MULTI-REWARD FUSION: LEARN FROM OTHER POLICIES BY DISTILLING

Anonymous authors
Paper under double-blind review

ABSTRACT

Designing rewards is crucial for applying reinforcement learning in practice. However, it is difficult to design a shaping reward which can accelerate agents' learning process without biasing the original task's optimization objective. Moreover, the low-dimensional representation of the reward and value function (i.e. scalar value) may also be an obstruction during the learning process. This paper contributes towards tackling these challenges, by proposing a new method, called Multi-Reward Fusion (MRF). MRF take as input a list of human designed rewards, which contains the information from multiple perspectives about the task, and learns separate policies for each component of the reward list. We formulate the problem of learning the target policy as a distillation task, propose a novel method which can selectively distills knowledge from the auxiliary policies, and theoretically show the feasibility of this method. We conduct extensive experiments and show that the MRF method performs better than state-of-the-art reward shaping methods.

1 INTRODUCTION

For applying reinforcement learning in real-world tasks, how to design a suitable reward function is a challenging problem. A common way for addressing this problem is reward shaping (RS), which transforms the human prior knowledge into shaping reward, so that the agents can be guided to learn faster and better with the combination of the original and new rewards. In early works, hand-crafted reward function had been used in robot behavior learning Dorigo & Colombetti (1994)Randløv & Alstrøm (1998). But the introduced shaping reward may deviate the converged policy away from the optimal policy of the original task. The potential-based reward shaping (PBRS) method Ng et al. (1999) firstly solved this problem by designing the shaping reward via the form of the difference of potential values, which guarantees the policy invariance. Although PBRS and its variantsDevlin & Kudenko (2012); Grzes & Kudenko (2008); Harutyunyan et al. (2015); Wiewiora et al. (2003) have good mathematical characteristics, sometimes it doesn't work due to its weak driving force. The phenomenon result from the invariance of the state-action value function by using PBRS, whose aid to policy learning is less straightforward. Moreover, the automatic shaping approaches Marthi (2007); Hu et al. (2020); Fu et al. (2019) learn to take advantage of multiple auxiliary shaping reward functions by adjusting the weight vector of the reward functions.

All of these exist works about reward shaping are trying to find a feasible way to introduce human prior knowledge into agents' learning process. However, when using these architectures we have to face such a dilemma: how to generate useful shaping rewards? The design of the rewards directly affect the results of training. For example, only when the shaping rewards transformed from prior knowledge are completely helpful, PBRS and its variants may work. Although, the automatic shaping approaches can alleviate this problem to some extent, their computational complexity is prohibitive.

We consider that the multiple sources of reward setting, recommended in hybrid reward architectures (HRA) Van Seijen et al. (2017) and RD^2 Lin et al. (2020), may be more suitable than the traditional scalar form of the reward. Because, a useful reward usually contains information from multiple perspective of the same task. For example, when we design reward functions for training a Doom agent Lample & Chaplot (2017), designers should consider in multiple perspectives such as object pickup, shooting, losing health, and losing ammo. These multi-perspective sources of rewards are

high-dimensional information, directly mapping them to a scalar is very rude and will lose a lot of information.

In this paper, we use a list of shaping rewards sourcing from different perspectives to train a list of critics. Each critic corresponds to a policy. And we use the relationship between the list of critics to decide how to learn from these policies by distillation. In this process, except the target policy and target critic, all of the other auxiliary critics and policies are trained in offline wayAn et al. (2021). The contributions of this paper are as follow:

- We use multi-perspective sources of rewards to train the agent to prevent information loss caused by dimensionality reduction (i.e., summation the shaping rewards into one scalar). In this process, we transform the optimization objective of optimization Haarnoja et al. (2018a) into a new form of policy distillation, and prove the equivalence of these two optimization objectives theoretically.
- We provide a gradient similarity-based regularization method to eliminate the effects of adverse rewards automatically. This regularization can improve convergence efficiency.
- Empirically, our method can make better trade-off between the policy invariance and driven power from the shaping rewards. Moreover, the auxiliary policies' offline training can proceed in parallel, which can spare training time.

2 BACKGROUND

2.1 SOFT ACTOR CRITIC

In this paper, we consider the soft actor critic framework Haarnoja et al. (2018a; 2017) of reinforcement learning and adopt Morkov decision process (MDP) as the mathematical model. Formally, MDP can be denoted as a tuple $\mathcal{M} =< S, A, P, r, p_0, \gamma >$, where S is the state space, A is the action space, $P : S \times A \times S \to [0, 1]$ denotes the state transition function, $r : S \times A \to \mathbb{R}$ is the reward function, $p_0 : S \to [0, 1]$ is the probability distribution of the initial state, and $\gamma \in [0, 1]$ is the discount rate. Normally, the purpose of reinforcement learning is to find a policy of an agent $\pi : S \times A \to [0, 1]$ in an MDP, which can maximize the expectation of the accumulative rewards $\mathbb{E}_{s\sim\rho^\pi, a\sim\pi}[r(s, a)]$. Here, $\rho^\pi(s) = \int_S \sum_{t=1}^{\infty} \gamma^{t-1} p_0(s')p(s' \to s, t, \pi)$ denotes the distribution of the state and $p(s' \to s, t, \pi)$ means the probability that state s is visited after t steps from state s' under policy π. Usually, we represent the policy π by a neural network and the parameters of this neural network can be denoted as θ.

