Dynamics Adaptive Safe Reinforcement Learning with a Misspecified Simulator

Ruiqi Xue[1,2], Ziqian Zhang[1,2], Lihe Li[1,2], Feng Chen[1,2], Yi-Chen Li[1,2], Yang Yu[1,2,3], and Lei Yuan[1,2,3,⋆]

[1] National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China
[2] School of Artificial Intelligence, Nanjing University, Nanjing, China
[3] Polixir Technologies, Nanjing, China

Abstract. Sim-to-real reinforcement learning offers the advantage of learning safe policies within simulators, circumventing the need for costly trial-and-error in the real world. Traditional approaches often rest on the assumption of consistent state-action transition between the simulator and the real-world environment. However, this assumption can be violated due to the poor fidelity of simulators, leading to a constrained trust region for effective policy learning. The limitation can be more pronounced when safety issues are considered, potentially resulting in threatening policies if no safe samples exist in the trust region. To overcome these challenges, we propose **D**ynamics **A**daptive **Sa**fe **R**einforcement Learning with a Misspecified Simulator (DASaR). Our approach begins by relaxing the assumption to expand the trust region and theoretically demonstrate the unbounded performance gap inherent in traditional methods. Subsequently, DASaR aligns the estimated value functions in the simulator and the real-world environment via inverse dynamics-based relabeling of reward and cost signals. Furthermore, to deal with the underestimation of cost value functions, DASaR employs uncertainty estimation to improve its conservatism, ensuring the safety of the learned policy. Experiments in various complex environments thoroughly demonstrate DASaR's outstanding ability to balance safety satisfaction and reward maximization across diverse dynamics gaps.

Keywords: Safe Reinforcement Learning · Domain Adaptation · Sim-to-real Reinforcement Learning.

1 Introduction

Learning optimal policies through reinforcement learning (RL) has demonstrated outstanding performance in various fields, including game playing [36], recommendation systems [2], and robotic control [19]. Nonetheless, the application of RL in real-world settings, which often come with multiple constraints, presents the critical challenge of ensuring policy safety. For instance, while an RL agent

⋆ Lei Yuan is the corresponding author.

can maximize rewards by driving autonomous vehicles at high speeds, it must also comply with speed limits to prevent collisions [20]. Safe RL [14], which balances reward maximization and constraint satisfaction, can effectively improve the safety of the learned policies. However, the real-world environment is not amenable to such trial-and-error learning due to numerous costly errors caused during policy exploration [25]. To reduce the risk and cost in the real-world environment, the strategy of training policies within a high-fidelity simulator before deployment has emerged as a viable and efficient paradigm for safe RL.

However, it is challenging to construct high-fidelity simulators of systems with complex physical laws, leading to significant dynamics gaps between the simulator and the real-world environment [32]. Directly deploying policies learned within such simulators can result in substantial performance declines [18]. To tackle the problem, sim-to-real RL enhances the robustness of the policies against different dynamics [38]. Domain randomization creates augmented environments with different physical parameters to enhance policy generalization [28; 32], but it necessitates expert knowledge of simulators and the training complexity scales with the number of variations. Alternatively, domain adaptation presumes access to a limited set of offline real-world samples [46]. It strives for better usage of online simulator explorations by training classifiers to differentiate between the simulator and the real-world transitions. Such classifiers then aid in providing intrinsic rewards or filtering data, steering the policy towards regions where the simulator and real-world dynamics align closely [10; 30].

Despite the effective adaptation to dynamics changes of past domain adaptation approaches, they all assume that every possible transition in the real world will also have a non-zero probability in the simulator [11]. This enables the estimation of the policy's value function within the simulated environment as a bound for real-world returns. However, it can be violated due to poor fidelity of the simulator, leading to unbounded performance gaps. In this case, only transitions in the intersection between offline data and the support set of the dynamics transition function in the simulator, referred to as the trust region, can be fully leveraged for policy optimization. Furthermore, the lack of safe transitions within this trust region could lead to significant safety issues.

Addressing the mentioned challenges, this paper presents the approach of learning safe policies in misspecified simulators using a limited amount of offline data from the real-world environment. Specifically, we propose a novel algorithmic framework named **D**ynamics **A**daptive **Sa**fe **R**einforcement Learning with a Misspecified Simulator (DASaR). We first relax the assumption to the state-transition level, not only allowing the method to be applied to a broader range of simulators but also serving as a foundation for the extension of the trust region. Afterward, to avoid the unbounded performance gap, we align the estimated value functions in the simulator and the real-world environment via inverse dynamics-based relabeling. Finally, as the inaccuracy of inverse dynamics models leads to the underestimation of cost critics, we strengthen the conservatism via uncertainty estimation, thus ensuring the safety of the learned policy. Through extensive experiments on safe tasks within MuJoCo environments [41],

we demonstrate that our approach can achieve superior performance in balancing safety satisfaction and reward maximization under various dynamics gaps.

2 Related Work

2.1 Safe Reinforcement Learning

Safety has been one of the major roadblocks in the way of deploying RL policies to the real world [14; 15]. To solve the problem, Safe RL typically models the environment as a Constrained Markov Decision Process (CMDP) [3] and employs constrained optimization methods for policy learning [5; 14]. Lagrangian-based methods are traditional approaches to solving constrained optimization problems and are widely used in Safe RL. These methods utilize a learnable multiplier to penalize violations of constraints [7; 37; 39]. Additionally, trust region methods have been proposed to maintain policies within a safe trust region via low-order Taylor expansions [1; 16; 44] or variational inference [24; 45]. Despite their advantages, both Lagrangian-based and trust region methods encounter difficulties in preventing unsafe interactions during the trial-and-error phase. An emerging approach is offline safe RL, which aims to learn policies ensuring safety exclusively from offline data, thus avoiding unsafe explorations [22; 23; 25; 43]. However, the efficacy of offline safe RL heavily depends on the quantity and quality of offline data, restricting the application in limited offline data scenarios.

2.2 Sim-to-real Reinforcement Learning

To deal with the dynamics gap between the simulator and the real-world environment, sim-to-real RL studies the problems in two contexts based on the accessibility of real-world data. Without samples from real-world environments, zero-shot sim-to-real RL unleashes the potential of simulators to ensure the robustness of polices [28]. As a widely-used solution, domain randomization necessitates expert knowledge of the simulator and creates augmented environments by randomizing the physical parameters of the simulator, thus training policies robust to dynamics changes [27; 28; 29; 40]. When real-world samples can be obtained, different methods are developed in few-shot sim-to-real RL. Among them, simulator calibration methods improve the accuracy of simulators by adjusting physical parameters based on offline data [6; 9; 12; 34]. However, the requirement of expert knowledge is not always feasible. Meanwhile, some approaches align the dynamics between the simulator and the real-world environment via inverse dynamics model learned from offline data [8], but the generalization error emerges as a critical challenge. Alternatively, domain adaptation methods have gained widespread attention in recent years. DARC [11] employs two classifiers to distinguish transitions between the simulator and the real-world environment, providing intrinsic rewards to guide the policy. H2O [30] combines the conservative regularization term of CQL [21] with the classifiers, filtering simulator samples with a small dynamics gap for policy optimization. However, the assumption of transition dynamics limits their applicability and neither of the mentioned methods takes the safety issue into consideration.

3 Problem Formulation

We model the standard safe RL problem as a Constrained Markov Decision Process (CMDP) denoted as $\mathcal{M} = (S, A, P_{\mathcal{M}}, R, C, \rho, \gamma, b)$, where S and A represent the state space and the action space, $R : S \times A \times S \to [R_{\min}, R_{\max}]$ and $C : S \times A \times S \to \{0, 1\}$ denote the reward and cost functions respectively. $P_{\mathcal{M}} : S \times A \to \Delta S$ is the transition dynamics function. ρ represents the initial state distribution, $\gamma \in (0, 1)$ is the discount factor, and b represents the safety constraint limit. Specifically, for a deterministic CMDP $\mathcal{M}$ where there is no uncertainty on the next state s' given s and a, we can describe the transition dynamics with a deterministic transition function $f_{\mathcal{M}} : S \times A \to S$. For the given policy $\pi : S \mapsto \Delta(A)$ which specifies the action distribution on state s, the expected reward return and cost return within the CMDP $\mathcal{M}$ can be expressed as $J^R_{\mathcal{M}}(\pi) = \mathbb{E}_{s_0 \sim \rho, a_t \sim \pi(\cdot|s_t), s_{t+1} \sim P_{\mathcal{M}}(\cdot|s_t, a_t)}[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t, s_{t+1})]$ and $J^C_{\mathcal{M}}(\pi) = \mathbb{E}_{s_0 \sim \rho, a_t \sim \pi(\cdot|s_t), s_{t+1} \sim P_{\mathcal{M}}(\cdot|s_t, a_t)}[\sum_{t=0}^{\infty} \gamma^t C(s_t, a_t, s_{t+1})]$, respectively.

The safe sim-to-real RL problem involves two CMDPs: $\mathcal{M}_S$ representing the source domain (simulator) and $\mathcal{M}_T$ representing the target domain (real world). The main difference between $\mathcal{M}_S$ and $\mathcal{M}_T$ lies in their transition dynamics functions $P_{\mathcal{M}_S}$ and $P_{\mathcal{M}_T}$, which we abbreviate as P_S and P_T. The objective is using interactions in $\mathcal{M}_S$ together with a small number of offline data $\mathcal{D}_T$ sampled from $\mathcal{M}_T$, under the given reward function R and cost function C, to acquire a policy π that serves as the optimal solution to the following constrained optimization problem:

$$\begin{aligned} &\max_{\pi} \ J^R_{\mathcal{M}_T}(\pi), \\ &s.t. \ J^C_{\mathcal{M}_T}(\pi) \leq b. \end{aligned} \tag{1}$$

To facilitate later analysis, we introduce the *discounted stationary state transition occupancy* $d^{\pi}_{P_{\mathcal{M}}}(s, s') = (1-\gamma)\mathbb{E}_{\rho, \pi, P_{\mathcal{M}}}[\sum_{t=0}^{\infty} \gamma^t \mathbb{P}(s_t = s, s_{t+1} = s')]$, and the *discounted stationary state-action transition occupancy* $\mu^{\pi}_{P_{\mathcal{M}}}(s, a, s') = (1-\gamma)\mathbb{E}_{\rho, \pi, P_{\mathcal{M}}}[\sum_{t=0}^{\infty} \gamma^t \mathbb{P}(s_t = s, a_t = a, s_{t+1} = s')]$. Intuitively, they measure the overall frequency of visiting a specific state(-action) transition. For simplicity, we will omit "discounted stationary" throughout.

As is introduced in Section 2.2, domain adaptation methods make the common assumption that every possible state-action transition (s, a, s') in the target domain also has non-zero probability in the source domain. However, this could be violated due to the poor fidelity of the simulator. In this paper, we loosen it and make the following assumption:

Assumption 1 *For every possible state transition in the deterministic target domain, it has a non-zero probability in the deterministic source domain.*

$$\forall s, s' \in S, (\exists a \in A, P_T(s'|s, a) > 0 \Rightarrow \exists a' \in A, P_S(s'|s, a') > 0). \tag{2}$$

Intuitively, by changing the restriction from state-action transition space to state transition space, the assumption can be satisfied more easily, allowing for broader application. We here only study the problem of deterministic CMDPs

for that it is hard to capture the probabilistic information with a small offline dataset, further studies in non-deterministic settings are encouraged.

4 Method

In this section, we will provide a detailed description of our proposed DASaR. This algorithm is designed to learn safe and high-performance policies within simulators having dynamics gaps compared to real-world environments. All proofs and implementation details like the value of hyperparameters are provided in the appendix. Additional theoretical results on the performance bound of DASaR's policy and the broadening of trust region will also be provided in the appendix.

4.1 Theoretical Motivation

In safe sim-to-real RL, we aim to learn the policy with high performance and safety guarantees in the real-world with only an offline dataset and a simulator. While direct performance evaluations may not be feasible, we can evaluate the performance gap between policies in the simulator and the real-world environment, as demonstrated in Proposition 1.

Proposition 1. *For any two policies π_1, π_2 and two CMDPs $\mathcal{M}_S, \mathcal{M}_T$, the following hold for expected reward and cost returns of policies within CMDPs:*

$$J^R_{\mathcal{M}_T}(\pi_2) \geq J^R_{\mathcal{M}_S}(\pi_1) - \frac{\sqrt{2}R_{max}}{1-\gamma}\sqrt{\mathbb{D}_{KL}(\mu^{\pi_1}_{P_S}(s,a,s')||\mu^{\pi_2}_{P_T}(s,a,s'))}, \tag{3}$$

$$J^C_{\mathcal{M}_T}(\pi_2) \leq J^C_{\mathcal{M}_S}(\pi_1) + \frac{\sqrt{2}}{1-\gamma}\sqrt{\mathbb{D}_{KL}(\mu^{\pi_1}_{P_S}(s,a,s')||\mu^{\pi_2}_{P_T}(s,a,s'))}, \tag{4}$$

where $\mathbb{D}_{KL}(\cdot||\cdot)$ is the KL divergence, $\mu^{\pi_1}_{P_S}(s,a,s')$ and $\mu^{\pi_2}_{P_T}(s,a,s'))$ are state-action transition occupancies of π_1 and π_2 within $\mathcal{M}_S$ and $\mathcal{M}_P$, respectively.

Intuitively, with a well-performed policy π_1 in the simulator, we could enhance the performance of π_2 in the real world by minimizing the KL divergence between two state-action transition occupancies. The cost return of π_2 can be minimized in the same way. Since the traditional domain adaptation methods make the assumption that every possible state-action transition in the real world also has a non-zero probability in the simulator, the KL divergence term can be effectively minimized within a broad trust region. However, such traditional approaches will fail under the loosened Assumption 1. To facilitate further analysis, we introduce the concept of inverse dynamics probability first:

Definition 1. ***Inverse Dynamics Probability.*** *The inverse dynamics probability $\rho^{\pi}_{\mathcal{M}}(a|s,s')$ given the policy π and CMDP $\mathcal{M}$ is defined as:*

$$\rho^{\pi}_{\mathcal{M}}(a|s,s') := \frac{P_{\mathcal{M}}(s'|s,a)\pi(a|s)}{\int_A P_{\mathcal{M}}(s'|s,\bar{a})\pi(\bar{a}|s)d\bar{a}}. \tag{5}$$

As $\rho^{\pi}_{\mathcal{M}}$ is injective and independent of π when $\mathcal{M}$ is deterministic, we denote it as $\rho_{\mathcal{M}}$ for brevity. Meanwhile, we can further define the ***deterministic inverse dynamics function*** *$g_{\mathcal{M}}(s, s')$ given the inverse dynamics probability $\rho_{\mathcal{M}}$:*

$$\forall s, a, s', (g_{\mathcal{M}}(s, s') = a \iff \rho_{\mathcal{M}}(a|s, s') = \delta(0)), \tag{6}$$

where δ is the Dirac delta function [4].

