Training and Transferring Safe Policies in Reinforcement Learning

Qisong Yang*
Delft University of Technology
Delft, The Netherlands
q.yang@tudelft.nl

Thiago D. Simão*
Radboud University
Nijmegen, The Netherlands
thiago.simao@ru.nl

Nils Jansen
Radboud University
Nijmegen, The Netherlands
nils.jansen@ru.nl

Simon H. Tindemans
Delft University of Technology
Delft, The Netherlands
s.h.tindemans@tudelft.nl

Matthijs T. J. Spaan
Delft University of Technology
Delft, The Netherlands
m.t.j.spaan@tudelft.nl

ABSTRACT

Safety is critical to broadening the application of reinforcement learning (RL). Often, RL agents are trained in a controlled environment, such as a laboratory, before being deployed in the real world. However, the target reward might be unknown prior to deployment. Reward-free RL addresses this problem by training an agent without the reward to adapt quickly once the reward is revealed. We consider the *constrained* reward-free setting, where an agent (the guide) learns to explore safely without the reward signal. This agent is trained in a controlled environment, which allows unsafe interactions and still provides the safety signal. After the target task is revealed, safety violations are not allowed anymore. Thus, the guide is leveraged to compose a safe sampling policy. Drawing from transfer learning, we also regularize a target policy (the student) towards the guide while the student is unreliable and gradually eliminate the influence from the guide as training progresses. The empirical analysis shows that this method can achieve safe transfer learning and helps the student solve the target task faster.

1 INTRODUCTION

Despite the numerous achievements of reinforcement learning [RL; 31, 41], safety still prevents the wide adoption of RL [8]. The lack of knowledge about the environment forces standard agents to rely on trial and error strategies. However, this approach is incompatible with safety-critical scenarios [11]. For instance, while operating a power network, an agent trying random actions could cause a blackout, which is strictly unacceptable [28, 39].

Developments in safe RL have allowed learning policies that respect safety constraints expressed by a constrained Markov decision process [CMDP; 3]. For instance, SAC-Lagrangian [15] combines the Soft Actor-Critic [SAC; 17, 18] algorithm with Lagrangian methods to learn a safe policy in an off-policy way. This method solves high-dimensional problems achieving a sample complexity lower than its on-policy counterparts. Unfortunately, it only finds a safe policy at the end of the training process and may be unsafe while learning.

Some knowledge about the safety dynamics can ensure safety during learning. One can precompute unsafe behavior and mask unsafe actions using a so-called shield [2, 22], or start from an initially safe baseline policy and gradually improve its performance while remaining safe [1, 45, 50]. However, this approach may require many interactions with the environment before it finds a reasonable policy [53]. Moreover, reusing a pre-trained policy can be harmful since the agent faces a new trajectory distribution as the policy changes [20]. Therefore, we focus on *how to efficiently solve a task without violating the safety constraints.*

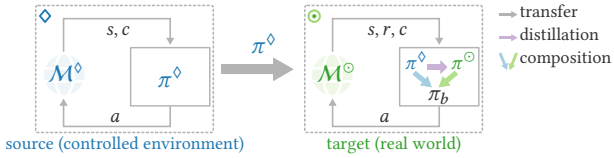


Figure 1: Transferring the Safe Guide (SaGui) policy $\pi^{\diamond}$.

This paper addresses this question drawing inspiration from transfer learning [44], where some prior knowledge from a source task speeds up learning on a target task. We propose to transfer a policy, the safe guide (SaGui, Figure 1), from the source task ($\diamond$) to the target task ($\odot$). This approach has three major steps: *i*) training the SaGui policy and *transferring* it to the target task; *ii*) *distilling* the guide's policy into a student policy, and *iii*) *composing* a behavior policy that balances safe exploration (using the guide) and exploitation (using the student).

To train the SaGui policy, we consider the reward-free constrained RL framework [30], where the agent only observes the cost function, and it does not have access to the reward function. This task-agnostic approach allows us to train a guide even when we do not know the reward of the target task, and this guide can be useful for different reward functions. Inspired by advances in robotics where an agent is trained under strict supervision, we assume the source task is a simulated/controlled environment [35, 48]. Therefore, safety is not required while training the SaGui policy. Once the target's reward is revealed, the SaGui policy safely collects the initial trajectories, and we start training the student's policy. To ensure the new policy quickly learns how to act safely, we also employ a policy distillation method, encouraging the student to imitate the SaGui policy.

The empirical analysis shows that this method almost completely prevents the violation of the safety constraints on the target task.

*Equal contribution.

Proc. of the Adaptive and Learning Agents Workshop (ALA 2022), Cruz, Hayes, da Silva, Santos (eds.), May 9-10, 2022, Online, https://ala2022.github.io/. 2022.

It also shows the exploration benefits of SaGui, which allows the agent to solve the target task faster than agents learning with a naive guide.

2 BACKGROUND

In this section, we formalize the safe reinforcement learning problem and describe how it typically is solved.

2.1 Constrained Markov Decision Processes

We consider tasks formulated using a constrained Markov decision process[CMDP; 3, 6]. A CMDP is defined as a tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, r, c, d, \gamma \rangle$: a state space $\mathcal{S}$, an action space $\mathcal{A}$, a probabilistic transition function $\mathcal{P} : \mathcal{S} \times \mathcal{A} \to Dist(\mathcal{S})$, a reward function $r : \mathcal{S} \times \mathcal{A} \to [r_{min}, r_{max}]$, a cost function $c : \mathcal{S} \times \mathcal{A} \to [c_{min}, c_{max}]$, a safety threshold $d \in \mathbb{R}^+$, and a discount factor $\gamma \in [0, 1)$. We also consider an initial state distribution $\iota : \mathcal{S} \to [0, 1]$. In a *constrained RL* problem an agent interacts with a CMDP, without knowledge about the transition, reward, and cost functions, generating a trajectory $\tau = \langle (s_0, a_0, r_0, c_0, s'_0), (s_1, a_1, r_1, c_1, s'_1), \cdots \rangle$. A trajectory starts from $s_0 \sim \iota(\cdot)$, then, at each timestep t the agent is in a state $s_t \in \mathcal{S}$, and takes an action $a_t \in \mathcal{A}$. Then, it gets a reward $r_t = r(s_t, a_t)$, a cost $c_t = c(s_t, a_t)$, and steps into a successor state $s'_t \sim \mathcal{P}(\cdot \mid s_t, a_t)$. This process repeats starting from $s_{t+1} = s'_t$, until some terminal condition is met and a new trajectory starts. The goal of the agent is to learn a policy π that maximizes the expected discounted return such that the expected discounted cost-return remains below the given threshold:

$$\max_{\pi} \mathop{\mathbb{E}}_{\tau \sim \rho_\pi} \left[\sum_{t=0}^{\infty} \gamma^t r_t \right] \text{s.t.} \mathop{\mathbb{E}}_{\tau \sim \rho_\pi} \left[\sum_{t=0}^{\infty} \gamma^t c_t \right] \leq d, \tag{1}$$

where ρ_π indicates the trajectory distribution induced by $s_0 \sim \iota(\cdot)$, $a_t \sim \pi(\cdot|s_t)$, and $s_{t+1} \sim \mathcal{P}(\cdot|s_t, a_t)$. We define the discounted *return* starting from $(s, a) \in \mathcal{S} \times \mathcal{A}$ and following π as

$$Q^r_\pi(s, a) = \mathop{\mathbb{E}}_{\tau \sim \rho_\pi} \left[\sum_{t=0}^{\infty} \gamma^t r_t \middle| s_0 = s, a_0 = a \right],$$

and, similarly, the discounted *cost-return* $Q^c_\pi(s, a)$.

From the safe RL perspective, if a policy has an expected cost-return lower than the safety-threshold d, then this policy is considered safe. Therefore, the objective of a safe RL agent is to find a policy among the safe policies that has the highest expected return.

2.2 Maximum Entropy Reinforcement Learning

A common strategy to improve the exploration and robustness of RL agents is to favor policies that induce diverse behaviors [9, 56]. We can incorporate it in the safe RL main objective by augmenting the problem with a term that aims to maximize the policy entropy [16]:

$$\begin{aligned} &\max_{\pi} \mathop{\mathbb{E}}_{\tau \sim \rho_\pi} \left[\sum_{t=0}^{\infty} \gamma^t \left(r_t + \alpha \mathcal{H}(\pi(\cdot \mid s_t)) \right) \right] \\ &\text{s.t.} \mathop{\mathbb{E}}_{\tau \sim \rho_\pi} \left[\sum_{t=0}^{\infty} \gamma^t c_t \right] \leq d, \end{aligned} \tag{2}$$

where $\mathcal{H}(\cdot)$ is the entropy of a distribution over a random variable and α is the entropy weight. In general, this objective encourages the agent to use maximally stochastic policies.

