GenSafe: A Generalizable Safety Enhancer for Safe Reinforcement Learning Algorithms Based on Reduced Order Markov Decision Process Model

Zhehua Zhou, Xuan Xie, Jiayang Song, Zhan Shu, and Lei Ma

***Abstract*—Although deep reinforcement learning has demonstrated impressive achievements in controlling various autonomous systems, e.g., autonomous vehicles or humanoid robots, its inherent reliance on random exploration raises safety concerns in their real-world applications. To improve system safety during the learning process, a variety of Safe Reinforcement Learning (SRL) algorithms have been proposed, which usually incorporate safety constraints within the Constrained Markov Decision Process (CMDP) framework. However, the efficacy of these SRL algorithms often relies on accurate function approximations, a task that is notably challenging to accomplish in the early learning stages due to data insufficiency. To address this problem, we introduce a Genralizable Safety enhancer (GenSafe) in this work. Leveraging model order reduction techniques, we first construct a Reduced Order Markov Decision Process (ROMDP) as a low-dimensional proxy for the original cost function in CMDP. Then, by solving ROMDP-based constraints that are reformulated from the original cost constraints, the proposed GenSafe refines the actions taken by the agent to enhance the possibility of constraint satisfaction. Essentially, GenSafe acts as an additional safety layer for SRL algorithms, offering broad compatibility across diverse SRL approaches. The performance of GenSafe is examined on multiple SRL benchmark problems. The results show that, it is not only able to improve the safety performance, especially in the early learning phases, but also to maintain the task performance at a satisfactory level.**

***Index Terms*—Safe Reinforcement Learning, Constrained Markov Decision Process, Model Order Reduction**

I. Introduction

OVER the past decade, Deep Reinforcement Learning (DRL) has achieved remarkable advancements in the control of a diverse array of autonomous systems, such as robotic manipulators [1], autonomous vehicles [2], and drones [3]. However, the inherent random exploration characteristic of DRL algorithms often undermines the safety of the learning process [4]. During the progression towards an optimal policy, the agent may exhibit numerous unsafe intermediate policies, posing risks not only to the system itself but also to its surrounding environment [5]. For example, an autonomous vehicle might repeatedly collide with obstacles prior to the successful determination of a DRL-based controller. Due to these safety concerns, the majority of DRL research findings have been confined to simulated environments [6], with fewer applications in real-world experiments [7]. This limitation becomes more evident in tasks where ensuring safety is of paramount importance.

To address this challenge, a variety of Safe Reinforcement Learning (SRL) approaches have been proposed [8], [9], aiming to realize a safe learning process for DRL algorithms. Since incorporating safety specifications naturally involves constraints, state-of-the-art SRL approaches often employ the Constrained Markov Decision Process (CMDP) [10] as their fundamental mathematical framework. In CMDP, a cost function is defined in addition to the standard reward function, which maps undesirable behavior at any given time step to a non-negative scalar cost. A system is considered safe if a cost-related constraint, e.g., the accumulated cost should remain below a predefined threshold, is satisfied [11]. This transforms the SRL problem into a constraint satisfaction problem and allows for the independent consideration of task performance and safety specifications [12]. However, although optimal solutions for finite and low-dimensional CMDPs can be obtained by linear programming [13], solving CMDPs for complex systems with high-dimensional and continuous state and action spaces remains a challenging problem.

Modern SRL techniques typically fall into two categories: model-based and model-free approaches [9]. Model-based approaches primarily focus on learning a system or cost model, which is then utilized to refine the current policy [14]–[21]. Although these approaches generally exhibit better sample efficiency compared to model-free ones, their performance is highly dependent on the accuracy of the learned model [17]. Conversely, model-free approaches usually employ policy gradient methods, necessitating function approximations for both reward and cost functions [22]–[25]. Accurate approximations are crucial in these methods for ensuring both monotonic performance improvement and constraint satisfaction [22]. However, obtaining reliable and precise function approximations is a challenging task, especially in the early learning stage, where the amount of available data is often insufficient. This inevitably diminishes the efficacy of model-free SRL methods [25].

In this work, we aim to enhance the performance of model-free SRL approaches, with an emphasis on improving their constraint satisfaction in the early learning stages. To achieve this, we introduce a **Gen**ralizable **Safe**ty enhancer (GenSafe) that is able to augment the safety performance of model-free SRL algorithms while maintaining the task performance at a satisfactory level. Drawing inspiration from research that employs model order reduction [26] to solve complex system

Zhehua Zhou, Xuan Xie, Jiayang Song, Zhan Shu are with University of Alberta, Canada (email: zhehua1@ualberta.ca, xxie9@ualberta.ca, jiayan13@ualberta.ca, zshu1@ualberta.ca).

Lei Ma is with The University of Tokyo, Japan and University of Alberta, Canada (email: ma.lei@acm.org).

dynamics [7], [27], [28], we first address the challenge of data insufficiency by approximating the CMDP's cost function with a low-dimensional Markov Decision Process (MDP) model. This approximation is achieved by reducing the dimensionality of the original state space, resulting in a Reduced Order Markov Decision Process (ROMDP) that serves as a proxy for the cost function and offers estimates on system safety. Then, the proposed GenSafe reinterprets the original cost constraint in CMDP into ROMDP-based constraints. By resolving these revised constraints, GenSafe modifies each action taken by the agent to enhance the possibility of constraint satisfaction. In essence, GenSafe acts as an additional safety layer for SRL algorithms [29] (see Fig. 1). Its compatibility with a diverse range of model-free SRL algorithms suggests its potential as a broadly generalizable tool for enhancing safety across various SRL applications and scenarios.

The contributions of this work are summarized as follows:

- We propose an innovative order reduction technique to construct a ROMDP that serves as a low-dimensional approximator for the cost function in CMDP. Such a ROMDP is able to effectively provide more accurate estimates of system safety in the early learning stages where data availability is limited, consequently leading to enhanced constraint satisfaction.
- Utilizing the constructed ROMDP, we present GenSafe, a novel approach for augmenting SRL algorithms. By modifying the actions of the agent, GenSafe enhances the probability of achieving constraint satisfaction while simultaneously maintaining task performance at a satisfactory level. Moreover, GenSafe is designed as an additional safety layer, which enables its compatibility and generalization across various SRL algorithms.
- We perform an extensive analysis of GenSafe's performance through experiments on a variety of SRL benchmark problems. The results validate the effectiveness of GenSafe, demonstrating its ability to enhance the performance of SRL algorithms.

II. Related Work

A. Model-free SRL

Constrained Policy Optimization (CPO) [22] is the first model-free SRL approach that employs the policy gradient method. It utilizes surrogate functions to approximate both objective and constraint functions. However, the inherent difficulty in achieving accurate function approximation diminishes the performance of CPO in certain state-of-the-art SRL benchmark problems, such as SafetyGym [12]. In [12], a Lagrangian relaxation of the SRL problem is introduced, which is integrated with Proximal Policy Optimization (PPO) [30] to create a PPO-Lagrangian algorithm, and with Trust Region Policy Optimization (TRPO) [31] for a TRPO-Lagrangian algorithm. Such a primal-dual method transforms the constrained problem into an unconstrained one by augmenting the objective with a sum of constraints, each weighted by its respective Lagrange multiplier [32]. Based on this concept, various primal-dual SRL approaches have been developed [23]–[25], [33]–[37]. For instance, [23] adopts a penalized reward function and

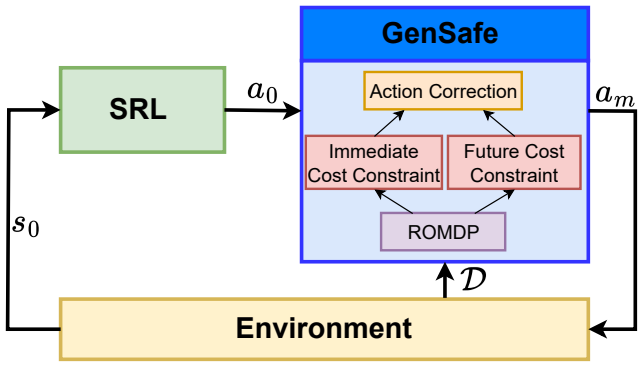


Fig. 1. SRL with GenSafe. At each timestep, the current SRL policy recommends an action a_0 based on the current state s_0. Then, the proposed GenSafe performs an action correction to identify a modified action a_m that is more likely to satisfy safety constraints. Such a corrective process involves resolving an optimization problem that considers both the immediate and future cost constraints, which are derived from the constructed ROMDP. We utilize the set of data samples $\mathcal{D}$ observed during the learning process to construct the ROMDP, which serves as a low-dimensional proxy for the original cost function in CMDP.

a two-timescale stochastic approximation scheme for policy optimization, where the Lagrange multiplier is updated on a slower timescale compared to the policy parameters. In [36], a PID-based technique is proposed for mitigating oscillations in Lagrangian methods, aiming to reduce constraint violations. Derived from CPO, [35] introduces the Projection-based Constrained Policy Optimization (PCPO), a two-step method that first solves the policy search problem via TRPO and then projects the policy back to a feasible region for satisfying safety constraints. Following this, [25] presents the First Order Constrained Policy Optimization (FOCOPS), which utilizes first-order optimization to achieve improved computational efficiency over PCPO. However, these primal-dual methods often exhibit sensitivity to the initialization of Lagrange multipliers and learning rates, which could result in considerable hyperparameter tuning efforts [38]. Moreover, the introduction of dual parameters may pose stability challenges at the algorithm's saddle points [9].

Another category of model-free SRL approaches is the primal methods [29], [39]–[42], which achieve constraint satisfaction through diverse designs of the objective function or the update process without the introduction of dual variables. For example, [42] introduces Interior Point Optimization (IPO), which utilizes logarithmic barrier functions to satisfy safety constraints. Similarly, Lyapunov functions have been employed in several SRL studies to maintain safety during the learning process [39]–[41], where the agent's actions are restricted by these functions. Nevertheless, these primal methods often require prior system knowledge for designing the barrier or Lyapunov functions, which can be inherently complex. Akin to our work, [29] also adds a safety layer to the policy network to ensure constraint satisfaction. However, [29] relies on pre-collected offline data to construct the safety layer and necessitates new datasets for each task, thereby limiting its applicability and generalizability. In contrast, the proposed GenSafe is an online learning method and does not demand pre-knowledge about the system or specific tasks.

A notable challenge with model-free SRL approaches is

their dependence on accurate function approximations, e.g., for cost and reward functions, to achieve satisfactory performance [8], [9]. This issue becomes more pronounced in the early learning phase, where limited data availability often hinders the creation of precise approximations, leading to reduced performance. To tackle this problem, we present GenSafe in this work, which utilizes a ROMDP to estimate the cost return more effectively. By employing order reduction, it mitigates the problem of data insufficiency, and as a result, improved constraint satisfaction is realized, particularly in the early learning stages.

B. Model-based SRL

Model-based SRL approaches [14]–[21] are generally considered more sample efficient than model-free ones. In [14], a model-based method that learns both the system dynamics and the cost function for addressing safety concerns is proposed. This method leverages the learned model to generate sample trajectories, which are then used to refine the policy via an adapted cross-entropy method. Similarly, [15] employs PILCO [43] for learning model dynamics and optimizes the policy using constraint functions based on Conditional Value at Risk (CVaR) [44]. Contrasting with these approaches, [16] utilizes a penalized reward function instead of a separate cost function for optimization purposes. However, the effectiveness of model-based SRL approaches often highly depends on the accuracy of the learned model. In practical tasks, obtaining an accurate system model could be a challenging task.

