SAFE MODEL-BASED REINFORCEMENT LEARNING WITH ROBUST CROSS-ENTROPY METHOD

Zuxin Liu[1], **Hongyi Zhou**[1], **Baiming Chen**[2], **Sicheng Zhong**[3], **Ding Zhao**[1]
[1]Carnegie Mellon University, [2] Tsinghua University, [3]University of Toronto
{zuxinl, hzhou3, baimingc, sichengz, dingzhao}@andrew.cmu.edu

ABSTRACT

This paper studies the constrained/safe reinforcement learning (RL) problem with sparse indicator signals for constraint violations. We propose a model-based approach to enable RL agents to effectively explore the environment with unknown system dynamics and environment constraints given a significantly small number of violation budgets. We employ the neural network ensemble model to estimate the prediction uncertainty and use model predictive control as the basic control framework. We propose the robust cross-entropy method to optimize the control sequence considering the model uncertainty and constraints. We evaluate our methods in the Safety Gym environment. The results show that our approach learns to complete the tasks with a much smaller number of constraint violations than state-of-the-art baselines. The code is available at https://github.com/liuzuxin/safe-mbrl.

1 INTRODUCTION

Reinforcement learning (RL) has achieved great success in a wide range of applications (Mnih et al., 2013; Filos et al., 2020). However, in the course of learning, it is usually hard to prevent the agent from getting into high-risk states which may lead to catastrophic results, especially for safety-critical applications. Therefore, it is important to develop safe reinforcement learning algorithms for real-world applications, which allow them to complete tasks while satisfying certain safety constraints. One example problem is the `PointGoal` task setting in the Safety Gym simulation environment (Ray et al., 2019), where a robot needs to navigate to the goal while avoiding all of the hazard areas. The dynamics model of the environment is unknown, and the robot only receives indicator signals when violating constraints. The observations of the robot are sensor data, such as a LiDAR point cloud, so it is hard to analytically express the mapping from observation space to the constraint violation. Thus we are interested in the hardest cases where both dynamics and constraints are needed to be learned from data without additional info.

The challenges of solving the above problem are threefold: First, pure model-free, safe RL algorithms, such as Lagrangian-based methods (Stooke et al., 2020; Altman, 1998) and projection-based optimization methods (Achiam et al., 2017) are not sample efficient. They need to constantly violate safety constraints and collect a large number of unsafe data to learn the policy, which restricts the application in safety-critical environments. Second, the task objective and the safety objective of an RL agent may contradict each other, which may corrupt the policy optimization procedure for methods that simply transform the original reward to the combination of reward and constraint violation cost (Geibel & Wysotzki, 2005; Gaskett, 2003). Finally, the black-box constraint function and unknown environment dynamics model make the problem hard to optimize. Therefore, most existing model-based safe RL approaches either assume a known prior dynamics or a known structure of the constraint functions (Berkenkamp et al., 2017; Koller et al., 2018; Pham et al., 2018). As far as we are aware, very little research has been done to investigate situations in which the dynamics and the constraint are both unknown.

We present a safe model-based RL algorithm with a robust cross-entropy (RCE) method to achieve near-optimal task performance with near-zero constraint violation rates. We propose a solution to learn both the dynamics model and constraint model from limited samples and weak constraint violation indicator signals. Our approach is able to achieve state-of-the-art performance in terms