According to the soft actor critic (SAC) method, the entropy of the policy also needs to be maximized to stimulate exploration. And the Bellman operator can be denoted as : $Q_\phi(s, a) = r(s, a) + \gamma\mathbb{E}_{s'\sim\rho^\pi}[V(s')]]$, where $V(s) = \mathbb{E}_{a\sim\pi}[Q(s, a) - \alpha \log \pi(a|s)]$, ϕ is the vector of parameter in critic neural network. Here, α is the temperature parameter. In Haarnoja et al. (2018b), α is used to maintain the constraint of the entropy of policy $\mathbb{E}_{s\sim\rho^\pi, a\sim\pi}[-\log \pi(a|s)] \geq \mathcal{H}$. Moreover, It is worth mentioning that the the policy parameters are learning to minimizing the expected KL-divergence $\mathbb{E}_{s\sim\rho^\pi}[D_{KL}(\pi_\theta(\cdot|s)||\frac{\exp(Q_\phi(s,\cdot))}{Z_\phi(s)})]$, where $Z_\phi(s)$ is a normalization term, which can be ignored during training.

2.2 RELATED WORK

2.2.1 REWARD SHAPING

The traditional reward shaping usually means that modifying the original reward with shaping reward functions which introduce domain knowledge, such as PBRS and its theoretical analysisWiewiora et al. (2003); Laud & DeJong (2003), automatic reward shaping Marthi (2007); Grzes & Kudenko (2008), multi-agent reward shaping Devlin & Kudenko (2011); Sun et al. (2018); Wang et al. (2022), belief reward shaping Marom & Rosman (2018), ethics shapingWu & Lin (2018), and reward shaping via meta learning Zou et al. (2019). The automatic reward shaping methods, such as the automatic successive reinforcement learning (ASR) framework Fu et al. (2019) and the bi-level optimization of parameterized reward shaping (BiPaRS) Hu et al. (2020), have the similar motivation of this paper, namely letting the agent learn from the knowledge that should be focused on. Instead

of considering how to adjust the weight of each component of the shaping reward, we directly learning multi-perspective state-action value functions and corresponding policies by multi-perspective rewards. It's meaningful to mention that each component of the traditional shaping reward can be seen as one source of the multi-perspective rewards.

2.3 HYBRID REWARD ARCHITECTURE

Hybrid Reward Architecture (HRA) Van Seijen et al. (2017) proposes a hybrid architecture to model the value functions for the rewards source from different perspectives. Their work justifies that learning from multi-perspective source of the rewards can improve sample efficiency empirically. HRA is built upon the Horde architecture Sutton et al. (2011) , which trains a separate general value function (GVF) for each pseudo-reward function. The work about reward decomposition Lin et al. (2020) also demonstrates that learning from multiple reward functions is beneficial. The structured policy iteration Boutilier et al. (1995) also supports this viewpoint. In these works, the multiple sources of rewards can be seen as auxiliary tasks, which serve as additional supervision for the agent to learn multi-perspective value functions and better representations of the task in different perspectives. However, in this paper, we not only learn multi-perspective value functions via multi-perspective sources of the rewards, but also learn the corresponding multi-perspective policies which are supplied for the distillation of the agent. Through our method, we only need to add the information we focused into the multi-perspective rewards' list without paying attention to how to adjust the weight of each reward's component. Furthermore, our method learning target policy by distilling from the auxiliary policies which is more direct than previous methods and has higher data efficiency.

3 METHOD

Given an Markov decision process $< S, A, P, r, p_0, \gamma >$ and a shaping reward function $\mathbf{f} : S \times A \rightarrow \mathbb{R}^{n\times 1}$, we denote the output of reward function as $\mathbf{f}(s, a) = [r_1, r_2, \cdots, r_n, 0]^{\mathrm{T}}$. The vector of multi-perspective rewards, which is composed by these shaping rewards and the original reward, can be formalized in the additive form as $\mathbf{r} = r_o + \mathbf{f}$. In this paper, we label derivatives (such as Q_o, π_o) of the original reward r_o with subscript o.