With the definition of inverse dynamics probability function, we can further decompose the KL divergence term as is shown in Theorem 1:

Theorem 1. *For any two policies π_1 and π_2, and any two CMDPs $\mathcal{M}_S$ and $\mathcal{M}_T$, the following holds:*

$$\begin{aligned}\mathbb{D}_{KL}(\mu^{\pi_1}_{P_S}(s, a, s')||\mu^{\pi_2}_{P_T}(s, a, s')) = &\underbrace{\mathbb{D}_{KL}(d^{\pi_1}_{P_S}(s, s')||d^{\pi_2}_{P_T}(s, s'))}_{term(a)} \\ &+ \underbrace{\mathbb{E}_{(s,s')\sim d^{\pi_1}_{P_S}}\left[\mathbb{D}_{KL}(\rho_{\mathcal{M}_S}(a|s, s')||\rho_{\mathcal{M}_T}(a|s, s'))\right]}_{term(b)}.\end{aligned} \tag{7}$$

Theorem 1 decomposes the KL divergence term between the state-action transition occupancy into terms (a) and (b). Although term (a) can be optimized to 0 under Assumption 1, term (b) is an uncontrollable factor. This implies that all past domain adaptation approaches that estimate $J^R_{\mathcal{M}_S}(\pi)$ and $J^C_{\mathcal{M}_S}(\pi)$ in the simulator and minimize the KL divergence term $\mathbb{D}_{\mathrm{KL}}(\mu^{\pi_1}_{P_S}(s, a, s')||\mu^{\pi_2}_{P_T}(s, a, s'))$ will fail to bound the performance gap.

4.2 Value Estimation Alignment with an Inverse Dynamics Model

DASaR is a framework-agnostic module designed for learning safe policies, any off-policy safe RL methods can be combined with it. We here choose a widely-used safe RL algorithm SAC_Lagrange [33] for its simplicity and efficiency. Specifically, it learns the reward and cost state-action value functions via the maximum entropy Bellman operator and standard Bellman operator respectively:

$$\hat{Q}^R(s, a) = R(s, a, s') + \gamma\mathbb{E}_{a'\sim\pi(\cdot|s')}[\hat{Q}^R(s', a') - \xi\log(\pi(a'|s'))], \tag{8}$$

$$\hat{Q}^C(s, a) = C(s, a, s') + \gamma\mathbb{E}_{a'\sim\pi(\cdot|s')}[\hat{Q}^C(s', a')], \tag{9}$$

where ξ is a hyperparameter. Then $\mathbb{E}_{s\sim\rho,a\sim\pi(\cdot|s)}[\hat{Q}^R(s, a)]$, $\mathbb{E}_{s\sim\rho,a\sim\pi(\cdot|s)}[\hat{Q}^C(s, a)]$ are used as approximations of $J^R_{\mathcal{M}}(\pi)$ and $J^C_{\mathcal{M}}(\pi)$ to guide the policy update. As is discussed in Section 4.1, directly applying SAC_Lagrange in the simulator may lead to an unbounded performance gap in both reward and cost return under Assumption 1. However, if we can have access to the critic values within the target domain, the policy can be optimized to directly maximize the reward return while restricting its cost value when deployed in real-world environments.

The difficulty lies in the lack of reward and cost signals from the real world. As the reward and cost functions are available in simulators, we can relabel the signals of transition (s, a, s') into $R(s, a', s')$ and $C(s, a', s')$ if given the appropriate action $a' = g_{\mathcal{M}_T}(s, s')$, which implies that (s, a', s') is possible in real-world environments. Accordingly, we first learn the deterministic inverse dynamics model g_I to mimic $g_{\mathcal{M}_T}$ using samples from offline datasets D_T:

$$\min_{g_I} \mathbb{E}_{(s,a,s')\sim D_T}[||g_I(s, s') - a||_2^2]. \tag{10}$$

With the well-trained inverse dynamics model g_I, we can now relabel the reward and cost signals of transition (s, a, s'), which is sampled from the simulator, by defining the corrected reward and cost functions as follows:

$$r' = R'(s, a, s') = R(s, g_I(s, s'), s'), \tag{11}$$

$$c' = C'(s, a, s') = C(s, g_I(s, s'), s'). \tag{12}$$

The relabeled transition (s, a, r', c', s') will then be used for the updates of state-action value functions $\hat{Q}^R(s, a)$ and $\hat{Q}^C(s, a)$ as introduced in Equation (8) and (9), thus achieving the alignment of value estimation.

Although the relabeled transition (s, a, r', c', s') helps align the value functions $\hat{Q}^R(s, a)$ and $\hat{Q}^C(s, a)$, the action $a \sim \pi(\cdot|s)$ cannot lead to the transition of s' in real-world environments due to the dynamics gap. Instead, we utilize the learned inverse dynamics model to derive the desired action via the composition of two models $\pi \circ g_I$, which is defined as:

$$\pi \circ g_I(a|s) = \int_A \pi(a'|s)\delta(g_I(s, f_S(s, a')) - a)da', \tag{13}$$

where f_S is the deterministic state transition function of $\mathcal{M}_S$. Intuitively, we first roll out the policy π in the simulator to get $s' = f_S(s, a')|_{a'\sim\pi(\cdot|s)}$ and derive the executable action in the real world via the inverse dynamics model $a = g_I(s, s')$. In other words, the state-action value functions are equivalently used to estimate the performance of the policy $\pi \circ g_I$ in real-world environments. We then theoretically prove that both the reward and cost returns of such policies can be bounded and further justify the approach. To promote the analysis, we define the Inverse Dynamics Induced CMDP as follows:

Definition 2. ***Inverse Dynamics Induced CMDP****. Given a deterministic inverse dynamics model $g(s, s')$, it can induce a CMDP with deterministic transition dynamics function f satisfying the following property:*

$$\forall s, a, s', (g(s, s') = a \;\Leftrightarrow\; f(s, a) = s'). \tag{14}$$

The reward and cost functions of such inverse dynamics induced CMDP can be designated the same as those in the simulator or the real-world environment.

It can be shown that, with the inverse dynamics model g_I that mimics $g_{\mathcal{M}_T}$ perfectly, the CMDP $\mathcal{M}_I$ induced by g_I is exactly the same as the target domain $\mathcal{M}_T$. Furthermore, we illustrate that the optimization of policies under the

aligned value functions is consistent with optimization in the induced CMDP $\mathcal{M}_I$ in Theorem 2.

Theorem 2. *For a given inverse dynamics model g_I and its induced CMDP $\mathcal{M}_I$ with transition dynamics function P_I, the following equations hold: $d^{\pi}_{P_S}(s,s') = d^{\pi\circ g_I}_{P_I}(s,s')$, $J^{R'}_{\mathcal{M}_S}(\pi) = J^{R}_{\mathcal{M}_I}(\pi \circ g_I)$, $J^{C'}_{\mathcal{M}_S}(\pi) = J^{C}_{\mathcal{M}_I}(\pi \circ g_I)$, where $J^{R'}_{\mathcal{M}_S}(\pi)$ and $J^{C'}_{\mathcal{M}_S}(\pi)$ are the expected return of π under the simulator $\mathcal{M}_S$ evaluated by the corrected reward and cost functions R' and C', defined in Equation 11.*

Although Theorem 2 proves that we can optimize the policy $\pi \circ g_I$ under the induced CMDP M_I through updates in simulators, the differences between $\mathcal{M}_I$ and the target domain $\mathcal{M}_T$ will hinder the performance improvement in the real world. The differences result from the inaccuracy and generalization ability of g_I, and the coverage of offline samples. To deal with the challenge, we prove that the cost and reward returns can both be bounded by $\mathbb{D}_{\mathrm{KL}}(d^{\pi\circ g_I}_{P_I}(s,s')||d^{\pi_B}_{P_T}(s,s'))$ with the adequate assumption, which is equivalent to $\mathbb{D}_{\mathrm{KL}}(d^{\pi}_{P_S}(s,s')||d^{\pi_B}_{P_T}(s,s'))$ through Theorem 2. Here, π^B is the behavioral policy for collecting offline samples, corresponding proofs can be found in the appendix. Intuitively, this result comes from that g_I has higher accuracy on its training samples. Leveraging advancements in imitation from observation [42], we introduce a binary discriminator $K(s,s')$ which is optimized through:

$$\max_{K} \ \mathbb{E}_{d^{\pi}_{P_S}}[\ln K(s,s')] + \mathbb{E}_{d^{\pi_B}_{P_T}}[\ln(1-K(s,s'))]. \tag{15}$$

Following the paradigm widely used in imitation learning, we introduce the additional reward signal $\Delta r = \ln \frac{1-K(s,s')}{K(s,s')}$ and optimize $\hat{Q}^R$ with $r' + \alpha\Delta r$ instead, where α is a hyperparameter. In this way, we successfully expand past methods' trust region based on state-action transitions to a new trust region based on state transitions, covering the whole offline dataset under Assumption 1.

4.3 Conservative Cost Critic Learning via Uncertainty Estimation

Although the value estimation alignment via inverse dynamics-based relabeling can effectively bound the performance gap between the simulator and the real-world environment under Assumption 1, the safety issue could still emerge as a challenge. On one hand, the inherent approximation errors in neural networks will affect the estimation of the cost critic. On the other hand, the inaccuracy of the inverse dynamics model may lead to misleading cost relabel. These might result in underestimation of the cost critic, making the policy execute unsafe actions with low cost critic values during deployment. To alleviate the problem, we introduce uncertainty estimation using model ensemble approaches.

To deal with the approximation errors of the cost critic's network, we train E_C base models, denoted as $\{\hat{Q}^{C,i}(s,a)\}_{i=1}^{E_c}$. Then, we use the upper confidence bound (UCB) [31] as the estimation of the expected cost return under (s,a):

$$\hat{Q}^{C,\mathrm{UCB}}(s,a) = \mathbb{E}_{i\in\{1,\ldots,E_C\}}[\hat{Q}^{C,i}(s,a)] + \beta_C \cdot \sqrt{\mathrm{Var}_{i\in\{1,\ldots,E_C\}}[\hat{Q}^{C,i}(s,a)]}, \tag{16}$$

where β_C is a hyperparameter.

As for the inaccuracy of the inverse dynamics model, we first analyze how it leads to the underestimation of the cost critic. We update the cost critic with relabeled cost value $C(s, g_I(s,s'), s')$, with the hope that $f_T(s, g_I(s,s')) = s'$ when g_I perfectly mimics $g_{\mathcal{M}_T}$ within the real world $\mathcal{M}_T$. However, the inaccuracy of g_I might lead to undesired state transition to $f_T(s, g_I(s,s')) = s' + e$ where e is the state transition error. Meanwhile, the aligned cost critic evaluates the expected cost return of policy $\pi \circ g_I$ under state-action pair $(s, g_I(s,s'))$ in the real world. Accordingly, the sample supposed to be used to update the critic is $(s, g_I(s,s'), s'+e)$ instead. As the lack of transition dynamics f_T of the real-world environment prevents us from attaining the state error e, we introduce the proxy error by model ensemble and pessimistic estimation.

First of all, we can approximate the action error $e^A = a' - g_I(s,s')$ with regards to the desired action $a' = g_{\mathcal{M}_T}(s,s')$ via uncertainty estimation. Specifically, we apply model ensemble by training E_I inverse dynamics models $\{g_I^i\}_{i=1}^{E_I}$. Without loss of generality, we assume the existence of β_T such that for any s, s', the following holds:

$$e^A \in [-\beta_T \cdot \sqrt{\mathrm{Var}_{i\in\{1,\dots,E_I\}}[g_I^i(s,s')]}, \beta_T \cdot \sqrt{\mathrm{Var}_{i\in\{1,\dots,E_I\}}[g_I^i(s,s')]}\ ]. \tag{17}$$

After determining the range of the action error e^A, we can then decide the range of state error e based on the assumption on the continuity of f_T and f_S as follows:

Assumption 2 *Given two CMDPs $\mathcal{M}_S, \mathcal{M}_T$, together with their deterministic inverse dynamics functions g_S, g_T and deterministic state transition functions f_S, f_T, and the range of action error e_T^A in $\mathcal{M}_T$, there exists $\beta_S > 0$ so that the range of state error in $\mathcal{M}_T$ can be included by that in $\mathcal{M}_S$:*

$$\{f_T(s, g_T(s,s') + e_T^A) | e_T^A \in [-l, l]\} \subseteq \{f_S(s, g_S(s,s') + e_S) | e_S^A \in [-\beta_S \cdot l, \beta_S \cdot l]\}, \tag{18}$$

where l is the range of action errors in $\mathcal{M}_T$.

Intuitively, Assumption 2 tells that we can get all possible state errors within the real world by rollouts in the simulator, it is actually a direct result of Assumption 1. When f_T and f_S are continuous, the parameter β_S will not need to be a large number. Combining Equation 17 and Assumption 2, we derive the range of action errors under simulators: $E_S^A = [-\beta_I \cdot \sqrt{\mathrm{Var}_{i\in\{1,\dots,E_I\}}[g_I^i(s,s')]}, \beta_I \cdot \sqrt{\mathrm{Var}_{i\in\{1,\dots,E_I\}}[g_I^i(s,s')]}\ ]$, where $\beta_I = \beta_T \times \beta_S$ and we set it as a hyperparameter. Since it is impossible to give a certain value of action error, we make the pessimistic estimation and update the cost critic based on the term with the highest cost:

$$\mathcal{B}^e \hat{Q}^{C,i}(s,a) = \max_{e^A \in E_S^A} (C(s, g_I^E(s,s'), s^{e^A}) + \gamma \mathbb{E}_{a'\sim\pi(\cdot|s^e)}[\hat{Q}^{C,i}(s^{e^A}, a')]), \tag{19}$$

where $g_I^E(s,s') = \mathbb{E}_{i\in\{1,\dots,E_I\}}[g_I^i(s,s')]$ is the expectation over all the inverse dynamics models, and $s^{e^A} = f_S(s, g_I^E(s,s') + e^A)$ is the simulated state under

action with the error. By improving the conservatism of the cost critics, the safety of the learned policy can be further guaranteed.

