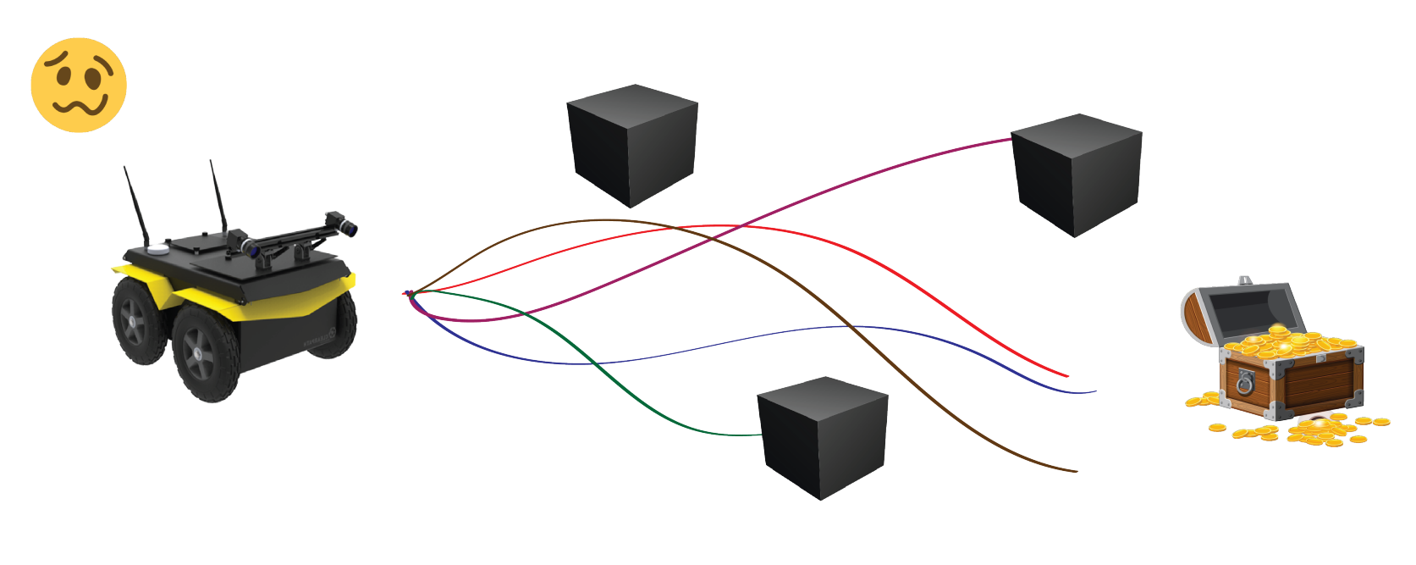
DiSProD

Differentiable Symbolic Propagation of Distributions for Planning

Palash Chatterjee, Ashutosh Chapagain, Weizhe Chen, Roni Khardon
Indiana University, Bloomington

