Supplementary Material

Zeyu Feng[1], Bowen Zhang[1], Jianxin Bi[1], Harold Soh[1]

I. Proof of theoretical results

A. Proof of Proposition 1

Before the formal proof, we first introduce the following revised lemmas of GPI [1].

Lemma I.1. (Generalized Policy Improvement.) *Let π_1^*, π_2^*, ..., π_N^* be N optimal decision policies of N tasks $\{M_i \in \mathcal{M}_\phi\}_{i=1}^N$, respectively, and let $\tilde{Q}^{\pi_1}$, $\tilde{Q}^{\pi_2}$, ..., $\tilde{Q}^{\pi_N}$ be approximations of their respective action-value functions such that*

$$\left|Q^{\pi_i^*}(s,a) - \tilde{Q}^{\pi_i}(s,a)\right| \leq \epsilon \text{ for all } s \in \mathcal{S}, a \in \mathcal{A} \text{ and } i \in [N]. \tag{1}$$

Define

$$\pi(s) \in \operatorname*{argmax}_a \max_i \tilde{Q}^{\pi_i}(s,a). \tag{2}$$

Then,

$$Q^{\pi}(s,a) \geq \max_i Q^{\pi_i^*}(s,a) - \frac{2}{1-\gamma}\epsilon. \tag{3}$$

Lemma I.2. *Let $\delta_{ij} = \max_{s,a} |r_i(s,a,s') - r_j(s,a,s')|$. Then,*

$$Q_i^{\pi_i^*}(s,a) - Q_i^{\pi_j^*}(s,a) \leq \frac{2\delta_{ij}}{1-\gamma}. \tag{4}$$

Now we are ready to prove Proposition 1.

Proof: Let $Q_{j,\lambda}^{\pi_i}(s,a) = Q_{j,r}^{\pi_i}(s,a) + \lambda\left(Q_{j,c}^{\pi_i}(s,a) - \tau\right)$ be the balanced action-value function of a parameter λ and a policy π_i on task $M_j^c \in \mathcal{M}_\phi^c$. As previously mentioned, it equals to the Q function of policy π_i with a reward function in the form of $r_j^\lambda(s,a,s') = r_j(s,a,s') + \lambda(c_j(s,a,s') - (1-\gamma)\tau)$. Then we can have

$$Q_{j,\lambda_j^*}^{\pi_j^*}(s,a) - Q_{j,\tilde{\lambda}_j}^{\pi}(s,a) \tag{5}$$

$$\leq Q_{j,\lambda_j^*}^{\pi_j^*}(s,a) - Q_{j,\tilde{\lambda}_j}^{\pi_i^*}(s,a) + \frac{2\epsilon}{1-\gamma}\left(1+\tilde{\lambda}_j\right) \tag{6}$$

$$= Q_{j,\lambda_j^*}^{\pi_j^*}(s,a) - Q_{j,\lambda_j^*}^{\pi_i^*}(s,a) - \tilde{\lambda}_j\left(Q_{j,c}^{\pi_i^*}(s,a) - \tau\right) + \lambda_j^*\left(Q_{j,c}^{\pi_i^*}(s,a) - \tau\right) + \frac{2\epsilon}{1-\gamma}\left(1+\tilde{\lambda}_j\right) \tag{7}$$

$$\leq \frac{2}{1-\gamma}\max_{s,a}\left|r_j^{\lambda_j^*}(s,a,s') - r_i^{\lambda_i^*}(s,a,s')\right| + \left|\lambda_j^* - \tilde{\lambda}_j\right|\frac{1}{1-\gamma} + \frac{2\epsilon}{1-\gamma}\left(1+\tilde{\lambda}_j\right) \tag{8}$$

$$\leq \frac{2}{1-\gamma}\Big(\max_{s,a}\left|\phi(s,a,s')^\intercal(w_{r,j} - w_{r,i}) + \phi(s,a,s')^\intercal\left(\lambda_j^* w_{c,j} - \lambda_i^* w_{c,i}\right)\right| + \tau(1-\gamma)\left|\lambda_j^* - \lambda_i^*\right|\Big) + \left|\lambda_j^* - \tilde{\lambda}_j\right|\frac{1}{1-\gamma} + \frac{2\epsilon}{1-\gamma}\left(1+\tilde{\lambda}_j\right) \tag{9}$$

$$\leq \frac{2}{1-\gamma}\Big(\phi_{\max}\|w_{r,j} - w_{r,i}\| + \phi_{\max}\|\lambda_j^* w_{c,j} - \lambda_i^* w_{c,i}\| + \left|\lambda_j^* - \tilde{\lambda}_j\right| + \epsilon\left(1+\tilde{\lambda}_j\right)\Big) + 2\tau\left|\lambda_j^* - \lambda_i^*\right|, \tag{10}$$

for any $i \in [N]$ and $\tilde{\lambda}_j > 0$, where Eq. (6) is due to Lemma I.1 and Eq. (8) is due to applying Lemma I.2, since there exists an optimal policy π_i^* maximizing $Q_{i,\lambda_i^*}^{\pi_i}$ following the strong duality Lemma 1.

□

B. Proof of Proposition 2

We first restate a convergence lemma in sub-gradient optimization [2].