Algorithm 1 Robust Cross-Entropy Method for RL

Input: Initial distribution parameter Θ ; number of samples N; number of elites k; initial state s_0
Output: Solution $\mathcal{X}^*$ with the highest reward

while The stop criteria is not satisfied **do**
  Draw N samples from the initial distribution: $\mathcal{X}_1, \mathcal{X}_2, ..., \mathcal{X}_N \sim \mathcal{N}(\Theta)$
  Evaluate each sample $\mathcal{X}_i$ by Eq.2 to get the estimation of reward $r(\mathcal{X}_i; s_0)$ and cost $c(\mathcal{X}_i; s_0)$
  Select the feasible set $\Omega \in \{\mathcal{X}_i\}_{i=1}^N$ based on the cost estimation
  if Ω is empty **then**
    Sort $\{\mathcal{X}_i\}_{i=1}^N$ in ascending order w.r.t the cost. Let Λ_k be the first k elements
  else
    Sort Ω in descending order w.r.t the reward.
    Let Λ_k be the first k elements of Ω if $|\Omega| > k$, otherwise let Λ_k be Ω
  end if
  Update Θ by maximizing the likelihood given Λ_k: $\Theta \leftarrow \arg\max_\theta \prod_{\mathcal{X} \in \Lambda_k} p(\mathcal{X}; \theta)$
end while
return $\mathcal{X}^*$ with highest reward in Λ_k

of constraint violation rate and accumulated expected reward in the Safety Gym environment (Ray et al., 2019).

2 METHOD

Model Learning. Since the dynamics $f(s_t, a_t)$ and the cost (constraint violation) model $c(s_{t+1})$ are both unknown, we need to infer them from collected samples. We use an ensemble of neural networks to learn the dynamics (Chua et al., 2018) and estimate the epistemic uncertainty (subjective uncertainty due to a lack of data) of the input data. Any binary classification model could be used to approximate the cost model because it is an indicator function. In light of robustness towards limited samples, as well as low computational burden, we adopt a gradient boosting decision tree-based ensemble method - LightGBM (Ke et al., 2017) - to learn the cost model.

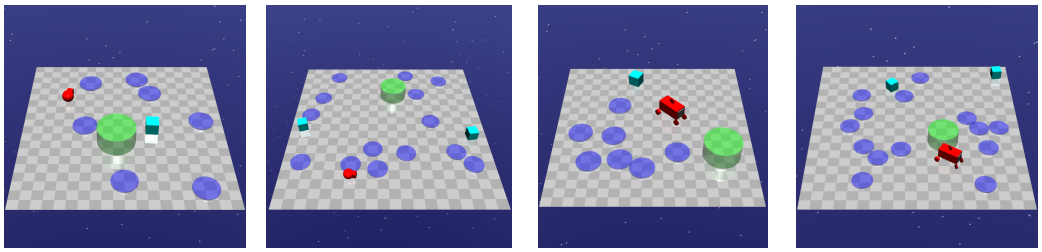
MPC with Learned Models. We use Model Predictive Control (MPC) as the basic control framework for our constrained model-based RL approach. The objective of MPC is to maximize the accumulated reward w.r.t a sequence of actions $\mathcal{X} = (a_0, ..., a_T)$, where T is the planning horizon. The first action is applied to the system, new observations are received, and the same optimization procedure is performed again. In our safe RL setting, additional constraints are introduced so that the original objective becomes a constrained optimization problem. Denote s_t as the state at time t, γ as the discount factor, $r(s_{t+1})$ as the reward function, we aim to solve the following problem:

$$\begin{aligned} \mathcal{X} &= \arg\max_{a_0,...,a_T} \quad \mathbb{E}\big[\sum_{t=0}^{T} \gamma^t r(s_{t+1})\big] \\ s.t. \quad & s_{t+1} = f(s_t, a_t), c(s_{t+1}) = 0, \quad \forall t \in \{0, 1, ..., T-1\} \end{aligned} \tag{1}$$