3.1 RELATIONSHIP BETWEEN AUXILIARY POLICY π_i AND TARGET POLICY π_o

Let Q_i represents the state-action value function which is learnt via r_i (i.e. the i^{th} component of $\mathbf{r}$). And $\mathbf{Q} = [Q_1, Q_2, \cdots, Q_n, Q_o]^{\mathrm{T}}$ corresponds to the state-action value function of $\mathbf{r}$. For the reason that all the rewards describe the same task from different perspectives, there must exist some implicit connection between each component of $\mathbf{Q}$. And we use the additive form to describe the relationship of the components of $\mathbf{Q}$ as:

$$Q_o^{\pi} = \sum_{i=1}^{n} w_i Q_i^{\pi} \tag{1}$$

where $w_i : S \times A \rightarrow \mathbb{R}$ is the weight of each Q_i (i.e. the i^{th} component of $\mathbf{Q}$). As long as $n \geq 2$, Eq.(1) must exist a solution $\mathbf{w} = [w_1, w_2, \cdots, w_n]$ Marcus & Minc (1992). When we use the SAC framework, the loss function of the policy π_o can be represented as $L_{\pi_o} = \mathbb{E}_{s\sim\rho^{\pi_o}}[D_{KL}(\pi_{\theta_o}(\cdot|s)||\frac{\exp(\frac{1}{\alpha}Q_{\phi_o}(s,\cdot))}{Z_{\phi_o}(s)})]$. Furthermore, via this loss function, the relationship between the policies $\boldsymbol{\pi} = [\pi_1, \pi_2, \cdots, \pi_n]$ and the target policy π_o can be described in the form of distillation as shown in Eq.(2).

$$\begin{aligned} L_{\pi_o} &= \mathbb{E}_{s\sim\rho^{\pi_o}}[D_{KL}(\pi_{\theta_o}(\cdot|s)||\frac{\exp(\frac{1}{\alpha}\sum w_i Q_i^{\pi_o}(\cdot|s))}{Z(s)})] \\ &\leq \mathbb{E}_{s\sim\rho^{\pi_o}}[\sum \lambda_i(s,\xi) D_{KL}(\pi_{\theta_o}||\pi_i) + \int_a \pi_{\theta_o}(\log \sum \lambda_i(s,a)\frac{\pi_i}{\exp\frac{q_i}{\alpha}}) \\ &\textbf{where} \quad \lambda_i = \frac{|w_i|}{\sum |w_j|} \qquad q_i = Q_i^{\pi_o}\frac{w_i \sum |w_j|}{|w_i|}] \end{aligned} \tag{2}$$

where λ_i means the confidence coefficient of the corresponding policy π_i, and $\xi \in A$ is the outcome of using the mean value theorem of integrals. And λ_i is normalized in the range of $[0, 1]$, which is helpful for using the Jensen's inequality Rudin (1987). For the reason that the determinacy of policy π_o is gradually increased in the training process, we approximate ξ by the mean of the distribution of π_o, which is usually represented by a Gaussian distribution, to simplify the calculation.

3.2 λ's Learning Architecture

The architecture of the λ calculator unit is shown in Fig.(1). To satisfy the equation constraint shown in Eq.(1), we want to describe the feasible set in an affine form as:

$$\begin{aligned} \hat{\mathbf{Q}} &= [Q_1, Q_2, \cdots, Q_n]^{\mathrm{T}} \\ \{\mathbf{w} | < \hat{\mathbf{Q}}, \mathbf{w} >= Q_o\} &= \{\mathbf{F}\mathbf{z} + \hat{\mathbf{w}} | \mathbf{z} \in \mathbb{R}^{(n-1)\times 1}\} \end{aligned} \tag{3}$$

where $\mathbf{F} \in \mathbb{R}^{n\times(n-1)}$, $\hat{\mathbf{w}} \in \mathbb{R}^{(n-1)\times 1}$, and $\mathbf{z} \in \mathbb{R}^{n\times 1}$ is the output of neural network. LU decomposition method Trefethen & Bau III (1997) can be used here to calculator $\mathbf{F}$ and $\hat{\mathbf{w}}$:

$$\begin{aligned} \hat{\mathbf{Q}} &= \mathbf{P}\mathbf{L}\mathbf{U} \quad \mathbf{L} = \begin{bmatrix} \mathbf{L_1} \\ \mathbf{L_2} \end{bmatrix} \\ \hat{\mathbf{w}} &= \mathbf{P} \begin{bmatrix} \mathbf{L_1}^{-\mathrm{T}}\mathbf{U}^{-\mathrm{T}}Q_o \\ \mathbf{0} \end{bmatrix} \quad \mathbf{F} = \mathbf{P} \begin{bmatrix} -\mathbf{L_1}^{-\mathrm{T}}\mathbf{L_2}^{\mathrm{T}} \\ \mathbf{I} \end{bmatrix} \end{aligned} \tag{4}$$

where $\mathbf{P} \in \mathbb{R}^{n\times n}$ is the permutation matrix, $\mathbf{L} \in \mathbb{R}^{n\times 1}$ is the unit lower triangular matrix ($\mathbf{L_1} \in \mathbb{R}^{1\times 1}$,$\mathbf{L_2} \in \mathbb{R}^{(n-1)\times 1}$), and $\mathbf{U} \in \mathbb{R}^{1\times 1}$ nonsingular upper triangular matrix.

