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

Idan Lev-Yehudi¹ Vadim Indelman^{2,3}

¹Technion Autonomous Systems Program (TASP), Technion – Israel Institute of Technology

²Stephen B. Klein Faculty of Aerospace Engineering, Technion – Israel Institute of Technology

³Faculty of Data and Decision Sciences, Technion – Israel Institute of Technology

idanlev@campus.technion.ac.il, vadim.indelman@technion.ac.il

Abstract

Planning under uncertainty in continuous domains is essential for autonomous systems, yet computationally demanding. Tree-based search methods such as Monte Carlo Tree Search (MCTS) remain popular, but their branching structure can require sampling budgets that grow exponentially with lookahead depth in the worst case. From a tree perspective, continuous state or action spaces become especially challenging, since the planner must decide where to search in an infinite branching hierarchy. We propose Graph Sparse Sampling (GSS), an online planning algorithm that shares sampled futures across many candidate decisions, rather than sampling separate successors for each candidate action. This branch-free graph exposes large GPU-friendly batches, while using heuristics to focus computation. We prove finite-sample performance guarantees for GSS covering full-rank or low-rank generative simulators via smoothed backups, and discrete or sampled continuous action spaces. Under suitable overlap, regularity, and action-coverage conditions, these bounds have polynomial dependence on the planning horizon, formalizing when shared futures can avoid the exponential horizon dependence of tree-shaped sparse sampling. We demonstrate continuous-control simulations where GSS substantially outperforms tree-based planners on long horizons or achieves near-optimal performance, supporting no-branching graph planning as a complementary design principle for online control.

1 Introduction

Planning under uncertainty is a fundamental problem in artificial intelligence, and continuous Markov Decision Processes (MDPs) model many physical systems. For discrete MDPs, exact policy computation is computationally constrained: finite-horizon, discounted, and average-cost variants are polynomial-time solvable but P-complete [Papadimitriou and Tsitsiklis, 1987]. Continuous MDPs add a difficulty, since neither the state space nor the action space can be enumerated directly. Sparse Sampling [Kearns et al., 2002] addresses large state spaces by using only a generative model to sample a sparse set of successor states at each state-action pair, yielding online planning bounds independent of the state dimension. However, these are exponential in the effective planning horizon, and worst-case tight when only a black-box simulator is available [Kearns et al., 2002].

Most practical continuous MDP planners build a search tree, and adapt how the tree is widened or how actions are selected. Double Progressive Widening (DPW) [Couëtoux et al., 2011] limits the number of state and action children in continuous Monte Carlo Tree Search; KR-UCT [Yee et al., 2016] shares value estimates between nearby continuous actions; Voronoi Optimistic Optimization (VOO) and VOOT [Kim et al., 2020] use Voronoi-based black-box optimization inside tree search; and

Code and data: <https://github.com/IdanLevYehudi/GraphSparseSampling>

VG-UCT [Lee et al., 2020] refines sampled actions using value gradients. Other non-tree approaches include Model Predictive Path Integral control (MPPI) [Williams et al., 2017], which optimizes sampled open-loop trajectories with importance-weighted path costs, and stochastic mesh method (SMM) [Broadie et al., 2004] and weighted stochastic mesh (WSM) [Belomestny et al., 2020, 2025], which reuse sampled states in problems such as high-dimensional option pricing and general Markov decision processes. These methods provide important ways to cope with continuous domains, but they either retain a tree-shaped planning object, optimize trajectories rather than shared Bellman layers, or are not designed around fixed-shape batched online control on GPUs.

We propose Graph Sparse Sampling (GSS), an online planner that replaces separate sampled subtrees with shared successor layers. Rather than drawing a different successor set for every state-action pair, GSS samples a common next-state layer and evaluates many candidate decisions against it. We design GSS to trade per-sample adaptivity for sampling throughput, so that GSS can scale to far larger sampling budgets in continuous MDPs. Our theoretical results formalize this intuition by showing that under suitable conditions, GSS can achieve polynomial error sample complexity in the planning horizon, while our experiments show it is effective in practice, reaching far higher sample counts than MCTS-based planners.

1.1 Main Contributions

Algorithmic contribution. We present Graph Sparse Sampling (GSS), a novel planning algorithm for continuous MDPs designed around state sample reuse in predictable shape operations. GSS is built for parallel processing on modern GPUs, and incorporates action and state sampling heuristics that focus budgets on more promising regions of the state and action spaces.