An alternative to maximizing the entropy is to encourage the policy to have at least a minimum entropy $\overline{\mathcal{H}}$ [18, 49]. This is achieved adding the following constraint to (1):

$$\mathop{\mathbb{E}}_{\tau \sim \rho_\pi} \left[-\log(\pi(a_t \mid s_t)) \right] \geq \overline{\mathcal{H}} : \quad \forall t, \tag{3}$$

where $\overline{\mathcal{H}}$ is the given entropy threshold to ensure a minimum degree of randomness. This approach allows the policy to converge to a more deterministic behavior than (2). Besides, it only requires the system's designer to define $\overline{\mathcal{H}}$ and it lets the RL agent automatically find a trade-off between the policy's entropy and rewards. Therefore, α becomes an internal parameter of the RL algorithm.

Inspired by Savas et al. [34], we will use (2) to train the guide in the source task with an auxiliary reward, so it achieves maximum entropy while remaining safe. Then, we train a student on a target task, optimizing (1) plus the constraint in (3), so it only needs to have a minimum entropy. Before we formalize these tasks, we describe the traditional way to solve these problems.

2.3 SAC-Lagrangian

In this section, we describe the SAC-Lagrangian [SAC-λ; 15] method designed for the maximum entropy RL with constraints (2) and (3). In general, SAC-λ is a SAC-based method that has two critics and uses adaptive safety weights to manage a trade-off between reward and safety. The reward critic estimates the expected return, possibly with an entropy to promote exploration, while the safety critic estimates the cost-return to encourage safety.

In SAC-λ, the constrained optimization problem is solved by Lagrangian methods [5], where an entropy weight α and a safety weight β (Lagrange-multipliers) are introduced to the constrained optimization:

$$\max_{\pi} \min_{\alpha \geq 0} \min_{\beta \geq 0} f(\pi) - \alpha e(\pi) - \beta g(\pi), \tag{4}$$

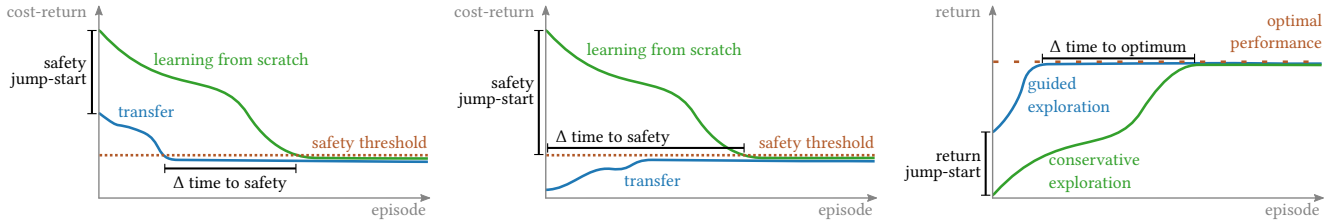
where

$$\begin{aligned} f(\pi) &= \mathop{\mathbb{E}}_{s_0 \sim \iota(\cdot), a_0 \sim \pi(\cdot|s_0)} \left[Q^r_\pi(s_0, a_0) \right] \\ e(\pi) &= \mathop{\mathbb{E}}_{s_t \sim \rho_\pi} \left[\log(\pi(\cdot \mid s_t)) + \overline{\mathcal{H}} \right], \text{and} \\ g(\pi) &= \mathop{\mathbb{E}}_{s_0 \sim \iota(\cdot), a_0 \sim \pi(\cdot|s_0)} \left[Q^c_\pi(s_0, a_0) - d \right]. \end{aligned}$$

In (4), the max-min optimization problem can be solved by gradient ascent on π, and descent on α and β.

Ha et al. [15] developed SAC-λ for local constraints, which means that the safety cost is constrained at each timestep. However, it can be easily generalized to constrain the expected cost-return[1]. In summary, the main components of the SAC-λ are: *i*) the policy π parameterized by θ; *ii*) the safety and reward critics Q^c and Q^r (parameterized by μ and ψ, respectively); and *iii*) the entropy and safety weights α and β (parameterized by χ_α and χ_β, such that $\alpha = \text{softplus}(\chi_\alpha)$ and $\beta = \text{softplus}(\chi_\beta)$, where $\text{softplus}(x) = \log(\exp(x) + 1)$. We provide a detailed description of how to learn each component including the losses in Appendix A. Throughout the paper we represent learning rates with η, replay buffers with $\mathcal{D}$ and losses with J. Notice that we only update α when a desirable $\overline{\mathcal{H}}$ is given. When we are considering (2) we do not have a $\overline{\mathcal{H}}$, so α is fixed.

[1] A similar approach can be found at https://github.com/openai/safety-starter-agents.



(a) Unsafe transfer. (b) Fully safe transfer. (c) Return transfer.

Figure 2: Transfer metrics for safe reinforcement learning.

3 SAFE AND EFFICIENT EXPLORATION

Naturally, to train reinforcement learning agents without violating the safety constraints, some prior knowledge is required [40]. Often, a safe initial policy is used to collect the initial trajectories [1, 45, 50]. However, these approaches often ignore how this policy is computed or what makes it effective. Therefore, we will explicitly consider how to obtain an initial safe policy and what makes it effective for learning under safety constraints.

We consider two agents, a *guide* and a *student*, with different responsibilities. The guide learns a policy to safely explore the environment. The guide interacts with a simulator or a controlled environment, therefore, it may violate the safety constraints while learning. The student learns a safe policy for a specific task, however it is not allowed to violate the safety constraints while learning since it interacts directly with the real-world. Therefore, the student may rely on the guide's policy to learn safely.

We will formalize the interaction between the guide and the student using the transfer learning (TL) framework. In general, transfer learning allows RL agents to utilize external expertise from other *source* tasks to benefit the learning process of the *target* task [44, 55]. The source tasks $\{\mathcal{M}^\lozenge\}$ provide some prior knowledge $\mathcal{K}^\lozenge$ that is accessible by an agent learning the target task $\mathcal{M}^\odot$, such that by leveraging the information from $\mathcal{K}^\lozenge$, the agent learns the target task faster. So, TL aims to learn an optimal policy $\pi^\odot$ for the target domain, by leveraging exterior information $\mathcal{K}^\lozenge$ from $\mathcal{M}^\lozenge$ as well as interior information $\mathcal{K}^\odot$ from $\mathcal{M}^\odot$. Notice that regular RL is a special case with $\mathcal{K}^\lozenge = \emptyset$.

In this section we formalize the problem of interest, describe the setting where it is applied, and specify how to evaluate the solutions for this problem.

3.1 Problem Statement

> We aim to maximize the *safety jump-start* (preventing safety violation) and reduce the *time to optimum* (improving exploration) when transferring the guide $\pi^\lozenge$ from a source task $\mathcal{M}^\lozenge$ to a target task $\mathcal{M}^\odot$ where we train the student $\pi^\odot$.

This setting has some particularities: *i*) the guide and the student are trained separately; *ii*) the guide is only trained once and can support the training of different students; and *iii*) the guide only has access to information regarding safety and has no knowledge about the student's task.

3.2 Problem Setting

For our transfer setting, we consider a single source task that only provides the safety signals, which we use to train the guide. Without the reward signal, the guide aims to explore the world safely and efficiently. We are interested in using the guide's safe exploration capabilities to train the student on the target task without violating the safety constraints.

Given a source task $\mathcal{M}^\lozenge = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \emptyset, c, d, \gamma \rangle$ we compute the guide's policy $\pi^\lozenge$ in the absence of a reward signal. This provides prior knowledge $\mathcal{K}^\lozenge = \{\pi^\lozenge\}$ that is used to train a student $\pi^\odot$ on the target task $\mathcal{M}^\odot = \langle \mathcal{S}^\odot, \mathcal{A}, \mathcal{P}^\odot, r^\odot, c, d, \gamma \rangle$.

We assume the source task $\mathcal{M}^\lozenge$ is the result of applying a Q^c_π-*irrelevant state abstraction* $\Xi : \mathcal{S}^\odot \to \mathcal{S}$ in the target task $\mathcal{M}^\odot$ [26]. Formally, $\Xi(s_1) = \Xi(s_2) \Rightarrow Q^c_\pi(s_1, a) = Q^c_\pi(s_2, a), \forall s_1, s_2 \in \mathcal{S}^\odot, \forall a \in \mathcal{A}, \forall \pi$, that is, if Ξ merges two states then their Q^c_π values are the same for all actions and all policies. Then, the source and target tasks have the same action space, but different state spaces.

This assumption ensures the source task provides enough safety information, such that any policy that is safe on the source task is also safe when deployed on the target task [37]. Note however, that the behavior required to accomplish the target task may not be defined on the source task, since the reward in the target task $r^\odot$ is not related to the state space of the source task $\mathcal{S}^\lozenge$. Consider, for instance, an agent with access to its position and the position of a threat. In each target task, the agent might have a different goal position, which is not defined on the source task. In this example, a safe policy may be conditioned only on the positions of the agent and the threat, but to achieve the target the agent must take the goal position into account.