There exist also model-based SRL approaches that are grounded in control-theoretical concepts [7], [27], [40], [45]–[52]. For example, in [45], reachability analysis is employed to devise a reliable backup policy that could rectify potentially hazardous actions taken by the agent. Lyapunov functions are also frequently used to maintain system stability during the learning process [40], [46]. Nevertheless, these approaches often necessitate the manual design of Lyapunov functions, which might be infeasible for complicated tasks. In [48], [49], the concept of the region of attraction is utilized to define a safe region, and the exploration during the learning process is confined to this specified region. Model Predictive Control (MPC) [53] is another tool widely used for safe learning [50]–[52]. These methods involve learning a system or disturbance model and applying robust MPC to determine safe actions [54]. However, approaches guided by control-theoretical concepts usually face computational difficulties [55], limiting their applicability predominantly to systems characterized by linear or low-dimensional dynamics [50].

III. Preliminary

In this section, we first present the fundamental concepts of CMDP (Sec. III-A). Then, we define the SRL problem within the CMDP framework (Sec. III-B). Finally, we introduce PPO-Lagrangian, which is usually regarded as a baseline method in SRL, to demonstrate the typical realization of SRL using model-free approaches (Sec. III-C). In Sec. V, we will use PPO-Lagrangian as a representative example to explain the integration of the proposed GenSafe with model-free SRL methods.

A. Constrained Markov Decision Process (CMDP)

The CMDP extends the standard MDP by incorporating a constraint function. A CMDP is defined by the tuple $\mathcal{M} = (S, A, T, R, C, \gamma, \mu)$, where S is the set of states, and A is the set of actions. $T : S \times A \times S \to [0, 1]$ denotes the state transition kernel, with $T(s, a, s')$ represents the probability of transitioning from state $s \in S$ to state $s' \in S$ under action $a \in A$. The reward function $R : S \times A \to \mathbb{R}$ and the cost function $C : S \times A \to \mathbb{R}$ assign single-stage reward r and cost c to each state-action pair, respectively. γ is the discount factor and μ is the initial state distribution. In general, both the reward and cost functions in CMDP are considered to be non-negative.

Remark 1. *In this work, we consider a single constraint function for simplicity. For scenarios that involve multiple constraint functions, the proposed method can be extended by replicating the computations used for the single constraint across each of the additional constraints.*

For a given state s, a stationary policy $\pi(s) : S \to A$ determines an action a for the agent to take. In deep learning contexts, the policy is typically represented by a neural network parameterized by θ, and is denoted as π_θ. Given a policy π_θ, the state value function $V_R^{\pi_\theta}(s)$, the state-action value function $Q_R^{\pi_\theta}(s, a)$, and the advantage function $A_R^{\pi_\theta}(s, a)$ for the reward are defined as

$$V_R^{\pi_\theta}(s) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \middle| s_0 = s\right], \tag{1}$$

$$Q_R^{\pi_\theta}(s, a) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \middle| s_0 = s, a_0 = a\right], \tag{2}$$

$$A_R^{\pi_\theta}(s, a) = Q_R^{\pi_\theta}(s, a) - V_R^{\pi_\theta}(s). \tag{3}$$

Similarly, for the cost, the state value function $V_C^{\pi_\theta}(s)$, the state-action value function $Q_C^{\pi_\theta}(s, a)$, and the advantage function $A_C^{\pi_\theta}(s, a)$ are given by substituting the reward function R with the cost function C. For simplicity, the notation π_θ will be omitted in the remainder of this paper when there is no ambiguity.

B. Safe Reinforcement Learning (SRL)

The goal of SRL is to maximize task performance while ensuring the satisfaction of safety requirements. Within CMDP, such an SRL problem can be defined through the use of infinite-horizon discounted reward $J_R(\pi_\theta)$ and cost $J_C(\pi_\theta)$ returns, leading to the following optimization problem:

$$\max_\theta J_R(\pi_\theta), \text{ s.t. } J_C(\pi_\theta) \le d, \tag{4}$$

where $J_R(\pi_\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)\right]$ and $J_C(\pi_\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{\infty} \gamma^t C(s_t, a_t)\right]$. d is a predefined threshold value. Hence, the optimal policy π_θ^* of the SRL problem can be determined by solving (4). Note that, alternative formulations, e.g., finite-horizon undiscounted returns, can also be applied to define the optimization problem (4). The proposed GenSafe is adaptable to these different definitions by modifying the reformulated ROMDP-based constraints accordingly, see Sec. V for more details.

C. PPO-Lagrangian

Directly solving (4) is generally infeasible since, on the one hand, both the reward function R and the cost function C are unknown and necessitate approximations. On the other hand, the constrained nature of the optimization problem poses computational challenges. Therefore, a widely employed method in model-free SRL is the introduction of Lagrangian relaxation. The Lagrangian of (4) can be written as

$$L(\theta, \lambda) = J_R(\pi_\theta) - \lambda(J_C(\pi_\theta) - d), \tag{5}$$

where $\lambda \in \mathbb{R}^+$ is the Lagrange multiplier. It then converts the constrained optimization problem (4) into the following unconstrained problem

$$L(\theta^*, \lambda^*) = \max_\theta \min_\lambda L(\theta, \lambda), \tag{6}$$

where the tuple (θ^*, λ^*) indicates the optimal saddle point. If $J_R(\pi_\theta)$ and $J_C(\pi_\theta)$ are known, (6) can then be solved via gradient search methods [17].

For this purpose, the PPO-Lagrangian method [12] employs the PPO clipped objectives [30] for the estimation of $J_R(\pi_\theta)$ and $J_C(\pi_\theta)$ as

$$J_R(\pi_\theta) = \mathbb{E}_t\left[\min(r_t(\theta)A_R^t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)A_R^t)\right], \tag{7}$$

$$J_C(\pi_\theta) = \mathbb{E}_t\left[\min(r_t(\theta)A_C^t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)A_C^t)\right], \tag{8}$$

where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ represents the likelihood ratio of selecting action a_t in state s_t under the parameter θ compared to θ_{old}. $\mathbb{E}_t$ indicates the empirical average over a finite batch of samples, and ϵ is the clip ratio. A_R^t and A_C^t are estimated advantages for the reward and cost, which are often computed via Generalized Advantage Estimation (GAE) [56] as

$$A_R^t = \sum_{l=0}^{k} (\gamma\bar{\lambda})^l \delta_R^{t+l}, \quad A_C^t = \sum_{l=0}^{k} (\gamma\bar{\lambda})^l \delta_C^{t+l} \tag{9}$$

where $\delta_R^t = r_{t+1} + V_R(s_{t+1}) - V_R(s_t)$ and $\delta_C^t = c_{t+1} + V_C(s_{t+1}) - V_C(s_t)$ denote the reward and cost temporal differences on the sample path, respectively. $\bar{\lambda}$ is a parameter that adjusts the bias-variance trade-off in GAE. In general, $V_R(s)$ and $V_C(s)$ are approximated using neural networks, which are trained with data from sampled trajectories. However, in the early learning phase, the limited data availability impedes the accuracy of these approximations, leading to diminished safety performance. To tackle this problem, we therefore propose GenSafe in this work.

IV. Reduced Order Markov Decision Process

The central idea of the proposed GenSafe involves utilizing a Reduced Order Markov Decision Process (ROMDP) as a low-dimensional proxy for approximating the cost function in CMDP. In this section, we explain how to construct the ROMDP using order reduction techniques that leverage data observed during the learning process (Sec. IV-A). The construction process (Sec. IV-B) consists of five steps: state abstraction (Sec. IV-B1), action abstraction (Sec. IV-B2), cost abstraction (Sec. IV-B3), transition abstraction (Sec. IV-B4), and policy abstraction (Sec. IV-B5). More details are presented as follows.

A. Data Samples

During the online learning process, we receive sampled data about the reward and cost returns. For better explaining the construction of the ROMDP, we first clarify how data samples are defined in this work.

We denote the entire set of currently available data as $\mathcal{D}$, which is defined as

$$\mathcal{D} = \left\{ \begin{array}{l} D_1, \\ D_2, \\ \dots \\ D_i, \\ \dots \\ D_n \end{array} \right\} = \left\{ \begin{array}{c} (s_1, a_1, s_1', r_1, c_1), \\ (s_2, a_2, s_2', r_2, c_2), \\ \dots \\ (s_i, a_i, s_i', r_i, c_i), \\ \dots \\ (s_n, a_n, s_n', r_n, c_n) \end{array} \right\} \tag{10}$$

where each data sample corresponds to one timestep in the learning process. $D_i = (s_i, a_i, s_i', r_i, c_i)$, $i = 1, \dots, n$ represents the i-th data sample that contains the current state s_i, the applied action a_i, the subsequent state s_i' for this given state-action pair, and the associated single-stage reward r_i and cost c_i. The size of the dataset is denoted by $|\mathcal{D}| = n$.

For brevity, we also denote the set of all current states included in $\mathcal{D}$ as $\mathcal{D}_s = \{s_1, \dots, s_n\}$. Similarly, we use $\mathcal{D}_a = \{a_1, \dots, a_n\}$, $\mathcal{D}_{s'} = \{s_1', \dots, s_n'\}$, $\mathcal{D}_r = \{r_1, \dots, r_n\}$, $\mathcal{D}_c = \{c_1, \dots, c_n\}$ to represent the set of applied actions, subsequent states, rewards and costs in $\mathcal{D}$, respectively. These datasets are then utilized to construct a low-dimensional ROMDP for approximating the cost function.

B. Construction of ROMDP

In complex autonomous systems, the state s usually comprises observations of numerous physical attributes, resulting in a high-dimensional state space $S \subseteq \mathbb{R}^{n_s}$. This high dimensionality presents computational challenges for function approximation, especially when data samples are insufficient. To address this problem and improve constraint satisfaction in the early learning phase, we apply order reduction techniques to transform the original state s into a new, simplified state s^r (referred to as the reduced state in this paper) with a reduced dimensionality of the state space $S^r \subseteq \mathbb{R}^{n_{s^r}}$, i.e., $n_{s^r} \ll n_s$. Following this transformation, we construct a ROMDP based on the reduced state s^r accordingly, which is defined as follows.