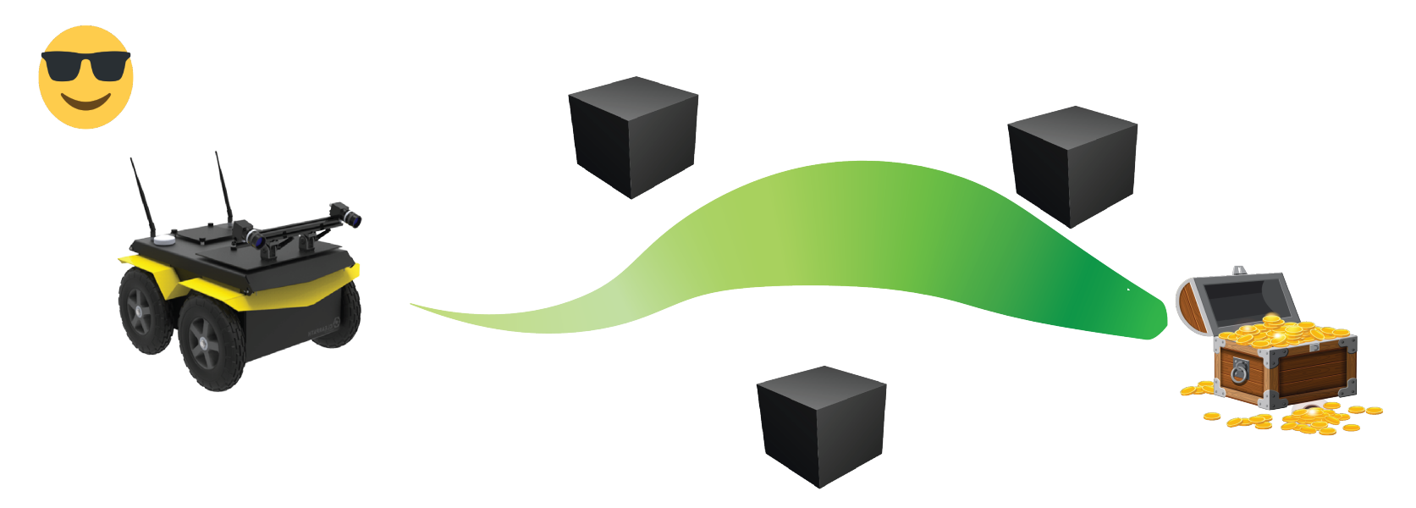
IJCAI 2023

I. Summary

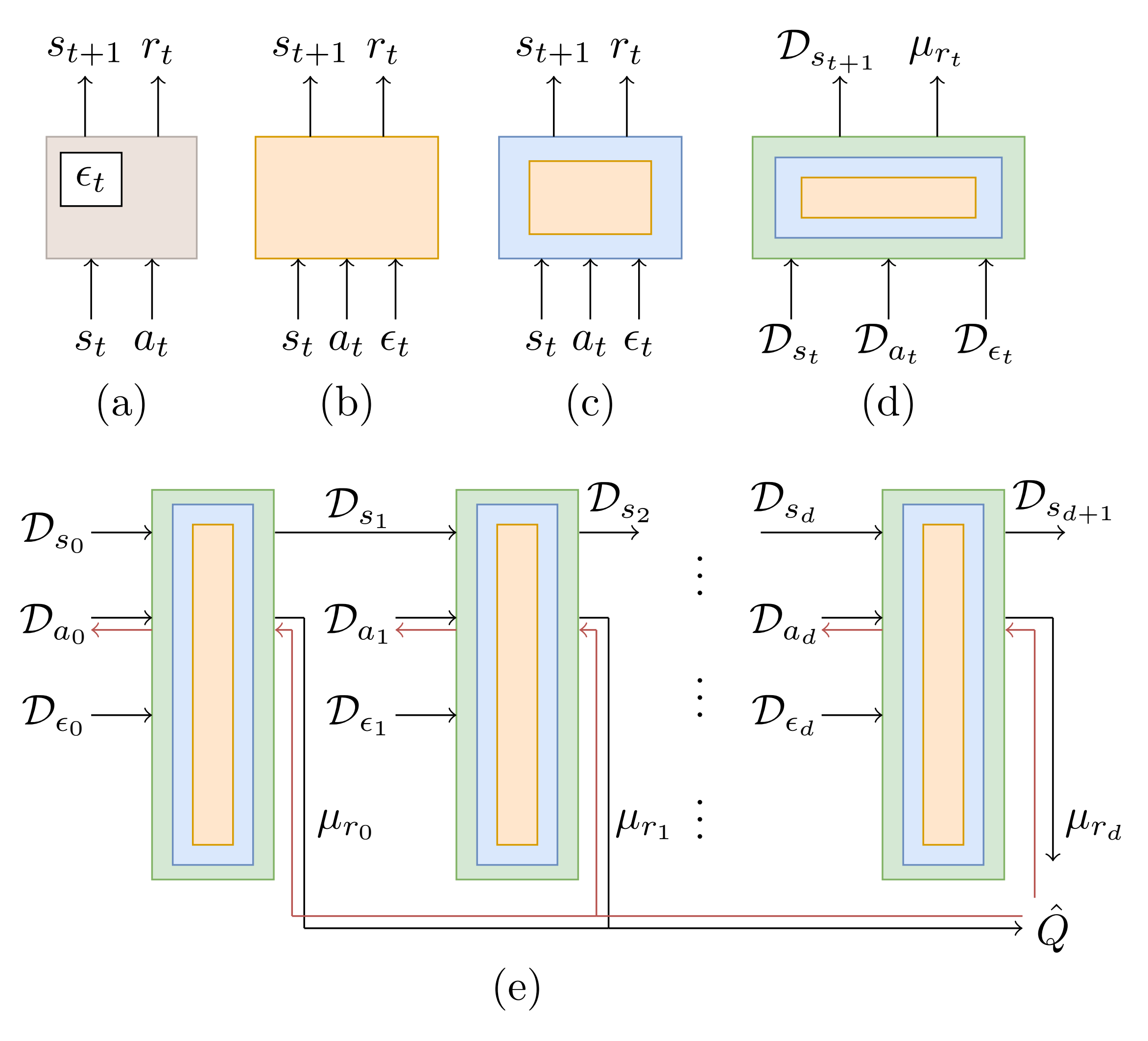


Motivation Planning in continuous/hybrid stochastic environments is challenging. Existing solutions use rollouts or make assumptions about the dynamics. Rollouts using the same $\{a_t\}$ can result in very different $\{s_{t+1}\}$ and lead to high variance estimates.

Contribution Construct a **symbolic computation graph** that captures the **distribution over future trajectories** allowing us to optimize the policy by performing **gradient search over the graph**. Use this in a robust search algorithm to choose the best action.



II. Method



1. Given a transition function **(a)**, first externalize the sampling of noise variables **(b)**.
2. Use Taylor's expansion for **(b)** to generate **(c)**. Till this point, all inputs and outputs are concrete variables.
3. They key idea is to combine **(c)** with propagation of distributions to yield **(d)** where the inputs and outputs are distributions over variables.
4. To obtain a computation graph, stack multiple such modules till desired depth **(e)**.

$$s_{t+1,j} \approx T_j(\hat{\boldsymbol{z}}_t) + \underbrace{\nabla T_j^\top (\boldsymbol{z}_t - \hat{\boldsymbol{z}}_t)}_{\text{Becomes zero}} + \tfrac{1}{2}(\boldsymbol{z}_t - \hat{\boldsymbol{z}}_t)^\top H_j (\boldsymbol{z}_t - \hat{\boldsymbol{z}}_t)$$

$$\mathbb{E}\left[s_{t+1,j}\right] \approx T_j(\hat{\boldsymbol{z}}_t) + \tfrac{1}{2}\left[\sum_k \left(\frac{\partial^2 T_j}{\partial s_{t,k}^2}\right)\hat{v}_{s,t,k} + \sum_\ell \left(\frac{\partial^2 T_j}{\partial a_{t,\ell}^2}\right)\hat{v}_{a,t,\ell} + \sum_i \left(\frac{\partial^2 T_j}{\partial \epsilon_{t,i}^2}\right)\hat{v}_{\epsilon,t,i}\right]$$