3.3 Transfer Metrics

An under-explored aspect of safe transfer RL is the *transfer metrics in safety*. Figure 2(a) presents a schematic of transfer metrics related to safety [inspired by 44]: *safety jump-start* indicates how much closer to the safety threshold the expected cost-return of an agent trained using the source knowledge is compared to the expected cost-return of an agent learning from scratch in the first episodes; and Δ *time to safety* is the difference in the number of interactions required to reach the safety threshold.

Notice that a trained agent might start with an expected cost-return lower than the safety threshold, for instance, when the safety threshold of the source task is lower than the safety threshold of the target task (Figure 2(b)). In this case, *safety jump-start* would

be the difference between the safety threshold and the cost-return of an agent learning from scratch. Similarly, the Δ *time to safety* would be the number of interactions an agent learning from scratch needs to satisfy the safety constraints.

Once we develop agents that learn without violating the safety constraints, we can also consider the usual metrics of transfer learning with respect to the reward [44]. For instance, Figure 2(c) shows the initial improvement in terms of performance which we call *return jump-start*, and the time necessary to reach an optimum performance, which we call the Δ *time to optimum*. We expect that the guide will help the student explore the environment, which can reduce the time to optimum. We stress that, in safety-critical tasks, these metrics have lower precedence than the safety metrics, so they can only compare agents that have no safety constraint violations [33].

4 GUIDED SAFE EXPLORATION

In this section, we consider how to train the *safe guide* (SaGui) policy. Then, we describe how the student learns to imitate the SaGui policy after the task is revealed, while learning to complete the target task. Finally, we investigate how to prevent safety violations while the student has not yet learned how to act safely.

4.1 Training the Safe Guide (SaGui)

Since the source task does not provide information regarding the reward of the target task, we adopt a reward-free exploration approach to train the guide. To efficiently explore the world, we first consider to maximize the policy entropy under the safety constraints. Then, we can solve the problem defined in Equation 2 with $r(s,a) = 0 : \forall s \in \mathcal{S}, a \in \mathcal{A}$ to get a guide MaxEnt. However, although MaxEnt tends to have diverse behaviors, that does not imply efficient exploration of the environment. Especially for continuous state and action spaces, it is possible that a policy provides limited exploration even if it has high entropy.

To enhance the exploration of the guide, we adopt an auxiliary reward that motivates the agent to visit novel states. To measure the difference between states, we first define the metric space $(\mathcal{S}, \delta)$, where $\mathcal{S}$ is the state space, and $\delta : \mathcal{S} \times \mathcal{S} \to [0, \infty)$ is a distance function, that is

$$\begin{aligned} &\delta(s,s') = 0 \Leftrightarrow s = s', \\ &\quad \delta(s,s') = \delta(s',s), \text{and} \\ &\delta(s',s'') \le \delta(s,s') + \delta(s,s''), \qquad \forall s, s', s'' \in \mathcal{S}. \end{aligned}$$

In practice, δ must be defined according to the observations of the underlying environment. Next, we define the auxiliary rewards as the expected distance from the current state and the successor state:

$$r'_t = \mathop{\mathbb{E}}_{s'_t \sim \mathcal{P}(\cdot|s_t,a_t)} \left[\delta(s_t, s'_t)\right], \qquad \forall s_t, a_t \in \mathcal{S} \times \mathcal{A}. \tag{5}$$

So, we train the *guide* agent by solving the constraint optimization problem (2) based on the auxiliary reward r'. Then, the SAC-λ algorithm can be directly employed to solve the problem (2), as Algorithm 1.

We could also consider more sophisticated reward-free exploration strategies such as maximizing the the entropy of the state occupancy distribution [19, 36, 43]. However, we leave it as future work and focus on how to use the guide to improve how the student learns.

Algorithm 1 Maximum exploration RL for *safe guide*

input Task $\mathcal{M}^\Diamond$
input Hyperparameters α, d
initialize: $\theta, \psi, \mu, \beta, \mathcal{D} \leftarrow \emptyset$
for each iteration **do**
  for each environment step **do**
    $a_t \sim \pi^\Diamond(\cdot \mid s_t)$
    $s'_t \sim \mathcal{P}(\cdot \mid s_t, a_t)$
    $r_t \leftarrow \delta(s_t, s'_t)$ ▷ *Auxiliary task* (5)
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{(s_t, a_t, r_t, c_t, s'_t)\}$ ▷ *Replay buffer*
  end for
  for each gradient step **do**
    Sample experience from replay buffer $\mathcal{D}$
    $\psi \leftarrow \psi - \eta_R \hat{\nabla}_\psi J_R(\psi)$ ▷ *Reward critic*
    $\mu \leftarrow \mu - \eta_C \hat{\nabla}_\mu J_C(\mu)$ ▷ *Safety critic*
    $\theta \leftarrow \theta - \eta_\pi \hat{\nabla}_\theta J_\pi(\theta)$ ▷ *Actor*
    $\chi_\beta \leftarrow \chi_\beta - \eta_\beta \hat{\nabla}_{\chi_\beta} J_s(\chi_\beta)$ ▷ *Safety weight*
  end for
end for
output Optimized parameters θ for $\pi^\Diamond$

4.2 Policy Distillation

When the agent is trained for a certain task, it is difficult to generalize when faced with a new task [20]. Similarly, it is not trivial to adjust the guide's policy that was trained to explore the environment to perform the target task. Therefore, we train a new policy, referred as the student, dedicated to the target task.

We can leverage the *guide* agent to make the student quickly learn how to act safely. Through the mapping function Ξ, the transferred policy can be used by most constrained RL algorithms to regularize the student policy $\pi^\odot$ towards the guide policy $\pi^\Diamond$ using KL divergences, as shown in Figure 3. So, with $\pi^\Diamond$ fixed, we have an augmented reward function

$$r'_t = r_t^\odot + \omega r_t^{\mathrm{KL}} + \alpha r_t^{\mathcal{H}},$$

where $r_t^{\mathrm{KL}} = \log \frac{\pi^\Diamond(a_t|\Xi(s_t))}{\pi^\odot(a_t|s_t)}$ and $r_t^{\mathcal{H}} = -\log \pi^\odot(a_t \mid s_t)$. The weights ω and α indicate the strengths of the KL and entropy regularization (respectively). Then, setting $r_t^\Diamond = \log \pi^\Diamond(a_t \mid \Xi(s_t))$, we can define the new objective to be maximized, i.e.,

$$\max_{\pi^\odot} \mathop{\mathbb{E}}_{\tau \sim \rho_{\pi^\odot}} \sum_{t=0}^{\infty} \gamma^t \left[r_t^\odot + \omega r_t^\Diamond + (\alpha + \omega) r_t^{\mathcal{H}} \right]. \tag{6}$$

To find an appropriate ω, our goal is to follow the guide more for safer exploration if the student's policy is unsafe, but eliminate the influence from the guide and focus more on the performance if the student's policy is safe. Therefore, we propose to set $\omega = \beta$ to determine the strength of the KL regularization since the adaptive safety weight β reflects the safety of the current policy.

In summary, we have an entropy regularized expected return with redefined (regularized) reward

$$r''_t = r_t^\odot + \beta r_t^\Diamond.$$

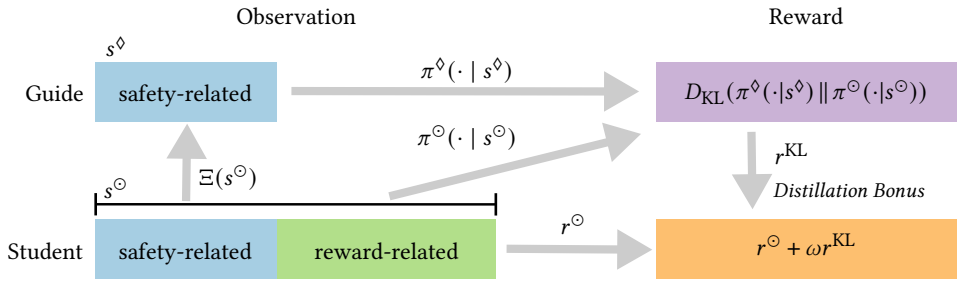


Figure 3: Overview of the policy distillation. The target policy $\pi^\odot$ may have an input space different from the guide policy $\pi^\Diamond$.

This augmented reward encourages the student to yield actions that are more likely to be generated by the guide. Then, SAC-λ can be directly used to solve (6) with the additional entropy constraint (Algorithm 2, lines 13-20).

Notice that the second use of β is fixed in the redefined reward to avoid any superfluous influence on the variable β, which is only updated based on the safety of $\pi^\odot$. Finally, as the student becomes safe, the influence of β vanishes, so our method still solves the original constrained optimization with $r'' = r^\odot$.

