miserably and even produce very negative scores on some datasets, such as halfcheetah-r.

Table 8: Normalized scores of RGM and offline RL baselines under incorrect rewards. All signs of the original D4RL rewards are flipped, and hence the corrupted rewards provide completely wrong and reversed learning signals. We denote (W) as learning under such incorrect rewards.

Dataset	TD3+BC (W)	CQL (W)	IQL (W)	RGM (W)
hopper-r	0.4	0.0	0.7	25.9
halfcheetah-r	-11.6	-38.4	2.2	0.2
walker2d-r	2.0	-0.0	-0.3	-0.1
hopper-m	37.3	11.2	35.5	54.6
halfcheetah-m	1.2	4.1	35.4	40.3
walker2d-m	17.2	3.3	22.0	38.4
hopper-m-r	16.3	0.0	0.7	44.5
halfcheetah-m-r	-1.0	-1.1	1.8	29.8
walker2d-m-r	-0.7	-0.0	-0.2	46.1
hopper-m-e	22.3	11.6	13.6	66
halfcheetah-m-e	1.9	11.1	35.8	78.4
walker2d-m-e	6.8	16.6	20.9	108.8
Mean score	7.7	-0.3	14.0	41.9

E.4 Experiments on noisy imperfect rewards

We add i.i.d Gaussian noises with different standard deviation σ to original D4RL rewards to construct noisy imperfect rewards with different degrees of imperfection. We set $\sigma = 1$ to construct partially correct rewards and $\sigma = 10$ as largely incorrect rewards, see Table 9 for detailed results.

Table 9 shows that RGM under perfect rewards slightly outperforms RGM with partially correct rewards, indicating that RGM can largely remedy the negative impacts caused by reward noises with $\sigma = 1$. Meanwhile, the highly noisy rewards ($\sigma = 10$) surely impact the performance, but its mean score is 45.0, which is still considerably higher than other Offline RL and IL methods under partially correct rewards with the largest mean value of 35.5 as showed in Table 1.

Table 9: Normalized scores of RGM on different degrees of noisy datasets.

Dataset	RGM(T)	RGM ($\sigma = 1$)	RGM ($\sigma = 10$)
hopper-r	29.6	8.5	9.8
halfcheetah-r	0.2	0.3	0.2
walker2d-r	3.9	0.6	-0.1
hopper-m	56.2	52.0	47.9
halfcheetah-m	40.4	41.2	38.4
walker2d-m	73.3	71.9	72
hopper-m-r	60.3	58.0	40.0
halfcheetah-m-r	37.9	38.3	28.1
walker2d-m-r	46.3	42.5	43.8
hopper-m-e	106.1	82.0	82.8
halfcheetah-m-e	85.6	88.7	69.1
walker2d-m-e	109.2	108.2	108.5
Mean score	54.1	49.4	45.0

E.5 Ablations on the number of expert trajectories

We add the ablations on the number of expert trajectories in $\mathcal{D}_E$ (N_E) for RGM, SMODICE and DWBC. Table 10, 11 and 12 show that RGM achieves better performance than offline IL methods designed for mixed-quality data (DWBC and SMODICE). It is found that RGM also enjoys a higher level of performance gains when the amount of expert data is increased.

Table 10: Normalized scores of RGM and offline IL baselines when $\mathcal{D}_E$ contains 10 expert trajectories.

Dataset	DWBC ($N_E = 10$)	SMODICE ($N_E = 10$)	RGM ($N_E = 10$)
hopper-r	52.5	1.3	30.8
halfcheetah-r	-0.3	2.1	0.2
walker2d-r	96.2	0.3	6.1
hopper-m	31.1	53.8	54.5
halfcheetah-m	5.0	40.9	41.4
walker2d-m	22.4	3.3	72.9
hopper-m-r	37.4	33.2	55.5
halfcheetah-m-r	3.9	36.7	34.9
walker2d-m-r	90.7	34.7	43.1
hopper-m-e	31.2	85.0	89.2
halfcheetah-m-e	10.9	86.6	79.4
walker2d-m-e	46.3	14.1	109.0
Mean score	35.6	32.7	51.4

Table 11: Normalized scores of RGM and offline IL baselines when $\mathcal{D}_E$ contains 40 expert trajectories.