Lemma I.3. *Consider the optimization problem*

$$\max_{x \in X} \quad f(x), \tag{11}$$

where $X \subset \mathbb{R}^n$ is a convex set and $f(\cdot) : \mathbb{R}^n \to \mathbb{R}$ is a concave function. Let X^* denote the set of optimal solutions. Suppose that the update $x^{(t+1)} = P_{\mathcal{X}}\left(x^{(t)} + \eta^{(t)}\gamma^{(t)}\right)$ is applied, where $\gamma^{(t)} \in \partial f\left(x^{(t)}\right)$ is a subgradient. If $\eta^{(t)} \to 0$, $\sum_{t=1}^\infty \eta^{(t)} = \infty$, $\sum_{t=1}^\infty \left[\eta^{(t)}\right]^2 \leq \infty$ and the sequence $\left\{\gamma^{(t)}\right\}$ is bounded, then $x^{(t)} \to x^* \in X^*$.

Then the proof of Prop. 2 is given below.

Proof: Recall that the optimization problem for dual estimation is

$$\tilde{\lambda}_j^\alpha \in \operatorname*{argmin}_{\lambda_j \geq 0} \max_{\pi_\alpha \in \Pi_c} V_{r,j}^{\pi_\alpha}(s) + \lambda_j\left(V_{c,j}^{\pi_\alpha}(s) - \tau\right). \tag{12}$$

By assumption 2, the primal problem is feasible and let $V_{r,j}^{\pi_\alpha^*}(s)$ denote the optimal value. Given the dual function according to the Eq. (4) in the main paper

$$d(\lambda_j) = \max_{\pi_\alpha \in \Pi_c} L(\pi_\alpha, \lambda_j) = \max_{\pi_\alpha \in \Pi_c} V_{r,j}^{\pi_\alpha}(s) + \lambda_j\left(V_{c,j}^{\pi_\alpha}(s) - \tau\right), \tag{13}$$

the dual optimization problem is then

$$\min_{\lambda_j \geq 0} d(\lambda_j). \tag{14}$$

By Lemma 1, strong duality holds for the problem in Eq. (12). Therefore, $d(\lambda_j) \geq V_{r,j}^{\pi_\alpha^*}(s)$ for every $\lambda_j \geq 0$ and the set of optimal solutions in Eq. (14) is nonempty. The dual function in Eq. (13) is also convex [3].

Let $\pi_{\alpha,t}^* \in \operatorname{argmax}_{\pi_\alpha \in \Pi_c} L(\pi_\alpha, \lambda^{(t)})$. The output of the update step in Eq. 10,

$$i^{(t+1)} = \operatorname*{argmax}_{i \in [N]} V_{r,j}^{\tilde{\pi}_i}(s) + \lambda^{(t)}\left(V_{c,j}^{\tilde{\pi}_i}(s) - \tau\right), \tag{15}$$

[1]Dept. of Computer Science, National University of Singapore {zeyu, bowenzhang, jianxin.bi, harold}@comp.nus.edu.sg

will give a value $L\left(\tilde{\pi}_{i^{(t+1)}}, \lambda^{(t)}\right) \geq L\left(\pi^*_{\alpha,t}, \lambda^{(t)}\right)$ due to the linearity of expectation in Π_α. Therefore, $L\left(\tilde{\pi}_{i^{(t+1)}}, \lambda^{(t)}\right) = L\left(\pi^*_{\alpha,t}, \lambda^{(t)}\right)$. Hence, the update step in Eq. 11,

$$\lambda^{(t+1)} = \mathrm{P}_{\mathbb{R}_{\geq 0}}\left(\lambda^{(t)} - \eta^{(t)}\left(V^{\tilde{\pi}_{i^{(t+1)}}}_{r,j}(s) - \tau\right)\right), \quad (16)$$

is optimizing with a subgradient $V^{\tilde{\pi}_{i^{(t+1)}}}_{r,j}(s) - \tau \in \partial d(\lambda^{(t)})$, which is also bounded since the value function is bounded. Finally by the condition of the sequence of the step sizes and Lemma I.3, which can be adapted to minimization of a convex function, we conclude the proof.

□

C. Proof of Proposition 3

Prop. 3 can be proved by similar steps of the Prop. 4.1 in [4].

II. Implementation details

In this section we describe in detail of the environmental setup and training details of our empirical studies. Four-Room and Reacher are two benchmarks for RL transfer and safety, and we adopted the similar configurations and hyper-parameters used in [5]. We also introduced a challenging task in SafetyGym, which has high state dimensions and complex physical dynamics.

A. Four-Room

The state in Four-Room consists of agent's 2-dimensional location and a binary vector indicating whether an object has been picked up, thus, $\mathcal{S} = J^2 \times \{0,1\}^{18}$ where $J = \{1, 2, \ldots, 13\} \subset \mathbb{N}$. The agent has 4 actions of moving up, down, left or right. Each task has unique values of the object rewards sampled uniformly between $[-1, +1]$. The agent receives zero rewards in empty locations. Therefore, some objects (with positive rewards) are beneficial while objects with negative rewards should be avoided by the agent. The goal state has a reward of 2, and the cost of the unsafe traps is -0.1, which is consistent with the expected cost $\mathbb{E}\left[c\left(s,a,s'\right)\right] = 5\% \times (-2)$ used in the uncertainty environment of RASFQL. Each task is trained for 20,000 interactions with the environment with an episode length of 200. All 128 tasks are sequentially trained. Thus, the total number of training iterations is 2.56 million. When started in a new task, the successor feature is initiated from the one learned in previous task. We then estimate the dual variable and use policy transfer during transfer learning on the new task. Some hyper-parameters of Q-learning in training are: discount factor $\gamma = 0.95$, epsilon-greedy for exploration $\epsilon = 0.12$, learning rates for successor feature, reward vector and λ are all $\alpha = 0.5$. For dual estimation during transfer, the learning rate $\eta^{(t)} = c \cdot \frac{1}{t}$, where $c = 1000$ is a constant. The threshold $\tau = -0.000005$. Successor features are represented with tables and computation is performed with CPUs. We report results using means and standard deviations calculated from 10 independent runs. Plots of per-task performance are results averaged over 8 tasks.