4.3 Composite Sampling

To enhance safety and improve the student during training (Algorithm 2, lines 3-12), we leverage a *composite sampling* strategy, which means our behavior policy (π_b) is a mixture of the guide's policy ($\pi^\Diamond$) and the student's policy ($\pi^\odot$). So, at each environment step, $a_t \sim \pi_b(\cdot \mid s_t), s_t \in \mathcal{S}^\odot$ where

$$\pi_b(\cdot \mid s_t) = \begin{cases} \pi^\Diamond(\cdot \mid \Xi(s_t)), & \text{if } b = \Diamond, \\ \pi^\odot(\cdot \mid s_t), & \text{otherwise.} \end{cases} \tag{7}$$

We investigate two strategies to define b (Appendix B provides more details):

Linear-decay. This strategy linearly decreases the probability of using $\pi^\Diamond$ with a constant decay rate after each iteration of the algorithm, conversely increasing the probability of using $\pi^\odot$. We have two modes with *linear-decay*: *step-wise*, where in each time step we may change π_b; and *trajectory-wise*, where π_b only changes at the start of a trajectory. The mode is decided before executing an episode, and smoothly switches from the complete *step-wise* to the complete *trajectory-wise* over the training process.

Control-switch. To generate more on-policy samples, this strategy starts each episode sampling from the student ($b = \odot$). When a first positive cost is encountered ($c_{t-1} > 0$), it switches to the guide's policy ($b = \Diamond$) and follows it until the end of the trajectory. Therefore, the guide policy serves as a *rescue policy* to improve safety during sampling. In addition, we use two replay buffers $\mathcal{D}^\Diamond$ and $\mathcal{D}^\odot$ to save the guide and student samples separately, so as to better control the proportion of the different types of samples when updating the target policy.

With the *composite sampling* strategy, the function approximation may diverge, because $\pi^\odot$ and π_b are too different, especially when we collect most data following $\pi^\Diamond$. We refer to this phenomenon as the *deadly triad* [42]. To eliminate its negative effect, we

Algorithm 2 Guided Safe Exploration

input Task $\mathcal{M}^\odot$, the guide's policy $\pi^\Diamond$
input Hyperparameters $\overline{\mathcal{H}}$, and d
1: **initialize** θ^*, ψ^*, μ^*, α^*, β^*, and $\mathcal{D} \leftarrow \emptyset$
2: **for** each iteration **do**
3: **for** each environment step **do**
4: $b \leftarrow \Diamond \vee \odot$ ▷ *Choose behavior policy (Appendix B)*
5: $a_t \sim \pi_b(\cdot \mid s_t)$ ▷ *Composite sampling (7)*
6: $\mathcal{I}_t \leftarrow \mathcal{I}(s_t, a_t)$ ▷ *IS ratio (8)*
7: $r_t \leftarrow r^\odot(s_t, a_t)$
8: $r_t^\Diamond \leftarrow \log \pi^\Diamond(a_t \mid \Xi(s_t))$
9: $s_{t+1} \sim \mathcal{P}^\odot(\cdot \mid s_t, a_t)$
10: $c_t \leftarrow c(\Xi(s_t), a_t)$
11: $\mathcal{D} \leftarrow \mathcal{D} \cup \{(s_t, a_t, r_t, r_t^\Diamond, c_t, \mathcal{I}_t, s_{t+1})\}$
12: **end for**
13: **for** each gradient step **do**
14: Sample experience from $\mathcal{D}$
15: $\psi^* \leftarrow \psi^* - \eta_R \hat{\nabla}_{\psi^*} \mathcal{I} J_R(\psi^*)$ ▷ *Reward critic*
16: $\mu^* \leftarrow \mu^* - \eta_C \hat{\nabla}_{\mu^*} \mathcal{I} J_C(\mu^*)$ ▷ *Safety critic*
17: $\theta^* \leftarrow \theta^* - \eta_\pi \hat{\nabla}_{\theta^*} \mathcal{I} J_\pi(\theta^*)$ ▷ *Actor*
18: $\chi_\alpha^* \leftarrow \chi_\alpha^* - \eta_\alpha \hat{\nabla}_{\chi_\alpha^*} J_e(\chi_\alpha^*)$ ▷ *Exploration weight*
19: $\chi_\beta^* \leftarrow \chi_\beta^* - \eta_\beta \hat{\nabla}_{\chi_\beta^*} J_s(\chi_\beta^*)$ ▷ *Safety weight*
20: **end for**
21: **end for**
output Optimized parameters θ^* for $\pi^\odot$

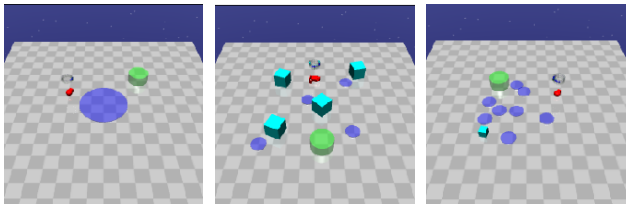
endow each sample with an *importance sampling* (IS) ratio:

$$\mathcal{I}(s, a) = \min\left(\max\left(\frac{\pi^\odot(a \mid s)}{\pi_b(a \mid s)}, \mathcal{I}_l\right), \mathcal{I}_u\right). \tag{8}$$

The clipping hyper-parameters $\mathcal{I}_u$ and $\mathcal{I}_l$ are introduced to reduce the variance of the off-policy TD target. Notice that if π_b is using the guide $\pi^\odot$ then $\mathcal{I}(s, a) = 1$. Here, in addition to use the IS ratio $\mathcal{I}$ for learning values (the *critics*), we also use it in the policy update, as shown in lines 15-19 of Algorithm 2.

5 EMPIRICAL ANALYSIS

We evaluate how well our method transfers from the reward-free setting using the SafetyGym engine [33], where a random-initialized robot navigates in a 2D map to reach target positions while trying to avoid dangerous areas and obstacles (Figure 4). These tasks are particularly complex due to the observation space; instead of



(a) Static (b) Semi-Dynamic (c) Dynamic

Figure 4: Navigation tasks with different complexity levels.

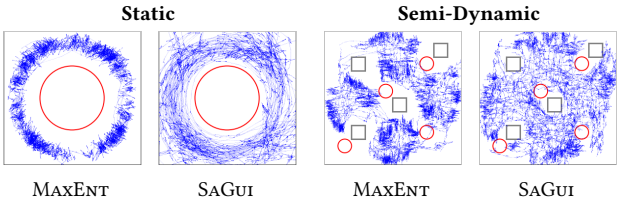


Figure 5: Exploration analysis. Trajectories collected by the guide agent, with and without the distance bonus, after training.

observing its location, the agent observes the other objects with a lidar sensor. We considered three environments with different complexity levels:

Static. A static environment with a point robot, and a hazard (Figure 4(a)). The locations of the hazard and goal are the same in all episodes.

Semi-Dynamic. A semi-dynamic environment with a car robot, four hazards, and four vases (Figure 4(b)). The locations of the hazards and vases are the same in all episodes. The location of the goal is random-initialized in each episode.

Dynamic. A dynamic environment with a point robot, eight hazards, and a vase (Figure 4(c)). The locations of the goal, vase, and hazards are random-initialized in each episode.

The *guide* agent is trained without the goals, and its auxiliary reward is the magnitude of displacement at each time step. Since our focus is on the target task and the guide is trained in a controlled environment, we do not consider the guide's training in the evaluation. In the target tasks, we use the original reward signal from Safety Gym, i.e., the distance towards the goal plus a constant for finishing the task [33]. In all environments: $c = 1$, if an unsafe interaction happens, and $c = 0$, otherwise. All experiments are performed over 10 runs with different random seeds and the plots show the mean and standard deviation of all runs.

To evaluate the performance of the algorithms during training, we use the following metrics: safety of the behavior policy (Cost-Return π_b), performance of the behavior policy (Return π_b), safety of the target policy (Cost-Return $\pi^{\odot}$), and performance of the target policy (Return $\pi^{\odot}$). To check the convergence of the target policy, we have a test process with 100 episodes after each epoch (in parallel to the training) to evaluate Return $\pi^{\odot}$ and Cost-Return $\pi^{\odot}$. Appendix C presents the evaluation of $\pi^{\odot}$ and Appendix D reports the hyperparameters used.

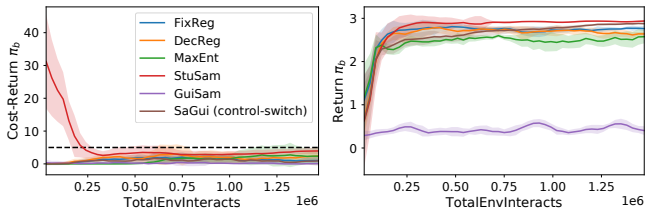


Figure 6: Ablation study in *Static* showing the safety (left) and performance (right) of the behavior policy. The black dashed line indicates the safety threshold.