Theoretical contribution. We provide a general derivation of finite-time, high-probability guarantees of action-value estimates that hold jointly for the entire sampled planning graph. Under density-ratio overlap, backup-stability, and action-coverage conditions, comparable to those of previous results, these guarantees depend polynomially on the planning horizon. Furthermore, we show extensions of the bounds to continuous action spaces, and to low-rank transition models of black-box simulators, under suitable conditions.

Empirical contribution. We demonstrate GSS in three continuous-control benchmarks. We compare it to tree-based planners in similar or extended time budgets, and to closed-loop analytic baselines. GSS is competitive with or improves over tree-based planners in the tested regimes, often at smaller measured planning times, and scales to larger batched sample counts. Furthermore, we show that GSS scales to high-dimensional and long-horizon settings in a physical simulation.

1.2 Related Work

Tree-based online planning with guarantees. Sparse Sampling [Kearns et al., 2002] and UCT [Kocsis and Szepesvári, 2006] are classical tree-based baselines with finite-sample or asymptotic guarantees in discrete-action MDPs. Continuous-domain tree planners add mechanisms for action and state expansion, including DPW [Couëtoux et al., 2011, Auger et al., 2013], kernel regression [Yee et al., 2016], Voronoi optimization [Kim et al., 2020, Lim et al., 2021], and value gradients [Lee et al., 2020]. Non-asymptotic theory for MCTS has been developed for finite or discrete-action settings [Shah et al., 2022, Barenboim and Indelman, 2026], and in continuous domains, for tree-based optimistic partitioning methods [Mao et al., 2020]. These methods remain tree-shaped and do not analyze shared successor layers or SNIS-style graph backups.

Graph-based online planning and sample reuse. Leurent and Maillard [2020] consider Monte Carlo Graph Search (MCGS), but in discrete MDPs, for repeating state samples. Kujanpää et al. [2024] considered a graph-based planner for continuous MDPs, but without formal guarantees. Lev-Yehudi et al. [2025] make reuse of state sample values via IS in a tree setting. Stochastic Mesh Method (SMM) [Broadie et al., 2004] and Weighted Stochastic Mesh (WSM) [Belomestny et al., 2020, 2025] are the closest methods to GSS in the literature, and Belomestny et al. [2025] prove finite-time bounds for WSM with polynomial dependence on the horizon. SMM targets optimal stopping problems, whose decisions are binary, and WSM assumes discrete or discretized action spaces with computable transition densities, so neither method directly applies to the continuous-action, low-rank, or deterministic-transition domains we test in Section 5. Its compact-state tractability result relies on global density lower/upper bounds and regularity assumptions, whereas our SNIS analysis is stated in

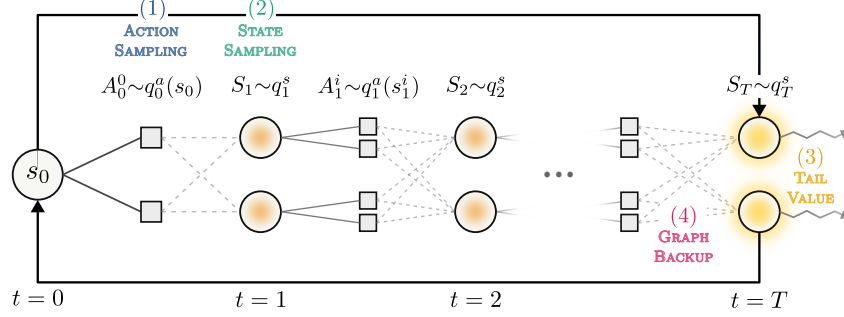


Figure 1: GSS builds a layered planning graph from the root state s_0 . Sampled states are shown as circles and actions as rectangles. In the forward pass, we draw candidate actions for each state using the (1) **action proposal**; the next shared state layer is then drawn by the (2) **state proposal**. At depth T , nodes are evaluated by a (3) **tail value**. The backward pass applies the (4) **graph backup** to each node-action pair, backpropagating the best action-value estimates at each node.

proposal-overlap terms through Rényi divergences, which can be considered as more general or not directly comparable. Furthermore, we incorporate search heuristics for action and state sampling, steering the algorithm towards closed-loop online planning on GPU hardware.