$$\mathrm{var}[s_{t+1,j}] \approx \sum_k \left(\frac{\partial T_j}{\partial s_{t,k}}\right)^2 \hat{v}_{s,t,k} + \sum_\ell \left(\frac{\partial T_j}{\partial a_{t,\ell}}\right)^2 \hat{v}_{a,t,\ell} + \sum_i \left(\frac{\partial T_j}{\partial \epsilon_{t,i}}\right)^2 \hat{v}_{\epsilon,t,i}$$

- Propagate distributions over states by computing the expectation and variance for each state variable, under an independence assumption.
- Optimize probabilistic policies by optimizing the expectation and variance for each action variable.

III. Algorithm

Online Planner At every timestep, optimize over a finite time horizon, but choose the first action from the solution.

Algorithm 1 One Step of DiSProD.

```
   Input: state
1: initialize actions (for all restarts)
2: build computation graph till depth $D$
3: while actions have not converged do
4:     loss = - $\sum_{\text{restart } k} \hat{\mathcal{Q}}(\boldsymbol{s_t}, \hat{\mu}^k_{\boldsymbol{a}}, \hat{v}^k_{\boldsymbol{a}})$
5:     loss.backward()
6:     actions ← safe-projected-gradient-update(actions)
7: save action-means $\hat{\mu}^{k^*}_{\boldsymbol{a}_{t+1:t+D}}$ from the best restart $k^*$
8: return action $\sim \mathcal{N}(\hat{\mu}^{k^*}_{\boldsymbol{a}_t}, \hat{v}^{k^*}_{\boldsymbol{a}_t})$
```

Initialization

$$\hat{\mu}_{a,\ell} \sim \mathcal{U}(a_{\ell,\min}, a_{\ell,\max})$$
$$\hat{v}_{a,\ell} = \min(a_{\ell,\max} - \hat{\mu}_{a,\ell}, \hat{\mu}_{a,\ell} - a_{\ell,\min})^2/12$$

Can use saved values from previous timestep.