5.1 Ablation Study

We investigate each component of the proposed SaGui algorithm individually to answer the following questions:

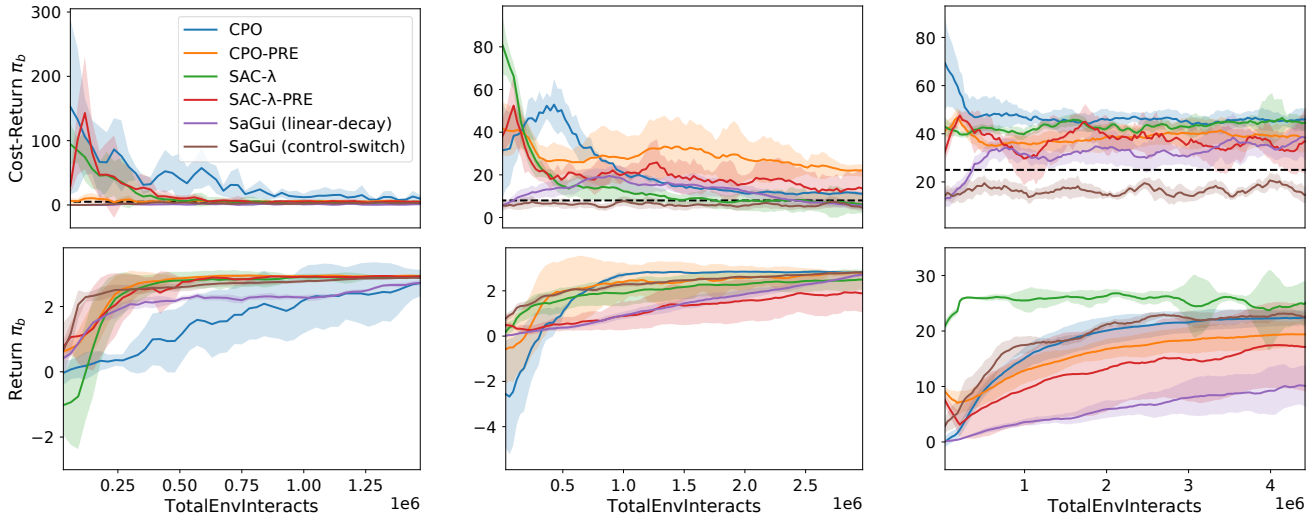
i) Does the *distance bonus* enlarge the exploration range?
ii) Does a better *guide* agent result in a better student in the target task?
iii) How does the *adaptive strength* of the KL regularization affect the performance?
iv) How does the *composite sampling* benefit the safe transfer learning?

i) Distance bonus leads to more diverse trajectories. We performed an ablation of our approach where no distance bonus is added while training the *guide* agent, called MaxEnt. We refer to the agent with the distance bonus as SaGui. This teases apart the role the designed auxiliary task plays in the exploration. In Figure 5, we can see that SaGui can explore larger areas in *Static* and *Semi-Dynamic*, which have the same layout in each episode. We notice that MaxEnt is safe, but the explored space is limited. That is also the case in *Dynamic*, as shown in the attached videos.

ii) An effective guide can speed up the student's training. We compare how these guides (MaxEnt and SaGui) affect the learning in the target task. In Figures 6 and 8 (Appendix C), we notice that both methods can collect samples safely but the agent using the distance bonus finds highly performing policies within fewer interactions with the environment.

iii) Safety-adaptive regularization improves the student's convergence rate. To combine the original reward with the bonus to follow the guide (ω), we have the following choices: fix the weights of the bonus and make it to be a hyperparameter to tune (FixReg); apply a decay rate to linearly decrease the weights during training (DecReg); and, adapt the weights of the bonus based on the safety performance (SaGui). In Figure 6 we observe that this weight does not affect the safety of the agent, but both FixReg and DecReg cause the student to converge slower in terms of performance (Figure 8 in Appendix C).

iv) Composite sampling enhances safety and final performance. We modify the composite sampling approach, sampling only from the guide (GuiSam) or the student (StuSam) instead. From the results in Figure 6, we can see that GuiSam can ensure safety, but the student does not learn a safe optimal policy (Figure 8 in Appendix C). Compared to our method, StuSam performs similarly converging to a safe target policy, but fails to satisfy the constraint



(a) Static (b) Semi-Dynamic (c) Dynamic

Figure 7: Evaluation of π_b for CPO, CPO-PRE, SAC-λ, SAC-λ-PRE, SAGUI (linear-decay), and SAGUI (control-switch) over 10 seeds. The black dashed lines indicate the safety thresholds.

at the early stage of training. So, *composite sampling* is necessary to avoid the dangerous actions from a naive policy and to ensure the target task is solved.

5.2 Comparison with Baselines

Finally, we compare our algorithms SAGUI (control-switch) and SAGUI (linear-decay) with four baselines. SAC-λ [15] shows the performance when starting to learn from scratch, representing an off-policy algorithm. CPO [1] is a well-known on-policy constrained RL algorithm that maximizes the reward in a small neighbourhood that enforces the constraints. CPO-PRE and SAC-λ-PRE are used to show how CPO and SAC-λ will perform after pre-trained in a task that replaces the original reward function of the target task by the distance bonus. So, we also encourage exploration in the task for pre-training, which shares the same observation space with the target task.

In Figures 7 and 9 (Appendix C), we observe that SAGUI (control-switch) is the only method that has a safe behavior during the full training process. However, SAGUI (linear-decay), which lacks samples from $\pi^{\odot}$ at the early stage of training, does not achieve similar performance. From Figures 7(b) and 7(c), it is obvious that *linear-decay* fails to compose π_b in a safe way. SAC-λ and CPO can learn safe policies in the relatively simpler environments (*Static* and *Semi-Dynamic*) but they violate the safety constraints at the beginning of training, which is expected. In *Dynamic*, SAC-λ and CPO fail to attain safe performance. However, with benefits from the *guide*, SAGUI (control-switch), on the basis of SAC-λ, attains a better balance between safety and performance. With pre-training, a safe initialization cannot benefit CPO-PRE and SAC-λ-PRE in safety, and may have negative effects. We infer that it is difficult to generalize a task when faced with a new reward signal [20]. Especially for SAC-λ-PRE with an initialized Q^r, the difficulty to adapt is evident.

6 RELATED WORK AND FUTURE DIRECTIONS

As we discussed in the introduction, safe RL has multiple facets [11], considering alternative optimization criteria [7, 49], and different types of prior knowledge to ensure safe exploration [1, 2, 22, 40, 51]. We will discuss alternatives to train the guide and how to adapt to new tasks using a pre-trained policy.

Yang et al. [51] proposed an on-policy framework to guide learning by different baseline policies often dedicated to the target task. But our framework can leverage the safe off-policy samples from the guide that is unaware of the target task.

Multiple algorithms have been proposed for generalising policies from reward-free RL for better performance in target tasks [13, 38, 54]. However, only Miryoosefi and Jin [30], Savas et al. [34] have considered the reward-free RL with constraints, focusing on tabular and linear settings, while we consider general function approximation algorithms.

While we considered a relatively simple strategy to achieve rich exploration, our framework allows any progress in reward-free RL to be translated into training the *guide* agent. For instance, we could adopt works with the entropy of the state density [19, 21, 24, 32, 36, 43, 47, 54]. Another option to improve exploration is to find a set of diverse policies to the same problem [12, 23, 52]. Our framework could easily combine multiple guides.

Work in transfer learning has leveraged meta-RL [10] for safe adaptation [14, 25, 27]. Our work is also related to curriculum learning [4, 29, 46]. We first train an agent to be safe and later solve a target task. However, our approach focuses on safe exploration and is able to transfer to tasks with different reward functions, so the guide's training is ignored. For curriculum learning, it would be interesting to consider when to stop training the guide and start training the student.

7 CONCLUSION

This work handles multiple challenges of reinforcement learning with safety constraints. It shows how we can use a safe policy (the guide) during data collection and gradually switch to a policy that is dedicated to the target task (the student). It tackles the off-policy issue that arises from collecting data with a policy different from the target policy. It shows how the student can make the best use of the guide's policy using an incentive to imitate the guide, which makes the student learn faster how to behave safely. It demonstrates that simply initializing an agent with a safe policy may not be as effective as learning a new policy dedicated to the target task through policy distillation. Finally, it proposes a method that can collect diverse trajectories, which reduces the sample complexity of the student on the target task. In summary, the framework proposed is a safe and sample-efficient way of training the agent on a target task.

ACKNOWLEDGMENTS

This research is funded by the Netherlands Organisation for Scientific Research (NWO), as part of the Energy System Integration: planning, operations, and societal embedding program and the grants NWO OCENW.KLEIN.187: "Provably Correct Policies for Uncertain Partially Observable Markov Decision Processes" and NWA.1160.18.238: "PrimaVera". Qisong Yang is supported by Xidian University.

REFERENCES

[1] Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. 2017. Constrained Policy Optimization. In *Proceedings of the 34th International Conference on Machine Learning*. PMLR, 22–31.