2 Background

Markov Decision Process (MDP): we consider MDPs in the form $\langle \mathcal{S}, \mathcal{A}, p, r, \gamma, L \rangle$. $\mathcal{S} \subseteq \mathbb{R}^{n_s}$, $\mathcal{A} \subseteq \mathbb{R}^{n_a}$ are the state and action spaces. We use primes for next-step variables: the transition density $p_t(s'|s, a)$ describes the next state after taking action $a \in \mathcal{A}$ at state $s \in \mathcal{S}$ at time t . We use the shorthand $s' \sim p_t(\cdot | s, a)$ to mean that s' is sampled from the distribution with this density, and use the same convention for other densities. The reward function $r_t(s, a, s') \in \mathbb{R}$ satisfies $|r_t(s, a, s')| \leq R_{\max}$, gives the immediate reward of transitioning from state s to s' by taking action a at time t , and the expected reward is $r_t(s, a) \triangleq \mathbb{E}_{s' \sim p_t(\cdot | s, a)}[r_t(s, a, s')]$. $\gamma \in (0, 1]$ is the discount factor. The MDP starts at time 0 and terminates after $L \in \mathbb{N} \cup \{\infty\}$ steps, so time indices satisfy $0 \leq t \leq L$, and if $L = \infty$ then we assume $\gamma < 1$. We consider (possibly time-dependent) deterministic policies $\pi = (\pi_t)_{t \in \mathbb{N}}$, i.e. $\pi_t : \mathcal{S} \rightarrow \mathcal{A}$, yet we note that our theoretical results can generalize to stochastic policies as well. The value function of a policy π at time t is $V_t^\pi(s_t) \triangleq \mathbb{E}^\pi[\sum_{i=t}^{L-1} \gamma^{i-t} r_i(s_i, \pi_i(s_i), s_{i+1}) | s_t]$, where $\mathbb{E}^\pi[\cdot | s_t]$ is taken over $s_{i+1} \sim p_i(\cdot | s_i, \pi_i(s_i))$ for $i \geq t$, and we use $m:n$ to denote the range $m, \dots, n \in \mathbb{Z}$. For any mapping $U : \mathcal{S} \rightarrow \mathbb{R}$, define the single-sample Bellman target $G_t^U(s, a, s') \triangleq r_t(s, a, s') + \gamma U(s')$. We define the action-value function as $Q_t^\pi(s, a) \triangleq \mathbb{E}_{s' \sim p_t(\cdot | s, a)}[G_t^{V_t^\pi+1}(s, a, s')]$. The optimal value and action-value functions are $V_t^*(s) \triangleq \sup_\pi V_t^\pi(s)$ and $Q_t^*(s, a) \triangleq \mathbb{E}_{s' \sim p_t(\cdot | s, a)}[G_t^{V_t^*+1}(s, a, s')]$ respectively, and we denote $G_t^*(s, a, s') \triangleq G_t^{V_t^*+1}(s, a, s')$. We write $\text{Argmax}_t(s) \triangleq \arg \max_{a \in \mathcal{A}} Q_t^*(s, a)$ for the exact optimal-action set. Our planning goal is to return a near-optimal root action at s_0 .

Importance Sampling (IS): IS [Kloek and Van Dijk, 1978] is a technique in Monte Carlo (MC) estimation where a proposal distribution q is used to generate samples. We use $\hat{\square}$ for empirical estimators and reserve $\tilde{\square}$ for approximate theoretical quantities. The IS estimator for $g(x) = \mathbb{E}_{x \sim p}[f(x)]$ is $\hat{g}_{q, \text{IS}} = N^{-1} \sum_{i=1}^N \rho_q^p(x^i) f(x^i)$, for importance ratios $\rho_q^p(x) \triangleq p(x)/q(x)$. $\hat{g}_{q, \text{IS}}$ is unbiased if $q(x) = 0$ implies $p(x) = 0$. Often, the self-normalized IS (SNIS) estimator is formulated when we only have access to an unnormalized version of p , or for variance reduction purposes, and is given by $\hat{g}_{q, \text{SNIS}} = (\sum_{i=1}^N \rho_q^p(x^i))^{-1} \sum_{i=1}^N \rho_q^p(x^i) f(x^i)$. While biased, under weak assumptions it is consistent [Owen, 2013, 9.2].