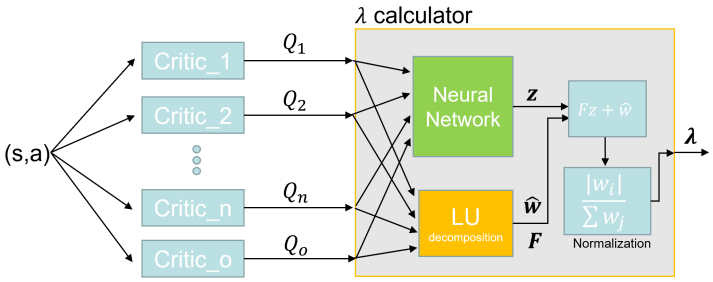


Figure 1: The architecture of λ calculator

3.3 The Similarity Regularization for Learning λ

Sometimes, some components of $\mathbf{r}$ are not helpful for the agent tackling the task. The corresponding state-action value function $Q_i \in \mathbf{Q}$ may also be diverging or getting meaningless, which may be an obstacle to the convergence of λ calculator. Hence, we propose a gradient similarity-based regularization to mitigate this trouble.

$$\begin{aligned} \mathbf{M} &= \frac{\partial \hat{\mathbf{Q}}}{\partial a} \cdot (\frac{\partial Q_o}{\partial a})^{\mathrm{T}} \\ &= \left[< \frac{\partial Q_1}{\partial a}, \frac{\partial Q_o}{\partial a} >, < \frac{\partial Q_2}{\partial a}, \frac{\partial Q_o}{\partial a} >, \cdots, < \frac{\partial Q_n}{\partial a}, \frac{\partial Q_o}{\partial a} > \right]^{\mathrm{T}} \\ L_{reg} &= < \mathbb{1}_{\mathbf{M}<\mathbf{0}}, \mathbf{w} \circ \mathbf{w} > \end{aligned} \tag{5}$$

where $\frac{\partial Q_i}{\partial a}$ $i \in [1, 2, \cdots, n, o]$ is the gradient of Q_i, $\mathbf{M} \in \mathbb{R}^{n\times 1}$ is the mask matrix, and $\mathbb{1}_{\mathbf{M}<\mathbf{0}} : \mathbb{R}^{n\times 1} \to \mathbb{R}^{n\times 1}$ is the indicator function. We use this form to restrict the output of λ calculator corresponding to the useless Q_i.

3.4 MULTI-REWARD FUSION ARCHITECTURE

MRF breaks through the traditional methods of using shaping rewards. Instead of working on the better representation of state-action value, we directly learning the policy by distillation of the auxiliary policies. On account of our method is based on SAC, the loss function of the critic can be formulated as:

$$L_Q(\boldsymbol{\phi}) = \sum_i \mathbb{E}_{(s,a,s')\sim D}[\frac{1}{2}\left(Q_{\phi_i}(s,a) - (r_i(s,a) + \gamma(\mathbb{E}_{a'\sim\pi_i}[Q_{\bar{\phi}_i}(s',a') - \alpha_i \log \pi_i(a'|s')])\right)^2] \\ where \quad i \in [1,2,\cdots,n,o] \tag{6}$$

where D represent the replay buffer, $\bar{\phi}_i$ is the parameter vector of the target state-action neural network. For the auxiliary policies, we design their loss function in the form of SAC:

$$L_{\pi_{aux}}(\boldsymbol{\theta}) = \mathbb{E}_{s\sim D}[\sum_i \mathbb{E}_{a\sim\pi_i}[\alpha_i \log(\pi_{\theta_i}(a|s) - Q_i(s,a))]] \\ where \quad i \in [1,2,\cdots,n] \tag{7}$$

In section 3.1, the upper bound of L_{π_o} have been demonstrated (More details can be seen in Appendix 3.1). Here, we will use this upper bound to optimize the target policy of the agent. And the optimization objective of policy can be replaced as:

$$L_{\pi_o}(\theta_o, \boldsymbol{\varphi}) = \mathbb{E}_{s\sim D}\left[\sum_i \lambda_{\varphi_i}(s,\bar{a}) D_{KL}(\pi_{\theta_o}(\cdot|s)||\pi_i(\cdot|s)) + \mathbb{E}_{a\sim\pi_o}[\log \sum \lambda_{\varphi_i}(s,a)\frac{\pi_i}{exp\,\frac{q_i}{\alpha_i}}]\right] \\ where \quad i \in [1,2,\cdots,n] \tag{8}$$

where $\boldsymbol{\varphi}$ is the parameter of λ calculator, and $\bar{a}$ is the mean of a Gaussian policy π_o, mentioned in section3.1. The optimization objective of the automatic temperature $\boldsymbol{\alpha}$ is shown in Eq:9.

$$L_\alpha(\boldsymbol{\alpha}) = \sum_i \mathbb{E}_{a\sim\pi_o}[-\alpha_i \log \pi_i(a|s) - \alpha_i \mathcal{H}] \tag{9}$$

where $\mathcal{H}$ is the target entropy. Through minimize Eq:9, the policy π_o can gradually satisfy the constraint $\mathbb{E}_{s\sim\rho^\pi, a\sim\pi}[-\log \pi(a|s)] \geq \mathcal{H}$, meanwhile, the auxiliary policies π_i will not be similar with π_o. We name the resulting actor-critic algorithm as Multi-Reward Fusion (MRF) and present the detailed procedure in Algorithm 1 and Fig.(2)