Robust Cross-Entropy Method. To directly solve the constrained optimization problem in Eq. 1, we propose the robust cross-entropy method (RCE) by using the trajectory sampling (TS) technique (Chua et al., 2018) to estimate reward and constraint violation cost. Given the initial state s_0, we can evaluate the accumulated reward and cost of the solution by:

$$r(\mathcal{X}; s_0) = \sum_{t=0}^{T} \gamma^t \big(\frac{1}{B} \sum_{b=1}^{B} r(s_{t+1}^b)\big), \quad c(\mathcal{X}; s_0) = \sum_{t=0}^{T} \beta^t \max_b c(s_{t+1}^b) \tag{2}$$

where $s_{t+1}^b = \tilde{f}_{\theta_b}(s_t^b, a_t), \forall t \in \{0, ..., T-1\}, \forall b \in \{1, ..., B\}$, γ and β are discounting factors, and B is the ensemble size of the dynamics model. The intuition of TS is to consider the worst-case scenario of constraint violations among all sampled future trajectories with dynamics prediction uncertainty. Our RCE method first selects the feasible set of solutions that satisfy the constraints based on the estimated cost in Eq. 2. Then, we sort the solutions in the feasible set and select the top k samples to use when calculating the parameters of the sampling distribution for the next iteration.



(a) Point Goal1 Env (b) Point Goal2 Env (c) Car Goal1 Env (d) Car Goal2 Env

Figure 1: Experiment Environments

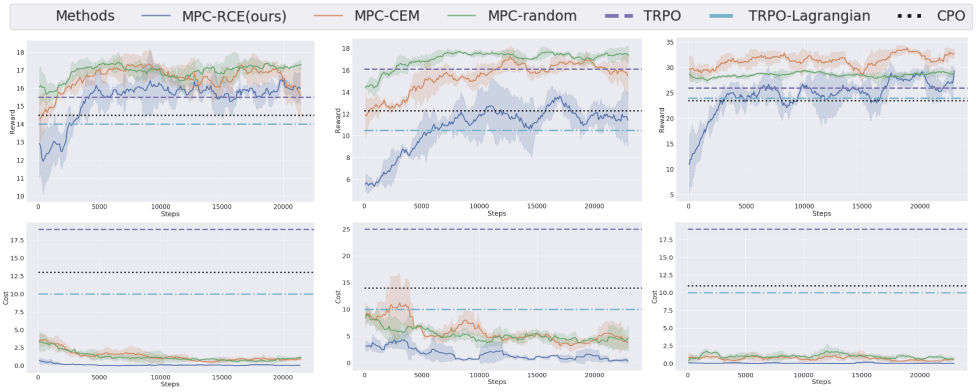


Figure 2: Learning curves. The upper row figures are the reward trends and the lower row figures are the cost trends. From left to right the tasks are: Point Goal1, Point Goal2, and Car Goal1.

If all the samples violate at least one constraint, we select the top k samples with the lowest costs. The RCE algorithm is shown in Algorithm 1. The entire training pipeline of our MPC with RCE is presented in Algorithm 2.

3 Experiment Validation

Algorithm 2 MPC with RCE

Input: Initial collected data $\mathcal{D}$; RCE parameters $\mathcal{P}$

while The performance is not converged **do**
  Train the dynamics $\tilde{f}$ and cost model $\tilde{c}$ given $\mathcal{D}$
  for Time $t = 0$ to EpisodeLength **do**
    Observe state s_t from the environment
    Optimize actions by Alg. 1: $\{a_i^*\}_{i=t}^{t+T} \leftarrow \text{RCE}(\mathcal{P}, s_t)$
    Apply the first action a_t^* in $\{a_i^*\}_{i=t}^{t+T}$ to the system
    Observe next state s_{t+1} and cost signal $c(s_{t+1})$
    Update data buffer: $\mathcal{D} \leftarrow \mathcal{D} \cup \{s_t, a_t, s_{t+1}, c(s_{t+1})\}$
  end for
end while

Simulation Environment. We evaluate our safe RL approach in the OpenAI Safety Gym environment (Ray et al., 2019) with the Goal task. Each experiment setting involves a robot (red object in Fig. 1) that must navigate a clustered environment to accomplish a task while avoiding contact with obstacles. When the robot enters the goal circle (green circle), the goal location is randomly reset. A bonus of $r_t = 1$ is given to the robot for reaching the goal. Hazards (blue circles) are dangerous areas to avoid. Vases (teal cube) are objects initialized to be stationary but movable upon touching. The agent is penalized for entering Hazards or touching Vases. If the agent violates the safety constraint, it will receive a cost equals 1. Level 2 tasks (Fig. 2bd) are more difficult to solve than level 1 (Fig. 2ac) task since there are more constraints presented.