Monte Carlo Tree Search (MCTS): MCTS is an algorithm used to quickly explore large state spaces [Browne et al., 2012]. It iteratively repeats four steps to build a search tree that approximates the action-values, using a best-first strategy. Often UCT [Kocsis and Szepesvári, 2006] is used to balance between exploration of new actions and exploitation of promising ones. Double Progressive

Algorithm 1 Graph Sparse Sampling

```

1: Set  $s_0^1 \leftarrow s_0$  and  $S_0 \leftarrow \{s_0^1\}$ .
2: procedure EVALUATELAYER( $t, S_t$ )
3:   if  $t = T$ , set  $\hat{V}_T^j \leftarrow \text{TailValue}_T(s_T^j)$  for all  $j \in 1:C_T$  and return
4:   For all  $i \in 1:C_t$ , draw  $a_t^{i,1:K_t} \sim q_t^a(\cdot \mid s_t^i)$ , set  $A_t \leftarrow \cup_{i \in 1:C_t} \{a_t^{i,1:K_t}\}$ 
5:   Fit  $q_t^s \leftarrow \text{FitProposal}_t(t, S_{0:t}, A_{0:t})$  on all available information
6:   Draw  $s_{t+1}^{1:C_{t+1}} \sim q_t^s$  and set  $S_{t+1} \leftarrow \{s_{t+1}^{1:C_{t+1}}\}$ 
7:   EVALUATELAYER( $t+1, S_{t+1}$ )
8:   For all  $i \in 1:C_t$  and  $k \in 1:K_t$ , set  $\hat{Q}_t^{i,k} \leftarrow \mathbf{B}_t(s_t^i, a_t^{i,k}; S_{t+1}, \hat{V}_{t+1}^{1:C_{t+1}}, q_t^s)$ 
9:   For all  $i \in 1:C_t$ , set  $\hat{V}_t^i \leftarrow \max_{k \in 1:K_t} \hat{Q}_t^{i,k}$ 
10:  if  $t = 0$  return  $a_0^{1,\hat{k}_0}$  for  $\hat{k}_0 \in \arg \max_{k \in 1:K_0} \hat{Q}_0^{1,k}$ 
11: end procedure
12: return EVALUATELAYER( $0, S_0$ ).

```

Widening (DPW) [Couëtoux et al., 2011] is a technique to limit the branching factor from being infinite in continuous settings. The number of children of a node is artificially limited to kN^α for N visitations, for fixed $k > 0$ and $0 < \alpha < 1$. VOO samples continuous actions by implicitly using Voronoi cells around evaluated actions to balance local and global search [Kim et al., 2020]. VPW uses VOO when progressive widening adds new actions, extending it to stochastic tree search in continuous or hybrid domains [Lim et al., 2021].

3 Graph Sparse Sampling (GSS)

3.1 Layered Sparse Graph Structure

GSS is a finite-lookahead planner from a root state $s_0 \in \mathcal{S}$, attempting to estimate $\pi_0^*(s_0)$. GSS operates by drawing a known number of state and action samples from time 0 until its terminal depth $T \leq L$. We design GSS around decoupling successor-state sampling from action-value estimation, separated into forward pass and backward pass procedures respectively. This decoupling allows GSS to achieve large, regular batches of operations that can be parallelized efficiently on modern GPUs. Algorithm 1 shows pseudocode for GSS, in which each line is a batched operation.

Forward pass. The forward pass alternates between action sampling and state sampling. Starting from the initial state s_0 , let $C_{0:T} \in \mathbb{N}^{T+1}$ be the state layer widths, denoting the i 'th state layer as $S_t = \{s_t^{1:C_t}\}$. Each state node s_t has an action-sampling budget of K_t candidate actions, resulting in a total of $C_t K_t$ candidate actions at depth t .

For action sampling, GSS uses an **action proposal** density over $\mathcal{A}$, denoted by $q_t^a(\cdot \mid s)$, in each state s_t^i to draw $a_t^{i,1:K_t} \stackrel{\text{i.i.d.}}{\sim} q_t^a(\cdot \mid s_t^i)$. For state sampling, a **state proposal** routine chooses a density $q_t^s \leftarrow \text{FitProposal}_t(t, S_{0:t}, A_{0:t})$ on $\mathcal{S}$, and GSS samples $s_{t+1}^{1:C_{t+1}} \stackrel{\text{i.i.d.}}{\sim} q_t^s$. We denote the sets $A_t^i \triangleq \{a_t^{i,1:K_t}\}$, $A_t \triangleq \cup_{i \in 1:C_t} A_t^i$, and $S_t = \{s_t^{1:C_t}\}$. At the terminal layer, GSS initializes $\hat{V}_T^j \triangleq \text{TailValue}_T(s_T^j)$ for $j \in 1:C_T$. The components q_t^a , FitProposal_t and TailValue_T may be tuned to incorporate domain heuristics, including rollout-based or learned heuristics.