Algorithm 1 Multi-Reward Fusion (MRF)

Initialize parameter vectors ϕ, $\bar{\phi}$, θ_{aux}, θ_o, φ.
for each iteration **do**
  for each environment step **do**
    $a \sim \pi_{\theta_o}(\cdot|s)$
    $s' \sim p(\cdot|s,a)$
    $D \leftarrow D \cup \{(s,a,\mathbf{r}(s,a),s')\}$, where $\mathbf{r}(s,a) \in \mathbb{R}^{(n+1)\times 1}$
  end for
  for each gradient step **do**
    $\phi_i \leftarrow \phi_i - \beta_Q \nabla_\phi L_{Q_i}$ for $i \in \{1,2\}$
    $\theta_{aux} \leftarrow \theta_{aux} - \beta_{\pi_{aux}} \nabla_{\theta_{aux}} L_{\pi_{aux}}$
    Fix θ_o, then $\varphi \leftarrow \varphi - \beta_\varphi \nabla_\varphi (L_{\pi_o} + L_{reg})$
    Fix φ, then $\theta_o \leftarrow \theta_o - \beta_{\pi_o} \nabla_{\theta_o} L_{\pi_o}$
    $\alpha \leftarrow \alpha - \beta_\alpha \nabla_\alpha L_\alpha$
    $\bar{\phi} \leftarrow \tau\phi + (1-\tau)\bar{\phi}$
  end for
end for

where β_Q, $\beta_{\pi_{aux}}$, β_φ, β_{π_o}, β_α, τ are the hyperparameters of MRF, more details can be seen in Appendix B.

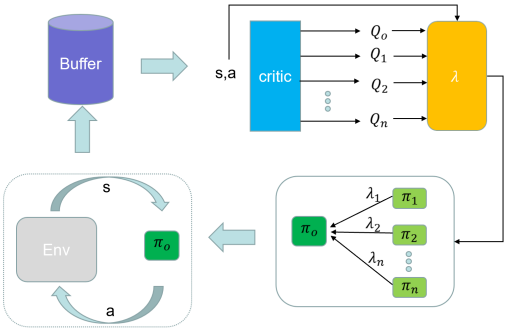


Figure 2: The architecture of MRF

4 EXPERIMENTS

4.1 EFFECTIVENESS OF COSINE SIMILARITY REGULARIZATION

This experiment we mainly discuss that if the similarity-based regularization proposed in section 3.3 is effective. To facilitate the research, we chose Random Walk task Sutton & Barto (2018) as the experimental scene. It is a simple discrete environment, where the agent only needs to choose turn left or turn right to approach the goal state. Except reaching the goal state where the agent will receive a $+100$ reward, each step the agent gets a reward -0.1 from the environment. The diagram of Random Walk environment is shown in Fig.(3). S_L and S_G are used to denote the leftmost terminal state and the goal state, respectively. We adopt the basic MRF algorithm as the base leaner, which do not set L_{reg} as the component of λ calculator's optimization objective. And compare this method's performance with the MRF which using similarity regularization.

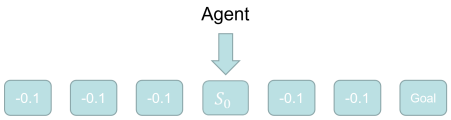


Figure 3: Random Walk Environment

Test Setting: The test of each method contains 1,000,000 training steps. During the training process, a 10-episode evaluation is conducted every 1,000 steps. The maximal length of an episode is 10. The shaping reward functions we designed is shown as follow:

$$
\begin{aligned}
r_1(s) &= -\|S_G - s\|_2 \\
r_2(s) &= \|S_L - s\|_2 \\
r_3(s) &= \|S_G - s\|_2 \\
\mathbf{r} &= r_o + [r_1, r_2, r_3, 0.0]^{\mathrm{T}}
\end{aligned}
\tag{10}
$$

where r_1 and r_2 encourage the agent approaching the goal state, r_3 is the interference term. Empirically speaking, r_1 is much direct than r_2. Because using r_2 may guide the agent unwilling to terminate the task.

Results: The performance of these two method are shown in Fig.(4)

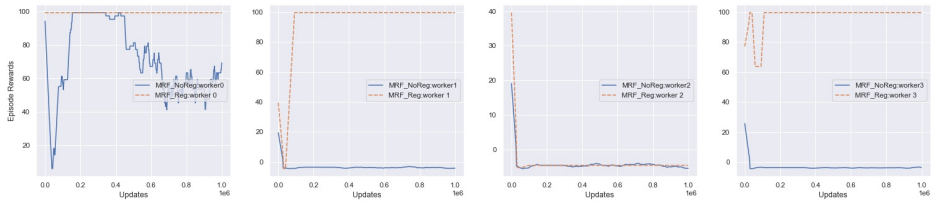
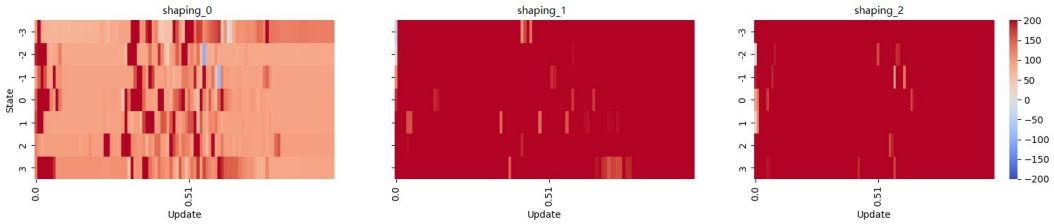