Table 1: Comparison of total constraint violation number for the first 10000 training steps.

Task \ Method	MPC-RCE	MPC-CEM	MPC-random
Point Goal 1	**16.00**	184.33	169.0
Point Goal 2	**231.00**	746.00	600.67
Car Goal 1	**7.00**	103.67	139.33
Car Goal 2	**96.00**	578.00	509.33

Baselines. We use the official baseline methods provided by the Safety Gym environment (Ray et al., 2019). The trust region policy optimization (TRPO) algorithm (Schulman et al., 2015) is an unconstrained baseline that could give us intuition about the performance and constraint violations when we only care about the reward and not the cost of violating constraints. We use the Lagrangian version of TRPO (TRPO-Lagrangian) and the constrained policy optimization (CPO) (Achiam et al., 2017) as two model-free constrained reinforcement learning baselines. For model-based safe RL baselines, we adopt the constrained extension of existing random shooting and the cross-entropy method (CEM) together with the MPC framework. We name the two methods as MPC-random and MPC-CEM.

Metrics and training. We compare different approaches in terms of episodic accumulated reward and episodic cost (Ray et al., 2019), which is defined as the total constraint violation number in each episode. **Method A is better than B if A could achieve lower episodic cost than B. If both A and B achieve the same cost, then the one with higher episodic reward is better.** We also compare the sample efficiency and the total cost during training. For each algorithm, we evaluate them for each task with 3 different random seeds. For the detail about specific hyper-parameters, please refer to the supplementary material.

Results. Since RL agents learn by trial and error and we assume no prior knowledge of the environment, it is inevitable to violate the constraints in the early stage of training. However, our goal is to reduce the unsafe samples as much as possible because collecting unsafe data could be expensive in some cases. Such setting is practical in real-world applications. Here we provide two examples: 1. The cost to enter an 'unsafe' zone is not deadly but is not desired. For example, a tomato-picking-up robot damages tomatoes, which is not harmful to itself and people, but we still regard it as unsafe because it should be avoided in reality. 2. We can train the robot by historical unsafe data (such as public dataset) or in the simulator, which is not harmful during training. Then we can deploy trained safe policies to real-world situations.

Fig. 2 shows the learning curves of reward and constraint violation cost in Point Goal1, Point Goal2, and Car Goal1 tasks. The reward and cost are averaged among 3 experiments with the same hyper-parameters but different random seeds. The solid line is the mean value, and the light shade represents the area within one standard deviation. We use dashed lines to represent the value at convergence for model-free approaches, as they require several orders of magnitude more interaction steps to obtain satisfactory performance.

From the figure, it is apparent that our MPC-RCE approach learns the underlying constraint function very quickly to avoid unsafe behaviors during the exploration and achieves the lowest constraint violation rate, though its reward is slightly lower than other methods. **It is reasonable because the best policy to maximize the task reward is to ignore the constraints and let the robot go straight towards the goal.** So MPC-CEM and MPC-random sacrifice the safety constraints satisfaction performance to obtain the gain on task reward. A more intuitive demonstration could be found in our supplementary video. However, as suggested in Safety Gym benchmark (Ray et al., 2019), we first compare the constraint satisfaction performance of different methods, and if they are the same, then we compare the task rewards. Therefore, our MPC-RCE is the best among all baselines because its constraint violation counts are always less than other methods, while the task rewards are comparable to other relatively 'unsafe' approaches.