Convergence Criterion Loose criterion meant to guide the search direction. Stop after 10 gradient updates or when $|\hat{\mu}_{\boldsymbol{a},t} - \hat{\mu}_{\boldsymbol{a},t-1}| \wedge |\hat{v}_{\boldsymbol{a},t} - \hat{v}_{\boldsymbol{a},t-1}| < 0.1$.

Safe Projected Gradient Descent Project the updated $\hat{\mu}_{\boldsymbol{a}}$ and $\hat{v}_{\boldsymbol{a}}$ into legal regions, and update only if $\mathcal{Q}$-value improves.

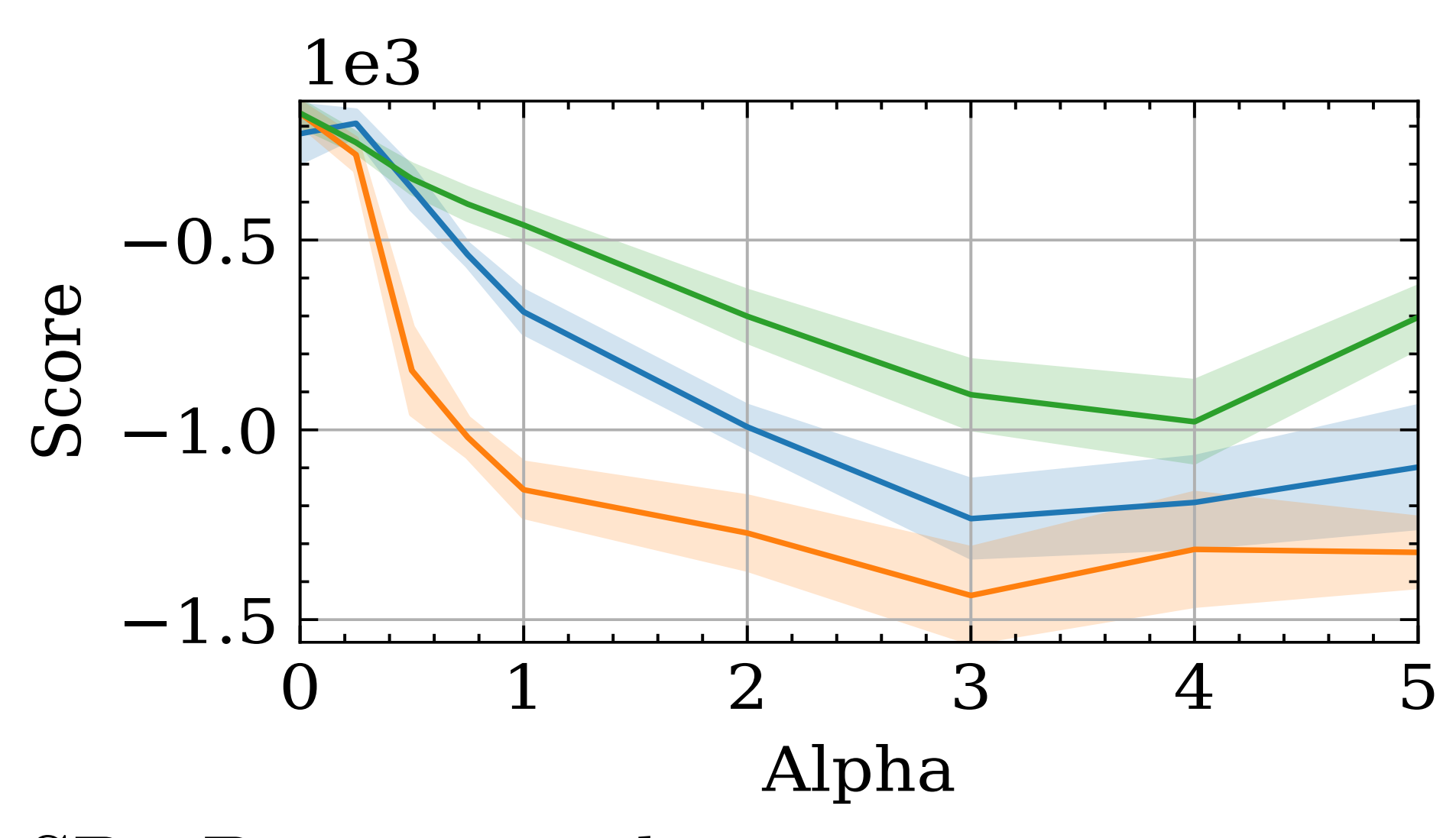
Best Restart Each random restart yields a computation graph approximating the $\mathcal{Q}$-value. Run multiple such restarts in parallel. Best restart (k^*) = Restart with highest $\mathcal{Q}$-value after convergence.