B. Reacher

The state space $\mathcal{S} \subset \mathbb{R}^4$ for the 2-link robot arm. Since Reacher is an environment with continuous state and action spaces, we discretize the action space to $\{-1, 0, 1\}^2$ by following prior work (SFQL and RaSFQL). The goals in training tasks are located in $(0.14, 0)$, $(0, 0.14)$, $(-0.14, 0)$ and $(0, -0.14)$. The locations for 8 test tasks are $(0.22, 0)$, $(0, 0.22)$, $(-0.22, 0)$, $(0, -0.22)$, $(0.1, 0.1)$, $(-0.1, 0.1)$, $(-0.1, -0.1)$ and $(0.1, -0.1)$. There are 6 unsafe round regions centered in $(0.14, 0)$, $(-0.14, 0)$, $(0.22, 0)$, $(-0.22, 0)$, $(0.1, 0.1)$ and $(-0.1, -0.1)$, respectively, whose radii are all 0.06. The reward is proportional to the negative distance d between the robot arm end effector and the goal location $r = 1 - 4d$. The cost of the unsafe region is -0.1. Each task is trained for 100,000 interactions with the environment with an episode length 500. We use a two-hidden-layer neural network for deep successor feature, where the size of the hidden layer is 256. The network is trained using stochastic gradient descent (SGD) with a learning rate of 0.001, a buffer size 400,000 and a batch size of 32. Other hyper-parameters are: discount factor $\gamma = 0.9$, epsilon-greedy for exploration $\epsilon = 0.1$, learning rate for the dual variable $\alpha = 10$ and threshold $\tau = -0.1$. The reacher environment is provided by the open-source PyBullet Gymperium packages [6]. The model is trained with GPUs and each trail (sequential training over 4 tasks) takes around 7 hours. We report results using means and standard deviations calculated from 10 independent runs.

C. SafetyGym

This environment has a larger state dimensionality $\mathcal{S} \subset \mathbb{R}^{77}$. The state space consists of the following components: readings from motor sensors of dimension 12, lidar readings of all 4 types of objects (button, goal, trap and wall) in 16 angles surrounding the robot, which bring up to a dimension of 64 in total, and finally a single-dimension indicator variable to represent whether the agent is inside a trap. We also discretize the action space in a similar fashion to Reacher. The two traps are located in $(0, 0)$ and $(0.5, -0.5)$. The goal is located in $(0.5, 0.5)$. The two buttons are located in $(-0.5, 0.5)$ and $(0.5, -0.5)$, therefore the former button is in a safe region while the latter is inside a trap, we refer to the former as 'safe button' and the latter 'risky button'. The robot starts in $(-0.5, -0.5)$. The reward in this environment is sparse, it is only obtained upon touching the buttons or the goal. The trap has a cost of 1 upon entrance. We train 4 tasks in total and in each task we alter the rewards of the buttons. The rewards of the 'safe button' and the 'risky button' in the 4 tasks are $(1, 4)$, $(2, 1)$, $(2, 5)$ and $(3, 3)$ respectively. Every task is trained for 200,000 steps with an episode length 500. We use the same network used for Reacher. We use a buffer size 100,000 and a batch size of 64. Other hyper-parameters are: discount factor $\gamma = 0.99$, epsilon-greedy for exploration $\epsilon = 0.25$, learning rate for the dual variable $\alpha = 10$ and threshold $\tau = -0.001$. We report results using means and standard deviations calculated from 10 independent runs.

III. Additional studies

A. Constraint satisfaction

Unlike the risk-aware method that learns to avoid any return with variance, the constraint formulation is more flexible in handling varying tolerance for violations. Fig. 1 shows that by setting different thresholds, SFT-CoP can satisfy the constraints during transfer learning (Fig. 1d exhibits the Q values are aligned with different thresholds). As expected, SFT-CoP has fewer failures when the threshold is tighter.

The reported curves over training task instances in this paper are training results when sequentially transferred to new tasks, with epsilon-greedy based exploration. Note that SFT-CoP can achieve almost zero constraint violation when the exploration parameter is set to 0, *e.g.*, Fig. 2 shows that after task 64 the failures curve becomes flat compared with continually increasing failures of epsilon-greedy based training.

B. Ablation study

In this section, we show the effect of our multi-source policy transfer strategy in Eq. (5) by comparing SFT-CoP against a transfer learning baseline that does not use this strategy. This baseline copies the successor feature from the previous task like SFT-CoP and estimates the optimal dual variable when learning on a new task during transfer. However, it does not use Eq. (5) for policy transfer. Fig. 3 shows that although it can achieve a similar level of failures during the sequential transfer learning, it collects fewer rewards. The performance does not improve in later tasks as shown by the flat curves. This suggests that without using policy transfer, the learned Q function gives a poorer policy that does not fully explore states.