5 Experiments

In this section, we present our experimental analysis conducted in four MuJoCo environments, each featuring three different simulators. The experiments are designed to address the following critical questions: (1) Can DASaR learn the policy that is both safe and high-performing in the real world via trial-and-error in different simulators (Sec. 5.2)? (2) What contributions do different components of DASaR make and how does the quantity of offline data impact its performance (Sec. 5.3)? (3) Does the policy learned by DASaR exhibit remarkable sim-to-real adaptation capabilities compared with other baselines (Sec. 5.4)?

For a thorough evaluation, we compare DASaR against multiple baselines. All experimental results within a simulator are averaged across ten evaluation episodes, five random seeds, and three different parameter values accompanied by standard deviation information. Detailed experimental information and additional results will be provided in the appendix.

5.1 Baselines and Environments

To thoroughly assess the performance of DASaR, we compare it against a range of baselines: (1) **SAC_Lagrange(sim)** [33] employs the Lagrangian version of SAC [17] to train a policy in the simulator and deploys it directly to the real world. (2) **SAC_Lagrange(id)** [8] also utilizes SAC_Lagrange for policy training in simulators but employs an inverse dynamics model learned from the offline dataset during real-world deployment. (3) **DARC_Lagrange** is the Lagrangian version of the sim-to-real approach, DARC [11], which utilizes two classifiers to differentiate transitions from the simulator and the real-world environment, thus constraining the policy learning to trust region with small dynamics gaps. (4) **H2O_Lagrange** also combines Lagrangian methods with H2O [30], which adds conservative regularization of CQL [21] into DARC's classifiers, filtering samples for policy updates. (5) **CPQ** [43] is a pure offline safe algorithm without explorations in simulators, emphasizing safety and conservatism via regularization.

For our evaluation, we chose four Gym MuJoCo environments [41]: `Ant`, `HalfCheetah`, `Hopper`, and `Walker`, and modified them for safety considerations. The offline dataset of each environment includes 100 trajectories, where 20 of them are collected from a safe policy and others are generated by diverse behavioral policies violating safety constraints. For each environment, we create three simulators, each differs from the real world in specific physical parameters, including `gravity`, `friction`, and `density`. The experiments are conducted within each simulator for three parameter values: 2.0, 1.5, and 0.5, indicating how much it deviates from the real-world parameter value 1.0.

Table 1. Average test return ± std across various environments. Rewards and costs are normalized via the offline dataset and safety constraint limit $b = 5$, respectively. If the normalized cost exceeds 1.0, the method with a lower cost is preferred, otherwise, the method with a higher reward is better. The method with the best performance in each simulator is emphasized in blue. Letters 'g', 'f', and 'd' represent simulators with different gravity, friction, and density parameter values compared with the real world.

Env		DASaR		SAC_Lagrange(id)		SAC_Lagrange(sim)	
		reward↑	cost↓	reward↑	cost↓	reward↑	cost↓
Ant	g	**72.8 ± 14.2**	**1.3 ± 1.2**	48.2 ± 16.5	3.4 ± 3.0	40.9 ± 29.3	5.9 ± 7.0
	f	**78.8 ± 8.8**	**1.1 ± 1.4**	52.8 ± 16.1	1.8 ± 1.4	51.6 ± 10.5	3.0 ± 2.2
	d	**97.0 ± 4.7**	**0.6 ± 0.5**	62.7 ± 16.5	1.1 ± 0.9	67.7 ± 20.9	2.5 ± 1.3
Cheetah	g	63.2 ± 8.4	1.7 ± 1.6	−184.9 ± 124.3	2.0 ± 2.6	**28.0 ± 63.6**	**1.2 ± 2.0**
	f	86.2 ± 8.5	1.2 ± 1.2	**-106.0 ± 171.6**	**0.8 ± 1.4**	65.1 ± 20.3	1.7 ± 1.9
	d	**83.3 ± 6.0**	**0.5 ± 0.5**	−215.9 ± 151.5	1.5 ± 2.7	61.4 ± 45.5	0.5 ± 0.8
Hopper	g	**44.5 ± 25.3**	**3.0 ± 2.9**	38.4 ± 19.0	5.2 ± 4.0	45.5 ± 19.1	7.5 ± 4.6
	f	**40.1 ± 26.7**	**4.6 ± 4.7**	35.5 ± 29.4	5.6 ± 6.2	49.5 ± 29.5	6.7 ± 4.1
	d	**59.2 ± 29.5**	**3.2 ± 3.2**	45.1 ± 20.2	4.8 ± 4.9	51.4 ± 18.0	6.5 ± 3.4
Walker	g	**62.5 ± 7.7**	**3.0 ± 4.1**	30.7 ± 29.6	7.0 ± 4.9	48.7 ± 18.9	5.3 ± 6.8
	f	**58.8 ± 23.8**	**2.5 ± 3.4**	21.6 ± 30.7	6.7 ± 5.3	73.1 ± 7.2	5.3 ± 4.3
	d	**65.4 ± 5.6**	**1.5 ± 1.8**	14.3 ± 30.5	6.3 ± 4.2	57.3 ± 21.0	8.1 ± 6.1
Overall		**67.7**	**2.0**	−13.1	3.9	53.4	4.5
		DARC_Lagrange		H2O_Lagrange		CPQ	
Ant	g	76.9 ± 33.5	3.6 ± 2.7	31.9 ± 28.3	13.7 ± 8.8	49.6 ± 6.0	9.7 ± 2.7
	f	97.2 ± 8.1	5.0 ± 2.6	56.3 ± 20.9	9.4 ± 6.6		
	d	111.2 ± 5.2	3.0 ± 1.7	27.4 ± 22.9	16.2 ± 8.1		
Cheetah	g	−21.3 ± 73.7	11.3 ± 6.7	−116.5 ± 228.1	46.5 ± 15.7	79.6 ± 3.7	8.3 ± 1.3
	f	2.5 ± 76.4	4.2 ± 9.2	−329.3 ± 0.3	60.0 ± 0.0		
	d	9.5 ± 72.4	1.5 ± 2.1	−301.0 ± 104.9	56.4 ± 13.4		
Hopper	g	63.5 ± 25.0	11.3 ± 10.9	22.3 ± 33.1	8.5 ± 10.8	98.0 ± 3.5	11.3 ± 2.6
	f	58.3 ± 37.5	11.1 ± 8.2	33.1 ± 52.0	11.6 ± 12.5		
	d	71.1 ± 32.0	11.8 ± 10.0	49.7 ± 42.0	13.1 ± 9.3		
Walker	g	65.5 ± 9.6	9.7 ± 11.3	45.5 ± 18.8	12.0 ± 4.5	72.7 ± 9.2	8.1 ± 3.1
	f	73.1 ± 3.1	5.5 ± 4.8	65.6 ± 16.5	16.8 ± 5.0		
	d	68.5 ± 7.3	12.5 ± 7.2	48.3 ± 17.1	17.0 ± 4.3		
Overall		56.3	7.5	−30.6	23.4	75.0	9.4

5.2 Overall Performance Comparison

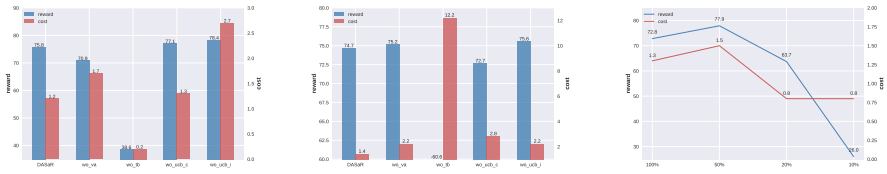
The detailed experimental results are shown in Table 1. Firstly, it is observed that in scenarios with limited and low-quality offline data, the pure offline safe algorithm CPQ severely violates safety constraints. This underscores the necessity to introduce online trial-and-error for data augmentation and policy learning.

When a certain dynamics gap exists in the simulator, SAC_Lagrange(sim) improves safety compared to CPQ but comes with a decrease in reward. Meanwhile, as it lacks theoretical guarantees, the performance of learned policies is highly dependent on the fidelity of simulators, emphasizing the importance of utilizing offline data to mitigate dynamics gaps. SAC_Lagrange(id) further incorporates an inverse dynamics model for adaptation in the real world, enhancing safety but causing a substantial drop in reward. This highlights the substantial negative impact of the inverse dynamics model's empirical and generalization errors on the policy.

Domain adaptation methods, DARC_Lagrange and H2O_Lagrange, fail to perform well when considering safety issues. While DARC_Lagrange achieves higher rewards in some settings, it significantly violates safety constraints. This validates the conflict between a trust region based on similar dynamics and safe data, with methods leaning towards the trust region as the primary constraint, tending to overlook safety constraints. H2O_Lagrange performs poorly in both reward and cost, diverging significantly compared to its outstanding performance in traditional environments without safety constraints. The issue arises from H2O's application of conservative regularization, which assigns transition samples with larger dynamics gaps lower rewards to avoid taking corresponding actions. However, in the safe scenario, the situation arises where samples with larger dynamics gaps, despite having lower rewards, also have lower estimated costs. Such actions are preferred compared to those with high costs in safe RL. Accordingly, these samples with larger dynamic gaps and low costs are used to update policy, leading to a significant performance gap.

Our method, DASaR, stands out by achieving the safest performance on most simulators while maintaining high reward returns. Compared to the safest baseline, DASaR improves safety performance by approximately 48%, and reward performance by approximately 80%. As for the approach with the highest reward, DASaR experiences a less than 10% decrease while achieving a remarkable 78% improvement in safety.

5.3 Ablation Studies and Data Sensitivity Study



(a) Ablation in Ant (b) Ablation in Cheetah (c) Data Sensitivity

Fig. 1. Ablation studies in two environments and data sensitivity study.

To illustrate the impact of different modules of DASaR on its policy performance, we conducted ablation studies in two environments: `Ant` and `HalfCheetah`. Results from two simulators parameterized with different gravity and friction are averaged for each environment. Detailed statistical data along with standard deviation information are provided in the appendix. We here present four different baselines: (1) **wo_va** does not use the inverse dynamics-based relabling. (2) **wo_tb** dismisses the discriminator reward signal. (3) **wo_ucb_c** uses the original cost critic value instead of UCB estimation. (4) **wo_ucb_i** applies the traditional Bellman operator into cost critic updating. The experimental results are shown in Fig. 1(a) and 1(b). It can be observed that wo_va, wo_ucb_c, and wo_ucb_i do not show significant differences in reward compared to DASaR, but they all exhibit an increase in cost, indicating the necessity of these modules for guaranteeing the safety of the learned policy. On the other hand, the performance of wo_tb varies between two environments. In Ant, while its reward decreases significantly, it achieves the lowest cost. However, in HalfCheetah, its reward drops while the cost increases significantly. This highlights the importance of constraining the policy within the trust region of higher accuracy inverse dynamics models for performance gap bounding.

Next, to verify the sensitivity of DASaR to the quantity of offline data, we conducted a data sensitivity study in the Ant environment's gravity simulator. We designated the original 100 trajectories as 100%, then reduced the data volume to 50%, 20%, and 10% while maintaining the same quality proportion, ensuring that only 20% of the data was sampled by the safe policy. As shown in Fig. 1(c), when the data volume is reduced from 100 trajectories to 50 or 20 trajectories, no significant undulation is observed. However, a significant performance decline happens when it is reduced to 10 trajectories, with only 2 of them safe. Despite the low cost, the failure in reward indicates the unbounded performance gap. This illustrates that DASaR exhibits adequate tolerance to reduced data volume, learning safe and high-performing policies under more limited offline data is promising in future work.

5.4 Visualization Analysis

To assess whether the policy learned in the simulator exhibits remarkable sim-to-real adaptation in the real world, we compare the state distribution induced by the policy in the simulator and the real-world environment. The state distribution of DASaR and other baselines in the Ant environment are visualized using t-SNE [26], as is shown in Fig. 2.

First of all, SAC_Lagrange(sim) displays notable discrepancies in state distribution between the simulator and the real world, indicating failure in sim-to-real adaptation. In contrast, SAC_Lagrange(id) shows increased similarity in state distribution compared to SAC_Lagrange(sim), despite that the discrepancy still exists. This underscores the necessity of employing sim-to-real techniques to enhance adaptation in the real-world environment. While H2O_Lagrange utilizes domain adaptation techniques, substantial disparities persist between simulated and real state distributions. This suggests the need for further adjustments in

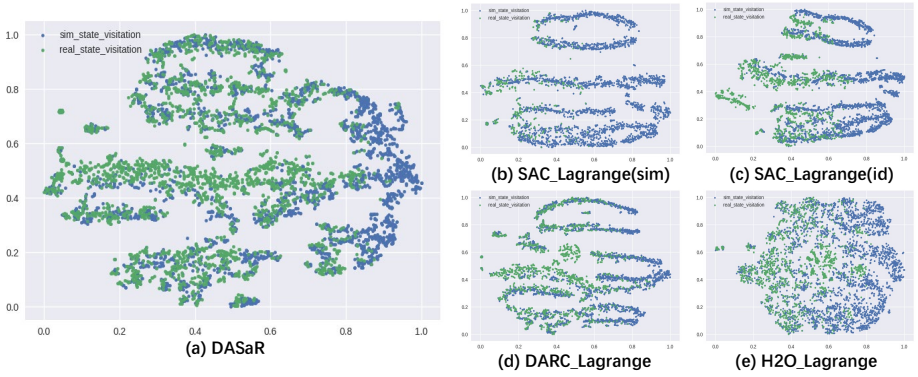


Fig. 2. Visualization of the state distributions in both simulator and the real.

safe RL. Both DASaR and DARC_Lagrange exhibit notably enhanced similarity in state distribution compared to other algorithms. This showcases the remarkable adaptation capabilities of these methods, which is in accordance with their higher performance in primary experiments. However, DARC_Lagrange's adaptation comes at the cost of compromising policy safety to some extent. Only DASaR achieves remarkable adaptation capabilities while maintaining the relative safety of the learned policy.