Action Selection Sample an action from the first action distribution $(\hat{\mu}^{k^*}_{\boldsymbol{a}_t}, \hat{v}^{k^*}_{\boldsymbol{a}_t})$.

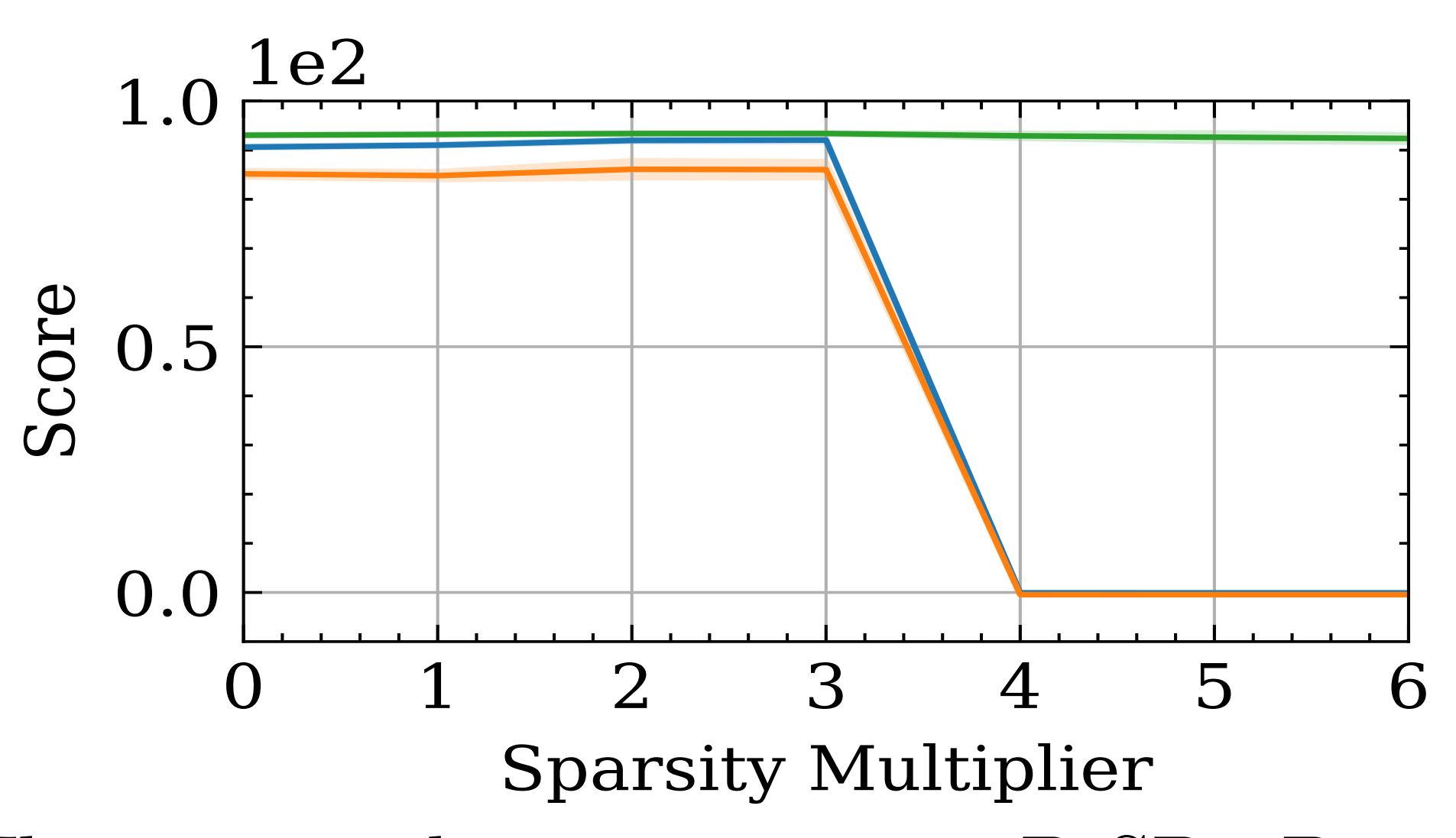
Saving Actions Save $(\hat{\mu}^{k^*}_{\boldsymbol{a}_{t+1:t+D}})$ to initialize one restart in the subsequent timestep.