Definition 1 (ROMDP)**.** *Given a CMDP $\mathcal{M} = (S, A, T, R, C, \gamma, \mu)$, a corresponding ROMDP $\mathcal{M}^r = (S^r, A^r, T^r, C^r, \gamma, \mu^r)$ is constructed by using the reduced state $s^r \in S^r$ and the reduced action $a^r \in A^r$, where $s^r = f_s(s)$ and $a^r = f_a(a)$. $f_s : S \to S^r$ and $f_a : A \to A^r$ are referred to as the state abstraction function and the action abstraction function, respectively. $T^r : S^r \times A^r \times S^r \to [0, 1]$ is the reduced transition kernel for the reduced states and actions. $C^r : S^r \times A^r \to \mathbb{R}$ denotes the reduced cost function that serves as a low-dimensional approximation of C. The discount factor γ remains the same as in the CMDP. The initial state distribution μ^r is derived by applying f_s to μ accordingly.*

Since the ROMDP is specifically employed to improve constraint satisfaction in SRL, it incorporates only the cost

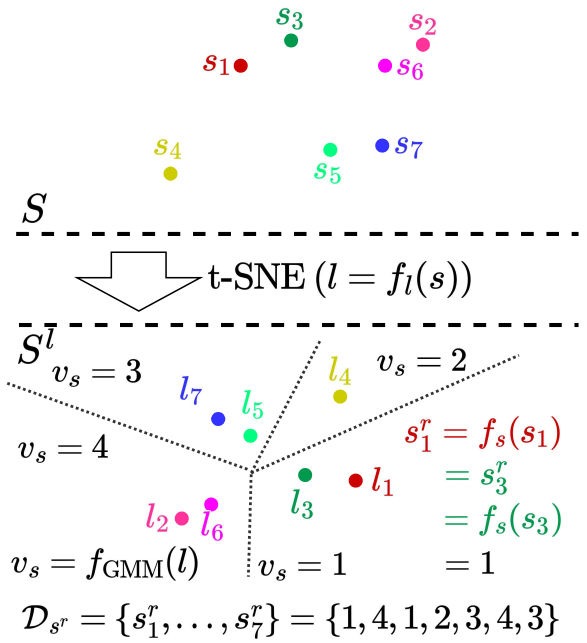


Fig. 2. Example of the state abstraction. Through applying t-SNE, a set of original states $\mathcal{D}_s = \{s_1, \ldots, s_7\}$ is transformed into a corresponding set of low-dimensional states $\mathcal{D}_l = \{l_1, \ldots, l_7\}$, where similar high-dimensional data points are represented by nearby low-dimensional states. The mapping function f_l, trained with $\mathcal{D}_s$ and $\mathcal{D}_l$, reduces the high-dimensional state space S to a two-dimensional state space S^l. The GMM classifier f_{GMM} then divides S^l into $k_s = 4$ regions, each assigned with an index $v_s \in \{1, 2, 3, 4\}$. The reduced state s^r is thus determined by using the state abstraction function $s^r = f_s(s) = f_{\text{GMM}}(f_l(s))$, e.g., we have $s_1^r = f_s(s_1) = s_3^r = f_s(s_3) = 1$ and $\mathcal{D}_{s^r} = \{s_1^r, \ldots, s_7^r\} = \{1, 4, 1, 2, 3, 4, 3\}$ in this example.

function. In the following, we present details about the construction of ROMDP, which consists of state abstraction, action abstraction, cost abstraction, transition abstraction and policy abstraction.

1) State Abstraction: To determine the state abstraction function f_s, we first apply an order reduction technique to transform the set of original states $\mathcal{D}_s = \{s_1, \ldots, s_n\}$ into a corresponding set of low-dimensional states. For this, we propose using t-distributed Stochastic Neighbor Embedding (t-SNE) [57], a method originally developed for visualizing high-dimensional data. Compared to traditional methods like Principal Component Analysis (PCA), t-SNE is able to more effectively distinguish and classify high-dimensional data. Through the usage of Euclidean distance as the similarity metric, t-SNE projects high-dimensional data points into a two- or three-dimensional state space S^l, in which similar high-dimensional data points are represented by nearby low-dimensional data points with high probability (see Fig. 2 for an example). In this work, we consider the order reduction to a two-dimensional state space, i.e., $S^l \subseteq \mathbb{R}^2$. The result of applying t-SNE is a set of low-dimensional states $D_l = \{l_1, \ldots, l_n\}$, where each state l_i is the reduced representation of its original state s_i. For a more comprehensive explanation of how t-SNE computes these low-dimensional states, please refer to Appendix A and [57].

Note that, while t-SNE effectively determines the values of the low-dimensional states, it does not provide an explicit expression for the mapping function $l = f_l(s) : S \to S^l$. To address this, we train a neural network to approximate this mapping function using the set of original states $\mathcal{D}_s$ and the corresponding set of low-dimensional states $\mathcal{D}_l$. Then, by leveraging the mapping function f_l, we are able to reduce the dimensionality of the original state space S to a two-dimensional state space S^l. However, dealing with a continuous state space often still presents computational challenges. Therefore, to further improve computational efficiency and make it more manageable for cost function approximation, we proceed to discretize the low-dimensional state space S^l.

A frequently used approach for discretization is dividing the entire state space S^l into grids of fixed sizes. However, this method may not be effective when data points are unevenly distributed across the state space (see Fig. 5a for an example). Therefore, in this work, we employ a Gaussian Mixture Model (GMM) classifier [58], which is trained on $\mathcal{D}_l$, for discretization. With a predefined number of clusters k_s, the GMM classifier segments the low-dimensional state space S^l into k_s distinct cluster regions, with each region being assigned an index $v_s \in \{1, 2, \ldots, k_s\}$. For any given low-dimensional state l, its corresponding cluster index is determined by $v_s = f_{\text{GMM}}(l)$, where $f_{\text{GMM}} : S^l \to \{1, 2, \ldots, k_s\}$ denotes the GMM-based classification process. Hence, the reduced state s^r can be effectively represented by the cluster index v_s, and the reduced state space S^r becomes the collection of all these cluster indices, i.e., we have $s^r = v_s \in S^r = \{1, \ldots, k_s\}$. Consequently, the state abstraction function f_s is defined as

$$s^r = f_s(s) = f_{\text{GMM}}(f_l(s)), \tag{11}$$

which indicates that the original state s is first transformed into a corresponding low-dimensional state l, and then is classified by the GMM classifier (see Fig. 2). We denote the set of reduced states that corresponds to $\mathcal{D}_s$ as $\mathcal{D}_{s^r} = \{s_1^r, \ldots, s_n^r\}$.

Remark 2. *While it is possible to directly learn a GMM classifier in the original state space S, accurately approximating a high-dimensional state space usually demands a considerable amount of data. In contrast, by introducing a low-dimensional state space S^l, we are able to reduce the data requirement. This strategy is based on the assumption that original states s that map to the same reduced state s^r are likely to have similar cost estimates. From this perspective, the reduced state s^r can be viewed as a low-dimensional safety feature that simplifies the computational complexity while still maintaining the capability to effectively capture critical safety characteristics.*

2) Action Abstraction: Normally, the action space $A \subseteq \mathbb{R}^{n_a}$ is of smaller dimensionality compared to the state space S, and the actions are usually distributed more evenly. Therefore, for the action abstraction, we opt for a straightforward approach of discretizing the action space A into grids of fixed sizes. Given a specified number of grids per dimension k_a, this results in a total of $k_a^{n_a}$ grids. We assign each of these grids an index $v_a \in \{1, 2, \ldots, k_a^{n_a}\}$. In a manner analogous to the reduced states, the reduced action a^r is thus represented by this index v_a, and the reduced action space A^r is the collection of all

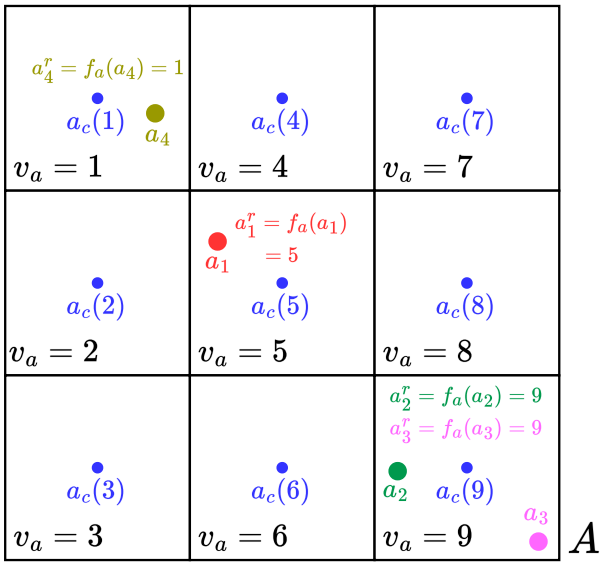


Fig. 3. Example of the action abstraction. For a two-dimensional action space $A \subseteq \mathbb{R}^2$, we discretize it using $k_a = 3$, which results in a total of $k_a^{n_a} = 9$ grids. Each grid cell is assigned an index $v_a \in \{1, \ldots, 9\}$. $a_c(1), \ldots, a_c(9)$ denote the center of each grid. For a set of original applied actions $\mathcal{D}_a = \{a_1, \ldots, a_4\}$, we thus have $a_1^r = f_a(a_1) = 5$, $a_2^r = f_a(a_2) = a_3^r = f_a(a_3) = 9$, $a_4^r = f_a(a_4) = 1$.

such indices, i.e., $a^r = v_a \in A^r = \{1, 2, \ldots, k_a^{n_a}\}$. The action abstraction function f_a is thus defined as

$$a^r = f_a(a) = \underset{v_a}{\operatorname{argmin}} \, ||a - a_c(v_a)||^2 \tag{12}$$

where $a_c(v_a)$ indicates the center of the grid cell indexed by v_a (see Fig. 3 for an example). f_a can be considered as a localization function that, for a given action a, determines the specific grid cell it falls within and then returns the corresponding index v_a. The set of reduced actions that corresponds to $\mathcal{D}_a$ are denoted as $\mathcal{D}_{a^r} = \{a_1^r, \ldots, a_n^r\}$.

Remark 3. *In this work, we assume that the action space A has a relatively small dimensionality, rendering it manageable for straightforward discretization using fixed grid sizes, e.g., $n_a = 2$ or $n_a = 3$. For scenarios where the action space possesses a higher dimensionality, a similar order reduction technique that is used for the state space can be applied to enhance computational efficiency.*

3) Cost Abstraction: After the state and action abstractions, we are now able to approximate the original cost function C with a reduced cost function C^r, which leverages the reduced states and actions. To achieve this, we first reorganize the set of observed costs $\mathcal{D}_c = \{c_1, \ldots, c_n\}$ by using its associated sets of reduced states $\mathcal{D}_{s^r}$ and reduced actions $\mathcal{D}_{a^r}$, which are obtained by applying the state and action abstraction functions to $\mathcal{D}_s$ and $\mathcal{D}_a$, respectively. For each specific pair of the reduced state and action, we organize the entire dataset $\mathcal{D}$ to identify a corresponding set of observed costs as follows

$$\mathcal{D}^C_{s^r,a^r} = \{c_i | f_s(s_i) = s^r, f_a(a_i) = a^r\}, \tag{13}$$

which includes all costs associated with the given reduced state s^r and action a^r. The size of this dataset is denoted as $|\mathcal{D}^C_{s^r,a^r}| = n_{s^r,a^r}$.

Then, the reduced cost function C^r is computed as follows

$$C^r(s^r, a^r) = \begin{cases} \dfrac{\sum_{c_i \in \mathcal{D}^C_{s^r,a^r}} c_i}{n_{s^r,a^r}}, & \text{if } n_{s^r,a^r} > 0, \\ \Delta, & \text{if } n_{s^r,a^r} = 0, \end{cases} \tag{14}$$

which indicates that the reduced cost is determined by averaging the observed costs within $\mathcal{D}^C_{s^r,a^r}$ if this set is not empty. When no observed costs are available for a given reduced state and action pair, the reduced cost is set to a predefined estimated value Δ. Thus, the value of the original cost function $C(s, a)$ can be approximated by using the reduced cost function C^r through converting the original state s and action a into their corresponding reduced state s^r and action a^r, i.e., $C(s, a) \approx C^r(s^r, a^r)$ with $s^r = f_s(s)$ and $a^r = f_a(a)$ (see also Example 1). Consequently, the reduced cost function C^r serves as a low-dimensional proxy for the original cost function C, enhancing the efficiency of estimation and computation processes.