6 Final Remarks

In this work, we present a novel algorithm, DASaR, designed to tackle the challenge of training a policy in a simulator with a dynamics gap while ensuring safe performance in the real. DASaR aligns the value estimation from the simulator with the real by incorporating an inverse dynamics model, resulting in a lower performance gap across the entire offline data distribution where the inverse dynamics model exhibits high accuracy. Additionally, DASaR employs uncertainty estimation to robustly model the conservative cost critic, addressing neural network approximation errors and further enhancing safety performance. Extensive experimental results demonstrate DASaR's remarkable adaptation capabilities across various simulators. However, communication with the simulator during deployment introduces additional overhead, future work can further apply a behavior cloning process to solve this problem. Additionally, the applicability of DASaR is somewhat constrained by the presence of Assumption 1, and expanding its scope remains a promising direction for further exploration.

Bibliography

[1] Achiam, J., Held, D., Tamar, A., Abbeel, P.: Constrained policy optimization. In: ICML. pp. 22–31 (2017)

[2] Afsar, M.M., Crump, T., Far, B.: Reinforcement learning based recommender systems: A survey. ACM Computing Surveys **55**(7), 1–38 (2022)

[3] Altman, E.: Constrained Markov decision processes. Routledge (2021)

[4] Arfken, G.B., Weber, H.J., Harris, F.E.: Mathematical methods for physicists: a comprehensive guide. Academic press (2011)

[5] Brunke, L., Greeff, M., Hall, A.W., Yuan, Z., Zhou, S., Panerati, J., Schoellig, A.P.: Safe learning in robotics: From learning-based control to safe reinforcement learning. Annual Review of Control, Robotics, and Autonomous Systems **5**, 411–444 (2022)

[6] Chebotar, Y., Handa, A., Makoviychuk, V., Macklin, M., Issac, J., Ratliff, N., Fox, D.: Closing the sim-to-real loop: Adapting simulation randomization with real world experience. In: ICRA. pp. 8973–8979 (2019)

[7] Chow, Y., Ghavamzadeh, M., Janson, L., Pavone, M.: Risk-constrained reinforcement learning with percentile risk criteria. Journal of Machine Learning Research **18**(167), 1–51 (2018)

[8] Christiano, P., Shah, Z., Mordatch, I., Schneider, J., Blackwell, T., Tobin, J., Abbeel, P., Zaremba, W.: Transfer from simulation to real world through learning deep inverse dynamics model. arXiv preprint arXiv:1610.03518 (2016)

[9] Collins, J., Brown, R., Leitner, J., Howard, D.: Traversing the reality gap via simulator tuning. In: ACRA. pp. 1–10 (2021)

[10] Desai, S., Durugkar, I., Karnan, H., Warnell, G., Hanna, J., Stone, P.: An imitation from observation approach to transfer learning with dynamics mismatch. In: NeurIPS. pp. 3917–3929 (2020)

[11] Eysenbach, B., Asawa, S., Chaudhari, S., Levine, S., Salakhutdinov, R.: Off-dynamics reinforcement learning: Training for transfer with domain classifiers. arXiv preprint arXiv:2006.13916 (2020)

[12] Farchy, A., Barrett, S., MacAlpine, P., Stone, P.: Humanoid robots learning to walk faster: From the real world to simulation and back. In: AAMAS. pp. 39–46 (2013)

[13] Fedotov, A.A., Harremoës, P., Topsoe, F.: Refinements of pinsker's inequality. IEEE Transactions on Information Theory **49**(6), 1491–1498 (2003)

[14] Garcıa, J., Fernández, F.: A comprehensive survey on safe reinforcement learning. Journal of Machine Learning Research **16**(1), 1437–1480 (2015)

[15] Gu, S., Kuba, J.G., Chen, Y., Du, Y., Yang, L., Knoll, A., Yang, Y.: Safe multi-agent reinforcement learning for multi-robot control. Artificial Intelligence **319**, 103905 (2023)

[16] Gu, S., Kuba, J.G., Wen, M., Chen, R., Wang, Z., Tian, Z., Wang, J., Knoll, A., Yang, Y.: Multi-agent constrained policy optimisation. arXiv preprint arXiv:2110.02793 (2021)

[17] Haarnoja, T., Zhou, A., Abbeel, P., Levine, S.: Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In: ICML. pp. 1861–1870 (2018)
[18] Höfer, S., Bekris, K., Handa, A., Gamboa, J.C., Mozifian, M., Golemo, F., Atkeson, C., Fox, D., Goldberg, K., Leonard, J., et al.: Sim2real in robotics and automation: Applications and challenges. IEEE transactions on automation science and engineering **18**(2), 398–400 (2021)
[19] Ibarz, J., Tan, J., Finn, C., Kalakrishnan, M., Pastor, P., Levine, S.: How to train your robot with deep reinforcement learning: lessons we have learned. The International Journal of Robotics Research **40**(4-5), 698–721 (2021)
[20] Kiran, B.R., Sobh, I., Talpaert, V., Mannion, P., Al Sallab, A.A., Yogamani, S., Pérez, P.: Deep reinforcement learning for autonomous driving: A survey. IEEE Transactions on Intelligent Transportation Systems **23**(6), 4909–4926 (2021)
[21] Kumar, A., Zhou, A., Tucker, G., Levine, S.: Conservative q-learning for offline reinforcement learning. In: NeurIPS. pp. 1179–1191 (2020)
[22] Le, H., Voloshin, C., Yue, Y.: Batch policy learning under constraints. In: ICML. pp. 3703–3712 (2019)
[23] Lee, J., Paduraru, C., Mankowitz, D.J., Heess, N., Precup, D., Kim, K.E., Guez, A.: Coptidice: Offline constrained reinforcement learning via stationary distribution correction estimation. arXiv preprint arXiv:2204.08957 (2022)
[24] Liu, Z., Cen, Z., Isenbaev, V., Liu, W., Wu, S., Li, B., Zhao, D.: Constrained variational policy optimization for safe reinforcement learning. In: ICML. pp. 13644–13668 (2022)
[25] Liu, Z., Guo, Z., Yao, Y., Cen, Z., Yu, W., Zhang, T., Zhao, D.: Constrained decision transformer for offline safe reinforcement learning. arXiv preprint arXiv:2302.07351 (2023)
[26] Van der Maaten, L., Hinton, G.: Visualizing data using t-sne. The Journal of Machine Learning Research **9**(11) (2008)
[27] Mehta, B., Diaz, M., Golemo, F., Pal, C.J., Paull, L.: Active domain randomization. In: CoRL. pp. 1162–1176. PMLR (2020)
[28] Mordatch, I., Lowrey, K., Todorov, E.: Ensemble-cio: Full-body dynamic motion planning that transfers to physical humanoids. In: IROS. pp. 5307–5314 (2015)
[29] Nagabandi, A., Clavera, I., Liu, S., Fearing, R.S., Abbeel, P., Levine, S., Finn, C.: Learning to adapt in dynamic, real-world environments through meta-reinforcement learning. In: ICLR (2018)
[30] Niu, H., Qiu, Y., Li, M., Zhou, G., HU, J., Zhan, X., et al.: When to trust your simulator: Dynamics-aware hybrid offline-and-online reinforcement learning. In: NeurIPS. pp. 36599–36612 (2022)
[31] Osband, I., Blundell, C., Pritzel, A., Van Roy, B.: Deep exploration via bootstrapped dqn. In: NeurIPS. pp. 4026–4034 (2016)
[32] Peng, X.B., Andrychowicz, M., Zaremba, W., Abbeel, P.: Sim-to-real transfer of robotic control with dynamics randomization. In: ICRA. pp. 3803–3810 (2018)

[33] Ray, A., Achiam, J., Amodei, D.: Benchmarking safe exploration in deep reinforcement learning. arXiv preprint arXiv:1910.01708 (2019)
[34] Ren, A.Z., Dai, H., Burchfiel, B., Majumdar, A.: Adaptsim: Task-driven simulation adaptation for sim-to-real transfer. arXiv preprint arXiv:2302.04903 (2023)
[35] Schulman, J., Levine, S., Abbeel, P., Jordan, M., Moritz, P.: Trust region policy optimization. In: ICML. pp. 1889–1897 (2015)
[36] Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al.: Mastering the game of go without human knowledge. nature **550**(7676), 354–359 (2017)
[37] Stooke, A., Achiam, J., Abbeel, P.: Responsive safety in reinforcement learning by pid lagrangian methods. In: ICML. pp. 9133–9143 (2020)
[38] Tan, J., Zhang, T., Coumans, E., Iscen, A., Bai, Y., Hafner, D., Bohez, S., Vanhoucke, V.: Sim-to-real: Learning agile locomotion for quadruped robots. arXiv preprint arXiv:1804.10332 (2018)
[39] Tessler, C., Mankowitz, D.J., Mannor, S.: Reward constrained policy optimization. arXiv preprint arXiv:1805.11074 (2018)
[40] Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W., Abbeel, P.: Domain randomization for transferring deep neural networks from simulation to the real world. In: IROS. pp. 23–30 (2017)
[41] Todorov, E., Erez, T., Tassa, Y.: Mujoco: A physics engine for model-based control. In: 2012 IEEE/RSJ international conference on intelligent robots and systems. pp. 5026–5033. IEEE (2012)
[42] Torabi, F., Warnell, G., Stone, P.: Generative adversarial imitation from observation. arXiv preprint arXiv:1807.06158 (2018)
[43] Xu, H., Zhan, X., Zhu, X.: Constraints penalized q-learning for safe offline reinforcement learning. In: AAAI. pp. 8753–8760 (2022)
[44] Yang, T.Y., Rosca, J., Narasimhan, K., Ramadge, P.J.: Projection-based constrained policy optimization. In: ICLR (2019)
[45] Yao, Y., Liu, Z., Cen, Z., Zhu, J., Yu, W., Zhang, T., Zhao, D.: Constraint-conditioned policy optimization for versatile safe reinforcement learning. NeurIPS **36** (2024)
[46] Zhou, K., Liu, Z., Qiao, Y., Xiang, T., Loy, C.C.: Domain generalization: A survey. IEEE Transactions on Pattern Analysis and Machine Intelligence **45**(4), 4396–4415 (2022)

Appendix

A Proofs for Theoretical Analysis

Assumption 1 *For every possible state transition in the deterministic target domain, it has a non-zero probability in the deterministic source domain.*

$$\forall s, s' \in S, (\exists a \in A, P_T(s'|s,a) > 0 \Rightarrow \exists a' \in A, P_S(s'|s,a') > 0). \tag{20}$$

Assumption 2 *Given two CMDPs $\mathcal{M}_S, \mathcal{M}_T$, together with their deterministic inverse dynamics functions g_S, g_T and deterministic state transition functions f_S, f_T, and the range of action error e_T^A in $\mathcal{M}_T$, there exists $\beta_S > 0$ so that the range of state error in $\mathcal{M}_T$ can be included by that in $\mathcal{M}_S$:*

$$\{f_T(s, g_T(s,s') + e_T^A) | e_T^A \in [-l, l]\} \subseteq \{f_S(s, g_S(s,s') + e_S) | e_S^A \in [-\beta_S \cdot l, \beta_S \cdot l]\}, \tag{21}$$

where l is the range of action errors in $\mathcal{M}_T$.

A.1 Proofs in the Main Paper

Proposition 1. *For any two policies π_1, π_2 and two CMDPs $\mathcal{M}_S, \mathcal{M}_T$, the following hold for expected reward and cost returns of policies within CMDPs:*

$$J^R_{\mathcal{M}_T}(\pi_2) \geq J^R_{\mathcal{M}_S}(\pi_1) - \frac{\sqrt{2}R_{max}}{1-\gamma}\sqrt{\mathbb{D}_{KL}(\mu^{\pi_1}_{P_S}(s,a,s') || \mu^{\pi_2}_{P_T}(s,a,s'))}, \tag{22}$$

$$J^C_{\mathcal{M}_T}(\pi_2) \leq J^C_{\mathcal{M}_S}(\pi_1) + \frac{\sqrt{2}}{1-\gamma}\sqrt{\mathbb{D}_{KL}(\mu^{\pi_1}_{P_S}(s,a,s') || \mu^{\pi_2}_{P_T}(s,a,s'))}. \tag{23}$$

Proof. First, following the proofs in TRPO [35], we rewrite the expected reward return as $J^R_{\mathcal{M}}(\pi) = \frac{1}{1-\gamma}\mathbb{E}_{\mu^{\pi}_{P_{\mathcal{M}}}}[R(s,a,s')]$. Therefore, we have:

$$\begin{aligned}
|J^R_{\mathcal{M}_T}(\pi_2) - J^R_{\mathcal{M}_S}(\pi_1)| &= \frac{1}{1-\gamma}|\mathbb{E}_{\mu^{\pi_2}_{P_T}}[R(s,a,s')] - \mathbb{E}_{\mu^{\pi_1}_{P_S}}[R(s,a,s')]| \\
&= \frac{1}{1-\gamma}|\int_{S\times A\times S}(R(s,a,s')\mu^{\pi_2}_{P_T}(s,a,s') - R(s,a,s')\mu^{\pi_1}_{P_S}(s,a,s'))dsdads'| \\
&= \frac{1}{1-\gamma}|\int_{S\times A\times S} R(s,a,s')(\mu^{\pi_2}_{P_T}(s,a,s') - \mu^{\pi_1}_{P_S}(s,a,s'))dsdads'| \\
&\leq \frac{2R_{\max}}{1-\gamma}\mathbb{D}_{\mathrm{TV}}(\mu^{\pi_2}_{P_T}(s,a,s') || \mu^{\pi_1}_{P_S}(s,a,s')) \\
&= \frac{2R_{\max}}{1-\gamma}\mathbb{D}_{\mathrm{TV}}(\mu^{\pi_1}_{P_S}(s,a,s') || \mu^{\pi_2}_{P_T}(s,a,s')) \\
&\leq \frac{2R_{\max}}{1-\gamma}\sqrt{\frac{\mathbb{D}_{\mathrm{KL}}(\mu^{\pi_1}_{P_S}(s,a,s') || \mu^{\pi_2}_{P_T}(s,a,s'))}{2}} \\
&= \frac{\sqrt{2}R_{\max}}{1-\gamma}\sqrt{\mathbb{D}_{\mathrm{KL}}(\mu^{\pi_1}_{P_S}(s,a,s') || \mu^{\pi_2}_{P_T}(s,a,s'))}.
\end{aligned} \tag{24}$$

The first inequality is an application of Holder's inequality, achieved by setting p to ∞ and q to 1. The second inequality is an application of the Pinsker inequality [13]. $J^C_{\mathcal{M}}(\pi)$ follows the same reason.