[2] Mohammed Alshiekh, Roderick Bloem, Rüdiger Ehlers, Bettina Könighofer, Scott Niekum, and Ufuk Topcu. 2018. Safe Reinforcement Learning via Shielding. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*. AAAI Press, 2669–2678.

[3] Eitan Altman. 1999. *Constrained Markov decision processes*. Vol. 7. CRC Press.

[4] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum Learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM, 41–48.

[5] Dimitri P Bertsekas. 1982. *Constrained Optimization and Lagrange Multiplier Methods*. Vol. 1. Academic press.

[6] Vivek S Borkar. 2005. An actor-critic algorithm for constrained Markov decision processes. *Systems & control letters* 54, 3 (2005), 207–213.

[7] Yinlam Chow, Mohammad Ghavamzadeh, Lucas Janson, and Marco Pavone. 2017. Risk-constrained reinforcement learning with percentile risk criteria. *The Journal of Machine Learning Research* 18, 1 (2017), 6070–6120.

[8] Gabriel Dulac-Arnold, Nir Levine, Daniel J. Mankowitz, Jerry Li, Cosmin Paduraru, Sven Gowal, and Todd Hester. 2021. Challenges of real-world reinforcement learning: definitions, benchmarks and analysis. *Machine Learning* 110, 9 (2021), 2419–2468.

[9] Benjamin Eysenbach and Sergey Levine. 2021. Maximum Entropy RL (Provably) Solves Some Robust RL Problems. *arXiv preprint arXiv:2103.06257*.

[10] Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. In *Proceedings of the 34th International Conference on Machine Learning*. PMLR, 1126–1135.

[11] Javier García and Fernando Fernández. 2015. A Comprehensive Survey on Safe Reinforcement Learning. *The Journal of Machine Learning Research* 16, 1 (2015), 1437–1480.

[12] Mahsa Ghasemi, Evan Scope Crafts, Bo Zhao, and Ufuk Topcu. 2021. Multiple Plans are Better than One: Diverse Stochastic Planning. In *Proceedings of the International Conference on Automated Planning and Scheduling*. AAAI Press, 140–148.

[13] Michael Gimelfarb, André Barreto, Scott Sanner, and Chi-Guhn Lee. 2021. Risk-Aware Transfer in Reinforcement Learning using Successor Features. *arXiv preprint arXiv:2105.14127*.

[14] Djordje Grbic and Sebastian Risi. 2020. Safe Reinforcement Learning through Meta-learned Instincts. In *ALIFE 2020 : The 2020 Conference on Artificial Life*. MIT Press, 183–291.

[15] Sehoon Ha, Peng Xu, Zhenyu Tan, Sergey Levine, and Jie Tan. 2020. Learning to Walk in the Real World with Minimal Human Effort. In *Proceedings of the 2020 Conference on Robot Learning*. PMLR, 1110–1120.

[16] Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. 2017. Reinforcement Learning with Deep Energy-Based Policies. In *Proceedings of the 34th International Conference on Machine Learning*. PMLR, 1352–1361.

[17] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. 2018. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. In *Proceedings of the 35th International Conference on Machine Learning*. PMLR, 1861–1870.

[18] Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, and Sergey Levine. 2018. Soft Actor-Critic Algorithms and Applications. *arXiv preprint arXiv:1812.05905*.

[19] Elad Hazan, Sham Kakade, Karan Singh, and Abby Van Soest. 2019. Provably Efficient Maximum Entropy Exploration. In *Proceedings of the 36th International Conference on Machine Learning*. PMLR, 2681–2691.

[20] Maximilian Igl, Gregory Farquhar, Jelena Luketina, Wendelin Boehmer, and Shimon Whiteson. 2021. Transient Non-stationarity and Generalisation in Deep Reinforcement Learning. In *9th International Conference on Learning Representations*. OpenReview.net, 1–9.

[21] Riashat Islam, Zafarali Ahmed, and Doina Precup. 2019. Marginalized State Distribution Entropy Regularization in Policy Optimization. *arXiv preprint arXiv:1912.05128*.

[22] Nils Jansen, Bettina Könighofer, Sebastian Junges, Alex Serban, and Roderick Bloem. 2020. Safe Reinforcement Learning Using Probabilistic Shields (Invited Paper). In *31st International Conference on Concurrency Theory*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 1–16.

[23] Saurabh Kumar, Aviral Kumar, Sergey Levine, and Chelsea Finn. 2020. One Solution is Not All You Need: Few-Shot Extrapolation via Structured MaxEnt RL. In *Advances in Neural Information Processing Systems 33*. Curran Associates, Inc., 8198–8210.

[24] Lisa Lee, Benjamin Eysenbach, Emilio Parisotto, Eric Xing, Sergey Levine, and Ruslan Salakhutdinov. 2019. Efficient Exploration via State Marginal Matching. *arXiv preprint arXiv:1906.05274*.

[25] Thomas Lew, Apoorva Sharma, James Harrison, and Marco Pavone. 2020. Safe Model-Based Meta-Reinforcement Learning: A Sequential Exploration-Exploitation Framework. *arXiv preprint arXiv:2008.11700*.

[26] Lihong Li, Thomas J Walsh, and Michael L Littman. 2006. Towards a Unified Theory of State Abstraction for MDPs. In *International Symposium on Artificial Intelligence and Mathematics*. ISAIM, 1–10.

[27] Michael Luo, Ashwin Balakrishna, Brijen Thananjeyan, Suraj Nair, Julian Ibarz, Jie Tan, Chelsea Finn, Ion Stoica, and Ken Goldberg. 2021. MESA: Offline Meta-RL for Safe Adaptation and Fault Tolerance. *arXiv preprint arXiv:2112.03575*.

[28] Antoine Marot, Benjamin Donnot, Camilo Romero, Balthazar Donon, Marvin Lerousseau, Luca Veyrin-Forrer, and Isabelle Guyon. 2020. Learning to run a power network challenge for training topology controllers. *Electric Power Systems Research* 189 (2020), 106635.

[29] Luca Marzari, Davide Corsi, Enrico Marchesini, and Alessandro Farinelli. 2021. Curriculum Learning for Safe Mapless Navigation. *arXiv preprint arXiv:2112.12490*.

[30] Sobhan Miryoosefi and Chi Jin. 2021. A Simple Reward-free Approach to Constrained Reinforcement Learning. *arXiv preprint arXiv:2107.05216*.

[31] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (2015), 529–533.

[32] Zengyi Qin, Yuxiao Chen, and Chuchu Fan. 2021. Density Constrained Reinforcement Learning. In *Proceedings of the 38th International Conference on Machine Learning*. PMLR, 8682–8692.

[33] Alex Ray, Joshua Achiam, and Dario Amodei. 2019. Benchmarking Safe Exploration in Deep Reinforcement Learning. https://cdn.openai.com/safexp-short.pdf.

[34] Yagiz Savas, Melkior Ornik, Murat Cubuktepe, and Ufuk Topcu. 2018. Entropy Maximization for Constrained Markov Decision Processes. In *56th Annual Allerton Conference on Communication, Control, and Computing*. IEEE, 911–918.

[35] Erik Schuitema, Martijn Wisse, Thijs Ramakers, and Pieter Jonker. 2010. The design of LEO: A 2D bipedal walking robot for online autonomous Reinforcement Learning. In *International Conference on Intelligent Robots and Systems*. IEEE, 3238–3243.

[36] Younggyo Seo, Lili Chen, Jinwoo Shin, Honglak Lee, Pieter Abbeel, and Kimin Lee. 2021. State Entropy Maximization with Random Encoders for Efficient Exploration. Presented at the ICLR 2021 Workshop on Self-Supervision for Reinforcement Learning.

[37] Thiago D. Simão, Nils Jansen, and Matthijs T. J. Spaan. 2021. AlwaysSafe: Reinforcement Learning Without Safety Constraint Violations During Training. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*. IFAAMAS, 1226–1235.

[38] Krishnan Srinivasan, Benjamin Eysenbach, Sehoon Ha, Jie Tan, and Chelsea Finn. 2020. Learning to be Safe: Deep RL with a Safety Critic. *arXiv preprint arXiv:2010.14603*.

[39] Medha Subramanian, Jan Viebahn, Simon H. Tindemans, Benjamin Donnot, and Antoine Marot. 2021. Exploring grid topology reconfiguration using a simple deep reinforcement learning approach. In *2021 IEEE Madrid PowerTech*. IEEE, 1–6.

[40] Yanan Sui, Alkis Gotovos, Joel Burdick, and Andreas Krause. 2015. Safe Exploration for Optimization with Gaussian Processes. In *Proceedings of the 32nd International Conference on Machine Learning*. PMLR, 997–1005.

[41] Richard S. Sutton and Andrew G. Barto. 2018. *Reinforcement Learning: An Introduction*. Vol. 2. MIT press.

[42] Richard S. Sutton, A. Rupam Mahmood, and Martha White. 2016. An Emphatic Approach to the Problem of Off-policy Temporal-Difference Learning. *The Journal of Machine Learning Research* 17, 1 (2016), 2603–2631.