Example 1. *Consider an example where the entire dataset $\mathcal{D}$ contains 5 data samples, resulting in the respective sets of original states $\mathcal{D}_s = \{s_1, \ldots, s_5\}$, original actions $\mathcal{D}_a = \{a_1, \ldots, a_5\}$ and observed costs $\mathcal{D}_c = \{c_1, \ldots, c_5\}$. Suppose that after applying the state and action abstraction functions, we obtain the corresponding sets of reduced states and actions as $\mathcal{D}_{s^r} = \{s_1^r, \ldots, s_5^r\} = \{1, 1, 2, 2, 3\}$ and $\mathcal{D}_{a^r} = \{a_1^r, \ldots, a_5^r\} = \{1, 1, 1, 2, 2\}$, where we assume that $S^r = \{1, 2, 3\}$ and $A^r = \{1, 2\}$. Thus, by sorting the dataset $\mathcal{D}$, we identify the following sets of observed costs $\mathcal{D}^C_{s^r,a^r}$ for each reduced state and action pair:*

$$\begin{aligned} &\mathcal{D}^C_{1,1} = \{c_1, c_2\}, \quad &&\mathcal{D}^C_{1,2} = \varnothing, \\ &\mathcal{D}^C_{2,1} = \{c_3\}, \quad &&\mathcal{D}^C_{2,2} = \{c_4\}, \\ &\mathcal{D}^C_{3,1} = \varnothing, \quad &&\mathcal{D}^C_{3,2} = \{c_5\}. \end{aligned} \tag{15}$$

Therefore, the reduced cost function is calculated as

$$\begin{aligned} &C^r(1,1) = \frac{c_1 + c_2}{2}, \quad &&C^r(1,2) = \Delta, \\ &C^r(2,1) = c_3, \quad &&C^r(2,2) = c_4, \\ &C^r(3,1) = \Delta, \quad &&C^r(3,2) = c_5. \end{aligned} \tag{16}$$

For any new state and action pair, the original cost function $C(s, a)$ is estimated by using the corresponding reduced state and action, e.g., if $f_s(s_6) = s_6^r = 2$ and $f_a(a_6) = a_6^r = 1$, then we have $C(s_6, a_6) \approx C^r(s_6^r, a_6^r) = C^r(2, 1) = c_3$.

Note that, while the single-stage cost can be directly estimated using the reduced cost function C^r, solving the SRL problem (4) often requires the estimation of future costs $J_C(\pi_\theta)$ under the current policy π_θ. Hence, to facilitate an accurate estimation of these future costs, we further incorporate transition abstraction and policy abstraction into the construction of ROMDP.

4) Transition Abstraction: To determine the reduced transition kernel T^r, we first transform the set of observed subsequent states $\mathcal{D}_{s'} = \{s'_1, \ldots, s'_n\}$ into a corresponding set of subsequent reduced states $\mathcal{D}_{s^{r'}} = \{s_1^{r'}, \ldots, s_n^{r'}\}$ using the state abstraction function f_s. Similar to the process of cost abstraction, we then identify a corresponding set of subsequent

reduced states for each reduced state and action pair by reorganizing $\mathcal{D}_{s^{r'}}$ alongside its associated $\mathcal{D}_{s^r}$ and $\mathcal{D}_{a^r}$

$$\mathcal{D}^{s^{r'}}_{s^r,a^r} = \{s_i^{r'} | f_s(s_i) = s^r, f_a(a_i) = a^r\}, \tag{17}$$

where each element represents an observed data sample for the transition path $s^r \times a^r \to s^{r'}$. Note that, the size of this dataset is the same as $\mathcal{D}^C_{s^r,a^r}$, i.e., $|\mathcal{D}^{s^{r'}}_{s^r,a^r}| = n_{s^r,a^r}$, as both datasets are derived by sorting the entire dataset $\mathcal{D}$ according to the reduced state and action pairs. Subsequently, we count the number of observed data samples for each possible transition path $s^r \times a^r \to s^{r'}$, which is denoted as $n_{s^r \times a^r \to s^{r'}}$. The reduced transition kernel T^r is thus computed as

$$T^r(s^r, a^r, s^{r'}) = \begin{cases} \frac{n_{s^r \times a^r \to s^{r'}}}{n_{s^r,a^r}}, & \text{if } n_{s^r,a^r} > 0, \\ \frac{1}{k_s}, & \text{if } n_{s^r,a^r} = 0, \end{cases} \tag{18}$$

where $k_s = |S^r|$ is the size of the reduced state space. This implies that when data samples are available for a specific reduced state and action pair, the probability of transitioning to a particular subsequent reduced state $s^{r'}$ is proportional to the observed frequency of this transition path $s^r \times a^r \to s^{r'}$ among all observed data samples for this pair (see Example 2). Conversely, if no data exists, all subsequent reduced states are assigned equal transition probabilities. Hence, for each given pair of the reduced state s^r and action a^r, we ensure that $\sum_{s^{r'} \in S^r} T^r(s^r, a^r, s^{r'}) = 1$ is satisfied.

Example 2. *Continued from Example 1, we assume that the corresponding set of subsequent reduced states is* $\mathcal{D}_{s^{r'}} = \{s_1^{r'}, \ldots, s_5^{r'}\} = \{1, 2, 2, 3, 1\}$. *Hence, we have*

$$\begin{aligned} &\mathcal{D}^{s^{r'}}_{1,1} = \{s_1^{r'}, s_2^{r'}\} = \{1, 2\}, \quad &&\mathcal{D}^{s^{r'}}_{1,2} = \varnothing, \\ &\mathcal{D}^{s^{r'}}_{2,1} = \{s_3^{r'}\} = \{2\}, &&\mathcal{D}^{s^{r'}}_{2,2} = \{s_4^{r'}\} = \{3\}, \\ &\mathcal{D}^{s^{r'}}_{3,1} = \varnothing, &&\mathcal{D}^{s^{r'}}_{3,2} = \{s_5^{r'}\} = \{1\}, \end{aligned} \tag{19}$$

with the counts of observed transition paths as $n_{1\times1\to1} = 1$, $n_{1\times1\to2} = 1$, $n_{2\times1\to2} = 1$, $n_{2\times2\to3} = 1$, $n_{3\times2\to1} = 1$, *and* $n_{s^r\times a^r\to s^{r'}} = 0$ *for any other transition paths. The reduced transition kernel* T^r *is thus calculated by using these counts. For instance, since* $n_{1,1} = 2$, $n_{1,2} = 0$, $n_{2,1} = 1$, *we have*

$$\begin{aligned} &T^r(1,1,1) = \frac{1}{2}, \quad T^r(1,1,2) = \frac{1}{2}, \quad T^r(1,1,3) = 0, \\ &T^r(1,2,1) = \frac{1}{3}, \quad T^r(1,2,2) = \frac{1}{3}, \quad T^r(1,2,3) = \frac{1}{3}, \\ &T^r(2,1,1) = 0, \quad T^r(2,1,2) = 1, \quad T^r(2,1,3) = 0. \end{aligned} \tag{20}$$

5) Policy Abstraction: To predict future costs, we also construct a reduced policy $\pi^r(s^r) : S^r \to A^r$ that emulates the original policy π_θ. It determines the probability of selecting a specific reduced action a^r for a given reduced state s^r. For this, we utilize the counts of observed data samples for each reduced state and action pair n_{s^r,a^r} that are determined when creating the datasets $\mathcal{D}^C_{s^r,a^r}$ and $\mathcal{D}^{s^{r'}}_{s^r,a^r}$. These counts reflect instances where, upon converting the original state and action to their reduced forms, the original policy π_θ selects a^r as the action for the reduced state s^r. The total number of observed pairs for a specific reduced state s^r is denoted as n_{s^r}, where we have $n_{s^r} = \sum_{a^r \in A^r} n_{s^r,a^r}$. Then, the reduced policy π^r is defined by the following probabilities

$$\mathbb{P}(\pi^r(s^r) = a^r) = \begin{cases} \frac{n_{s^r,a^r}}{n_{s^r}}, & \text{if } n_{s^r} > 0, \\ \frac{1}{k_a^{n_a}}, & \text{if } n_{s^r} = 0, \end{cases} \tag{21}$$

where $k_a^{n_a} = |A^r|$ is the size of the reduced action space. Similarly, this indicates that when data are available, the probability of choosing the reduced action a^r for the reduced state s^r is proportional to its observed frequency relative to the total observations for this state (see Example 3). For reduced states s^r without data, we adopt an equal probability distribution across all possible reduced actions. Therefore, for each given reduced state s^r, we have $\sum_{a^r \in A^r} \mathbb{P}(\pi^r(s^r) = a^r) = 1$.

Example 3. *Based on Example 1, we have the following counts for each reduced state and action pair:* $n_{1,1} = 2$, $n_{1,2} = 0$, $n_{2,1} = 1$, $n_{2,2} = 1$, $n_{3,1} = 0$, $n_{3,2} = 1$. *Hence, we have* $n_1 = n_{1,1} + n_{1,2} = 2$, $n_2 = n_{2,1} + n_{2,2} = 2$, $n_3 = n_{3,1} + n_{3,2} = 1$. *The reduced policy* π^r *is thus represented by the following probabilities*

$$\begin{aligned} &\mathbb{P}(\pi^r(1) = 1) = 1, \quad \mathbb{P}(\pi^r(1) = 2) = 0, \\ &\mathbb{P}(\pi^r(2) = 1) = \frac{1}{2}, \quad \mathbb{P}(\pi^r(2) = 2) = \frac{1}{2}, \\ &\mathbb{P}(\pi^r(3) = 1) = 0, \quad \mathbb{P}(\pi^r(3) = 2) = 1. \end{aligned} \tag{22}$$

Through the aforementioned comprehensive abstraction processes, we develop a ROMDP model that functions as a low-dimensional proxy for the cost function in CMDP. It is capable of efficiently predicting not only the single-stage costs but also the future costs under a given policy π_θ. In the next section, we explain how to integrate this constructed ROMDP into our proposed GenSafe for improving the safety performance of SRL algorithms.

V. Generalizable Safety Enhancer

The fundamental concept behind the proposed Generalizable Safety Enhancer (GenSafe) is to reformulate the original cost constraint in CMDP into constraints based on the ROMDP. Then, GenSafe enhances the performance of SRL algorithms by modifying each applied action in accordance with these revised constraints. In essence, GenSafe acts as an additional safety layer for SRL algorithms (see Fig. 1). Further details are provided in the remaining part of this section.

A. Action Correction Based on ROMDP

Solving the SRL problem (4) necessitates the prediction of future costs $J_C(\pi_\theta)$ under a given policy π_θ. However, accurately estimating $J_C(\pi_\theta)$ is a challenging task and often demands complex computations, e.g., (8)-(9) in PPO-Lagrangian. To enable a more reliable and efficient SRL process during the early learning phase, we utilize the constructed ROMDP to approximate cost returns. Nonetheless, it should be noted that a certain degree of information loss is inevitable in the order reduction process, resulting in discrepancies between the estimated and the actual values. Such inaccuracies can accumulate over time, especially with longer prediction horizons. This complicates the task of performing a reliable policy-level

Algorithm 1 Value iteration for calculating $V^r_{C^r}(s^r)$

Require: The constructed ROMDP, tolerance δ
Initialize $V^r_{C^r}(s^r) = 0$ for all $s^r \in S^r$
$\Delta_V = \delta$
while $\Delta_V \geq \delta$ **do**
  $\Delta_V = 0$
  for $s^{r,i} \in S^r$, $i = 1, \ldots, k_s$ **do**
    $V_{\mathbf{current}} = V^r_{C^r}(s^{r,i})$
    Update $V^r_{C^r}(s^{r,i})$ according to (27)-(28)
    $\Delta_V = \max(\Delta_V, |V_{\mathbf{current}} - V^r_{C^r}(s^{r,i})|)$
  end for
end while

modification, i.e., directly revising the policy π_θ, based on ROMDP predictions.