Theorem 1. *For any two policies π_1 and π_2, and any two CMDPs $\mathcal{M}_S$ and $\mathcal{M}_T$, the following holds:*

$$\begin{aligned}\mathbb{D}_{KL}(\mu^{\pi_1}_{P_S}(s,a,s')||\mu^{\pi_2}_{P_T}(s,a,s')) = \mathbb{D}_{KL}(d^{\pi_1}_{P_S}(s,s')||d^{\pi_2}_{P_T}(s,s')) \\ +\mathbb{E}_{(s,s')\sim d^{\pi_1}_{P_S}}[\mathbb{D}_{KL}(\rho_{\mathcal{M}_S}(a|s,s')||\rho_{\mathcal{M}_T}(a|s,s'))].\end{aligned} \tag{25}$$

Proof. First, we have:

$$\begin{aligned}&\mathbb{D}_{\mathrm{KL}}(d^{\pi_1}_{P_S}(s,s')||d^{\pi_2}_{P_T}(s,s')) \\ &= \int_{S\times S} d^{\pi_1}_{P_S}(s,s')(\log\frac{d^{\pi_1}_{P_S}(s,s')}{d^{\pi_2}_{P_T}(s,s')})dsds' \\ &= \int_{S\times S}(\frac{\int_A \mu^{\pi_1}_{P_S}(s,a,s')da}{d^{\pi_1}_{P_S}(s,s')})d^{\pi_1}_{P_S}(s,s')(\log\frac{d^{\pi_1}_{P_S}(s,s')}{d^{\pi_2}_{P_T}(s,s')})dsds' \\ &= \int_{S\times S}(\int_A \frac{\mu^{\pi_1}_{P_S}(s,a,s')}{d^{\pi_1}_{P_S}(s,s')}da)d^{\pi_1}_{P_S}(s,s')(\log\frac{d^{\pi_1}_{P_S}(s,s')}{d^{\pi_2}_{P_T}(s,s')})dsds' \\ &= \int_{S\times A\times S}\frac{\mu^{\pi_1}_{P_S}(s,a,s')}{d^{\pi_1}_{P_S}(s,s')}d^{\pi_1}_{P_S}(s,s')(\log\frac{d^{\pi_1}_{P_S}(s,s')}{d^{\pi_2}_{P_T}(s,s')})dsdads' \\ &= \int_{S\times A\times S}\mu^{\pi_1}_{P_S}(s,a,s')(\log\frac{d^{\pi_1}_{P_S}(s,s')}{d^{\pi_2}_{P_T}(s,s')})dsdads'.\end{aligned} \tag{26}$$

With the equation, we can obtain:

$$\begin{aligned}&\mathbb{D}_{\mathrm{KL}}(\mu^{\pi_1}_{P_S}(s,a,s')||\mu^{\pi_2}_{P_T}(s,a,s')) - \mathbb{D}_{\mathrm{KL}}(d^{\pi_1}_{P_S}(s,s')||d^{\pi_2}_{P_T}(s,s')) \\ &= \int_{S\times A\times S}\mu^{\pi_1}_{P_S}(s,a,s')(\log\frac{\mu^{\pi_1}_{P_S}(s,a,s')}{\mu^{\pi_2}_{P_T}(s,a,s')}\times\frac{d^{\pi_2}_{P_T}(s,s')}{d^{\pi_1}_{P_S}(s,s')})dsdads' \\ &= \int_{S\times A\times S}\mu^{\pi_1}_{P_S}(s,a,s')(\log\frac{\rho_{\mathcal{M}_S}(a|s,s')}{\rho_{\mathcal{M}_T}(a|s,s')})dsdads' \\ &= \int_{S\times A\times S}\frac{\mu^{\pi_1}_{P_S}(s,a,s')}{\rho_{\mathcal{M}_S}(a|s,s')}\rho_{\mathcal{M}_S}(a|s,s')(\log\frac{\rho_{\mathcal{M}_S}(a|s,s')}{\rho_{\mathcal{M}_T}(a|s,s')})dsdads' \\ &= \int_{S\times A\times S}d^{\pi_1}_{P_S}(s,s')\rho_{\mathcal{M}_S}(a|s,s')(\log\frac{\rho_{\mathcal{M}_S}(a|s,s')}{\rho_{\mathcal{M}_T}(a|s,s')})dsdads' \\ &= \mathbb{E}_{(s,s')\sim d^{\pi_1}_{P_S}}[\mathbb{D}_{\mathrm{KL}}(\rho_{\mathcal{M}_S}(a|s,s')||\rho_{\mathcal{M}_T}(a|s,s'))].\end{aligned} \tag{27}$$

Finally, performing rearrangements, we have:

$$\begin{aligned}\mathbb{D}_{\mathrm{KL}}(\mu^{\pi_1}_{P_S}(s,a,s')||\mu^{\pi_2}_{P_T}(s,a,s')) = \mathbb{D}_{\mathrm{KL}}(d^{\pi_1}_{P_S}(s,s')||d^{\pi_2}_{P_T}(s,s')) \\ +\mathbb{E}_{(s,s')\sim d^{\pi_1}_{P_S}}[\mathbb{D}_{\mathrm{KL}}(\rho_{\mathcal{M}_S}(a|s,s')||\rho_{\mathcal{M}_T}(a|s,s'))].\end{aligned} \tag{28}$$

Lemma 1. *The state transition occupancy of $\pi\circ g_I$ on $\mathcal{M}_I$ is the same with that of π on $\mathcal{M}_S$, that is:*

$$d^{\pi}_{P_S}(s,s') = d^{\pi\circ g_I}_{P_I}(s,s'). \tag{29}$$

Proof. We prove this by induction.

First of all, due to the problem formulation, $\mathcal{M}_S$ and $\mathcal{M}_T$ have the same initial state distribution ρ, therefore, $\mathcal{M}_S$ and $\mathcal{M}_I$ also have the same initial state distribution ρ.

Then, given the same current state s, suppose the next state distribution of $\pi \circ g_I$ in $\mathcal{M}_I$ is $p_I^{\pi \circ g_I}(s'|s)$, and the next state distribution of π in $\mathcal{M}_S$ is $p_S^{\pi}(s'|s)$, then the following holds:

$$
\begin{aligned}
p_I^{\pi \circ g_I}(s'|s) &= \int_A \pi \circ g_I(a|s) P_I(s'|s,a) da \\
&= \int_A \pi \circ g_I(a|s) \delta(s' - f_I(s,a)) da \\
&= \int_A \pi \circ g_I(a|s) \delta(0) \mathbb{I}(f_I(s,a) = s') da \\
&= \int_A \int_A \pi(a'|s) \delta(0) \mathbb{I}(g_I(s, f_S(s,a')) = a) \delta(0) \mathbb{I}(f_I(s,a) = s') da' da \\
&= \int_A \int_A \pi(a'|s) \delta(0) \mathbb{I}(f_S(s,a') = f_I(s,a)) \delta(0) \mathbb{I}(f_I(s,a) = s') da' da \\
&= \int_A \int_A \pi(a'|s) \delta(0) \mathbb{I}(f_S(s,a') = s') \delta(0) \mathbb{I}(f_I(s,a) = s') da' da \\
&= \int_A \pi(a'|s) \delta(0) \mathbb{I}(f_S(s,a') = s') da' \int_A \delta(0) \mathbb{I}(f_I(s,a) = s') da \\
&= \int_A \pi(a'|s) \delta(s' - f_S(s,a)) da' \\
&= p_S^{\pi}(s'|s).
\end{aligned} \tag{30}
$$

Therefore, by induction we have $d_{P_S}^{\pi_s}(s,s') = d_{P_I}^{\pi_s \circ g_I}(s,s')$.

Theorem 2. *For a given inverse dynamics model g_I and its induced CMDP $\mathcal{M}_I$ with transition dynamics function P_I, the following equations hold: $d_{P_S}^{\pi}(s,s') = d_{P_I}^{\pi \circ g_I}(s,s')$, $J_{\mathcal{M}_S}^{R'}(\pi) = J_{\mathcal{M}_I}^{R}(\pi \circ g_I)$, $J_{\mathcal{M}_S}^{C'}(\pi) = J_{\mathcal{M}_I}^{C}(\pi \circ g_I)$, where $J_{\mathcal{M}_S}^{R'}(\pi)$ and $J_{\mathcal{M}_S}^{C'}(\pi)$ are the expected return of π under the simulator $\mathcal{M}_S$ evaluated by the corrected reward and cost functions R' and C'.*

Proof. Lemma 1 has proved that $d_{P_S}^{\pi}(s,s') = d_{P_I}^{\pi \circ g_I}(s,s')$. Therefore, we continue to prove the rest. Due to the definition, $J_{\mathcal{M}_S}^{R'}(\pi)$ can be expressed as $J_{\mathcal{M}_S}^{R'}(\pi) =$

$\frac{1}{1-\gamma}\mathbb{E}_{(s,s')\sim d^{\pi}_{P_S}(s,s'),a'\sim\rho_{M_I}(s,s')}[R(s,a',s')]$. Therefore, we have:

$$\begin{aligned} J^{R'}_{\mathcal{M}_S}(\pi) &= \frac{1}{1-\gamma}\mathbb{E}_{(s,s')\sim d^{\pi}_{P_S}(s,s'),a'\sim\rho_{M_I}(a'|s,s')}[R(s,a',s')] \\ &= \frac{1}{1-\gamma}\int_{S\times A\times S} R(s,a',s')d^{\pi}_{P_S}(s,s')\rho_{M_I}(a'|s,s')dsda'ds' \\ &= \frac{1}{1-\gamma}\int_{S\times A\times S} R(s,a',s')d^{\pi\circ g_I}_{P_I}(s,s')\rho_{M_I}(a'|s,s')dsda'ds' \\ &= \frac{1}{1-\gamma}\int_{S\times A\times S} R(s,a',s')\mu^{\pi\circ g_I}_{P_I}(s,a',s')dsda'ds' \\ &= J^{R}_{\mathcal{M}_I}(\pi\circ g_I), \end{aligned} \tag{31}$$

where the third equality holds because of Lemma 1. Similarly, we have $J^{C'}_{\mathcal{M}_S}(\pi) = J^{C}_{\mathcal{M}_I}(\pi\circ g_I)$.

A.2 More Theoretical Results

To obtain a performance bound for DASaR, we first make the following assumption:

Assumption 3 *The inverse dynamics model g_I has zero empirical error when sufficiently optimized. And let π_B be the behavior policy of the offline dataset $\mathcal{D}_T$, for any policy π and $\epsilon \geq 0$, if it satisfies $\mathbb{D}_{KL}(d^{\pi}_{P_I}(s,s')||d^{\pi_B}_{P_T}(s,s')) \leq \epsilon$, then its state-action transition occupancy satisfies $\mathbb{D}_{KL}(\mu^{\pi}_{P_I}(s,a,s')||\mu^{\pi}_{P_T}(s,a,s')) \leq k\epsilon^2$, where k is a positive constant.*

This assumption actually assumes that when the state visitation distribution of the policy π is similar to the behavior policy π_B, there will be more accurate inference of the inverse dynamics model, thus $\mathcal{M}_I$ will be more similar to $\mathcal{M}_T$ in such areas. This is an intuitive result at least when $\epsilon = 0$. For $\epsilon > 0$, a large enough k can make it possible. Then, we have the following results:

Lemma 2. *The optimal $K^*(s,s')$ satisfies the following condition:*

$$\ln\frac{1-K^*(s,s')}{K^*(s,s')} = \ln\frac{d^{\pi_B}_{P_T}}{d^{\pi}_{P_I}}. \tag{32}$$

Proof. Let $J(K,\pi) = \mathbb{E}_{d^{\pi}_{P_I}}[\ln K(s,s')] + \mathbb{E}_{d^{\pi_B}_{P_T}}[\ln(1-K(s,s'))]$, then we have:

$$\begin{aligned} J(K,\pi) &= \mathbb{E}_{d^{\pi}_{P_I}}[\ln K(s,s')] + \mathbb{E}_{d^{\pi_B}_{P_T}}[\ln(1-K(s,s'))] \\ &= \int_{S\times S}(\ln K(s,s')d^{\pi}_{P_I}(s,s') + \ln(1-K(s,s'))d^{\pi_B}_{P_T}(s,s'))dsds'. \end{aligned} \tag{33}$$

Therefore, to maximize $J(K,\pi)$, $\ln K(s,s')d^{\pi}_{P_I}(s,s') + \ln(1-K(s,s'))d^{\pi_B}_{P_T}(s,s')$ needs to be maximized at every (s,s'). Let $x = K(s,s')$, $f(x) = \ln(x)d^{\pi}_{P_I}(s,s') +$

$\ln(1-x)d_{P_T}^{\pi_B}(s,s')$, then we have:

$$\frac{\partial f(x)}{\partial x} = \frac{d_{P_I}^{\pi}(s,s')}{x} - \frac{d_{P_T}^{\pi_B}(s,s')}{1-x}. \tag{34}$$

It is clear that $\frac{\partial f(x)}{\partial x}$ is a decreasing function for $x \in (0,1)$, therefore the optimal x^* satisfies:

$$\frac{x^*}{1-x^*} = \frac{d_{P_I}^{\pi}(s,s')}{d_{P_T}^{\pi_B}(s,s')}, \tag{35}$$

as a result:

$$\ln \frac{1-K^*(s,s')}{K^*(s,s')} = \ln \frac{d_{P_T}^{\pi_B}}{d_{P_I}^{\pi}}. \tag{36}$$

Proposition 2. *For any $\epsilon > 0$, there exists $\alpha > 0$ such that using a standard safe RL algorithm to optimize in the simulator using $r' + \alpha\Delta r$ and c', the theoretically optimal policy π is the solution to the following optimization problem:*

$$\begin{aligned} &\max_{\pi} J_{\mathcal{M}_I}^{R}(\pi \circ g_I) \\ &s.t.\ J_{\mathcal{M}_I}^{C}(\pi \circ g_I) \le b, \\ &\quad \mathbb{D}_{KL}(d_{P_I}^{\pi \circ g_I}(s,s') || d_{P_T}^{\pi_B}(s,s')) \le \epsilon. \end{aligned} \tag{37}$$

Proof. Firstly, due to Theorem 2, employing standard safe RL algorithms for policy optimization based on r' and c' in the simulator results in the theoretically optimal policy π which optimizes that:

$$\begin{aligned} &\max_{\pi} J_{\mathcal{M}_I}^{R}(\pi \circ g_I) \\ &s.t.\ J_{\mathcal{M}_I}^{C}(\pi \circ g_I) \le b. \end{aligned} \tag{38}$$

