Graph Sparse Sampling: Breaking the Curse of the Horizon in Continuous MDP Planning

Idan Lev-Yehudi [1] Vadim Indelman [2,3]

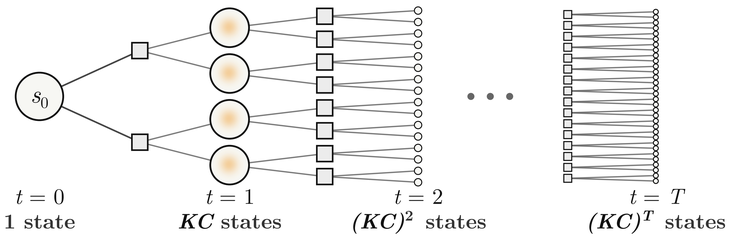
[1]Technion Autonomous Systems Program [2]Faculty of Aerospace Engineering [3]Faculty of Data and Decision Sciences
Technion – Israel Institute of Technology

GSS shares sampled states across actions instead of branching, for **long-horizon planning** in continuous MDPs on **GPUs**

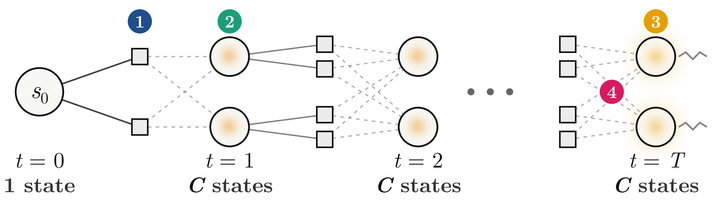
1. Graph Sparse Sampling: Constant Sampling Width

Full tree: $(KC)^T$ states vs. GSS: $C \cdot T$ states

Sparse Sampling (tree; Kearns et al., 2002)



Graph Sparse Sampling (GSS)



Sharing successor layers trades per-sample adaptivity for sampling throughput.

Graph Sparse Sampling Components

1. **Action sampling:** K candidate actions per state from a proposal q^a.
2. **State sampling:** one shared layer of C states from a proposal q^s.
3. **Tail value:** states at depth T are valued by a rollout policy.
4. **Graph backup:** each action value uses the whole next layer, via importance sampling, kernel density, nearest-neighbour or learned backups.

Graph Backup: Self-Normalized Importance Sampling (SNIS)

We analyze SNIS backups, whose normalized weights make the guarantees tractable:

$$Q(s,a) = \frac{\sum_j \rho_j \left(r(s,a,s'_j) + \gamma V(s'_j) \right)}{\sum_j \rho_j}, \qquad \rho_j = \frac{p(s'_j \mid s,a)}{q^s(s'_j)}$$

All actions reweight the same sampled states, assuming proposals that cover the relevant actions and states.

2. Finite-Sample Guarantees

Theorem 1 (Graph recursive bounds). With high probability, estimated values $\hat{Q}$ stay close to the optimal Q^* at every layer, so the returned root action $\hat{a}$ is near-optimal:

$$Q^*(s_0,a^*) - Q^*(s_0,\hat{a}) \;\le\; 2\Big(\sum_{t<T} \gamma^t \big(\underbrace{\lambda_t}_{\text{backup}} + \underbrace{\Gamma_t}_{\text{actions}} \big) + \gamma^T \underbrace{\kappa_{\text{tail}}}_{\text{tail}} \Big)$$

Errors accumulate layer by layer, discounted by γ.

Polynomial Horizon Dependence (Corollary 4.3)

For finite actions ($\Gamma_t = 0$) and bounded overlap $d_\infty = \sup p/q^s$, the loss above reaches ϵ with high probability using

$$N_{\text{GSS}} = \widetilde{O}\left(\frac{R_{\max}^2 d_\infty^2 T^5}{\epsilon^2} \right)$$

simulator draws, whereas Sparse Sampling needs a budget exponential in T. If $d_\infty = O(T^n)$, the bound remains polynomial in T.

Extensions Beyond Full-Rank Transitions and Finite Actions

Each extension changes one term of Theorem 1.

Low-dimensional manifolds (Proposition 1). Noise of lower dimension than the state leaves the transition without a density; smoothing with bandwidth τ restores one:

$$\lambda_t \;\to\; \lambda_t + L\, m_\beta(\tau), \qquad m_\beta(\tau) \xrightarrow{\tau \to 0} 0$$

Continuous action spaces (Proposition 2). K actions from a proposal with mass $m(\varepsilon)$ within ε of an optimal action, Q Hölder in the action:

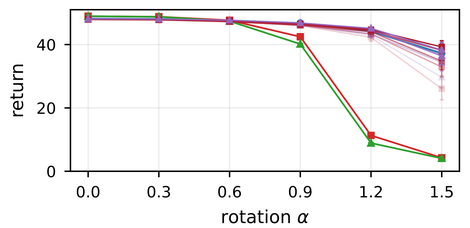
$$\Gamma_t \le L\, \varepsilon^\beta \quad \text{with probability} \;\ge 1 - \big(1 - m(\varepsilon)\big)^K$$

Paper (arXiv)

Code (GitHub)

3. Results on Continuous Control

GSS is compared with Monte Carlo tree search (MCTS) planners, Double Progressive Widening (DPW; Coüetoux et al., 2011) and Voronoi Progressive Widening (VPW; Lim et al., 2021), and with analytic controllers. At equal or longer planning time, **GSS samples more than $6\times$ as many states per decision**.

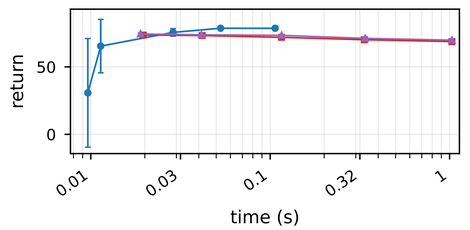


Rotating Double Integrator, 8D

GSS LQR LQG DPW VPW

- Rotation α moves the optimal path away from the analytic LQR and LQG solutions
- DPW, VPW: budgets up to 1 s per decision; GSS: at most 0.7 s

GSS stays consistent as the optimal path diverges from the analytic solutions.

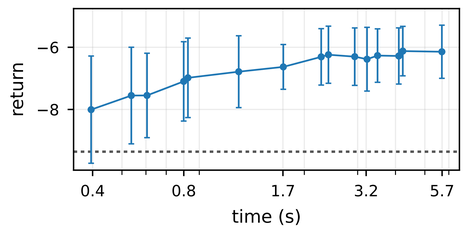


Lunar Lander, 6D

GSS DPW VPW

- Singular, nonlinear dynamics
- One shared rollout policy

GSS reaches 78.6 within 0.05 s and stays consistent; best MCTS: 74.4, then degrades.



Reacher, 10D

GSS PD controller

- PD controller is the base policy: GSS proposes its actions and rolls it out for tail values
- Graph shapes from 256 states × 24 actions to 2048 × 64, plotted by planning time

Every GSS size beats its PD base policy; return keeps climbing with planning time.