Table 1 demonstrates the constraint satisfaction performance during the training procedure. For model-free methods, the number of constraint violations is several orders of magnitude larger than model-based approaches, so we do not list the result here. From the table, we can see our approach achieves much lower cost than the other methods, which means the MPC-RCE agent requires the minimum number of unsafe samples to converge. As far as we are aware, our method can achieve the best constraint satisfaction performance in these Safety Gym tasks.

ACKNOWLEDGMENTS

The authors acknowledge the support from the Manufacturing Futures Initiative at Carnegie Mellon University made possible by the Richard King Mellon Foundation which supported Zuxin Liu and Baiming Chen. Zuxin Liu is also in part supported by Carnegie Mellon University's Mobility21 National University Transportation Center, which is sponsored by the US Department of Transportation.

REFERENCES

Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained policy optimization. *arXiv preprint arXiv:1705.10528*, 2017.

Eitan Altman. Constrained markov decision processes with total cost criteria: Lagrangian approach and dual linear program. *Mathematical methods of operations research*, 48(3):387–417, 1998.

Felix Berkenkamp, Matteo Turchetta, Angela Schoellig, and Andreas Krause. Safe model-based reinforcement learning with stability guarantees. In *Advances in neural information processing systems*, pp. 908–918, 2017.

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Advances in Neural Information Processing Systems*, pp. 4754–4765, 2018.

Angelos Filos, Panagiotis Tigas, Rowan McAllister, Nicholas Rhinehart, Sergey Levine, and Yarin Gal. Can autonomous vehicles identify, recover from, and adapt to distribution shifts? *arXiv preprint arXiv:2006.14911*, 2020.

Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pp. 1050–1059, 2016.

Chris Gaskett. Reinforcement learning under circumstances beyond its control. 2003.

Peter Geibel and Fritz Wysotzki. Risk-sensitive reinforcement learning applied to control under constraints. *Journal of Artificial Intelligence Research*, 24:81–108, 2005.

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. In *Advances in neural information processing systems*, pp. 3146–3154, 2017.

Torsten Koller, Felix Berkenkamp, Matteo Turchetta, and Andreas Krause. Learning-based model predictive control for safe exploration. In *2018 IEEE Conference on Decision and Control (CDC)*, pp. 6059–6066. IEEE, 2018.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Tu-Hoa Pham, Giovanni De Magistris, and Ryuki Tachibana. Optlayer-practical constrained optimization for deep reinforcement learning in the real world. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6236–6243. IEEE, 2018.

Alex Ray, Joshua Achiam, and Dario Amodei. Benchmarking safe exploration in deep reinforcement learning. *arXiv preprint arXiv:1910.01708*, 2019.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pp. 1889–1897, 2015.

Adam Stooke, Joshua Achiam, and Pieter Abbeel. Responsive safety in reinforcement learning by pid lagrangian methods. *arXiv preprint arXiv:2007.03964*, 2020.

A APPENDIX

A.1 DISCUSSION ABOUT THE SAFETY GYM ENVIRONMENT AND THE TASK SETTING

Safety Gym environments use the MuJoCo physics engine as the backbone simulator. Each environment and task is inspired by a practical safety issue in robotics control. The observation spaces used in the original Safety Gym environment includes standard robot sensors (accelerometer, gyroscope, magnetometer, and velocimeter) and pseudo-lidar (each lidar sensor perceives objects of a single kind and is computed by filling bins with appropriate values). The observation space used in our approach is different from the default Safety Gym options in that we pre-process the sensor data to get rid of some noisy and unstable sensors, such as the z-axis data of accelerometer. We use the relative coordinates of the perceived objects instead of the pseudo-lidar readings because the former representation is more friendly to dynamics model learning, which is important for model-based RL.

Both robots used in our experiment have two-dimensional continuous action spaces and all actions are linearly scaled to $[-1, +1]$. We also performed careful hand-tuning of some MuJoCo actuator parameters during sensor analysis, since robust and responsive control is critical to robot operations in both the simulation environment and the real world.