Next, for that $\mathbb{D}_{\mathrm{KL}}(d_{P_S}^{\pi}(s,s') || d_{P_T}^{\pi_B}(s,s')) = \mathbb{E}_{d_{P_S}^{\pi}(s,s')}[\ln \frac{d_{P_S}^{\pi}}{d_{P_T}^{\pi_B}}]$, by applying the Lagrange multiplier method and Lemma 2, it follows that for any $\epsilon > 0$, there exists $\alpha > 0$ such that using a standard safe RL algorithm to optimize in the simulator using $r' + \alpha\Delta r$ and c', the theoretically optimal policy π optimizes:

$$\begin{aligned} &\max_{\pi} J_{\mathcal{M}_I}^{R}(\pi \circ g_I) \\ &s.t.\ J_{\mathcal{M}_I}^{C}(\pi \circ g_I) \le b, \\ &\quad \mathbb{D}_{\mathrm{KL}}(d_{P_S}^{\pi}(s,s') || d_{P_T}^{\pi_B}(s,s')) \le \epsilon. \end{aligned} \tag{39}$$

Furthermore, due to Assumption 1, it is evident that any policy that entirely constrains the state transition occupancy measure to the offline dataset is a solution satisfying $\mathbb{D}_{\mathrm{KL}}(d_{P_S}^{\pi}(s,s') || d_{P_T}^{\pi_B}(s,s')) = 0 \le \epsilon$. Additionally, since there exist trajectories in the offline dataset that satisfy safety constraints, this optimization problem is feasible. Ultimately, applying Lemma 1, it is concluded that π

optimizes:

$$\begin{aligned} &\max_{\pi} J^{R}_{\mathcal{M}_I}(\pi \circ g_I) \\ &s.t.\ J^{C}_{\mathcal{M}_I}(\pi \circ g_I) \leq b, \\ &\quad \mathbb{D}_{\mathrm{KL}}(d^{\pi \circ g_I}_{P_I}(s,s') || d^{\pi_B}_{P_T}(s,s')) \leq \epsilon. \end{aligned} \tag{40}$$

Theorem 3. *Let $J^{R'}_{\mathcal{M}_S}(\pi)$ and $J^{C'}_{\mathcal{M}_S}(\pi)$ be the expected value returns of the policy π in $\mathcal{M}_S$ evaluated by the corrected reward and cost functions R' and C', g_I be the given inverse dynamics model. When Assumption 3 holds, for any $\epsilon > 0$, there exists $\alpha > 0$ such that using a standard safe RL algorithm to optimize in the simulator using $r' + \alpha \Delta r$ and c', the theoretically optimal policy π obtained meets the following condition for expected value return in the real world:*

$$J^{R}_{\mathcal{M}_T}(\pi \circ g_I) \geq J^{R'}_{\mathcal{M}_S}(\pi) - \frac{\sqrt{2k}\epsilon R_{max}}{1-\gamma}, \tag{41}$$

$$J^{C}_{\mathcal{M}_T}(\pi \circ g_I) \leq J^{C'}_{\mathcal{M}_S}(\pi) + \frac{\sqrt{2k}\epsilon}{1-\gamma} \leq b + \frac{\sqrt{2k}\epsilon}{1-\gamma}. \tag{42}$$

Proof. This is a direct result combining Assumption 3, Proposition 1, Proposition 2, and Theorem 2. Due to Proposition 2, for any $\epsilon > 0$, there exists $\alpha > 0$, such that $\mathbb{D}_{\mathrm{KL}}(d^{\pi \circ g_I}_{P_I}(s,s') || d^{\pi_B}_{P_T}(s,s')) \leq \epsilon$. By Assumption 3, we have $\mathbb{D}_{\mathrm{KL}}(\mu^{\pi}_{P_I}(s,a,s') || \mu^{\pi}_{P_T}(s,a,s')) \leq k\epsilon^2$. Substituting this expression into Proposition 1 yields:

$$J^{R}_{\mathcal{M}_T}(\pi \circ g_I) \geq J^{R}_{\mathcal{M}_I}(\pi \circ g_I) - \frac{\sqrt{2k}\epsilon R_{\max}}{1-\gamma}, \tag{43}$$

$$J^{C}_{\mathcal{M}_T}(\pi \circ g_I) \leq J^{C}_{\mathcal{M}_I}(\pi \circ g_I) + \frac{\sqrt{2k}\epsilon}{1-\gamma}. \tag{44}$$

Using Proposition 2 again, we have:

$$J^{R}_{\mathcal{M}_T}(\pi \circ g_I) \geq J^{R}_{\mathcal{M}_I}(\pi \circ g_I) - \frac{\sqrt{2k}\epsilon R_{\max}}{1-\gamma}, \tag{45}$$

$$J^{C}_{\mathcal{M}_T}(\pi \circ g_I) \leq J^{C}_{\mathcal{M}_I}(\pi \circ g_I) + \frac{\sqrt{2k}\epsilon}{1-\gamma} \leq b + \frac{\sqrt{2k}\epsilon}{1-\gamma}. \tag{46}$$

Finally, combining Theorem 2, we have:

$$J^{R}_{\mathcal{M}_T}(\pi \circ g_I) \geq J^{R'}_{\mathcal{M}_S}(\pi) - \frac{\sqrt{2k}\epsilon R_{\max}}{1-\gamma}, \tag{47}$$

$$J^{C}_{\mathcal{M}_T}(\pi \circ g_I) \leq J^{C'}_{\mathcal{M}_S}(\pi) + \frac{\sqrt{2k}\epsilon}{1-\gamma} \leq b + \frac{\sqrt{2k}\epsilon}{1-\gamma}. \tag{48}$$

Theorem 3 provides bounds on the expected value return of the policy deployed in the real world under Assumption 3. Combining the results of Theorem 2 and Proposition 2, it can be inferred that, given ϵ, DASaR optimizes the lower bound of the expected reward return and the upper bound of the expected cost return in the real world. The value of ϵ is directly related to the hyperparameter α; when α is large, while ϵ is small, the search space for the policy π will also be smaller. Conversely, when α is small, while ϵ is large, the search space for the policy π will also be larger. Therefore, the choice of this hyperparameter is one of the key factors influencing performance.

Additionally, we theoretically show that the effective training data of DASaR for deterministic CMDPs will not be less than that of the past domain adaptation methods even when Assumption 1 does not hold. First, we formally define the following trust regions:

- The trust region based on state-action transitions $\mathcal{T}_{sa}$: $\forall(s,a,s') \in \mathcal{T}_{sa}$, $P_S(s'|s,a) > 0,\ P_T(s'|s,a) > 0$, and the action probability $p(a) > 0$.
- The trust region based on state transitions $\mathcal{T}_s$: $\forall(s,s') \in \mathcal{T}_s$, let $p_S(s'|s)$ and $p_T(s'|s)$ denote the probability of state transition (s,s') in the simulator and the real-world, then $p_S(s'|s) > 0$ and $p_T(s'|s) > 0$.

Theorem 4. *$\mathcal{T}_{sa}$ is the trust region based on state-action transitions, $\mathcal{T}_s$ is the trust region based on state transitions, and $\mathcal{D}_T$ is the offline dataset. Then*

$$(s,a,s') \in \mathcal{T}_{sa} \bigcap \mathcal{D}_T \Rightarrow (s,s') \in \mathcal{T}_s \bigcap \mathcal{D}_T. \tag{49}$$

Proof. For any transition $(s,a,s') \in \mathcal{T}_{sa}$, due to the definition, $P_S(s'|s,a) > 0,\ P_T(s'|s,a) > 0$, also $p(a) > 0$. Therefore, we have:

$$\begin{aligned} p_S(s'|s) &= \int_A P_S(s',a'|s)da' \\ &= \int_A P_S(s'|s,a')p(a')da' \\ &> P_S(s'|s,a)p(a) \\ &> 0. \end{aligned} \tag{50}$$

Similarly, we have $p_T(s'|s) > 0$. Which means (s,s') can occur in both the simulator and the real-world environment. Therefore, $(s,s') \in \mathcal{T}_s$. Thus, we have

$$(s,a,s') \in \mathcal{T}_{sa} \Rightarrow (s,s') \in \mathcal{T}_s. \tag{51}$$

Further, we can obtain that:

$$(s,a,s') \in \mathcal{T}_{sa} \bigcap \mathcal{D}_T \Rightarrow (s,s') \in \mathcal{T}_s \bigcap \mathcal{D}_T. \tag{52}$$

Also, we will provide an example yo show that for a real-world environment, there exists a simulator such that the trust region based on state transitions is larger than the trust region based on state-action transitions in Sec. E.

B Detailed Description of the Environments and Baselines

B.1 Environments

MuJoCo [41] is a high-fidelity physics engine designed for detailed and efficient rigid body simulations with contacts. It is widely used for benchmarking RL algorithms. Agents in MuJoCo environments receive vectorial state inputs and

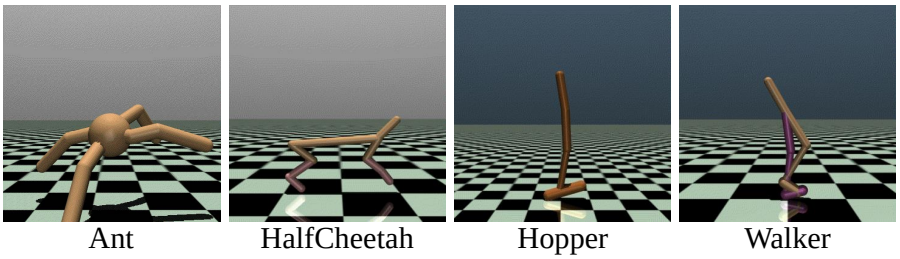


Fig. 3. MuJoCo environments used in this paper.

output continuous actions. For a comprehensive evaluation in our study, we have selected four tasks from the Gym MuJoCo suite and made small modifications to their reward and cost functions. Additionally, the maximum episode length of all environments is reduced to 300 for more efficient comparisons.

- `Ant`: The Ant is a 3D robot consisting of one torso with four articulated legs, each with two links. The objective is to coordinate the movements of the legs to navigate towards a specified direction.
  Due to the design of the cost function, we removed the action-related reward from the reward function and reduced the alive reward to 0.2. The cost function of this environment consists of three parts. The first part is the obj_cost, which is only related to the state. Let x and y be the coordinates of the Ant robot's position, then the obj_cost can be expressed as:

$$\text{obj\_cost}(x,y) = \begin{cases} 1 - \mathbb{1}(0.5x - 4 \le y \le 0.5x + 4) & 0 \le x \le 20 \\ 1 - \mathbb{1}(16 - 0.5x \le y \le 24 - 0.5x) & 20 < x \le 40 \\ 1 - \mathbb{1}(-3 \le y \le 3) & \text{otherwise} \end{cases}. \quad (53)$$

  The second part is the action_cost, which is only related to the action. Let the action a have a dimension of d_a; then, the action_cost can be expressed as:

$$\text{action\_cost}(a) = \mathbb{1}(\frac{||a||_1}{d_a} > 0.8). \quad (54)$$

  The final part is the done_cost, which is 1 if the task stops due to reasons other than reaching the episode length limit. The ultimate cost is the sum of these three parts, clipped to the range $[0, 1]$.
- `HalfCheetah`: The HalfCheetah is a 2-dimensional robot consisting of 9 links forming a spine, with 8 joints allowing articulation. The challenge is to exert torques on the joints to propel the cheetah forward as swiftly as possible.
  Due to the necessity of cost computation, we have added the x-coordinate of the robot to the observations and also removed the ctrl reward related to actions from the reward function. The cost in this environment is divided into two parts. The first part is the vel_cost, which is solely related to the state. If the x-coordinate of the robot at the previous time step is x_0, the

current time step is x_1, and the fixed time difference between the two steps is dt, then the vel_cost can be expressed as:

$$\text{vel\_cost}(x_0, x_1) = \mathbb{1}(\frac{x_1 - x_0}{dt} > 2.1). \tag{55}$$

The second part is the action_cost, which is similar across all environments. Let the action a have a dimension of d_a; then, the action_cost can be expressed as:

$$\text{action\_cost}(a) = \mathbb{1}(\frac{||a||_1}{d_a} > 0.7). \tag{56}$$

- `Hopper`: The Hopper is a two-dimensional one-legged figure that consists of four main body parts - the torso at the top, the thigh in the middle, the leg in the bottom, and a single foot on which the entire body rests. The goal is to make hops that move in the forward direction.
  The modifications made to the observation and reward aspects for Hopper and Walker are the same as those for HalfCheetah, and are thus omitted here. The first two parts of the cost for Hopper are similar to those for HalfCheetah:

$$\text{vel\_cost}(x_0, x_1) = \mathbb{1}(\frac{x_1 - x_0}{dt} > 1.4), \tag{57}$$

$$\text{action\_cost}(a) = \mathbb{1}(\frac{||a||_1}{d_a} > 0.7). \tag{58}$$

  While the third part of the cost is the same as the third part of Ant's cost.
- `Walker`: The Walker is a two-dimensional two-legged figure that consists of four main body parts - a single torso at the top, two thighs in the middle, two legs in the bottom, and two feet attached to the legs. The goal is to coordinate both sets of feet, legs, and thighs to move in the forward direction.
  The three components of Walker's cost are all similar to Hopper, with the expressions for the first two parts being:

$$\text{vel\_cost}(x_0, x_1) = \mathbb{1}(\frac{x_1 - x_0}{dt} > 1.6), \tag{59}$$

$$\text{action\_cost}(a) = \mathbb{1}(\frac{||a||_1}{d_a} > 0.7). \tag{60}$$

B.2 Simulators

For each environment, we construct three different simulators, representing variations in gravity, friction, and density parameters compared to the real environment. To thoroughly assess DASaR's adaptation ability, each simulator is tested with three different parameter values: 2.0, 1.5, and 0.5. These parameters can be adjusted in the MuJoCo environment by modifying the XML files. For instance, in MuJoCo, changing the gravity parameter of Ant can be achieved by adjusting the value associated with the "gravity" keyword in ant.xml.