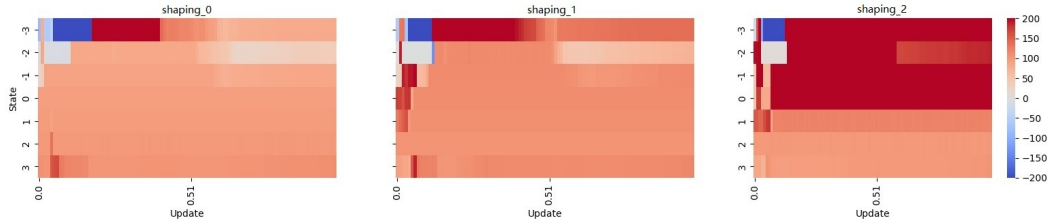
Figure 4: The training process of all perspectives policies

The 'worker:i' $i \in [0, 1, 2]$ corresponds to the multi-perspective auxiliary policies, and 'worker:3' is the behavior policy. Firstly, we recognize that different shaping rewards may result in different scales of the value function, as shown in Fig.(4.1). The scale difference would result in the λ calculator convergence difficulties, which have been mentioned in section 3.3. The phenomenon, demonstrated in Fig.(4.1) and Fig.(4.1), can also imply that the large gap between the input Q may make the $lamda$ calculator ineffective. With the huge difference of the input values, λ calculator can not make the right decision about which shaping reward needs to be attention.

After using the regularization we proposed, we can find that the λ calculator are easily to learn which policy needs to be distilled. It's worth mentioning that the mean similarity in Fig.(4.1) and Fig.(4.1) are calculated by Eq 5. It means that the higher mean similarity is, the corresponding auxiliary policy is more likely to be ignored, when we using the regularization.



(a) $V(s)$ of MRF without regularization



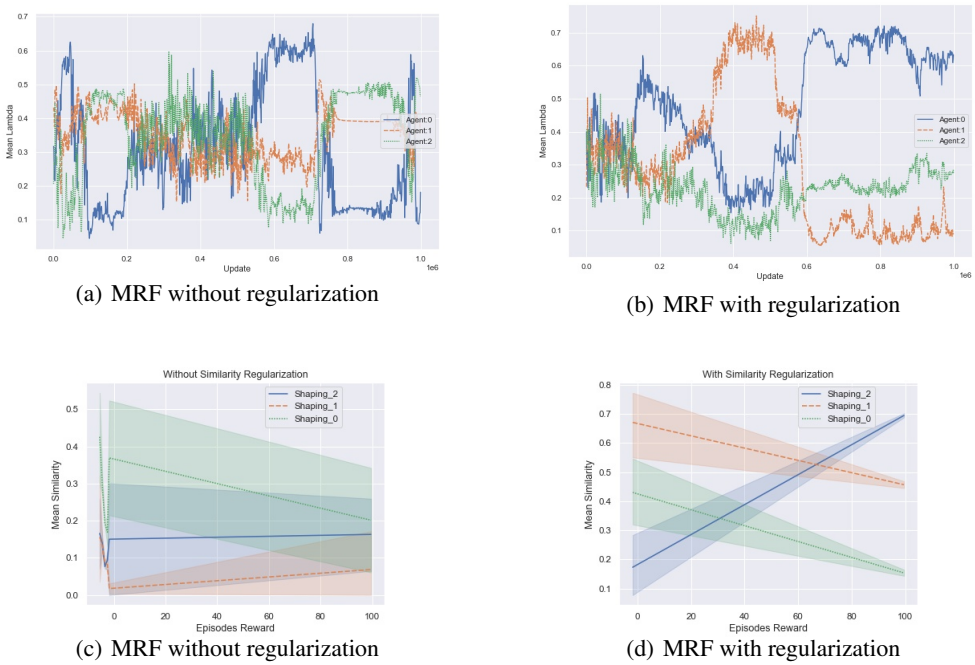
(b) $V(s)$ of MRF with regularization

Figure 5: $V(s)$ result from MRF with and without regularization, during 10^6 training process

4.2 Effect of the Shaping Rewards' Number

This part we will find the effect of the number of shaping rewards. We suspect that quantity will not directly affect performance. Because we regard the essence of reward shaping as introducing effective information, and the amount of effective information for a task is limited. We choose Hungry-Thirsty Singh et al. (2009) as the experiment environment, which isn't adequate to devise a good reward signal via the intuition alone.