IV. Experiments

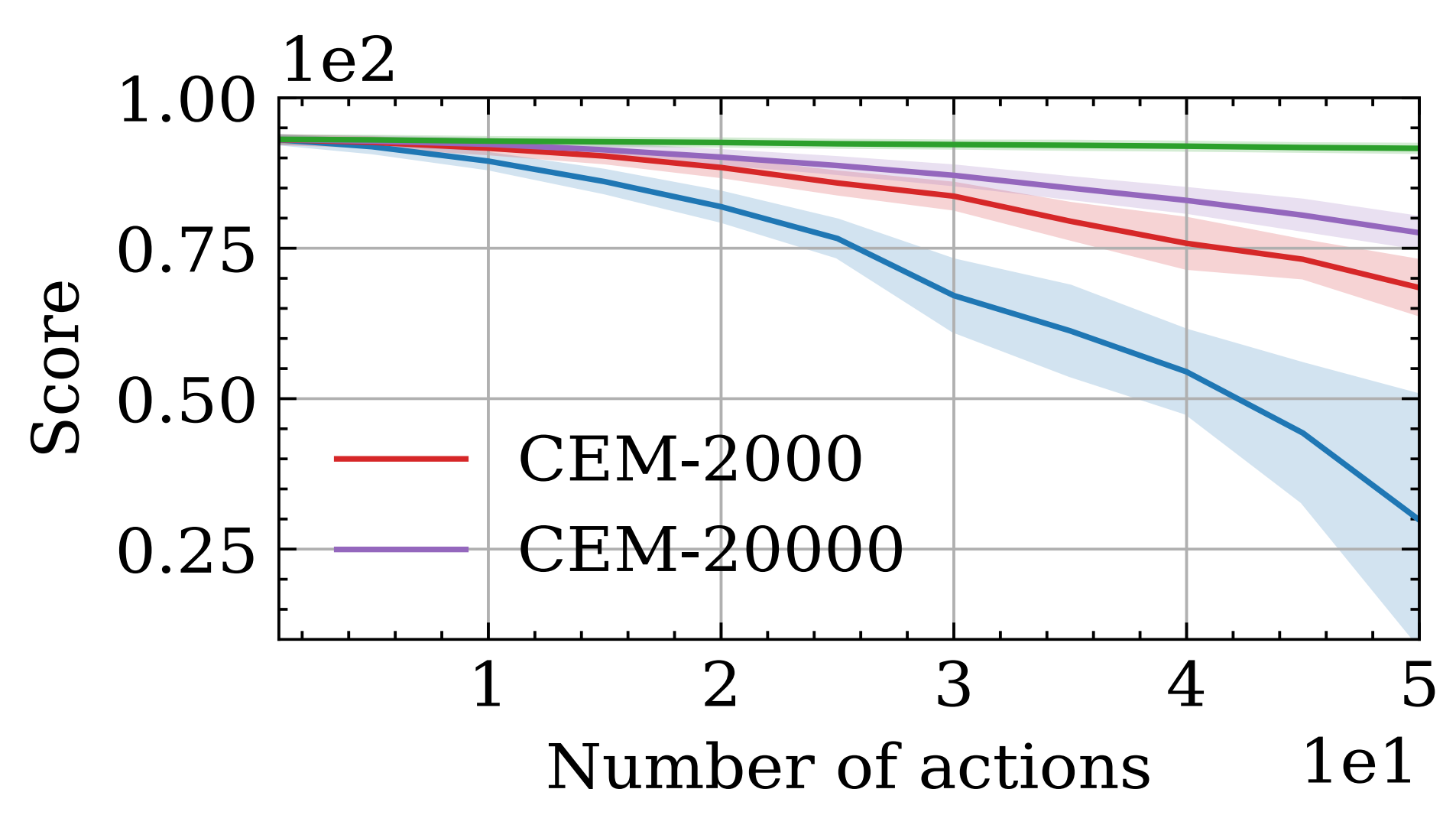
🟦 CEM 🟧 MPPI 🟩 DiSProD



DiSProD is more robust to increasing noise in environments than CEM/MPPI. (Env: Pendulum)