B.3 Baselines

- **SAC_Lagrange(sim)** employs SAC_Lagrange, a classic off-policy safe RL algorithm, for training in the simulator and direct deployment in the real world. SAC_Lagrange builds upon the classical maximum entropy RL algorithm SAC [17] by additionally training a cost critic network $\hat{Q}^C$ and a Lagrange multiplier λ. It updates λ using the value estimates from $\hat{Q}^C$ and finally updates the policy using $\hat{Q}^R - \lambda\hat{Q}^C$, where $\hat{Q}^R$ is the reward critic network. We choose SAC_Lagrange as the main baseline algorithm because it serves as the backbone algorithm for our method, DASaR. In fact, DASaR can be combined with various off-policy safe RL algorithms such as CVPO [24]. However, for clarity and to highlight the unique advantages of DASaR, we chose the relatively simple SAC_Lagrange as the backbone algorithm in this paper.
- **SAC_Lagrange(id)** is trained in the simulator and obtains a policy the same as SAC_Lagrange(sim). The only difference lies in its deployment strategy, which aligns with that of DASaR, using an inverse dynamics model as the final action output. In simple terms, considering the simulator policy as π and the inverse dynamics model as g_I, with the current real state as s, the process is as follows: first, set the simulator state to the real state s and use π to output an action a to transition to simulator state s'. Then, use g_I to compute $a' = g_I(s, s')$ as the actual action executed in the real world. The choice of this baseline demonstrates that relying solely on the inverse dynamics model during deployment is insufficient, leading to suboptimal utilization of offline data.
- **CPQ** is a pure offline safe RL algorithm built upon the classic offline algorithm CQL [21]. It incorporates the conservative regularization operator from CQL onto the cost critic, treating out-of-distribution samples as unsafe. Additionally, CPQ no longer utilizes the Lagrangian multiplier method for policy updates. Instead, for unsafe (s, a) pairs, it directly truncates the reward critic to 0, preventing the execution of unsafe policies. This algorithm has become one of the most common baselines in offline safe algorithms and represents the state-of-the-art in RL-based offline safe algorithms.
- **DARC** is a classic baseline in sim-to-real RL domain adaptation methods. It trains two binary classifiers, $q_{sas}(target|s_t, a_t, s_{t+1})$ and $q_{sa}(target|s_t, a_t)$, to approximate $p_{\text{target}}(s_{t+1}|s_t, a_t)$ and $p_{\text{source}}(s_{t+1}|s_t, a_t)$. Ultimately, it employs $\Delta r = \log p_{\text{target}}(s_{t+1}|s_t, a_t) - \log p_{\text{source}}(s_{t+1}|s_t, a_t)$ as an additional reward to train the policy. This additional reward guides the policy to explore parts of the simulator with dynamics similar to the real world, thereby reducing the dynamics gap when deploying the policy to the real world.
- **H2O** is the latest state-of-the-art algorithm in domain adaptation. Similar to DARC, it also computes the probability ratio $\frac{p_{\text{target}}(s_{t+1}|s_t,a_t)}{p_{\text{source}}(s_{t+1}|s_t,a_t)}$. However, unlike DARC, this term is not utilized as an additional reward. H2O has a higher utilization of offline data: it directly uses offline data as training data for the policy. For simulator data, it employs $\frac{p_{\text{target}}(s_{t+1}|s_t,a_t)}{p_{\text{source}}(s_{t+1}|s_t,a_t)}$ as the basis for data selection. For parts of the simulator with dynamics similar to the real

world, it has a smaller weight for the conservative regularization operator. In contrast, for regions with significant dynamics gaps, a larger weight is assigned to the conservative regularization operator, ensuring lower reward estimates for these samples to prevent a substantial performance gap in the policy. Additionally, $\frac{p_{\text{target}}(s_{t+1}|s_t,a_t)}{p_{\text{source}}(s_{t+1}|s_t,a_t)}$ is used to weight the Bellman operator during the update of simulator data, mitigating the Bellman error caused by dynamics gap.

C Practical Implementation and Overall Algorithm

In this section, we describe the detailed practical designs and techniques used for DASaR. We also provide the hyperparameters used in DASaR for consistency.

C.1 Practical Implementation

In practical implementation, we made certain adjustments to the algorithm for stability and efficiency during training.

- **Discriminator**. We train the discriminator similarly to GAIL, with the only difference being a change in input from (s, a) to (s, s'). To prevent overfitting of the discriminator, we introduced Gaussian noise sampled from $\mathbb{N}(0, 0.1)$ into the discriminator's input. Additionally, to stabilize the training of the discriminator and prevent overly concentrated outputs, we incorporated a binomial entropy loss with a weight of -0.001. Additionally, for additional reward $\Delta r = \ln \frac{1-K(s,s')}{K(s,s')}$, we clip it to the range of $[-20, 20]$ and add a constant 20 to it to ensure it is positive.
- **Conservative cost function learning**. To address the issue of cost underestimation caused by the approximation errors in the inverse dynamics model function, we use Formula (61) as the update target:

$$\mathcal{B}^e \hat{Q}^{C,i}(s,a) = \max_{e^A \in E_S^A} (C(s, g_I^E(s,s'), s^{e^A}) + \gamma \mathbb{E}_{a' \sim \pi(\cdot|s^e)}[\hat{Q}^{C,i}(s^{e^A}, a')]), \quad (61)$$

To approximate the max operator in continuous spaces, we sample $n = 20$ candidates e^A from E_S^A as the candidate set. However, applying this formula for updates would require the simulator to perform n dynamics computations and the policy π to conduct an additional n inference operations. Compared to the original single-step Bellman operator computation, this formula significantly increases computational costs. Therefore, due to computational resource constraints, we simplified this formula in our practical implementation:

$$\mathcal{B}^{e'} \hat{Q}^{C,i}(s,a) = c(s, g_I^E(s,s'), s') + \gamma \max_{e^A \in E_S'^A} (\mathbb{E}_{a' \sim \pi(\cdot|s')}[\hat{Q}^{C,i}(s', a' + e^A)]), \quad (62)$$

where $E_S'^A = [-\beta_I \cdot \sqrt{\text{Var}_{i \in \{1,\ldots,E_I\}}[g_I^i(s',s'')]}, \beta_I \cdot \sqrt{\text{Var}_{i \in \{1,\ldots,E_I\}}[g_I^i(s',s'')]}\ ]$ and $s'' = f_S(s', a')$. Formula (62) simplifies the robustness update from the

entire temporal difference target to the target Q network, eliminating the need for additional n dynamics computations and requiring only one additional computation to obtain s''. While this simplification weakens the conservatism and robustness of the cost critic, it significantly reduces computational costs. Furthermore, considering that in continuous state spaces, single-step state transitions do not introduce significant changes to the state, there is not a substantial difference between s and s', as well as between s' and s''. Therefore, we can further simplify Formula (62) to obtain:

$$\mathcal{B}^{e''}\hat{Q}^{C,i}(s,a) = c(s, g_I^E(s,s'), s') + \gamma \max_{e^A \in E_S^A} (\mathbb{E}_{a' \sim \pi(\cdot|s')}[\hat{Q}^{C,i}(s', a' + e^A)]). \quad (63)$$

The simplification of Formula (63) completely eliminates the need for additional dynamics computations in the simulator, reducing the additional overhead to neural network calculations that can be processed in parallel. However, if computational resources are abundant, and there are no concerns about training speed, we still recommend using Formula (61) for updates to achieve better policy performance. Additionally, to ensure a more stable learning process of the cost critic, the operator $\mathcal{B}^{e''}$ is only used after 40 learning epochs, before which we use the traditional Bellman operator to update the cost critic.

C.2 Overall Algorithm

In this section, we will provide an overview of the overall algorithmic process.

During training, we will first train the ensemble inverse dynamics model to minimize the following supervised learning loss:

$$\mathcal{L}_{\text{id}} = \mathbb{E}_{(s,a,s') \sim D_T}[\sum_{i=1}^{E_I} ||g_I^i(s,s') - a||_2^2]. \quad (64)$$

Next, based on the learned ensemble inverse dynamics model, we perform value relabeling to obtain:

$$r' = R'(s,a,s') = R(s, g_I^E(s,s'), s'), \quad (65)$$

$$c' = C'(s,a,s') = C(s, g_I^E(s,s'), s'), \quad (66)$$

where $g_I^E = \mathbb{E}_{i \in \{1,\ldots,E_I\}}[g_I^i(s,s')]$. To constrain the policy within the offline dataset where g_I^E possesses high accuracy, we utilize the following loss to learn a binary discriminator $K(s,s')$:

$$\mathcal{L}_{\text{dis}} = -\mathbb{E}_{(s,s') \sim D_S}[\ln K(s,s')] - \mathbb{E}_{(s,s') \sim D_T}[\ln(1 - K(s,s'))]. \quad (67)$$

Based on this discriminator, we have:

$$\Delta r = \ln \frac{1 - K(s,s')}{K(s,s')}. \quad (68)$$

Algorithm 1 DASaR Training

Input: safe RL algorithm SAC_Lagrange, simulator, real-world offline data D_T, discount factor γ, safety constraint limit b, reward function R, cost function C, hyperparameters α, β_C, β_I.

Initialize: empty buffer D_S, ensemble inverse dynamics model $\{g_I^i\}_{i=1}^{E_I}$, discriminator K, policy π_θ, reward critic $\hat{Q}^R$, ensemble cost critic $\{\hat{Q}^{C,i}\}_{i=1}^{E_c}$, Lagrangian multiplier λ.

for step in supervised training steps **do**
  Sample a batch $\{(s, a, s')\}$ from D_T.
  Compute $\mathcal{L}_{\text{id}}$ according to Eqn. 64.
  Update $\{g_I^i\}_{i=1}^{E_I}$ to minimize $\mathcal{L}_{\text{id}}$.
end for
for step in RL training steps **do**
  for each environment step **do**
    Observe current simulator state s.
    Compute $a \sim \pi_\theta(\cdot|s)$.
    Step action a in the simulator and observe the next simulator state s'.
    Update $D_S \leftarrow D_S \bigcup \{(s, a, R(s, a, s'), C(s, a, s'), s')\}$.
  end for
  for each discriminator gradient step **do**
    Sample a batch $\{(s_S, s'_S)\}$ from D_S.
    Sample a batch $\{(s_T, s'_T)\}$ from D_T.
    Compute $\mathcal{L}_{\text{dis}}$ according to Eqn. 67.
    Update K to minimize $\mathcal{L}_{\text{dis}}$.
  end for
  for each policy gradient step **do**
    Sample a batch $\{(s, a, R(s, a, s'), C(s, a, s'), s')\}$ from D_S.
    Compute relabeled reward and cost r', c' according to Eqn. 65.
    Compute Δr according to Eqn. 68.
    Update $\hat{Q}^R$ with $r' + \alpha \Delta r$ by SAC_Lagrange.
    Compute $\mathcal{L}_{\text{qc}}$ according to Eqn. 69.
    Update $\{\hat{Q}^{C,i}\}_{i=1}^{E_c}$ to minimize $\mathcal{L}_{\text{qc}}$.
    Compute $\hat{Q}^{C,\text{UCB}}$ according to Eqn. 70.
    Update π_θ, λ with $\hat{Q}^R$ and $\hat{Q}^{C,\text{UCB}}$ by SAC_Lagrange.
  end for
end for
Return $\pi_\theta, \{g_I^i\}_{i=1}^{E_I}$.

Then, update the reward critic $\hat{Q}^R$ based on SAC using $r' + \alpha \Delta r$, and update the ensemble cost critic with the following loss:

$$\mathcal{L}_{\text{qc}} = \mathbb{E}_{(s,a,s') \sim D_S} [\sum_{i=1}^{E_C} (\hat{Q}^{C,i}(s, a) - \mathcal{B}^{e''} \hat{Q}^{C,i}(s, a))^2]. \tag{69}$$

Finally, we have the upper confidence bound of cost critic as:

$$\hat{Q}^{C,\text{UCB}}(s, a) = \mathbb{E}_{i \in \{1,\ldots,E_C\}} [\hat{Q}^{C,i}(s, a)] + \beta_C \cdot \sqrt{\text{Var}_{i \in \{1,\ldots,E_C\}} [\hat{Q}^{C,i}(s, a)]}, \tag{70}$$

Algorithm 2 `DASaR Deployment`

Input: simulator, real-world environment, simulator-based policy π_θ, ensemble inverse dynamics model $\{g_I^i\}_{i=1}^{E_I}$.

for step=1 to $T_{\max}$ **do**
  Observe the current state of the real-world environment s.
  Set the simulator's current state to s.
  Compute $a \sim \pi_\theta(\cdot|s)$.
  Step action a in the simulator and observe the next simulator state s'.
  Compute $a' = \mathbb{E}_{i\in\{1,\ldots,E_I\}}[g_I^i(s,s')]$.
  Step action a' in the real-world environment.
end for

During deployment, the simulator, the real-world environment, the simulator-based policy π and the inverse dynamics model will work together. When the current state of the real-world environment is s, DASaR will first set the simulator state to s and π will put out an action a to let the state in the simulator transition to s'. Then, a real action $a' = g_I^E(s,s')$ will be computed to take place in the real-world environment.

Detailed training and deployment pseudo codes are provided in Algorithm 1 and Algorithm 2, respectively.

D Experimental Details

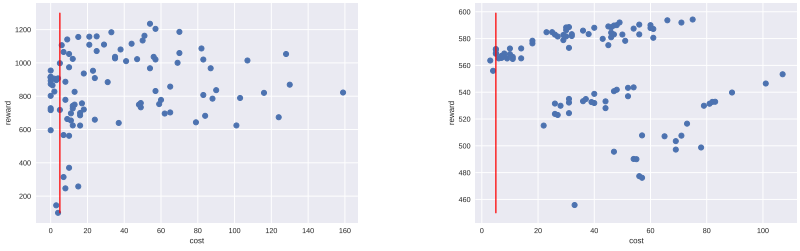
D.1 Offline Data Collection

The offline datasets for each environment are obtained by sampling from five behavioral policies within the environment learned by CPO [1]. One behavioral policy is trained using CPO with the safety constraint limit $b = 5$. Two other behavioral policies are trained with $b = 25$, and the remaining two are trained with $b = 50$. Let the maximum training step for the policy with $b = 5$ be denoted as $T_{\max}$. Then, one of the policies trained with $b = 25$ has a training step limit of $\frac{T_{\max}}{3}$, while the other has a limit of $T_{\max}$. The policies trained with $b = 50$ follow the same pattern. To be specific, the maximum training step of Ant is $T_{\max} = 3e7$ while that of the other environments is $T_{\max} = 1.8e7$.