To address this issue and fully utilize the prediction capability of the ROMDP, we draw inspiration from [29] and opt for action-level corrections rather than policy-level modifications. At each timestep of the learning process, the current SRL policy proposes an action $a_0 = \pi_\theta(s_0)$ based on the current state s_0. Then, we solve the following optimization problem to refine this action a_0 such that the modified action a_m is more likely to satisfy the safety constraints

$$a_m = \underset{a}{\operatorname{argmin}} ||a - a_0||^2, \tag{23}$$

$$\text{s.t.} \quad C^r(s^r_0, a^r) \leq d_s, \tag{24}$$

$$C^r(s^r_0, a^r) + \gamma V^r_{C^r}(s^{r'}_0) \leq d, \tag{25}$$

where $s^r_0 = f_s(s_0)$ and $a^r = f_a(a)$ are the corresponding reduced state and action, respectively. $s^{r'}_0$ denotes the subsequent reduced state resulting from applying a^r to s^r_0, i.e., $s^r_0 \times a^r \rightarrow s^{r'}_0$. The goal of this optimization is to find an action a_m that remains as close as possible to the initially determined action a_0 while satisfying the constraints (24) and (25). (24) imposes an immediate cost constraint that the single-stage cost estimated by the ROMDP should not exceed a predefined threshold d_s. This constraint is motivated by the fact that, in various autonomous systems, e.g., autonomous vehicles or robotic manipulators, the immediate cost can significantly impact system safety [29]. Therefore, we additionally incorporate this constraint into the optimization process. (25) mirrors the original cost constraint $J_C(\pi_\theta) = C(s_0, a_0) + \gamma V_C(s_0^{'}) \leq d$, where $s_0 \times a_0 \rightarrow s_0^{'}$ and the state, action and cost are substituted with their reduced forms contained in the ROMDP, respectively. $V^r_{C^r}(s^r)$ denotes the reduced value function for the reduced state s^r and cost C^r under the reduced policy π^r. Taking into account the reduced transition kernel T^r, (25) can be further reformulated as

$$C^r(s^r_0, a^r) + \gamma \sum_{s^r \in S^r} T^r(s^r_0, a^r, s^r) V^r_{C^r}(s^r) \leq d. \tag{26}$$

However, although the reduced cost C^r is identified during the ROMDP construction, tackling (26) requires the computation of the reduced value function $V^r_{C^r}(s^r)$ for each reduced state s^r. For this, we adapt the value iteration algorithm [59] by replacing the traditional maximum operation with an average operation that accounts for the probabilities defined by the reduced policy π^r (see Algorithm 1). After initializing the values of all reduced states to zero, we iteratively update $V^r_{C^r}(s^r)$ for each reduced state $s^{r,i}$, $i = 1, \ldots, k_s$ contained in the reduced state space S^r by utilizing the Bellman equation

$$V^r_{C^r}(s^{r,i}) = \sum_{a^r \in A^r} \mathbb{P}(\pi^r(s^{r,i}) = a^r) V^r_{s^{r,i}, a^r}, \tag{27}$$

with

$$V^r_{s^{r,i}, a^r} = C^r(s^{r,i}, a^r) + \gamma \sum_{s^r \in S^r} T^r(s^{r,i}, a^r, s^r) V^r_{C^r}(s^r), \tag{28}$$

where the values are calculated through averaging the probabilities given by the reduced transition kernel T^r and policy π^r, rather than the maximum operation employed in standard value iteration. The resulting reduced value function $V^r_{C^r}(s^r)$ estimates the future cost returns for any given reduced state s^r under the reduced policy π^r. The update is terminated when the difference in the value function between two successive iterations falls beneath a predefined threshold δ.

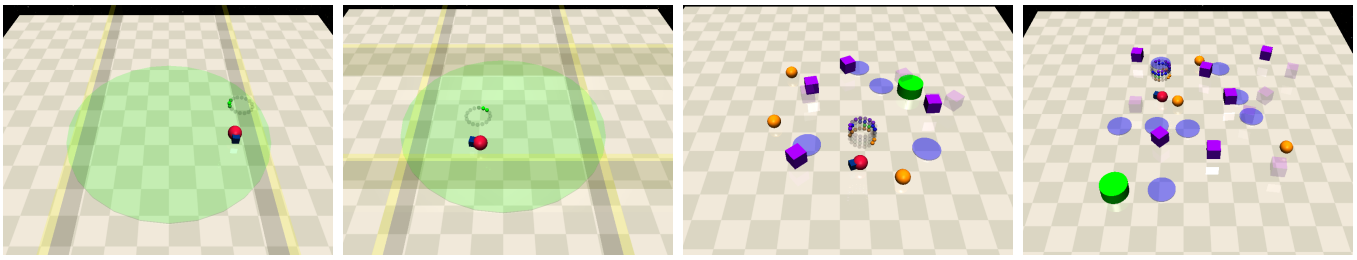
Once the reduced value function $V^r_{C^r}(s^r)$ is determined, verifying constraints (24) and (26) for any chosen action a becomes straightforward, akin to referencing a look-up table. However, due to the non-differentiable nature of both the reduced cost C^r and value function $V^r_{C^r}(s^r)$, solving the optimization problem (23) through gradient-based methods is not feasible. Therefore, we employ derivative-free optimization techniques to find the modified action a_m. Specifically, we use Particle Swarm Optimization (PSO) [60] in this work. Consequently, the proposed GenSafe incorporates this action correction mechanism, grounded in ROMDP predictions, to enhance the safety performance of SRL algorithms.

Algorithm 2 SRL with GenSafe

Require: Number of training epochs k_e, number of timesteps per each epoch k_t
Initialize SRL policy π_θ, ROMDP $\mathcal{M}^r = \varnothing$
$i = 0$, $\mathcal{D} = \varnothing$
while $i < k_e$ **do** ▷ Training loop
  $j = 0$
  while $j < k_t$ **do** ▷ Collect Data
    $a_0 = \pi_\theta(s_0)$ ▷ SRL policy
    if $\mathcal{M}^r \neq \varnothing$ **then**
      Find a_m by solving (23) ▷ Action correction
    else
      $a_m = a_0$
    end if
    Add observed data sample $D = (s_0, a_m, s_0', r, c)$ to $\mathcal{D}$
  end while
  Construct the ROMDP $\mathcal{M}^r$ based on $\mathcal{D}$
  Update the policy π_θ with the selected SRL algorithm
end while

B. SRL with GenSafe

The integration of the proposed GenSafe with a chosen model-free SRL algorithm, e.g., PPO-Lagrangian, is delineated in Algorithm 2. Throughout the learning process, at each timestep, the SRL policy π_θ determines an action a_0 based on the current state s_0. Subsequently, provided that the ROMDP has been constructed, i.e., except during the initial epoch, the proposed GenSafe solves the optimization problem (23) to identify a modified action a_m. This action is then applied to the system, with the resultant observed data sample being



(a) Circle Level 1 (b) Circle Level 2 (c) Button Level 1 (d) Button Level 2

Fig. 4. SRL tasks. (a)-(b) *Circle Level 1* and *Circle Level 2*. The red dot represents the agent robot, which operates based on point dynamics. The green circle signifies the target circular path that the robot aims to follow. Meanwhile, the yellow lines depict the walls that the robot should avoid colliding with. (c)-(d) *Button Level 1* and *Button Level 2*. The red dot symbolizes the robot. The orange spheres denote the buttons, among which only one is designated as the goal button. The green pillar represents a fixed obstacle, and the purple cubes depict moving obstacles. The blue circles outline the hazardous zones.

incorporated into the dataset $\mathcal{D}$. Upon reaching a predefined number of timesteps within each epoch, the ROMDP is reconstructed leveraging the most recent dataset $\mathcal{D}$. Meanwhile, the SRL policy π_θ is updated according to the selected SRL algorithm. In essence, GenSafe acts as an additional safety layer to SRL algorithms (see also Fig. 1). Given its focus on action-level corrections, GenSafe is inherently compatible with a broad spectrum of model-free SRL algorithms, enhancing their safety performance without necessitating modifications to their core methodologies.

C. Dynamic Activation of GenSafe

As the dataset $\mathcal{D}$ enlarges, the prediction accuracy of the ROMDP is expected to improve. However, due to the inherent information loss in the order reduction process, discrepancies between the estimated and actual costs could persist regardless of the data volume. In contrast, with a sufficient amount of data, traditional methods of approximating value functions $V_R(s)$ and $V_C(s)$ via neural networks are able to provide accurate estimations, thereby leading to satisfactory performance of SRL algorithms. This rationale forms the basis for our strategy of applying the ROMDP for enhancing the safety performance of SRL algorithms, particularly in the early learning phases where data availability is often limited.

Recognizing this characteristic, we therefore introduce a dynamic activation mechanism for GenSafe. For instance, with PPO-Lagrangian as the SRL algorithm, we assess the accuracy of the current cost value function $V_C(s)$ approximation after each epoch update. This can be done by determining whether the loss value from updating the neural network falls below a predetermined threshold. If it does, we deactivate GenSafe, meaning no action corrections will be applied, and the action chosen by the SRL policy π_θ is directly utilized. Subsequently, we continue to monitor the loss value associated with approximating the cost value function $V_C(s)$ throughout the learning process. If this value again exceeds a certain threshold, indicating a degradation in approximation accuracy, GenSafe is reactivated. Such a dynamic activation mechanism not only enhances computational efficiency but also ensures that, in later learning phases, the agent's performance will not be overly affected by the information loss caused by the order reduction process.

Remark 4. *Different criteria, e.g., the number of training epochs completed or the observed average cost, can also be employed to determine the deactivation or reactivation of GenSafe. Choosing these criteria should consider the specific requirements and properties of the task, alongside the characteristics of the selected SRL algorithm, to ensure a satisfying performance of GenSafe tailored to the learning context.*

D. Practical Implementation Details

During the practical implementation of GenSafe, we adopt several strategies to improve computational efficiency. Firstly, instead of reconstructing the ROMDP at every epoch, we establish a periodic update interval, meaning that the ROMDP is reconstructed only after a specified number of epochs. Secondly, considering the available computational resources and memory constraints, we impose a limit on the maximum number of data samples that can be stored in the dataset $\mathcal{D}$. Upon reaching this limit, we prioritize the retention of newer data samples and discard older entries to maintain the dataset size. Thirdly, given that the computational cost of t-SNE increases with more data samples, we also limit the maximum number of data samples employed in t-SNE computations. If the number of available data samples surpasses this limit, we randomly select a subset from $\mathcal{D}$ to use, ensuring manageable computational loads. These measures collectively enhance the computational efficiency of GenSafe. However, the determination of the parameter values should take into account the available computational resources as well as the specific requirements of the tasks at hand.

VI. Experimental Results

In this section, we evaluate the performance of the proposed GenSafe on various SRL benchmark problems. We first explain the employed experimental setup. Then, we demonstrate the outcomes of the model order reduction process, illustrating the effectiveness of utilizing t-SNE for identifying the reduced state space. Finally, we present the safety and task performance, i.e., the reward and cost returns, observed throughout the learning process. More details are provided as follows.