C. Comparison with RASFQL

Fig. 4 and Fig. 5 compare SFT-CoP with RASFQL in the uncertain scenario on Four-Room and Reacher using the same risk configuration as in [5], where upon stepping into a trap or unsafe region, the agent only receives costs when they become activated (with a probability of 5% and 3.5% for these two environments, respectively). Results show that SFT-CoP provides a similar level of safety as RASFQL.

D. Additional results

Here we present all cumulative and per-task results of failures and different kinds of rewards in support of Fig. 2 and 3 in the main paper. Fig. 6 plots additional results of non-transfer comparison. We have tuned the baseline methods and find that PDQL and CPO can only converge after a larger number of steps (*e.g.* 350,000 steps on reacher for CPO) to learn a safe policy. Fig. 7, Fig. 8 and Fig. 9 plot the transfer learning results for Four-Room, Reacher and SafetyGym, respectively. Fig. 10 present the test results of averaged episode failures and rewards on the 8 test tasks of Reacher.

References

[1] A. Barreto, D. Borsa, J. Quan, T. Schaul, D. Silver, M. Hessel, D. Mankowitz, A. Zidek, and R. Munos, "Transfer in deep reinforcement learning using successor features and generalised policy improvement," in *Proceedings of the 35th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, J. Dy and A. Krause, Eds., vol. 80. PMLR, 10–15 Jul 2018, pp. 501–510.

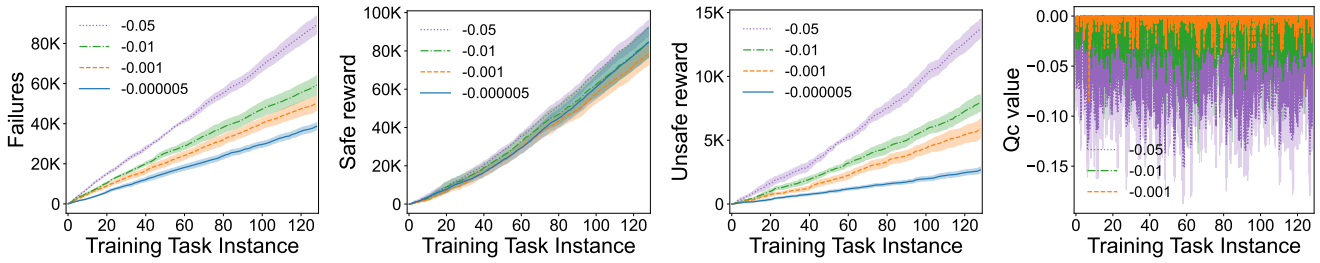
[2] K. M. Anstreicher and L. A. Wolsey, "Two "well-known" properties of subgradient optimization," *Mathematical Programming*, vol. 120, no. 1, pp. 213–220, 2009.

[3] S. Boyd, S. P. Boyd, and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[4] A. Shapiro and A. Kleywegt, "Minimax analysis of stochastic problems," *Optimization Methods and Software*, vol. 17, no. 3, pp. 523–542, 2002.

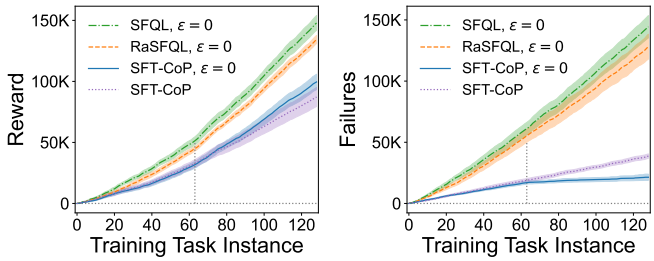
[5] M. Gimelfarb, A. Barreto, S. Sanner, and C.-G. Lee, "Risk-aware transfer in reinforcement learning using successor features," in *Advances in Neural Information Processing Systems*, 2021.

[6] B. Ellenberger, "Pybullet gymperium," https://github.com/benelot/pybullet-gym, 2018–2019.



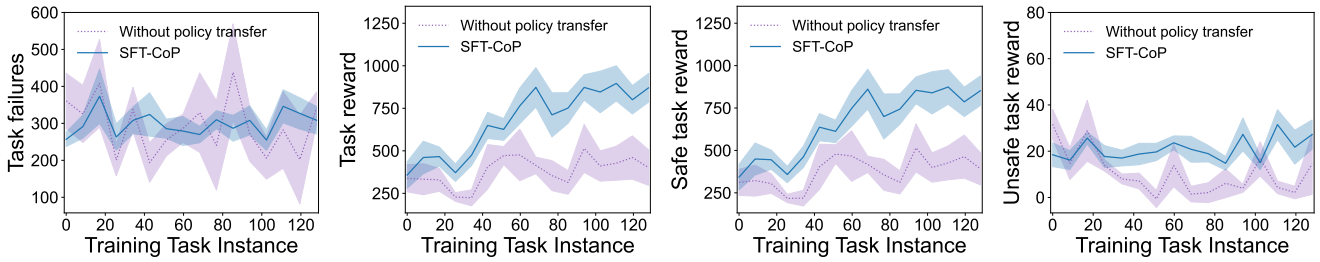
(a) Accumulated failures. (b) Accumulated rewards from safe objects. (c) Accumulated rewards from unsafe objects. (d) Value of $Q^{\pi}_{c,j}$.