In this environment, the agent has movement actions and two special actions available:
a) eat—with which the agent can consume food at the food location
b)drink—with which the agent can consume water at the water location
When the agent eats food, it becomes not-hungry for one time step, after which it becomes hungry



(a) MRF without regularization

(b) MRF with regularization

(c) MRF without regularization

(d) MRF with regularization

Figure 6: The effect of regularization on λ calculator

again. When the agent drinks water, it becomes not-thirsty for a random period of time (when not-thirsty, it becomes thirsty with probability 0.1 at each successive time step).

Test Setting: The maximal episode length is set as 200, and the training process lasts 1,000,000 update steps. The shaping rewards are designed as follow:

$$\begin{aligned} &\rho_0 = [\mathbb{1}_{eat}, \mathbb{1}_{drink}]^{\mathrm{T}}, \quad \rho_1 = [\mathbb{1}_{HT}, \mathbb{1}_{H\bar{T}}, \mathbb{1}_{\bar{H}T}, \mathbb{1}_{\bar{H}\bar{T}}]^{\mathrm{T}}, \\ &w_1 = [N_{eat}, N_{drink}]^{\mathrm{T}}, \; w_2 = [-0.05, -0.01, 1.0, 0.5]^{\mathrm{T}}, \\ &r_1 =< w_1, \rho_0 >, \; r_2 =< w_1, w_1 >, \; r_3 =< w_2, \rho_1 >, \\ &r_4 = -\|S_{eat} - s\|_2, \; r_5 = -\|S_{drink} - s\|_2, \end{aligned} \tag{11}$$

where N means the count of the events, namely eat food and drink water, subscript H and T represent the agent is hungry or thirsty, respectively. We mainly compare the performance of the MRF using different number of shaping rewards. We set MRF_0 (namely SAC without shaping rewards), MRF_5 ($r_o + [r_1, r_2, r_3, 0.0, 0.0]$), MRF_7 ($r_o + [r_1, r_2, r_3, r_4, r_5, 0.0, 0.0]$), where r_o is the original reward of the task. We usually set two '0.0' in the reward list. Because the multi-perspective rewards corresponds to $Q_1, Q_2 \cdots, Q_n, Q_o$, and we want to make it easy for the policy to be polarized to the auxiliary policy π_n under the constraint Eq. 1.

Results: The performance of MRF using different numbers of shaping rewards are shown in Fig(7). We can find that the introduce information can encourage the agent learning, but the performance gap between MRF_5 and MRF_7 is limited. It means that an appropriate amount of decoupled information can promote agent learning, but the effect of information introduced will become saturated as the number of shaping rewards increasing.

4.3 Performance of MRF

We believe that our method can show advantage in the problem of energy optimal control. Because the commonality of these problems is the sparse rewards, only at terminal state their rewards include the information about the completion of tasks. Hence, we choose two environments: Mountain Car and Lunar Lander. These tasks are all the continuous version in Gym (Brockman et al. (2016)) and

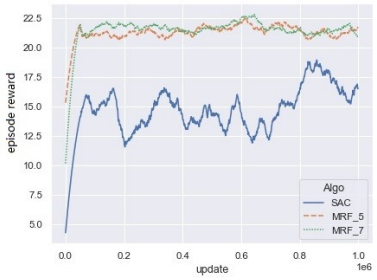
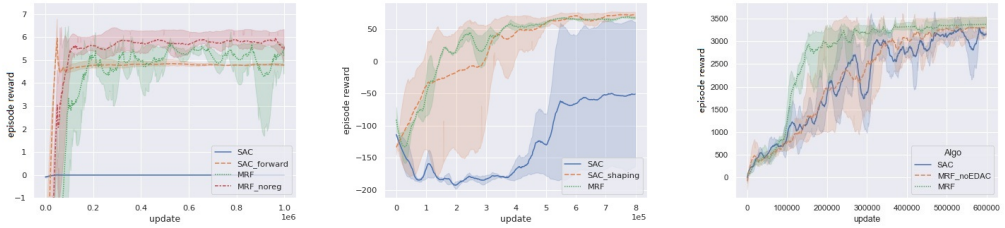


Figure 7: The performance of different shaping reward number

we have converted these tasks into the form of energy optimal control problem, namely the environment's rewards become sparse. The performance of MRF are shown in Fig.(4.3) and Fig.(4.3).



(a) Mountain Car (b) Lunar Lander (c) Hopper

Figure 8: The performance of MRF in different tasks

In Mountain Car, we provide two shaping rewards, the form of these rewards are likely to the form we mentioned in section 4.1. The total figure is shown in Appendix C. Here, for demonstrating the performance of our method, we don't exhibit the 'away' shaping reward's corresponding performance (whose episode reward converge to about -20). Combining Fig.(4.3) and Fig.(4.3), we can find that MRF can perform better than the traditional reward shaping and the basic SAC. However, the regularization imported causes some performance loss. It implies that when we have confidence of multi-perspective rewards' qualities, we'd better not use the regularization. Although, any form of MRF performs better than other methods.

We also test the performance of MRF in another style of tasks, like Hopper, where the rewards are dense enough that shaping rewards may be useless. When we face relatively complex tasks, the OOD problem may be gradually serious. We use the trick of EDAC An et al. (2021), and the performance is shown in Fig.(4.3). We can find that with the correction from EDAC our method can also improve the data efficiency.