Our work in the Safety Gym environment has implications for real-world applications. The Goal task in our experiment resembles the setting of the delivery robot and other domestic robots, where the robot has to navigate around static obstacles such as furniture to reach the goal. Additionally, since the state representation in our experiments is directly derived from sensor information and the control input of our environment to the robot is very similar to that of real-world situations, our model-based RL approach in the simulation environment could serve as an important pre-training for the real-world applications. Given that a certain amount of unsafe data is required to train our model, it would be unrealistic to have the real robot repeatedly violate the constraints to collect such data. Therefore, the training in the simulator is an important step for the model to be transferable to real wold safety-critical applications.

A.2 TRAINING DETAIL

Dynamics model: We use the same architecture and hyper-parameters of each neural network in the ensemble dynamics model. Each neural network is of 3 layers with ReLU activation and each layer is of 1024 neurons. All the training parameters for one task are the same for MPC-RCE, MPC-CEM, and MPC-random. The batch size is 256, the learning rate is 0.001, the training epochs are 70, and the optimizer is Adam. The ensemble number is 4 for Point robot-related tasks and is 5 for Car robot-related tasks. Each neural network model in the ensemble is trained with 80% of the training data to prevent overfitting.

LightGBM classifier: We use LightGBM to predict the constraint violation given a state in our RCE method and all the model-based baselines. We use the default `gdbt` boosting type and 400 base estimators. Each base estimator has a maximum depth of 8 and 12 leaves. The learning rate is 0.3 and all other hyper-parameters are the default value.

RCE, CEM, and random optimizer: We use the same hyper-parameters for RCE and CEM except that RCE has a discount value of $\gamma = 0.98$ for reward and discount value of $\beta = 0.4$ for cost while CEM only has one discount value $\gamma = 0.98$ for the combination of reward and cost. We sample $N = 500$ solutions for each iteration of RCE and CEM and select top $k = 12$ elite samples to estimate the distribution parameters for the next iteration. If the iteration number exceeds 8 or the sum of the variance of elite samples is less than $\epsilon = 0.01$, the optimization procedure stops and returns the best solution that has been found so far. To fairly compare with RCE and CEM, we use 5000 samples for the random shooting method so that the maximum number of samples is at the same order of magnitude. The planning horizon is $T = 8$ for all methods.

TRPO, TRPO-Lagrangian, and CPO: We use the same hyper-parameters offered in the open-sourced code from the baseline method for the Safety Gym simulation environment (Ray et al., 2019). All hyperparameters are kept the same for all three model-free baseline methods. The actor-critic neural network model has 2 linear layers of 256 hidden neurons in each. The discount factor $\gamma = 0.99$. The target cost limit is 10 with penalty term λ initialized to be 1 and a penalty term learning rate of 0.05. The target KL divergence is 0.01, and for the value function learning, the

learning rate is 0.001 with 80 iterations. For each experiment, the total number of environment interactions is $1e7$ and $3e4$ steps for each training epoch.

A.3 More Results

The influence of the target cost value for TRPO-Lagrangian and CPO. Since the target cost limit value must be set in advance before training, we empirically study the performance of TRPO-Lagrangian and CPO with different target values in the Point Goal2 environment. The learning curves are shown in Fig 3 and Fig 4. We can see that the task performance is negatively correlated with the target cost, and there is a dramatic task performance drop if we limit the target cost to a small value. Compared with model-based approaches, CPO and TRPO-Lagrangian can hardly achieve comparable task performance with the same level of constraint violation rate.