A. Experimental Setup

We assess the performance of GenSafe on four representative SRL tasks that are given in Safety-Gymnasium [61], a

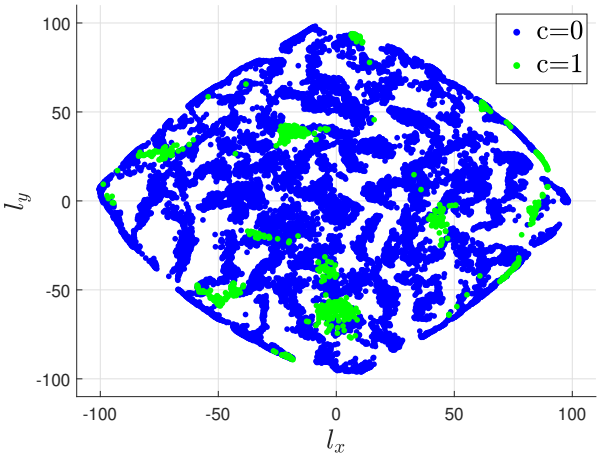
collection of SRL environments that builds upon the widely recognized Safety-Gym [12] benchmark. The tasks we used are: *Circle Level 1*, *Circle Level 2*, *Button Level 1*, and *Button Level 2* (see Fig. 4). In the circle tasks, a robot needs to follow a circular path while avoiding collisions with the surrounding walls. *Circle Level 1* has two walls, whereas *Circle Level 2* includes four walls, increasing the complexity of the task (see Fig. 4a and Fig. 4b). In the button tasks, the robot's objective is to press a designated goal button while circumventing obstacles and hazardous zones. Meanwhile, it should also not activate any incorrect buttons. *Button Level 2* has an increased number of obstacles and hazardous zones compared to *Button Level 1* (see Fig. 4c and Fig. 4d). The state space for the circle tasks is 28-dimensional, while the button tasks present a more complex 76-dimensional state space. All tasks employ a two-dimensional action space. For more details about these task environments, the readers may refer to [61].

For each task, we compare the performance of five different learning methods: (1) *PPO*, which serves as the reference method to illustrate the standard performance of DRL algorithms; (2) *PPO-Lagrangian (PL)*, which is a conventional SRL approach typically used as the baseline for SRL algorithm assessments; (3) *FOCOPS (FO)*, a state-of-the-art SRL method proposed in [25], which is considered to have a better performance than PPO-Lagrangian; (4) *PPO-Lagrangian with GenSafe (PL-G)* and *(5) FOCOPS with GenSafe (FO-G)*, which integrate the PPO-Lagrangian and FOCOPS algorithms with the proposed GenSafe, respectively. The PL-G and FO-G methods are utilized to demonstrate the effectiveness of GenSafe on enhancing the performance of existing SRL algorithms. To ensure a thorough comparison, each method is executed on each task using five randomly selected seeds, and the results are averaged accordingly.

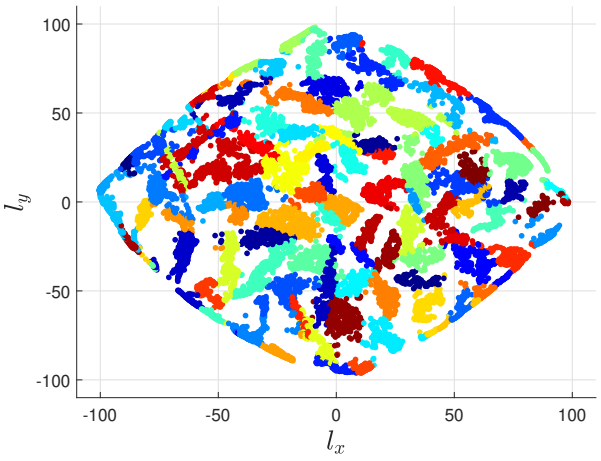
The following parameters are used in our experiments: number of clusters for the state abstraction $k_s = 100$, number of grids for the action abstraction $k_a = 20$, estimated cost for unobserved reduced state-action pairs $\Delta = 0.5$, immediate cost threshold $d_s = 0.3$, future cost threshold $d = 25$, number of timesteps per each epoch $k_l = 20000$, number of training epochs $k_e = 200$, tolerance for calculating the reduced value function $\delta = 0.01$. The thresholds for deactivating and reactivating GenSafe are respectively set to 2 and 5. The neural network, which approximates the mapping function f_l, has two layers with 64 neurons in each layer. The ROMDP is reconstructed every two epochs, and the maximum size of the dataset $\mathcal{D}$ is selected as 1000000. The maximum number of data samples employed for t-SNE computations is limited to 200000. For solving (23), PSO is employed with a swarm size of 20, a maximum iteration count of 50, and a termination tolerance of $1e^{-4}$.

B. Model Order Reduction

We utilize the initial construction of ROMDP in the *Button Level 1* task as an example to illustrate the results of model order reduction, i.e., the application of t-SNE for deriving a reduced state space. Given a dataset of original states $\mathcal{D}_s = \{s_1, \ldots, s_n\}$, we employ t-SNE to map these states into a corresponding set of low-dimensional states $D_l = \{l_1, \ldots, l_n\}$, with $n = 20000$ representing the first iteration. The results are depicted in Fig. 5a, where each 76-dimensional original state s_i is transformed into a two-dimensional state l_i. Note that, similar to outcomes of PCA, the dimensions of the low-dimensional states, i.e., l_x and l_y, do not possess actual physical meanings but serve only as abstract features. The coloring within the figure corresponds to the observed cost for each data sample. As the button tasks are designed to return a binary cost, each data sample is associated with a cost of either $c = 0$ or $c = 1$, where $c = 0$ and $c = 1$ denote safe and unsafe data samples, respectively. It can be observed that, t-SNE effectively reduces the original high-dimensional state

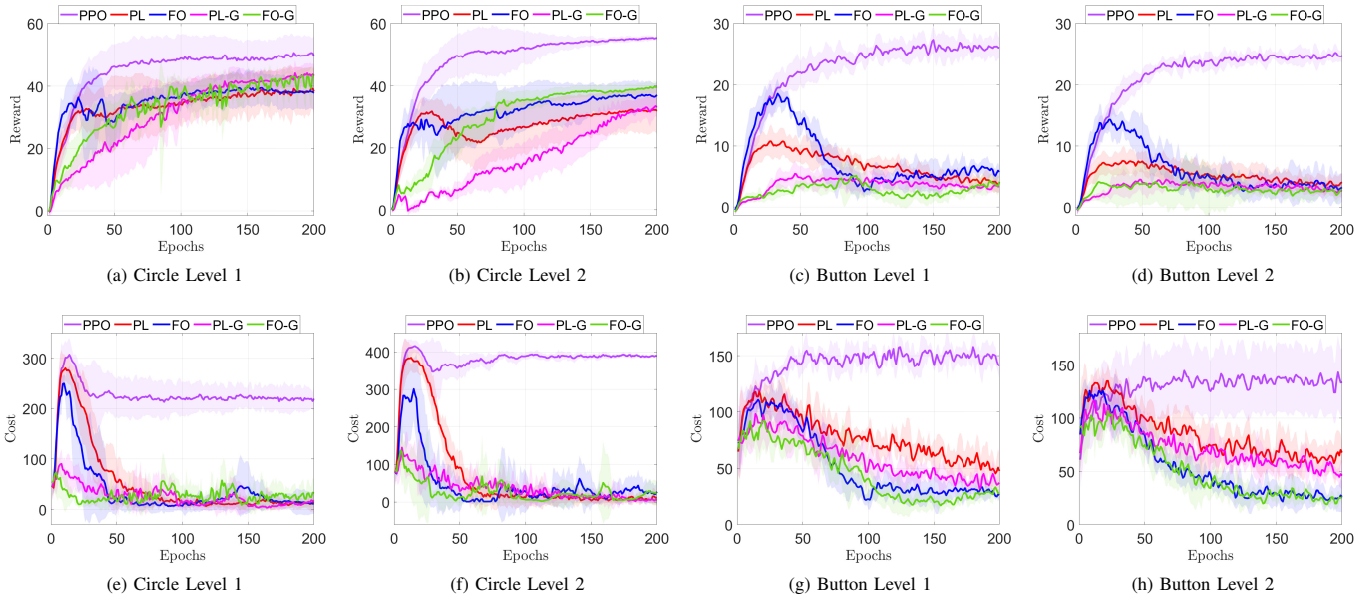


(a)



(b)

Fig. 5. (a) The set of low-dimensional states $D_l = \{l_1, \ldots, l_{20000}\}$ distributed among a two-dimensional state space S^l, which is derived by applying t-SNE on the set of original states $\mathcal{D}_s = \{s_1, \ldots, s_{20000}\}$. The dimensions l_x and l_y represent two abstract features of these low-dimensional states. Each data sample is associated with an observed cost of either $c = 0$ (blue) or $c = 1$ (green), where $c = 0$ and $c = 1$ denote safe and unsafe observations, respectively. (b) Results of the GMM classifier. Each low-dimensional state l_i is categorized into one of the $k_s = 100$ cluster regions, with each region represented by a different color and assigned an index $v_s \in \{1, 2, \ldots, 100\}$. The reduced state s^r corresponding to each original state s is therefore given by the cluster index, i.e., $s^r = v_s = f_s(s) = f_{\text{GMM}}(l) = f_{\text{GMM}}(f_l(s))$. This results in a dataset of reduced states $\mathcal{D}_{s^r} = \{s^r_1, \ldots, s^r_{20000}\}$.



(a) Circle Level 1 (b) Circle Level 2 (c) Button Level 1 (d) Button Level 2

(e) Circle Level 1 (f) Circle Level 2 (g) Button Level 1 (h) Button Level 2

Fig. 6. Rewards and costs of the selected learning methods on different SRL tasks. (a)-(d) The reward returns for the *Circle Level 1*, *Circle Level 2*, *Button Level 1*, and *Button Level 2* tasks, respectively. (e)-(h) The corresponding cost returns for the same tasks.

space S into a representative low-dimensional state space S^l, grouping safe and unsafe data samples closely. This supports the assumption that states with proximate reduced states are likely to incur similar costs, thereby formulating the basis for employing such a reduced state space to approximate the original cost function.

After determining the set of low-dimensional states $D_l = \{l_1, \ldots, l_n\}$ via t-SNE, we proceed to train a GMM classifier on this dataset, choosing the number of clusters as $k_s = 100$. The corresponding outcomes of the GMM classifier are illustrated in Fig. 5b, where each low-dimensional state l_i is categorized into one of the 100 cluster regions. We assign each cluster region an index $v_s \in \{1, 2, \ldots, 100\}$, with different colors in the figure representing different clusters. Consequently, the reduced state s^r for each original state s is given by the corresponding cluster index, i.e., $s^r = v_s = f_s(s) = f_{\text{GMM}}(l) = f_{\text{GMM}}(f_l(s))$, leading to the set of reduced states $\mathcal{D}_{s^r} = \{s^r_1, \ldots, s^r_n\}$. The reduced state space is, therefore, the collection of all these indices $S^r = \{1, 2, \ldots, 100\}$. Utilizing the identified $\mathcal{D}_{s^r}$, we accordingly construct the ROMDP. More details about the performance of GenSafe based on the constructed ROMDP are presented in the next subsection.