Fig. 1: Performance of SFT-CoP with different threshold τ on the Four-Room domain. We compute accumulated (a) failures, (b) rewards from safe objects, (c) rewards from unsafe objects and (d) the value of $Q^{\pi}_{c,j}$, over the training task instances.



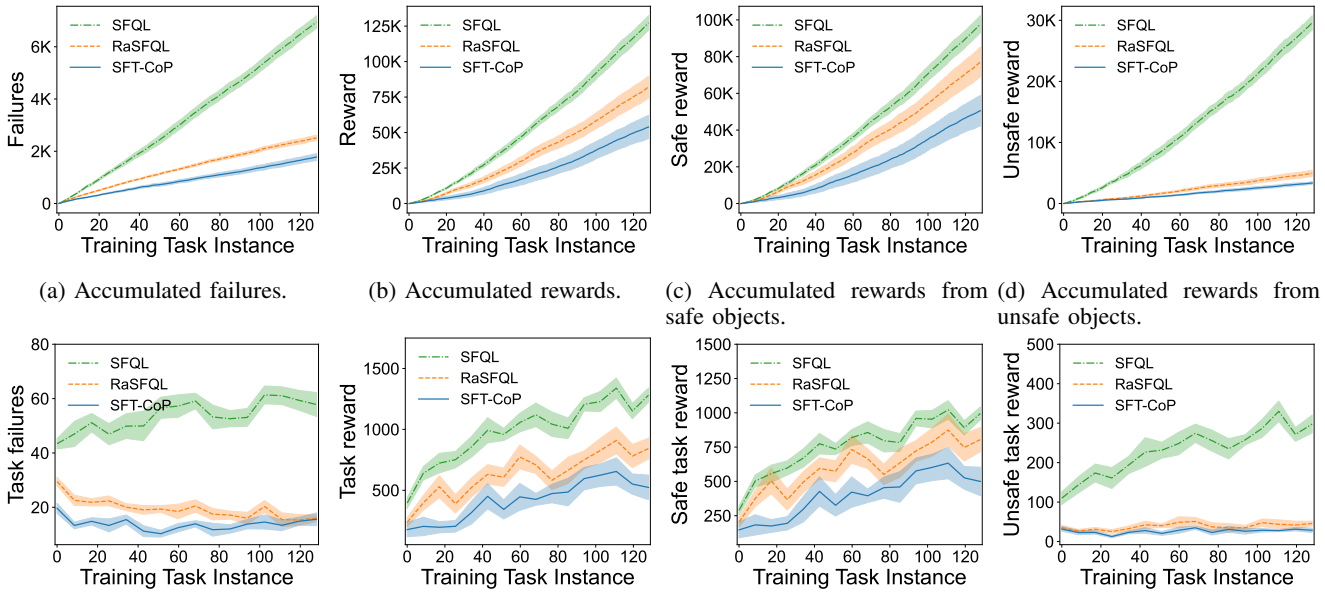
(a) Accumulated rewards. (b) Accumulated failures.

Fig. 2: Accumulated (a) rewards and (b) failures of SFQL, RASFQL ($\beta = 2$) and SFT-CoP on the Four-Room domain with the parameter of epsilon-greedy exploration ϵ set to 0 after task 64.



(a) Task failures. (b) Task rewards. (c) Task rewards from safe objects. (d) Task rewards from unsafe objects.

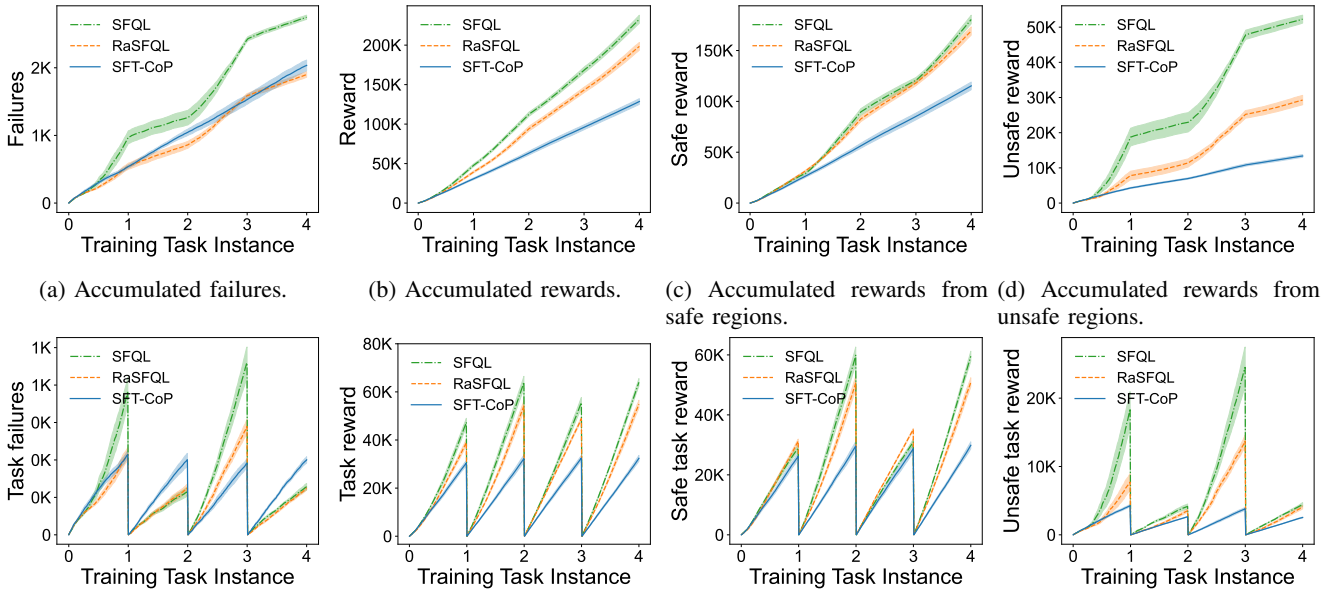
Fig. 3: Performance of SFT-CoP w. and w.o. multi-source policy transfer during sequential transfer learning on the Four-Room domain. We compute accumulated (a) failures, (b) total rewards, (c) rewards from safe objects and (d) rewards from unsafe objects, over the training task instances.