[43] Oleg Svidchenko and Aleksei Shpilman. 2021. Maximum Entropy Model-based Reinforcement Learning. *arXiv preprint arXiv:2112.01195*.

[44] Matthew E. Taylor and Peter Stone. 2009. Transfer Learning for Reinforcement Learning Domains: A Survey. *The Journal of Machine Learning Research* 10, 56 (2009), 1633–1685.

[45] Chen Tessler, Daniel J. Mankowitz, and Shie Mannor. 2019. Reward Constrained Policy Optimization. In *7th International Conference on Learning Representations*. OpenReview.net, 1–9.

[46] Matteo Turchetta, Andrey Kolobov, Shital Shah, Andreas Krause, and Alekh Agarwal. 2020. Safe Reinforcement Learning via Curriculum Induction. In *Advances in Neural Information Processing Systems 33*. Curran Associates, Inc., 12151–12162.

[47] Giulia Vezzani, Abhishek Gupta, Lorenzo Natale, and Pieter Abbeel. 2019. Learning latent state representation for speeding up exploration. *arXiv preprint arXiv:1905.12621*.

[48] Zhaoming Xie, Patrick Clary, Jeremy Dao, Pedro Morais, Jonathan W. Hurst, and Michiel van de Panne. 2019. Learning Locomotion Skills for Cassie: Iterative Design and Sim-to-Real. In *3rd Annual Conference on Robot Learning*. PMLR, 317–329.

[49] Qisong Yang, Thiago D. Simão, Simon H. Tindemans, and Matthijs T. J. Spaan. 2021. WCSAC: Worst-Case Soft Actor Critic for Safety-Constrained Reinforcement Learning. In *Thirty-Fifth AAAI Conference on Artificial Intelligence*. AAAI Press, 10639–10646.

[50] Tsung-Yen Yang, Justinian Rosca, Karthik Narasimhan, and Peter J. Ramadge. 2020. Projection-Based Constrained Policy Optimization. In *8th International Conference on Learning Representations*. OpenReview.net, 1–10.

[51] Tsung-Yen Yang, Justinian Rosca, Karthik Narasimhan, and Peter J Ramadge. 2021. Accelerating Safe Reinforcement Learning with Constraint-mismatched Baseline Policies. In *International Conference on Machine Learning*. PMLR, 11795–11807.

[52] Tom Zahavy, Brendan O'Donoghue, André Barreto, Sebastian Flennerhag, Volodymyr Mnih, and Satinder Singh. 2021. Discovering Diverse Nearly Optimal Policies with Successor Features. ICML 2021 Workshop on Unsupervised Reinforcement Learning.

[53] Moritz A. Zanger, Karam Daaboul, and J. Marius Zöllner. 2021. Safe Continuous Control with Constrained Model-Based Policy Optimization. In *IEEE/RSJ International Conference on Intelligent Robots and Systems IROS*. IEEE, 3512–3519.

[54] Jesse Zhang, Brian Cheung, Chelsea Finn, Sergey Levine, and Dinesh Jayaraman. 2020. Cautious Adaptation For Reinforcement Learning in Safety-Critical Settings. In *Proceedings of the 37th International Conference on Machine Learning*. PMLR, 11055–11065.

[55] Zhuangdi Zhu, Kaixiang Lin, and Jiayu Zhou. 2020. Transfer Learning in Deep Reinforcement Learning: A Survey. *arXiv preprint arXiv:arXiv:2009.07888*.

[56] Brian D Ziebart. 2010. *Modeling Purposeful Adaptive Behavior with the Principle of Maximum Causal Entropy*. Carnegie Mellon University.

A SAC-LAGRANGIAN

In this section, we present how we learn the parameters in SAC-λ.

Similar to the formulation used by Haarnoja et al. [18], we can get the actor loss:

$$J_\pi(\theta) = - \underset{\substack{s_t \sim \mathcal{D} \\ a_t \sim \pi_\theta(\cdot|s_t)}}{\mathbb{E}} \left[Q^r_{\pi_\theta}(s_t, a_t) - \alpha \log \pi_\theta(a_t \mid s_t) - \beta Q^c_{\pi_\theta}(s_t, a_t) \right], \tag{9}$$

where $\mathcal{D}$ is the replay buffer and θ is the parameters of the policy π.

The safety and reward critics (including a bonus for the policy entropy) are, respectively, trained to minimize

$$J_C(\mu) = \underset{(s_t, a_t) \sim \mathcal{D}}{\mathbb{E}} \left[\frac{1}{2} \left(Q^c_\mu(s_t, a_t) - (c_t + \gamma Q^c_\mu(s_{t+1}, a_{t+1})) \right)^2 \right] \tag{10}$$

and

$$J_R(\psi) = \underset{(s_t, a_t) \sim \mathcal{D}}{\mathbb{E}} \left[\frac{1}{2} (Q^r_\psi(s_t, a_t) - (r_t + \gamma (Q^r_\psi(s_{t+1}, a_{t+1}) - \alpha \log(\pi_\theta(a_{t+1} \mid s_{t+1})))))^2 \right], \tag{11}$$

where $a_{t+1} \sim \pi(\cdot \mid s_{t+1})$, Q^c and Q^r are parameterized by μ and ψ, respectively.

Finally, let χ_α and χ_β be the parameters learned for the exploration and safety weight such that $\alpha = \text{softplus}(\chi_\alpha)$ and $\beta = \text{softplus}(\chi_\beta)$, where

$$\text{softplus}(x) = \log(\exp(x) + 1).$$

We can learn α and β by minimizing the loss functions:

$$J_e(\chi_\alpha) = \underset{\substack{s_t \sim \mathcal{D} \\ a_t \sim \pi_\theta(\cdot|s_t)}}{\mathbb{E}} \left[-\alpha (\log(\pi_\theta(a_t \mid s_t)) + \overline{\mathcal{H}}) \right], \tag{12}$$

and

$$J_s(\chi_\beta) = \underset{\substack{s_t \sim \mathcal{D} \\ a_t \sim \pi_\theta(\cdot|s_t)}}{\mathbb{E}} \left[\beta (d - Q^c_{\pi_\theta}(s_t, a_t)) \right]. \tag{13}$$

So the corresponding weight will be adjusted if the constraints are violated, that is, if we estimate that the current policy is unsafe or if it does not have enough entropy.

B TWO STRATEGIES IN COMPOSITE SAMPLING

B.1 Linear-decay

We show the *linear-decay* strategy to compose π_b in Algorithm 3. In this case, we have two modes: *step-wise* and *trajectory-wise*. We linearly decrease the probability to execute the *step-wise* and use the *guide* with a constant decay rate after each iteration of the algorithm, conversely increasing the probability of executing the *trajectory-wise* and using the student policy. So, we initialize the probabilities $P_\pi = 1$ to determine π_b, and $P_{wise} = 1$ to determine the mode at the beginning (line 1). We linearly decrease them with a constant decay rate υ (lines 12 and 20), determined by the training length. At the beginning of each episode, we sample $\kappa_{wise} \sim U(0, 1)$, so if $\kappa_{wise} < P_{wise}$, we will execute *step-wise*, or we are in *trajectory-wise* (lines 3-11). Under *step-wise*, at each time step, we sample from the *guide* $\pi^\Diamond$ with probability P_π, and sample from the student $\pi^\odot$ with probability $1 - P_\pi$ (lines 14-18). Under *trajectory-wise*, we only make a decision once at the beginning of the trajectory (line 10).

B.2 Control-switch

We show the *control-switch* strategy to compose π_b in Algorithm 4. In this case, the guide policy serves as a *rescue policy* to ensure the safety during sampling. To balance between the safe exploration and the sample efficiency (the samples from the target policy is relatively more valuable), the student policy keeps sampling, i.e., $\pi_b = \pi^\odot$ at the start of a trajectory (line 3); after we meet the first $c_{t-1} > 0$, we have $\pi_b = \pi^\Diamond$ until the end of the trajectory (lines 13-16). In addition, we leverage two replay buffers $\mathcal{D}^\Diamond$ and $\mathcal{D}^\odot$ to save the guide and student samples separately (lines 8-12), so as to control the probability $P_{\mathcal{D}^\odot}$ to use the more on-policy samples in $\mathcal{D}^\odot$. Thus, we have the probability $P_{\mathcal{D}^\Diamond} = 1 - P_{\mathcal{D}^\odot}$ to sample from $\mathcal{D}^\Diamond$.

