Experiment Setting We implement our algorithm to solve **MuJoCo's Half-cheetah** simulation. This setting is more realistic than our toy setting and aims at learning how to control a simplified cross-section of a cheetah robot. The setting uses continuous state-action spaces, which further increases the complexity of the problem. The actions here are the torques to apply at each joint of the model while the state contains the information on the speed along which the robot is heading, rotational speed, and momenta. The goal of the simulation is to make the robot run as fast as possible to the right. We solve the problem with our proposed algorithm as well as TRC and PbOP as baselines.

Training Setting We run both policies in the MuJoCo setting until 1,000 timesteps have passed. We repeat this for 100 episodes, saving the interactions and final preference based on the cumulative reward after each episode. Finally, we use the data to perform gradient descent for a pre-defined number of repetitions. We repeat the cycle of interaction and training 100 times. In total, the algorithm sees 10,000,000 timesteps and 10,000 preference reward signals.

RA-PbRL Implementation For our implementation of RA-PbRL, we iterate through the trajectories, using an initial state, action taken, and outcome state. We perform stochastic gradient descent to find the best vectors that parameterize the transition and reward functions with mean-squared error. After obtaining the best transition and reward functions, we use their parameterization to create a parameterization of the risk-aware value function of our proposed algorithm. For this, we simply concatenate the vectors parameterizing the transition and reward functions alongside a vector parameterizing the policies. The policy vector used depends on the policy being optimized (the best or exploratory policies.) We then compute the α-CVaR over the trajectory preferences from the interaction. We finally optimize the value function parameterization using this data for training and perform stochastic gradient descent.

To follow the theoretical bounds we establish, we compute the distance between the initial transition and reward parameterizations before and after optimization was performed. If the distance is larger than what is established by the theoretical bounds, we undo the optimization, as the vectors have abandoned the confidence bound.

Result Figure 1 shows our algorithm improving faster than both baselines we use. We compute the CVaR regret by using the cumulative reward obtained by a deep-RL algorithm, mainly for comparison. These results meet the expectations create by the results from our toy example in the paper.

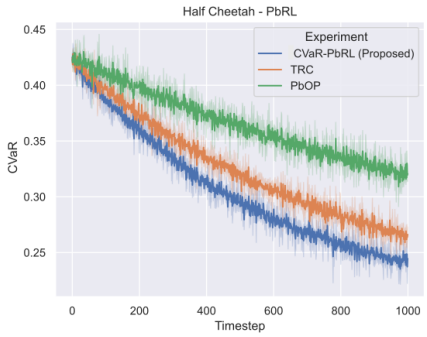

Figure 1: MuJoCo's Half-cheetah experiment with risk aversion set to $\alpha = 0.1$

.