Dataset	DWBC ($N_E = 40$)	SMODICE ($N_E = 40$)	RGM ($N_E = 40$)
hopper-r	54.6	67.2	36.9
halfcheetah-r	8.8	14.8	18.7
walker2d-r	78.7	92.9	-0.1
hopper-m	13.5	54.2	57.0
halfcheetah-m	5.6	44.5	40.9
walker2d-m	16.9	3.5	74.3
hopper-m-r	54.3	47.2	54.3
halfcheetah-m-r	46.1	54.6	46.1
walker2d-m-r	87.8	35.7	61.2
hopper-m-e	34.5	75.9	92.4
halfcheetah-m-e	3.4	85.1	84.7
walker2d-m-e	57.2	17.1	108.6
Mean score	38.5	44.9	56.5

Table 12: Normalized scores of RGM and offline IL baselines when $\mathcal{D}_E$ contains 80 expert trajectories.

Dataset	DWBC ($N_E = 80$)	SMODICE ($N_E = 80$)	RGM ($N_E = 80$)
hopper-r	65.1	92.7	47.3
halfcheetah-r	2.3	48.1	40.4
walker2d-r	86.1	98.8	109.1
hopper-m	8.8	53.4	59.7
halfcheetah-m	6.7	51.2	42.5
walker2d-m	36.5	2.9	73.4
hopper-m-r	35.3	46.6	66.2
halfcheetah-m-r	36.1	59.6	54.0
walker2d-m-r	85.8	32.8	64.9
hopper-m-e	8.8	83.7	97.1
halfcheetah-m-e	12.1	87.4	83.9
walker2d-m-e	64.5	43.7	108.6
Mean score	37.5	58.4	70.6

E.6 Experiments on the sparse reward setting

In this section, we further validate the performance of RGM under the sparse reward setting. Specifically, we construct experiments on the well-known Antmaze task, where an ant not only needs to learn to walk but also navigates from the goal to the destination in a maze. The ant can only get a +1 reward signal when reaching the goal. This task is extremely difficult due to the non-markovian and mixed-quality offline dataset, the stochastic property of environments, and the high dimensional state-action space (Fu et al., 2020). Note that in typical offline RL papers, like IQL (Kostrikov et al., 2021b) and CQL (Kumar et al., 2020), they turn the original sparse rewards into dense rewards by applying the reward substraction trick (minus 1 on every rewards, so the reward become negative except at the goal). These offline RL methods can obtain non-zero successful rates with the substracted dense rewards but fail miserably when trained on the original sparse reward.

In this experiment, the expert dataset $\mathcal{D}_E$ of RGM is composed of 30 successful trajectories (which may be suboptimal) that are collected by training IQL with dense rewards. To give a fair comparison, we also add these successful demonstrations to the original D4RL datasets to train the other offline RL methods. Table 13 shows that RGM is the only approach that can solve such a difficult task.

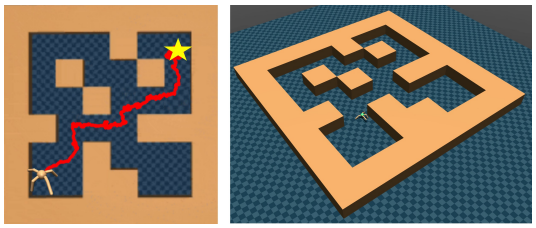


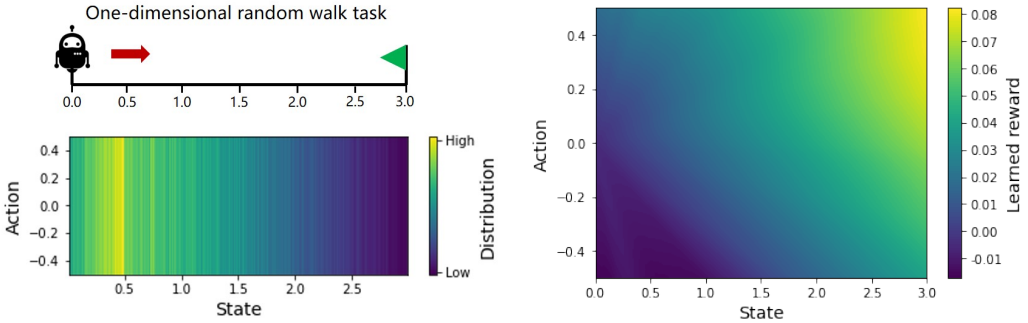
Figure 10: Illustration of the Antmaze task.