Algorithm 3 Composite sampling (linear-decay)

input $\pi^\Diamond$, $\pi^\odot$, υ
1: **initialize** $P_\pi \leftarrow 1$, $P_{wise} \leftarrow 1$
2: **for** each iteration **do**
3: Sample $\kappa_{wise} \sim U(0, 1)$
4: **if** $\kappa_{wise} < P_{wise}$ **then**
5: *step-wise* $\leftarrow$ *true*
6: **else**
7: *step-wise* $\leftarrow$ *false*
8: $P_b(\Diamond) = P_\pi$
9: $P_b(\odot) = 1 - P_\pi$
10: $b \sim P_b$
11: **end if**
12: $P_{wise} = P_{wise} - \upsilon$
13: **for** each environment step **do**
14: **if** *step-wise* **then**
15: $P_b(\Diamond) = P_\pi$
16: $P_b(\odot) = 1 - P_\pi$
17: $b \sim P_b$
18: **end if**
19: **end for**
20: $P_\pi = P_\pi - \upsilon$
21: **end for**
output π_b

Algorithm 4 Composite sampling (control-switch)

input $\pi^\Diamond$, $\pi^\odot$
1: **initialize** $\mathcal{D}^\Diamond \leftarrow \emptyset$, $\mathcal{D}^\odot \leftarrow \emptyset$
2: **for** each iteration **do**
3: $b \leftarrow \odot$
4: *control-switch*$(t) \leftarrow false$
5: **for** each environment step **do**
6: $a_t \sim \pi_b(\cdot \mid s_t)$
7: Generate $(s_t, a_t, r_t, r_t^\Diamond, c_t, \mathcal{I}_t, s_{t+1})$
8: **if** $b = \Diamond$ **then**
9: $\mathcal{D}^\Diamond \leftarrow \mathcal{D}^\Diamond \cup \{(s_t, a_t, r_t, r_t^\Diamond, c_t, \mathcal{I}_t, s_{t+1})\}$
10: **else**
11: $\mathcal{D}^\odot \leftarrow \mathcal{D}^\odot \cup \{(s_t, a_t, r_t, r_t^\Diamond, c_t, \mathcal{I}_t, s_{t+1})\}$
12: **end if**
13: **if** $\neg$ *control-switch*$(t) \wedge c_t > 0$ **then**
14: $b \leftarrow \Diamond$
15: *control-switch*$(t) \leftarrow true$
16: **end if**
17: **end for**
18: **end for**
output π_b

C EVALUATION OF THE TARGET POLICY

Ablation study. In Figure 6, we only show the safety and performance of the behavior policy π_b in ablation study. In this section, we evaluate the safety and performance of the target policy $\pi^{\odot}$ in Figure 8. About *Guide effect*, we can observe that MaxEnt converge quite slowly in safety, and does not achieve a high return compared to SaGui. So, the quality of the guide also has a quite obvious influence on the target policy. As to *regularization*, with the adaptive safety weight, SaGui converge faster to a safe optimal target-policy compared to FixReg and DecReg. In terms of *composite sampling*, if it is ablated to be GuiSam, the corresponding target policy will diverge, and fail to get a safe policy that can finish the task well. If only the student samples (StuSam) are used, the target policy will have competitive performance in both return and cost-return compared to SaGui, but it cannot ensure the safety during training, as shown in Figure 6.

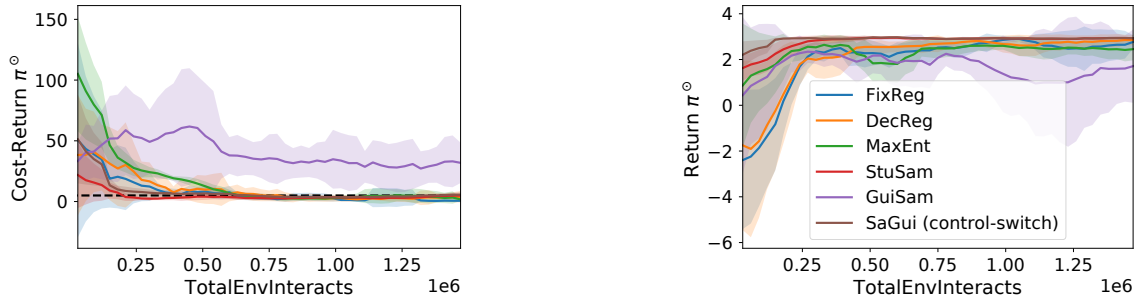
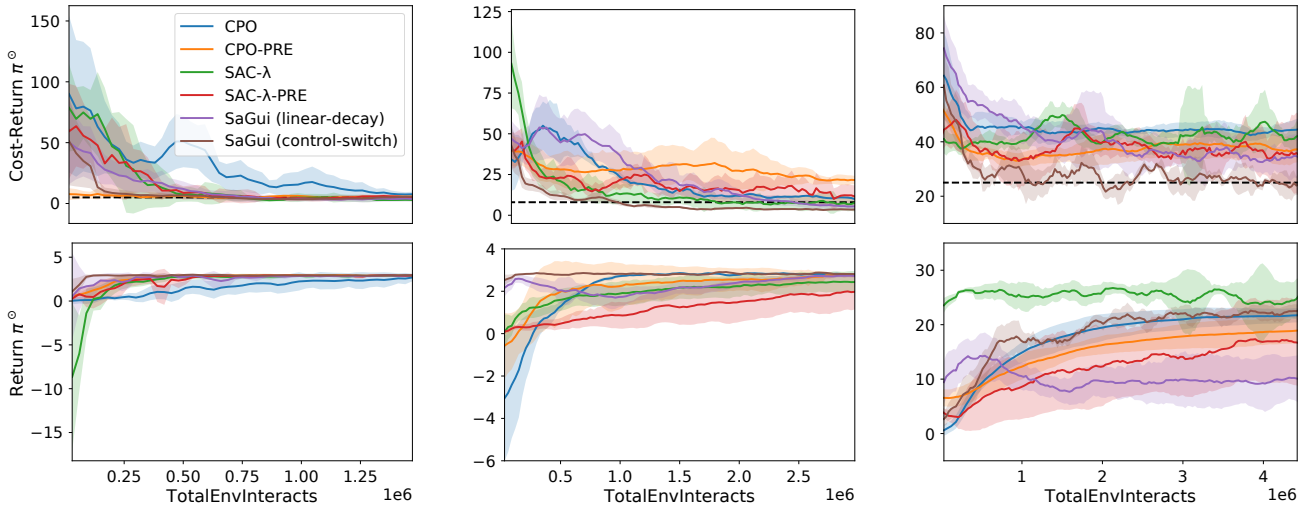


Figure 8: Ablation study in *Static* showing the safety (left) and performance (right) of the target policy.

Comparison with baselines. In Figure 7, we evaluate the behavior policy π_b for all algorithms: CPO, SAC-λ, CPO-pre, SAC-λ-pre, and SaGui. So, in Figure 9, we show how their resulted target policy will perform during training. In all these algorithms, SaGui (control-switch) is the only one that can find a safe optimal target-policy in all environments. However, SaGui (linear-decay) cannot achieve similar performance, especially in *Semi-dynamic* and *Dynamic*. We infer that SaGui (linear-decay) lack samples from the target policy, especially at the early stage of training. As to the pre-training baselines, CPO-pre and SAC-λ-pre do not attain obvious improvement compared to CPO and SAC-λ that are trained from scratch. Instead, pre-training may have some negative impacts on getting a good target policy. The only exception is that CPO-pre is largely improved in the relatively simple environment *Static*.



(a) Static (b) Semi-Dynamic (c) Dynamic

Figure 9: Evaluation of $\pi^{\odot}$ for CPO, CPO-pre, SAC-λ, SAC-λ-pre, SaGui (linear-decay), and SaGui (control-switch) over ten seeds. The solid lines are the average of all runs, and the shaded area is the standard deviation. The black dashed lines indicate the safety thresholds.

D HYPERPARAMETERS

We list the hyperparameters used in SaGui, which are summarized in Table 1. As to the baselines, we use the default hyperparameters in https://github.com/openai/safety-starter-agents. All runs in the experiment use separate feedforward Multilayer Perceptron (MLP) actor and critic networks. The size of the neural network (all actors and critics of the algorithms) depend on the complexity of the tasks. We use a replay buffer of size 10^6 for each off-policy algorithm to store the experience. The discount factor is set to be $\gamma = 0.99$, the target smoothing coefficient is set to be 0.005 to update the target networks, and the learning rate to 0.001. The clipping intervale hyper-parameters $[\mathcal{I}_l, \mathcal{I}_u]$ is set to $[0.1, 2.0]$, while the sampling probabilities $P_{\mathcal{D}^\diamond}$ and $P_{\mathcal{D}^\odot}$ are set to 0.25 and 0.75, respectively. The maximum episode length is 1000 steps in all experiments. We set the safety constraint d based on the problem. The rest of the hyperparameters are explained in the Empirical Analysis part of the paper.

Parameter	Static	Semi-Dynamic	Dynamic	Note
Size of networks	$(32, 32)$	$(64, 64)$	$(256, 256)$	
Size of replay buffer	10^6	10^6	10^6	$\lvert\mathcal{D}\rvert$
Batch size	32	64	256	
Number of epochs	50	100	150	
Safety constraint	5	8	25	d

Table 1: Summary of hyperparameters in SaGui.