The detailed distribution of trajectory rewards and costs for each environment's offline dataset is illustrated in Figure 4.

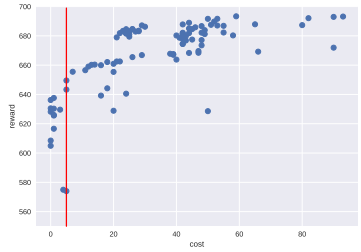
D.2 The Hyperparameter Choice of DASaR

The training progress of DASaR involves supervised learning and reinforcement learning, these two parts all depend on the choices of hyperparameters to ensure the stability and effectiveness of training. Therefore, we provide the hyperparameters used in the experiment of DASaR in Table 2 to guarantee the reproducibility of the performance. We implement DASaR based on SAC within the

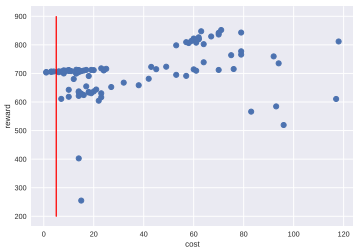


(a) Ant

(b) HalfCheetah



(c) Hopper



(d) Walker

Fig. 4. Visualization of the reward and cost distribution of the trajectories for different environments. The points to the left of the red lines represent trajectories that adhere to the safety constraint limit.

RLKit [4] framework. Default parameters in the framework are used for those hyperparameters not mentioned in the following table. In these hyperparameters, reward scaling means we give the total reward $r' + \alpha \Delta r$ a weight ω when learning the reward critic. Lagrange multiplier update warm up epoch means we keep the Lagrange multiplier unchanged for some epochs until the estimation of critics does not have a large error. Other hyperparameters not mentioned in the table are the same as the default hyperparameters for SAC.

E Additional Experimental Results

E.1 Case Study: When The Trust Region Conflicts With Safe Data

To better illustrate the conflict between the trust region based on state-action transition and safe data mentioned in the main text, and to confirm the existence of environments where the trust region based on state transition is larger than the trust region based on state-action transition, we designed an easily understandable toy environment. The environment is roughly depicted in Fig. 5(a), where the agent starts from the left starting point and aims to reach the goal

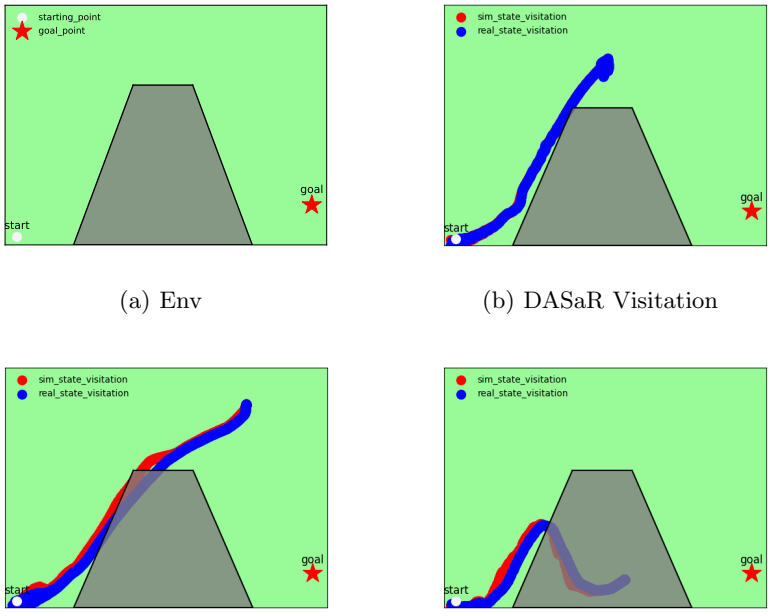
[4] https://github.com/rail-berkeley/rlkit

Table 2. Hyperparameter choices of DASaR. Values wrapped in {} means it differs in {Ant,HalfCheetah,Hopper,Walker}.

	Hyperparameter	Value
Inverse Dynamics Model	network hidden layers	[256, 256]
	ensemble number E_I	7
	batch size	512
	maximum learning epochs	2000
	early stop validation ration	0.1
	learning rate	0.0001
Discriminator	network hidden layers	[256, 256]
	source batch size	512
	target batch size	512
	update step per RL step	1
	learning rate	0.0001
Policy Learning	safety constraint limit b	5
	α	0.5
	β_C	{2.0, 2.0, 6.0, 6.0}
	β_I	1.0
	reward scaling	0.5
	cost critic ensemble number E_C	4
	error e^A sampling number n	20
	network hidden layers	[256, 256]
	batch size	512
	training epochs	400
	steps per epoch	5000
	Lagrange multiplier update warm up epoch	40
	policy learning rate	0.0001
	critic learning rate	0.0003
	Lagrange multiplier learning rate	0.00001
	γ	0.99

point on the right. However, there is a trapezoidal shadow area in the middle of the environment where moving is unsafe and incurs a cost of 1. The agent's state has two dimensions representing the current x and y coordinates, and the action also has two dimensions representing the distance moved in the x and y directions, with each dimension ranging from $[-1, 1]$. However, there are discrepancies in the construction of the environment's simulator, where the dynamics transition is correct in the first dimension (x-axis) of the state but not in the second dimension, causing a doubling of the magnitude of state transitions. That is, suppose the current state is (x_1, y_1), and the action taken is $(\Delta x, \Delta y)$, then in the simulator, the state at the next time step would be $(x_1 + \Delta x, y_1 + 2\Delta y)$.

In this environment, the trust region based on state-action transition only includes state-action transitions in the real-world environment where there is no change in the y direction. However, to traverse the shadow area in the mid-



(a) Env (b) DASaR Visitation

(c) DARC Visitation Pattern 1 (d) DARC Visitation Pattern 2

Fig. 5. Visualization of a toy environment and how DASaR and DARC_Lagrange perform in it. (We breviate DARC_Lagrange as DARC.)

dle, which is necessary for a safe trajectory, changes in the y direction are inevitable. Consequently, the trust region defined by state-action transition conflicts with the safe data. Moreover, for any transition in the real environment $((x_1, y_1), (\Delta x_1, \Delta y_1), (x_1 + \Delta x_1, y_1 + \Delta y_1)), \Delta x_1, \Delta y_1 \in [-1, 1]$, there always exists an action $(\Delta x_1, \frac{\Delta y_1}{2})$ in the simulator that validates the state transition $((x_1, y_1), (x_1 + \Delta x_1, y_1 + \Delta y_1))$, thus the trust region based on state transition can cover the entire real environment, which is much larger than the trust region based on state-action transition.

To show DASaR's better adaptation ability, we compare it with the typical domain adaptation algorithm DARC_Lagrange. Fig. 5(b) shows the state visitation of DASaR in both the simulator and the real-world environment. It is clear that DASaR can achieve nearly the same state visitation in the simulator and the real-world environment and maintain safety simultaneously. However, DARC_Lagrange exhibits two distinct behavioral patterns, as illustrated in Fig.5(c) and5(d). In the first pattern, it disregards the constraint of the trust region and instead attempts to satisfy the safety constraint. However, in this scenario, although DARC shows small safety violations in the simulator, it incurs more safety violations when deployed in the real-world environment due

to the presence of the dynamics gap. The second pattern emerges as the policy progresses, attempting to comply with the trust region constraint by approaching the target point directly with more horizontal trajectories. However, due to the conflict between safe data and the trust region, it already exhibits numerous safety violations in the simulator, resulting in continued safety violations in the real-world environment. These results vividly illustrate the limitations of traditional domain adaptation methods, highlighting the rationality of DASaR's design under Assumption 1.

E.2 Detailed Results of Ablation Studies and Data Sensitivity Study

In the main paper's ablation studies and data sensitivity study, we only present the average results of each baseline without standard deviation information. Therefore, in this section, we present the detailed results of both the ablation studies and the data sensitivity study in Tab. 3 and 4.

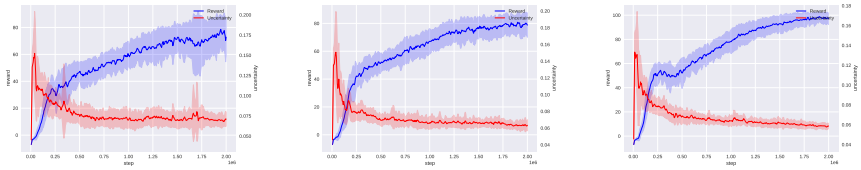
Table 3. Detailed results of the ablation studies. The highest performance in each row is emphasized in blue.

Environment		Ant		Cheetah		Overall
		g	f	g	f	
DASaR	reward↑	72.8 ± 14.2	78.8 ± 8.8	**63.2 ± 8.4**	86.2 ± 8.5	**75.3**
	cost↓	1.3 ± 1.2	1.1 ± 1.4	**1.7 ± 1.6**	1.2 ± 1.1	**1.3**
wo_va	reward↑	67.8 ± 17.2	74.0 ± 10.7	63.1 ± 9.9	87.2 ± 8.9	73.0
	cost↓	1.3 ± 1.1	2.1 ± 2.6	2.3 ± 2.6	2.0 ± 3.4	1.9
wo_tb	reward↑	**34.8 ± 12.5**	**42.5 ± 14.1**	−46.5 ± 144.5	−74.7 ± 218.2	−11.0
	cost↓	**0.2 ± 0.2**	**0.2 ± 0.6**	10.8 ± 12.0	13.6 ± 16.8	6.2
wo_ucb_c	reward↑	73.8 ± 17.8	80.4 ± 8.2	57.0 ± 49.6	**88.3 ± 6.1**	74.9
	cost↓	1.5 ± 1.3	1.2 ± 1.1	4.6 ± 6.7	**1.0 ± 1.0**	2.1
wo_ucb_i	reward↑	72.0 ± 18.4	84.8 ± 6.5	66.4 ± 8.0	84.8 ± 10.3	77.0
	cost↓	3.1 ± 2.3	2.3 ± 3.4	2.8 ± 2.4	1.7 ± 1.5	2.5

Table 4. Detailed results of the data sensitivity study in Ant gravity.

Environment		Ant gravity
100%	reward↑	72.8 ± 14.2
	cost↓	1.3 ± 1.2
50%	reward↑	77.9 ± 17.1
	cost↓	1.5 ± 1.5
20%	reward↑	63.7 ± 30.0
	cost↓	0.8 ± 0.5
10%	reward↑	26.0 ± 24.2
	cost↓	0.8 ± 1.3

E.3 Relationship Between Reward and Inverse Dynamics Model Uncertainty



(a) Ant gravity (b) Ant friction (c) Ant density

Fig. 6. Visualization of the curves depicting the changes in the real reward and uncertainty during the training process.

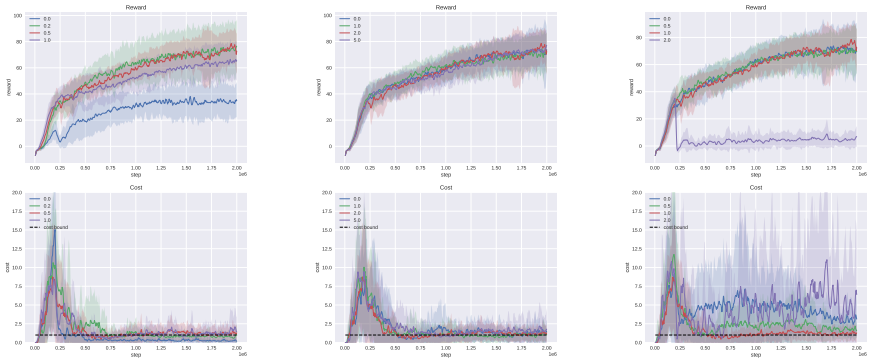
The performance of DASaR's policy in the real world is closely related to the accuracy of the inverse dynamics model. For an ensemble inverse dynamics model, its accuracy can be reflected using the uncertainty of the model ensemble. Therefore, we visualize the changes in the reward of the policy in three simulators of the Ant environment and the uncertainty of the inverse dynamics model, as shown in Fig. 6.

Two conclusions can be drawn from this. First, the introduction of additional reward based on the discriminator does indeed constrain the policy within regions where the inverse dynamics model has higher accuracy, thus providing some evidence for the correctness of Proposition 2. Second, there exists a negative correlation between the reward in the real world and uncertainty. As uncertainty gradually decreases, the reward in the real world also tends to increase. Conversely, when uncertainty exhibits certain fluctuations, the reward in the real world also fluctuates accordingly, which is particularly evident in Ant gravity.

E.4 Parameter Sensitivity Studies

To assess the sensitivity of DASaR to different hyperparameter selections, we conduct hyperparameter sensitivity studies in the Ant gravity simulator for three hyperparameters: α, β_C, and β_I. Specifically, α is varied between 0.0, 0.2, 0.5, and 1.0; β_c is varied between 0.0, 1.0, 2.0, and 5.0; and β_I is varied between 0.0, 0.5, 1.0, and 2.0. Results are shown in Fig. 7.

Firstly, concerning the hyperparameter α, it is evident that when α is 0, the policy yields a significantly lower reward return, indicating a considerable performance gap. As α increases to 0.2 and 0.5, the reward return experiences a substantial increase while maintaining safety performance within an acceptable range. However, as α further increases to 1.0, both the reward and safety performance of the policy decrease. This underscores the importance of both reducing the performance gap and expanding the optimization search space of the policy to achieve better performance in real-world environments.



(a) α Sensitivity (b) β_C Sensitivity (c) β_I Sensitivity

Fig. 7. Visualization of the curves depicting the changes in the reward and costs during the training process of different parameter values.

Moving on to the hyperparameter β_C, it is observed that in the Ant gravity environment, the values of β_C do not have a significant impact on the performance of the policy. This suggests that in this environment, the learning of the cost critic is relatively accurate, thus the upper confidence bound of the cost critic will not undergo significant changes regardless of the value of β_C.

Finally, regarding the hyperparameter β_I, it is observed that as β_I gradually increases from 0 to 1.0, the safety performance of the policy also gradually improves. However, when β_I further increases to 2.0, the policy learning collapses, with the reward essentially dropping to near 0 and safety no longer guaranteed. This phenomenon primarily occurs because an excessively large β_I makes the cost critic overly conservative, potentially preventing the discovery of a policy that satisfies the cost critic's estimation being less than $b = 5$. Consequently, the Lagrange multiplier continues to increase, leading to the collapse of policy learning. This underscores the importance of selecting β_I appropriately.