Table 13: Successful rates in Antmaze medium tasks

Dataset	TD3+BC	CQL	IQL	RGM (Ours)
Antmaze-m-p	0%	0%	0%	13.7%
Antmaze-m-d	0%	0%	0%	3.3%

E.7 ILLUSTRATIVE EXAMPLE FOR THE NON-TABULAR SCENARIOS

The results of the 8×8 grid world experiments in Section 5.3 and Appendix C.2 illustrate the potential benefits of the learned rewards in the tabular case. In this subsection, we consider an one-dimensional random walk task in the non-tabular case and provide the visualization of the learned corrected rewards $\hat{r}$. In this task, the state space is a straight line from [0, +3] and the agent can move at each step in the range of [-0.5, 0.5]. If the agent go beyond the edge ($s < 0$ or $s > +3$), then we keep it at the edge ($s = 0$ or $s = +3$). The agent needs to start from state $s = 0$ and reach the destination located at $s = 3$ as fast as possible. The expert dataset $\mathcal{D}^E$ consists of one trajectory where the expert takes action $a = 0.5$ at every state. The offline datatset $\mathcal{D}^O$ consists of 1000 trajectories generated by a completely random policy where the agent takes action uniformly from [-0.5, 0.5] at every state. The sparse rewards $\tilde{r} = +10$ is set when reaching the destination while $\tilde{r} = 0$ anywhere else. The visualization of learned rewards $\hat{r}$ at each state-action pair is shown in Figure 11.



(a) Empirical distribution of $\mathcal{D}^O$

(b) Visualization of learned rewards $\hat{r}$

Figure 11: (a) The empirical distribution of offline dataset $\mathcal{D}^O$ in a continuous one-dimensional random walk task. Most states in the offline dataset are distributed near the starting point. (b) At each state (at each vertical line), the learned reward $\hat{r}$ get larger value when the action get closer to 0.5. The expert data has contain 7 state point ($s = 0.0, 0.5, 1.0, 1.5, 2.0, 2.5, 3.0$), but the learned rewards can still generalize well in the state space even in regions that not covered by the expert data. Similar to the 8×8 grid world experiment, we can successfully navigate to the destination by only maximizing per-step reward $\hat{r}$, which means that the learned rewards also encode long-horizon information.

F Discussion on the Applicability to Online Settings

It should be noted that the proposed RGM framework can also be applied to the online setting. This can be achieved by simply setting $\alpha = 0$ in Eq. (6-7), and we have the bi-level objective of the online version of RGM:

$$\begin{aligned} \Delta r^* &= \arg\min_{\Delta r} \ \mathrm{D}_f \left(d^{\pi^*_{\hat{r}}} \| d^E \right) \\ \text{s.t.} \quad \pi^*_{\hat{r}} &= \ \arg\max_{\pi} \ \mathbb{E}_{(s,a)\sim d^{\pi_{\hat{r}}}} [\hat{r}(s,a)] \end{aligned} \tag{51}$$

Since we could get online samples from $d^{\pi^*_{\hat{r}}}$ in online setting, so we don't have to eliminate $d^{\pi^*_{\hat{r}}}$. One can use the existing popular online RL algorithms to solve the lower level problem, while leveraging the online samples from $d^{\pi^*_{\hat{r}}}$ to solve the upper level problem. Hence the online version of RGM can be perceived as a reduced and simplified version of the original RGM. The core idea of the reward correction has not been changed in the online setting, which illustrates that to some extent, our proposed RGM is an unified policy optimization method for imperfect rewards.