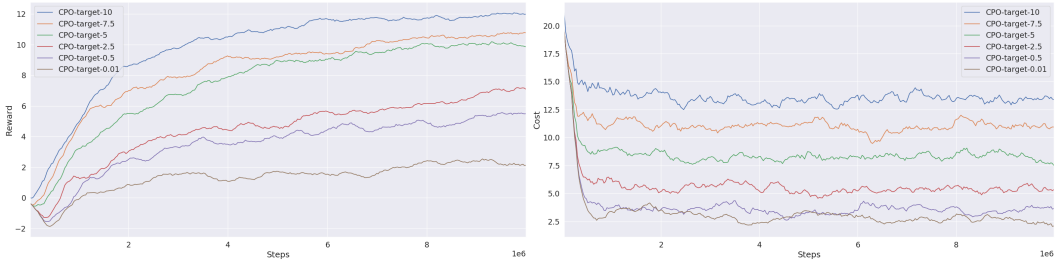


Figure 3: Learning curves of reward and cost for CPO with different target cost value.

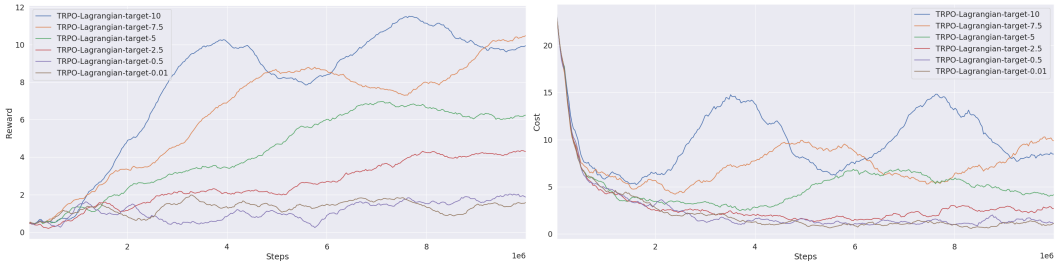


Figure 4: Learning curves of reward and cost for TRPO-Lagrangian with different target cost value.

RCE, CEM, and random optimizer comparison. To better compare the performance of RCE, CEM, and random optimizer, we fix the dynamics model, cost model, and random seed to test in the same Point Goal1 environment. The smoothed reward and cost curves are shown in Fig. 5. We can see our RCE approach achieves the lowest cost throughout the testing phase while maintaining comparable task performance compared to CEM and random. It is interesting to note that there is a reward drop for RCE and a cost jump for CEM and random methods at around 1000 steps, which means the environment layout in this episode is difficult. Compared to CEM and random methods, which fail to explore the environment safely, our RCE approach is able to achieve zero constraint violation.

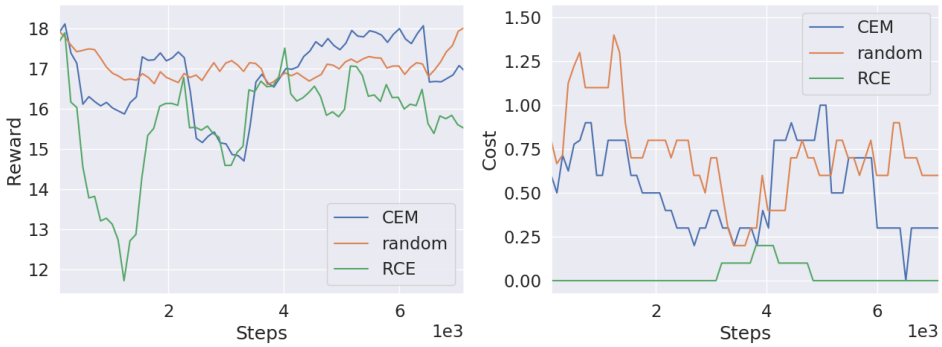


Figure 5: Testing curves of reward and cost with fixed learned models.

Uncertainty-aware dynamics model selection. The dynamics model module in this paper can be replaced by any other uncertainty-aware models in principle, and there are some alternatives other than the ensemble method, such as the dropout-based approximate Bayesian inference method (Gal & Ghahramani, 2016). We implemented and tested the dropout-based method but found that the prediction performance is worse than the ensemble method, so we use the ensemble method in our cases.