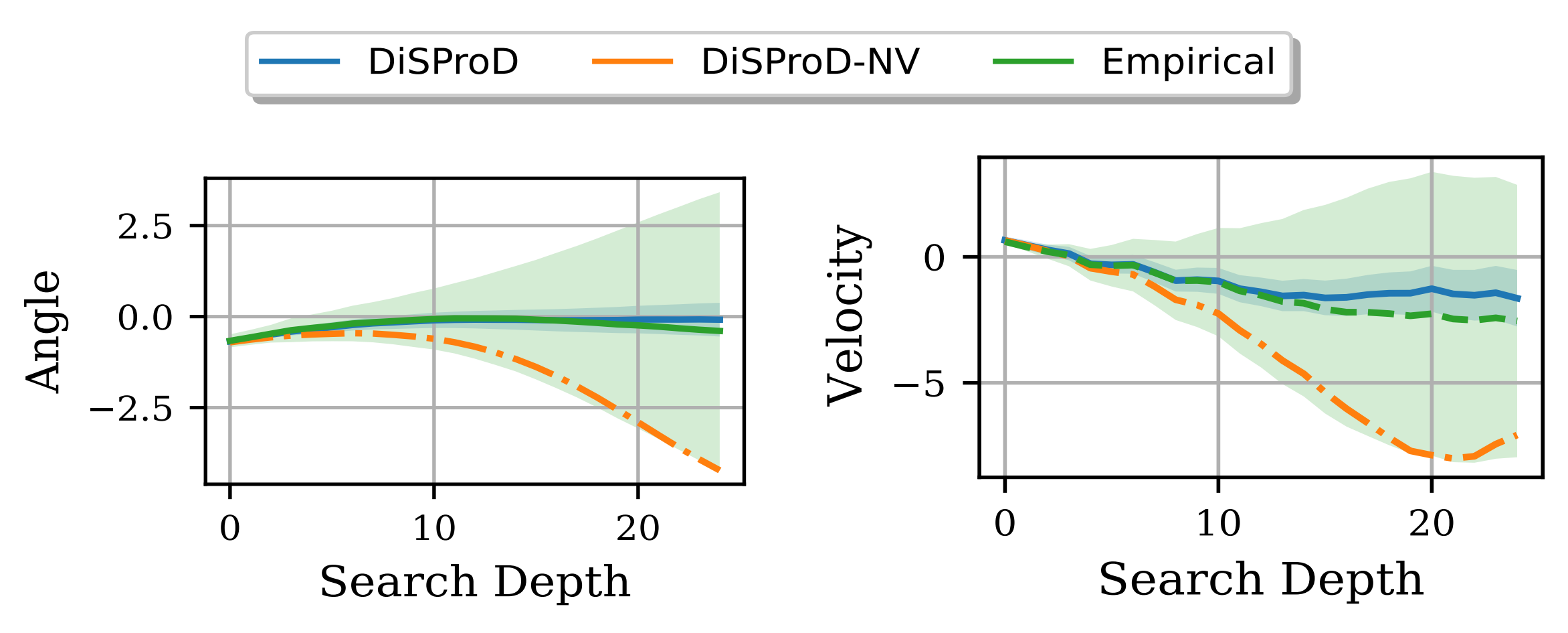


When rewards are sparse, DiSProD performs optimally by searching deep, while CEM/MPPI fail. (Env: Modified Mountain Car)



CEM requires 100x samples to achieve decent performance when action space increases, while DiSProD performs optimally without any extra resources. (Env: Modified Mountain Car)

DiSProD controls Jackal successfully despite the transition model ignoring physical properties (like friction, weight of car) and needing to deal with asynchronous execution of actions.



Using the variance terms is crucial to get a good approximation of the empirical trajectory distribution.

More experiments with CartPole, MountainCar, Pendulum and some modified versions of these (in Gym), Dubins' car model (in Gym and ROS) and two physical robots are in the paper.