(a) Accumulated failures. (b) Accumulated rewards. (c) Accumulated rewards from safe objects. (d) Accumulated rewards from unsafe objects.

(e) Task failures. (f) Task rewards. (g) Task rewards from safe objects. (h) Task rewards from unsafe objects.

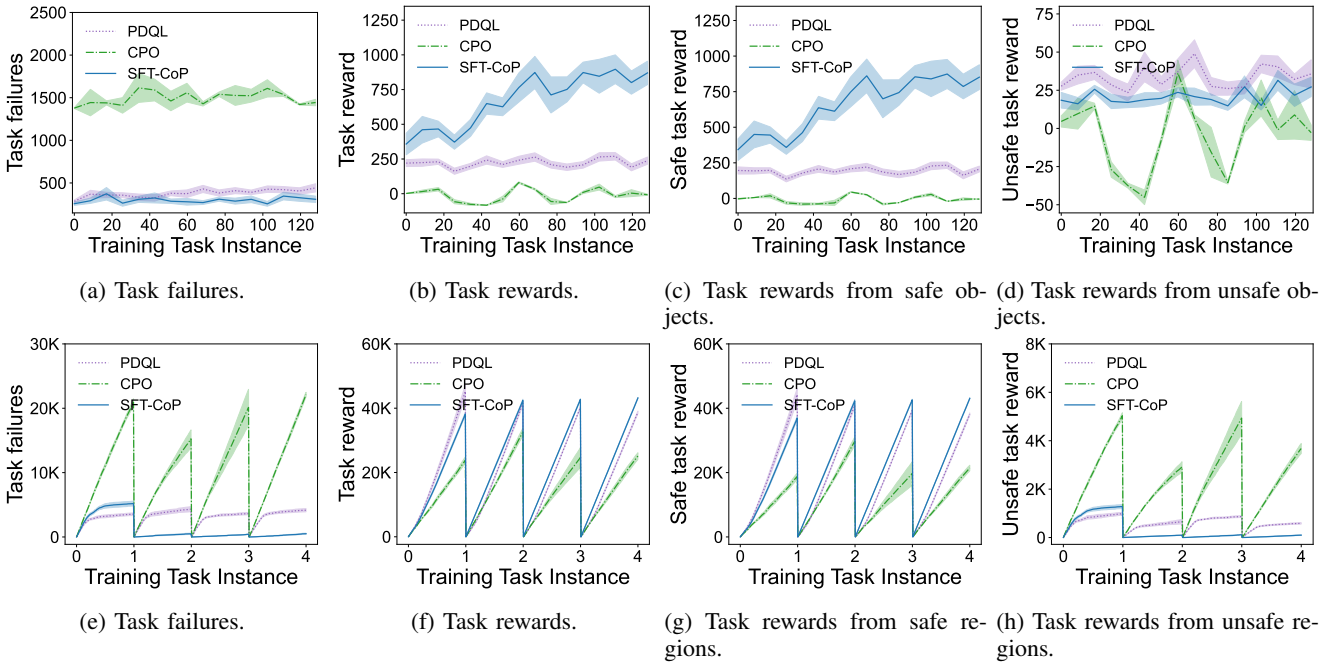
Fig. 4: Performance of SFQL, RASFQL ($\beta = 2$) and SFT-COP on the Four-Room domain with probabilistic traps. We compute accumulated and per-task (a, e) failures, (b, f) total rewards (c, g) rewards from safe objects and (d, h) rewards from unsafe objects, over the training task instances.



(a) Accumulated failures. (b) Accumulated rewards. (c) Accumulated rewards from safe regions. (d) Accumulated rewards from unsafe regions.

(e) Task failures. (f) Task rewards. (g) Task rewards from safe regions. (h) Task rewards from unsafe regions.