C. Task and Safety Performance

We compare the performance of the selected learning methods on all four SRL tasks. The results are presented in Fig. 6. For the *Circle Level 1* and *Circle Level 2* tasks, it can be observed that, compared to the baseline PPO algorithm, which does not incorporate explicit safety mechanisms, all other approaches achieve a substantial reduction in cost returns throughout the learning phase (see Fig. 6e and Fig. 6f). Moreover, the proposed GenSafe notably enhances the safety performance of existing SRL methods (see PL vs. PL-G and FO vs. FO-G). While all four methods eventually converge to a final cost nearing zero, the cost returns are significantly reduced by using GenSafe during the early learning stages (prior to 50 epochs), which aligns with GenSafe's design principle.

Meanwhile, the algorithms augmented by GenSafe, i.e., PL-G and FO-G, achieve similar final rewards to their respective original versions, albeit at a slower pace (see Fig. 6a and Fig. 6b). A primary factor contributing to the slower increase in reward is that, to realize a safer learning process, GenSafe restricts the exploration of learning algorithms to regions deemed reliably safe. While this improves safety performance, it naturally reduces the speed of reward growth, as the search areas for potentially higher-reward policies are constrained. This phenomenon is also observable when comparing PPO to other SRL algorithms, which, despite achieving higher reward returns, does so at the expense of safety due to its unrestricted exploration. Such a trade-off between safety and task performance is recognized in various SRL studies [7]–[9]. Nevertheless, GenSafe is still able to achieve final rewards comparable to those of the original SRL algorithms, demonstrating its ability to enhance safety while maintaining task performance at a satisfactory level.

Similar results can also be observed for the *Button Level 1* and *Button Level 2* tasks. As demonstrated in Fig. 6g and Fig. 6h, GenSafe realizes a notable cost reduction, especially in the early learning phases, for both tasks. However, its effectiveness is slightly diminished in the *Button Level 2* task as compared to the *Button Level 1* task. A major reason for this is that, due to the increased task complexity, achieving a safer learning process becomes more challenging, resulting in lesser cost reductions. Nevertheless, the algorithms augmented by GenSafe still outperform the original SRL algorithms in terms of safety performance. The influence of task complexity can also be noted in the trade-off between safety and task performance. Since the transition from circle tasks to button tasks brings an increase in difficulty, achieving safer behaviors

also poses more challenges. Hence, the button tasks exhibit a more obvious trade-off than the circle tasks, where achieving a safer learning process necessitates a greater compromise on reward returns (see Fig. 6c and Fig. 6d). This suggests that, for more complicated tasks, the balance between safety and task performance should be carefully determined to meet the actual requirements and goals of the task.

VII. Discussion

In this section, we discuss several important aspects of the proposed approach, as well as its limitations and possible future directions.

A. Action-level Correction

In this work, we employ action-level corrections instead of policy-level modifications. Such a strategy offers easier implementation and a broader compatibility with various SRL algorithms, thereby enhancing the applicability of GenSafe. Moreover, the proposed action correction incorporates an immediate cost constraint. This facilitates the realization of safe behaviors that require quick responses, e.g., collision avoidance in navigation problems, providing better performance for such tasks. However, there are also drawbacks to the action-level correction. Firstly, limited by the ROMDP's capability in predicting long-term costs, the ability of the proposed action correction to accurately account for the long-term effects of actions is constrained. This limitation could impact the effectiveness of GenSafe in tasks where long-term outcomes are crucial. Secondly, since GenSafe acts as an additional safety layer, it is separated from the policy update process of the utilized SRL algorithm. This may potentially lead to a less efficient policy optimization. Given these considerations, the efficacy of GenSafe is therefore influenced by the characteristics and requirements of the tasks.

B. Trade-off between Safety and Task Performance

As aforementioned, determining the trade-off between maximizing reward returns and ensuring system safety is a challenging problem in SRL. In the proposed GenSafe, this trade-off is modifiable through parameters such as the estimated cost for unobserved reduced state-action pairs Δ, the immediate cost threshold d_s, and the future cost threshold d. Setting a lower estimated cost Δ tends to classify more unexplored regions as safe. Then, together with higher thresholds for d_s and d, a more expansive exploration is allowed, thereby leading to a more efficient policy search and improved rewards. However, this increased exploratory flexibility also raises the possibility of encountering unsafe behaviors, which, in turn, results in higher cost returns. How to determine the optimal balance between safety and task performance remains an open research question. Currently, decisions often rely on experience-driven manual adjustments that are based on prior knowledge of system characteristics and task requirements. Further research is needed to develop more reliable and systematic approaches for managing this trade-off.

C. Limitations and Future work

One major limitation of the proposed GenSafe is the increased computational demand stemming from activities like constructing the ROMDP and performing action corrections. Although parallel computing techniques can be used to mitigate this issue, they inevitably prolong the training duration and increase the requirements for computational resources. One possible solution to this problem is to reformulate the constraints within the optimization (23) into differentiable forms, thereby enabling the usage of gradient-based methods or the derivation of analytical solutions. However, devising a method for accurately and reliably transforming these constraints needs more investigation. Another limitation of GenSafe, echoing a common issue among model-free SRL algorithms, is the challenge of providing absolute safety guarantees, i.e., complete avoidance of unsafe behaviors. Given its reliance on a data-driven approach, the algorithm has to encounter unsafe regions to identify and subsequently avoid them, which inevitably leads to unsafe actions during the learning phase. To achieve absolute safety guarantees, model-based SRL methods that leverage precise system models and control-theoretical concepts may offer a viable path. Nevertheless, this area remains open for in-depth research and development.

For future work, we plan to extend the current approach to policy-level modifications. This could be done by, e.g., integrating the cost predictions generated by the ROMDP into the policy update mechanism of the selected SRL algorithm. However, this necessitates improving the ROMDP's long-term prediction accuracy to ensure a reliable policy-level adjustment. Another direction involves incorporating the proposed GenSafe and ROMDP into model-based SRL algorithms. In such a scenario, the ROMDP could serve as a simplified version of the system model, providing estimates for use within model-based SRL methods. Nevertheless, this requires further research to investigate and improve the construction of ROMDP and GenSafe to align with the characteristics and application requirements of model-based SRL algorithms.

VIII. Conclusion

In this work, we present the Generalizable Safety Enhancer, referred to as GenSafe, for improving the safety performance of SRL algorithms. Leveraging data collected throughout the learning process, we first apply a model order reduction technique to transform the original high-dimensional state space into a representative two-dimensional state space. Then, based on this reduced state space, we construct a ROMDP through a comprehensive abstraction process that includes state, action, cost, transition, and policy abstractions. The resultant ROMDP serves as a low-dimensional proxy for the original cost function in CMDP. Subsequently, by reformulating the original cost constraint into ROMDP-based constraints, we propose an action correction mechanism that adjusts each applied action to enhance the probability of constraint satisfaction. Through this action-level correction strategy, the proposed GenSafe acts as an additional safety layer for SRL algorithms, providing a broad compatibility with a wide range of SRL algorithms. We evaluate the performance of GenSafe across various SRL

benchmark problems. The results illustrate the effectiveness of GenSafe in improving the safety performance of SRL algorithms, especially in early learning stages, while also maintaining the task performance at a satisfactory level. This validates the utility and applicability of GenSafe, suggesting its potential as a versatile tool for enhancing safety across diverse SRL applications and scenarios.

REFERENCES

[1] R. Liu, F. Nageotte, P. Zanne, M. de Mathelin, and B. Dresp-Langley, "Deep reinforcement learning for the control of robotic manipulation: a focussed mini-review," *Robotics*, vol. 10, no. 1, p. 22, 2021.

[2] B. R. Kiran, I. Sobh, V. Talpaert, P. Mannion, A. A. Al Sallab, S. Yogamani, and P. Pérez, "Deep reinforcement learning for autonomous driving: A survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 6, pp. 4909–4926, 2021.

[3] A. T. Azar, A. Koubaa, N. Ali Mohamed, H. A. Ibrahim, Z. F. Ibrahim, M. Kazim, A. Ammar, B. Benjdira, A. M. Khamis, I. A. Hameed *et al.*, "Drone deep reinforcement learning: A review," *Electronics*, vol. 10, no. 9, p. 999, 2021.

[4] J. Garcıa and F. Fernández, "A comprehensive survey on safe reinforcement learning," *Journal of Machine Learning Research*, vol. 16, no. 1, pp. 1437–1480, 2015.

[5] P. Ladosz, L. Weng, M. Kim, and H. Oh, "Exploration in deep reinforcement learning: A survey," *Information Fusion*, vol. 85, pp. 1–22, 2022.

[6] Y. Duan, X. Chen, R. Houthooft, J. Schulman, and P. Abbeel, "Benchmarking deep reinforcement learning for continuous control," in *International Conference on Machine Learning*. PMLR, 2016, pp. 1329–1338.

[7] Z. Zhou, O. S. Oguz, M. Leibold, and M. Buss, "A general framework to increase safety of learning algorithms for dynamical systems based on region of attraction estimation," *IEEE Transactions on Robotics*, vol. 36, no. 5, pp. 1472–1490, 2020.

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

[9] S. Gu, L. Yang, Y. Du, G. Chen, F. Walter, J. Wang, Y. Yang, and A. Knoll, "A review of safe reinforcement learning: Methods, theory and applications," *ArXiv preprint arXiv:2205.10330*, 2022.

[10] E. Altman, *Constrained Markov decision processes*. Routledge, 2021.

[11] A. Wachi and Y. Sui, "Safe reinforcement learning in constrained markov decision processes," in *International Conference on Machine Learning*. PMLR, 2020, pp. 9797–9806.

[12] A. Ray, J. Achiam, and D. Amodei, "Benchmarking safe exploration in deep reinforcement learning," *ArXiv preprint arXiv:1910.01708*, vol. 7, no. 1, p. 2, 2019.

[13] R. J. Vanderbei *et al.*, *Linear programming*. Springer, 2020.

[14] Z. Liu, H. Zhou, B. Chen, S. Zhong, M. Hebert, and D. Zhao, "Safe model-based reinforcement learning with robust cross-entropy method," *ArXiv preprint arXiv:2010.07968*, 2020.

[15] A. I. Cowen-Rivers, D. Palenicek, V. Moens, M. A. Abdullah, A. Sootla, J. Wang, and H. Bou-Ammar, "Samba: Safe model-based & active reinforcement learning," *Machine Learning*, vol. 111, no. 1, pp. 173–203, 2022.

[16] G. Thomas, Y. Luo, and T. Ma, "Safe reinforcement learning by imagining the near future," *Advances in Neural Information Processing Systems*, vol. 34, pp. 13 859–13 869, 2021.

[17] A. K. Jayant and S. Bhatnagar, "Model-based safe deep reinforcement learning via a constrained proximal policy optimization algorithm," *Advances in Neural Information Processing Systems*, vol. 35, pp. 24 432–24 445, 2022.

[18] H. Sikchi, W. Zhou, and D. Held, "Learning off-policy with online planning," in *Conference on Robot Learning*. PMLR, 2022, pp. 1622–1633.

[19] Y. J. Ma, A. Shen, O. Bastani, and J. Dinesh, "Conservative and adaptive penalty for model-based safe reinforcement learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 5, 2022, pp. 5404–5412.

[20] Y. As, I. Usmanova, S. Curi, and A. Krause, "Constrained policy optimization via bayesian world models," *ArXiv preprint arXiv:2201.09802*, 2022.