5 Conclusion

In this work, we propose a novel reward shaping architecture, called the Multi-Reward Fusion (MRF), which learns the policy by distilling from a series of auxiliary policies. We formulate the relationship between auxiliary policies and the policy to be optimized, via the multi-perspective state-value function. Furthermore, we propose a gradient similarity-based regularization to reduce the influence of useless shaping rewards. The results in Mountain Car and some other tasks show that our algorithm can exploit the information from shaping rewards without deviating the optimization objective of the original task. Moreover, with the similarity regularization we proposed, the unbeneficial shaping rewards can be ignored.

REFERENCES

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.

Craig Boutilier, Richard Dearden, Moisés Goldszmidt, et al. Exploiting structure in policy construction. In *IJCAI*, volume 14, pp. 1104–1113, 1995.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

Sam Devlin and Daniel Kudenko. Theoretical considerations of potential-based reward shaping for multi-agent systems. In *The 10th International Conference on Autonomous Agents and Multiagent Systems*, pp. 225–232. ACM, 2011.

Sam Michael Devlin and Daniel Kudenko. Dynamic potential-based reward shaping. In *Proceedings of the 11th international conference on autonomous agents and multiagent systems*, pp. 433–440. IFAAMAS, 2012.

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

Zhao-Yang Fu, De-Chuan Zhan, Xin-Chun Li, and Yi-Xing Lu. Automatic successive reinforcement learning with multiple auxiliary rewards. In *IJCAI*, pp. 2336–2342, 2019.

Marek Grzes and Daniel Kudenko. Learning potential for reward shaping in reinforcement learning with tile coding. In *Proceedings AAMAS 2008 Workshop on Adaptive and Learning Agents and Multi-Agent Systems (ALAMAS-ALAg 2008)*, pp. 17–23, 2008.

Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. In *International conference on machine learning*, pp. 1352–1361. PMLR, 2017.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pp. 1861–1870. PMLR, 2018a.

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

Anna Harutyunyan, Sam Devlin, Peter Vrancx, and Ann Nowé. Expressing arbitrary reward functions as potential-based advice. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29, 2015.

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.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Guillaume Lample and Devendra Singh Chaplot. Playing fps games with deep reinforcement learning. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

Adam Laud and Gerald DeJong. The influence of reward on the speed of reinforcement learning: An analysis of shaping. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pp. 440–447, 2003.

Zichuan Lin, Derek Yang, Li Zhao, Tao Qin, Guangwen Yang, and Tie-Yan Liu. Rd˜2: Reward decomposition with representation decomposition. *Advances in Neural Information Processing Systems*, 33:11298–11308, 2020.

Marvin Marcus and Henryk Minc. *A survey of matrix theory and matrix inequalities*, volume 14. Courier Corporation, 1992.

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.

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.

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

Walter Rudin. Real and complex analysis (mcgraw-hill international editions: Mathematics series). 1987.

Satinder Singh, Richard L Lewis, and Andrew G Barto. Where do rewards come from. In *Proceedings of the annual conference of the cognitive science society*, pp. 2601–2606. Cognitive Science Society, 2009.

Fan-Yun Sun, Yen-Yu Chang, Yueh-Hua Wu, and Shou-De Lin. Designing non-greedy reinforcement learning agents with diminishing reward shaping. In *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*, pp. 297–302, 2018.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Richard S Sutton, Joseph Modayil, Michael Delp, Thomas Degris, Patrick M Pilarski, Adam White, and Doina Precup. Horde: A scalable real-time architecture for learning knowledge from unsupervised sensorimotor interaction. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pp. 761–768, 2011.

Lloyd N Trefethen and David Bau III. *Numerical linear algebra*, volume 50. Siam, 1997.

Harm Van Seijen, Mehdi Fatemi, Joshua Romoff, Romain Laroche, Tavian Barnes, and Jeffrey Tsang. Hybrid reward architecture for reinforcement learning. *Advances in Neural Information Processing Systems*, 30, 2017.

Li Wang, Yupeng Zhang, Yujing Hu, Weixun Wang, Chongjie Zhang, Yang Gao, Jianye Hao, Tangjie Lv, and Changjie Fan. Individual reward assisted multi-agent reinforcement learning. In *International Conference on Machine Learning*, pp. 23417–23432. PMLR, 2022.

Eric Wiewiora, Garrison W Cottrell, and Charles Elkan. Principled methods for advising reinforcement learning agents. In *Proceedings of the 20th international conference on machine learning (ICML-03)*, pp. 792–799, 2003.

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.

Runzhe Yang, Xingyuan Sun, and Karthik Narasimhan. A generalized algorithm for multi-objective reinforcement learning and policy adaptation. *Advances in Neural Information Processing Systems*, 32, 2019.

Haosheng Zou, Tongzheng Ren, Dong Yan, Hang Su, and Jun Zhu. Reward shaping via meta-learning. *arXiv preprint arXiv:1901.09330*, 2019.