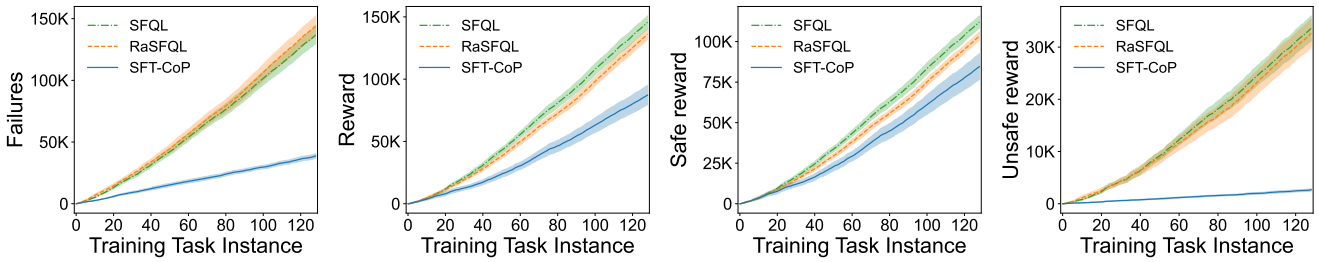
Fig. 5: Performance of SFQL, RASFQL ($\beta = 2$), and SFT-COP on the Reacher domain with probabilistic unsafe regions. We report accumulated and per-task (a, e) failures, (b, f) total rewards, (c, g) rewards from safe regions and (d, h) rewards from unsafe regions, over the training tasks.



(a) Task failures. (b) Task rewards. (c) Task rewards from safe objects. (d) Task rewards from unsafe objects.

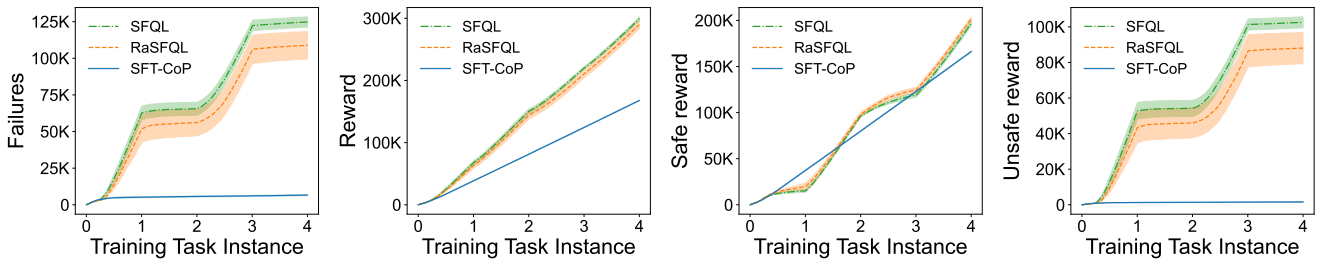
(e) Task failures. (f) Task rewards. (g) Task rewards from safe regions. (h) Task rewards from unsafe regions.

Fig. 6: Performance of PDQL and SFT-CoP on the Four-Room (top row) and Reacher (bottom row) domains. We report per-task (a, e) failures, (b, f) total rewards, (c, g) rewards from safe regions and (d, h) rewards from unsafe regions, over the training tasks.

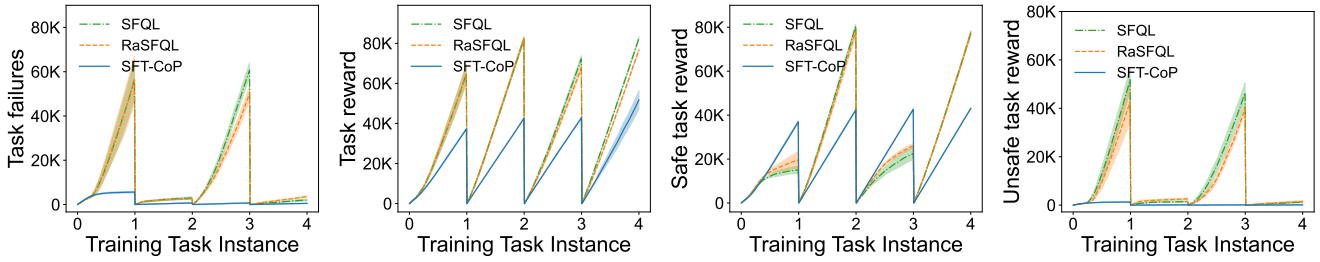


(a) Accumulated failures. (b) Accumulated rewards. (c) Accumulated rewards from safe objects. (d) Accumulated rewards from unsafe objects.

Fig. 7: Performance of SFQL, RASFQL ($\beta = 2$) and SFT-CoP on the Four-Room domain. We compute accumulated (a) failures, (b) total rewards, (c) rewards from safe objects and (d) rewards from unsafe objects, over the training task instances.

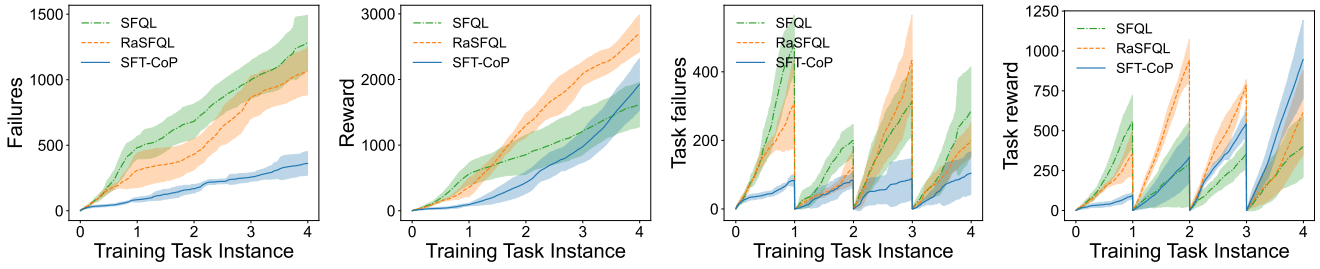


(a) Accumulated failures. (b) Accumulated rewards. (c) Accumulated rewards from safe regions. (d) Accumulated rewards from unsafe regions.