[21] M. Wen and U. Topcu, "Constrained cross-entropy method for safe reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[22] J. Achiam, D. Held, A. Tamar, and P. Abbeel, "Constrained policy optimization," in *International Conference on Machine Learning*. PMLR, 2017, pp. 22–31.

[23] C. Tessler, D. J. Mankowitz, and S. Mannor, "Reward constrained policy optimization," *ArXiv preprint arXiv:1805.11074*, 2018.

[24] M. Yu, Z. Yang, M. Kolar, and Z. Wang, "Convergent policy optimization for safe reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[25] Y. Zhang, Q. Vuong, and K. Ross, "First order constrained optimization in policy space," *Advances in Neural Information Processing Systems*, vol. 33, pp. 15 338–15 349, 2020.

[26] W. H. Schilders, H. A. Van der Vorst, and J. Rommes, *Model order reduction: theory, research aspects and applications*. Springer, 2008, vol. 13.

[27] Z. Zhou, O. S. Oguz, M. Leibold, and M. Buss, "Learning a low-dimensional representation of a safe region for safe reinforcement learning on dynamical systems," *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[28] Y.-M. Chen and M. Posa, "Optimal reduced-order modeling of bipedal locomotion," in *IEEE International Conference on Robotics and Automation*. IEEE, 2020, pp. 8753–8760.

[29] G. Dalal, K. Dvijotham, M. Vecerik, T. Hester, C. Paduraru, and Y. Tassa, "Safe exploration in continuous action spaces," *ArXiv preprint arXiv:1801.08757*, 2018.

[30] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *ArXiv preprint arXiv:1707.06347*, 2017.

[31] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International Conference on Machine Learning*. PMLR, 2015, pp. 1889–1897.

[32] D. P. Bertsekas, *Constrained optimization and Lagrange multiplier methods*. Academic Press, 2014.

[33] Q. Liang, F. Que, and E. Modiano, "Accelerated primal-dual policy optimization for safe reinforcement learning," *ArXiv preprint arXiv:1802.06480*, 2018.

[34] S. Paternain, M. Calvo-Fullana, L. F. Chamon, and A. Ribeiro, "Safe policies for reinforcement learning via primal-dual methods," *IEEE Transactions on Automatic Control*, vol. 68, no. 3, pp. 1321–1336, 2022.

[35] T.-Y. Yang, J. Rosca, K. Narasimhan, and P. J. Ramadge, "Projection-based constrained policy optimization," *ArXiv preprint arXiv:2010.03152*, 2020.

[36] A. Stooke, J. Achiam, and P. Abbeel, "Responsive safety in reinforcement learning by pid lagrangian methods," in *International Conference on Machine Learning*. PMLR, 2020, pp. 9133–9143.

[37] D. Ding, X. Wei, Z. Yang, Z. Wang, and M. Jovanovic, "Provably efficient safe exploration via primal-dual policy optimization," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2021, pp. 3304–3312.

[38] T. Xu, Y. Liang, and G. Lan, "Crpo: A new approach for safe reinforcement learning with convergence guarantee," in *International Conference on Machine Learning*. PMLR, 2021, pp. 11 480–11 491.

[39] T. J. Perkins and A. G. Barto, "Lyapunov design for safe reinforcement learning," *Journal of Machine Learning Research*, vol. 3, no. Dec, pp. 803–832, Dec. 2002.

[40] Y. Chow, O. Nachum, E. Duenez-Guzman, and M. Ghavamzadeh, "A lyapunov-based approach to safe reinforcement learning," in *Advances in Neural Information Processing Systems*, Dec. 2018, pp. 8103–8112.

[41] Y. Chow, O. Nachum, A. Faust, E. Duenez-Guzman, and M. Ghavamzadeh, "Lyapunov-based safe policy optimization for continuous control," *ArXiv preprint arXiv:1901.10031*, 2019.

[42] Y. Liu, J. Ding, and X. Liu, "Ipo: Interior-point policy optimization under constraints," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, 2020, pp. 4940–4947.

[43] M. Deisenroth and C. E. Rasmussen, "Pilco: A model-based and data-efficient approach to policy search," in *International Conference on Machine Learning*, 2011, pp. 465–472.

[44] Y. Chow and M. Ghavamzadeh, "Algorithms for cvar optimization in mdps," *Advances in Neural Information Processing Systems*, vol. 27, 2014.

[45] J. F. Fisac, A. K. Akametalu, M. N. Zeilinger, S. Kaynama, J. Gillula, and C. J. Tomlin, "A general safety framework for learning-based control in uncertain robotic systems," *IEEE Transactions on Automatic Control*, vol. 64, no. 7, pp. 2737–2752, Jul. 2019.

[46] F. Berkenkamp and A. P. Schoellig, "Safe and robust learning control with gaussian processes," in *European Control Conference*. IEEE, 2015, pp. 2496–2501.
[47] B. Thananjeyan, A. Balakrishna, S. Nair, M. Luo, K. Srinivasan, M. Hwang, J. E. Gonzalez, J. Ibarz, C. Finn, and K. Goldberg, "Recovery rl: Safe reinforcement learning with learned recovery zones," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4915–4922, 2021.
[48] F. Berkenkamp, M. Turchetta, A. Schoellig, and A. Krause, "Safe model-based reinforcement learning with stability guarantees," in *Advances in Neural Information Processing System*, Dec. 2017, pp. 908–919.
[49] F. Berkenkamp, R. Moriconi, A. P. Schoellig, and A. Krause, "Safe learning of regions of attraction for uncertain, nonlinear systems with gaussian processes," in *IEEE 55th Conference on Decision and Control*, Dec. 2016, pp. 4661–4666.
[50] M. Zanon and S. Gros, "Safe reinforcement learning using robust mpc," *IEEE Transactions on Automatic Control*, vol. 66, no. 8, pp. 3638–3652, 2020.
[51] S. Gros and M. Zanon, "Learning for mpc with stability & safety guarantees," *Automatica*, vol. 146, p. 110598, 2022.
[52] B. Lütjens, M. Everett, and J. P. How, "Safe reinforcement learning with model uncertainty estimates," in *IEEE International Conference on Robotics and Automation*. IEEE, 2019, pp. 8662–8668.
[53] B. Kouvaritakis and M. Cannon, "Model predictive control," *Switzerland: Springer International Publishing*, vol. 38, 2016.
[54] A. Aswani, H. Gonzalez, S. S. Sastry, and C. Tomlin, "Provably safe and robust learning-based model predictive control," *Automatica*, vol. 49, no. 5, pp. 1216–1226, 2013.
[55] J. F. Fisac, N. F. Lugovoy, V. Rubies-Royo, S. Ghosh, and C. J. Tomlin, "Bridging hamilton-jacobi safety analysis and reinforcement learning," in *IEEE International Conference on Robotics and Automation*, May 2019, pp. 8550–8556.
[56] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," *ArXiv preprint arXiv:1506.02438*, 2015.
[57] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne." *Journal of Machine Learning Research*, vol. 9, no. 11, 2008.
[58] D. A. Reynolds *et al.*, "Gaussian mixture models." *Encyclopedia of biometrics*, vol. 741, no. 659-663, 2009.
[59] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. The MIT Press, 2018. [Online]. Available: http://incompleteideas.net/book/the-book-2nd.html
[60] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in *IEEE International Conference on Neural Networks*, 1995, pp. 1942–1948.
[61] J. Ji, B. Zhang, J. Zhou, X. Pan, W. Huang, R. Sun, Y. Geng, Y. Zhong, J. Dai, and Y. Yang, "Safety-gymnasium: A unified safe reinforcement learning benchmark," *ArXiv preprint arXiv:2310.12567*, 2023.

APPENDIX A
COMPUTATIONS OF T-SNE

Given a set of original states $\mathcal{D}_s = \{s_1, \ldots, s_n\}$, t-SNE first computes the conditional probability $p_{j|i}$ as

$$p_{j|i} = \frac{\exp(-||s_i - s_j||^2/2\sigma_i^2)}{\sum_{k\neq i}\exp(-||s_i - s_k||^2/2\sigma_i^2)} \tag{29}$$

where $||\cdot||$ is the Euclidean distance. σ_i represents the variance of the Gaussian distribution centered on the state s_i. This probability $p_{j|i}$ can be interpreted as the probability that the state s_i picks the state s_j as its neighbor according to their probability density under Gaussian distribution.

For any given variance value σ_i, it results in a distribution P_i of the conditional probability $p_{j|i}$ across all other data points. A binary search is then conducted to find the variance value σ_i that yields a distribution P_i corresponding to a perplexity level defined by the user. The perplexity is defined as

$$\text{Perplexity}(P_i) = 2^{H(P_i)} \tag{30}$$

where $H(P_i)$ is the Shannon entropy of the distribution P_i measured as

$$H(P_i) = -\sum_j p_{j|i}\log_2 p_{j|i} \tag{31}$$

As described in [57], perplexity is often interpreted as a smooth measure of the effective number of neighbors.

To alleviate the identification problem caused by outliers, the similarity between two data points in the training dataset is defined using a joint probability p_{ij}

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n} \tag{32}$$

where $n = |\mathcal{D}_s|$. Since we are focused on pairwise similarities, we define $p_{ij} = 0$ when $i = j$.

Subsequently, we calculate the set of low-dimensional states $\mathcal{D}_l = \{l_1, \ldots, l_n\}$ to best represent the similarity p_{ij}. This is achieved through a similar joint probability q_{ij} for two low-dimensional states l_i and l_j, which is defined as

$$q_{ij} = \frac{(1 + ||l_i - l_j||^2)^{-1}}{\sum_{k\neq i}(1 + ||l_i - l_j||^2)^{-1}} \tag{33}$$

where we have $q_{ij} = 0$ if $i = j$. The similarity is measured using a heavy-tailed Student t-distribution. The values of the low-dimensional states $l_1, \ldots, l_n$ are determined by minimizing a cost function $C_{\text{t-SNE}}$ given as

$$C_{\text{t-SNE}} = \text{KL}(P||Q) = \sum_i\sum_j p_{ij}\log\frac{p_{ij}}{q_{ij}} \tag{34}$$

where $\text{KL}(\cdot)$ is the Kullback-Leibler divergence. P and Q are the joint probability distributions in the high-dimensional and low-dimensional state spaces, respectively. The cost function $C_{\text{t-SNE}}$ evaluates the effectiveness of the identified low-dimensional states in replicating the similarities between different training data points.

The minimization is performed using a gradient descent method, where the gradient is computed as

$$\frac{\partial C}{\partial l_i} = 4\sum_j(p_{ij} - q_{ij})(l_i - l_j)(1 + ||l_i - l_j||^2)^{-1} \tag{35}$$

The initial values for the low-dimensional states $l_1, \ldots, l_n$, denoted as $\mathcal{Q}^{(0)}$, are obtained by randomly sampling points from an isotropic Gaussian distribution. To speed up the optimization and avoid poor local minimum, the update of the solution is performed with a momentum term

$$\mathcal{Q}^{(t)} = \mathcal{Q}^{(t-1)} + \eta\frac{\partial C}{\partial \mathcal{Q}^{(t-1)}} + m^{(t)}\left(\mathcal{Q}^{(t-1)} - \mathcal{Q}^{(t-2)}\right) \tag{36}$$

where $\mathcal{Q}^{(t)}$ is the solution at iteration t, η is the learning rate, and $m^{(t)}$ is the momentum at iteration t. More implementation details of t-SNE are presented in [57].