(e) Task failures. (f) Task rewards. (g) Task rewards from safe regions. (h) Task rewards from unsafe regions.

Fig. 8: Performance of SFQL, RASFQL ($\beta = 2$), and SFT-COP on the Reacher domain. We report accumulated and per-task (a, e) failures, (b, f) total rewards, (c, g) rewards from safe regions and (d, h) rewards from unsafe regions, over the training tasks.



(a) Accumulated failures. (b) Accumulated rewards. (c) Task failures. (d) Task rewards.

Fig. 9: Performance of SFQL, RASFQL ($\beta = 2$) and SFT-COP on the SafetyGym domain. We report accumulated (a) failures and (b) rewards, and per-task (c) failures and (d) rewards, over the training tasks.

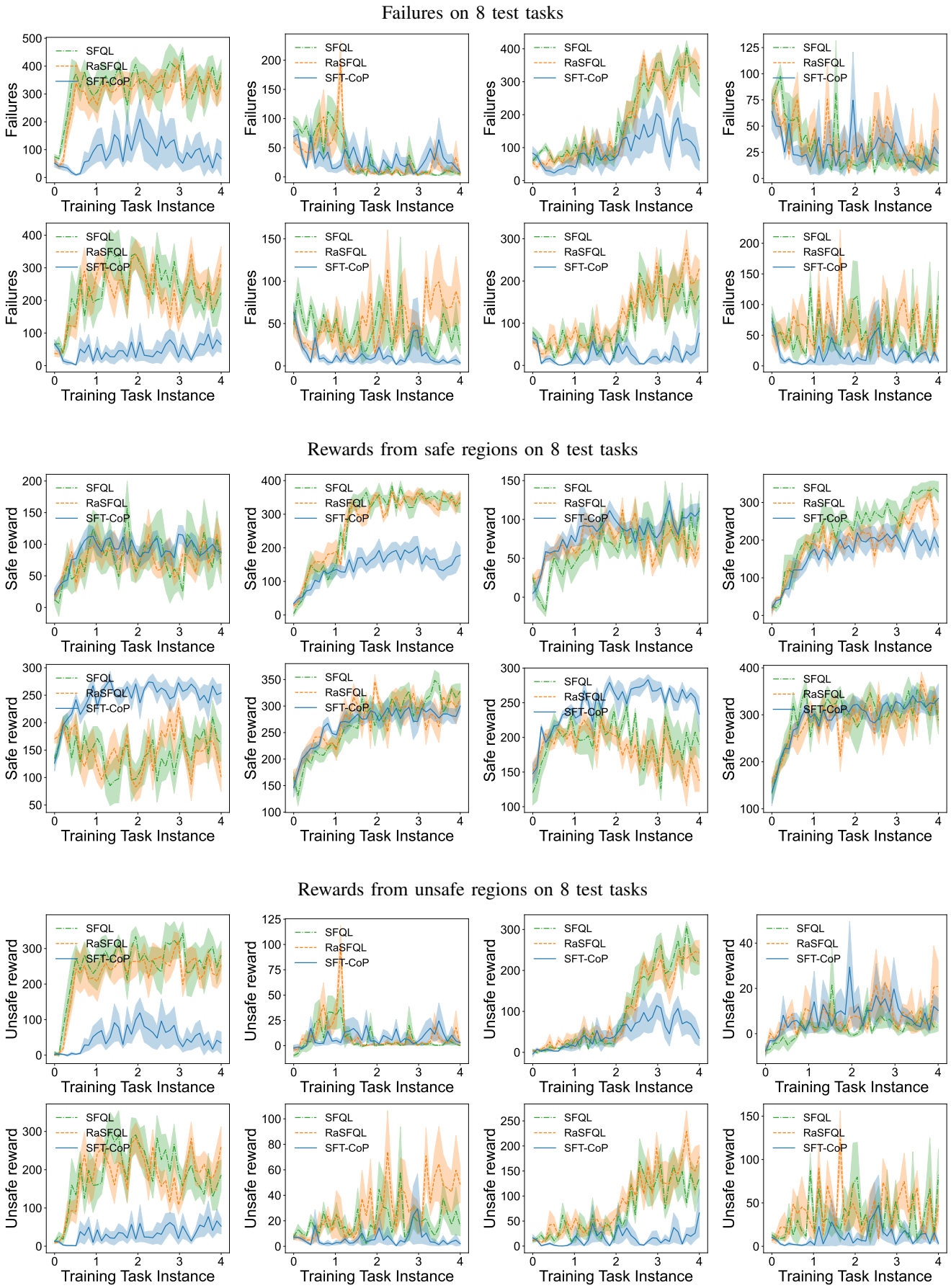


Fig. 10: Performance of SFQL, RASFQL ($\beta = 2$), and SFT-COP on the Reacher domain tested on 8 test tasks. We report averaged episode failures (top 3 rows), rewards from safe regions (middle 3 rows) and rewards from unsafe regions (bottom 3 rows), during the training course.