Backward pass. The backward pass proceeds from $t = T - 1$ down to $t = 0$, alternating between a **graph backup** and a max-reduction at each layer. **Graph backup** sets the action-value estimates for each $i \in 1:C_t$ and $k \in 1:K_t$ via ${}^1\hat{Q}_t^{i,k} \triangleq \mathbf{B}_t(s_t^i, a_t^{i,k}; S_{t+1}, \hat{V}_{t+1}^{1:C_{t+1}}, q_t^s) \in \mathbb{R}$, given the next state layer S_{t+1} , its value estimates $\hat{V}_{t+1}^{1:C_{t+1}}$, and the state proposal density q_t^s . The node value estimate is the maximum over sampled actions, i.e. $\hat{V}_t^i \triangleq \max_{k \in 1:K_t} \hat{Q}_t^{i,k}$. At the root, the returned action is $a_0^{1,\hat{k}_0}$ where $\hat{k}_0 \in \arg \max_{k \in 1:K_0} \hat{Q}_0^{1,k}$, which is then executed by the online planner before the next planning episode. We denote the short-hand Bellman target estimates, $\hat{G}_t^{i,k,j} \triangleq r_t(s_t^i, a_t^{i,k}, s_{t+1}^j) + \gamma \hat{V}_{t+1}^j$, and give example backups:

¹While we could backup $\hat{Q}_t^{i,k}$ over a sparse subset $E_t^{i,k} \subseteq S_{t+1}$ to reduce the time complexity below $O(C_t \cdot A_t \cdot C_{t+1})$, we find that the dense assignment $E_t^{i,k} = S_{t+1}$ is both simpler in theory and runs faster for simple GPU implementations.

$$\hat{Q}_{t,\text{SNIS}}^{i,k} = \eta_\rho^{-1} \sum_{j=1}^{C_{t+1}} \rho_t^{i,k}(j) \hat{G}_t^{i,k,j} \quad (1.A) \quad \hat{Q}_{t,\text{KDE}}^{i,k} = \frac{\sum_{j=1}^{C_{t+1}} w_{t,h}^{i,k}(j) \hat{G}_t^{i,k,j}}{\sum_{j=1}^{C_{t+1}} w_{t,h}^{i,k}(j)} \quad (1.B)$$

$$\hat{Q}_{t,\text{NN}}^{i,k} = r_t(s_t^i, a_t^{i,k}, \tilde{s}_{t+1}^{i,k}) + \gamma \hat{V}_{t+1}^{j_{\text{NN}}} \quad (1.C) \quad \hat{Q}_{t,\text{learn}}^{i,k} = F_{t,\phi}(s_t^i, a_t^{i,k}; \hat{V}_{t+1}^{1:C_{t+1}}, \mathcal{D}_t) \quad (1.D)$$

For SNIS (1.A), $\rho_t^{i,k}(j) \triangleq p_t(s_{t+1}^j | s_t^i, a_t^{i,k})/q_t^s(s_{t+1}^j)$ and $\eta_\rho = \sum_{j=1}^{C_{t+1}} \rho_t^{i,k}(j)$. When exact transition densities are available, the SNIS backup (1.A) is a convenient choice for GSS. As we show in Section 4, it allows to derive finite-time performance guarantees with a polynomial error sample complexity in the horizon in Corollary 4.3, and it is the basis of the backups in our experiments.

When importance weights are unavailable or impractical to use online, one can use simulator draws in several ways. KDE approximation (1.B) [Wand and Jones, 1994] draws $x_{t+1}^{i,k,1:M} \sim p_t(\cdot | s_t^i, a_t^{i,k})$ and forms $\hat{p}_{t,h}^{i,k}(s) \triangleq M^{-1} \sum_{\ell=1}^M K_h(s - x_{t+1}^{i,k,\ell})$; in (1.B), $w_{t,h}^{i,k}(j) \triangleq \hat{p}_{t,h}^{i,k}(s_{t+1}^j)/q_t^s(s_{t+1}^j)$ are the resulting SNIS weights. Nearest-neighbor approximations (1.C) avoid density estimation completely by drawing $\tilde{s}_{t+1}^{i,k} \sim p_t(\cdot | s_t^i, a_t^{i,k})$ and setting $j_{\text{NN}} \in \operatorname{argmin}_{j \in 1:C_{t+1}} d(\tilde{s}_{t+1}^{i,k}, s_{t+1}^j)$. In general, any learned approximation may be used as in (1.D), conditioning the action-value estimates with offline and online available data $\mathcal{D}_t$.

3.2 Smoothed SNIS Graph Backups for Low-Rank Generative Simulators

In many continuous simulators, the transition model is only given by the generative process

$$s' = f_t(s, a, \xi), \quad \xi \sim p_{\xi,t}(\cdot | s, a), \quad (2)$$

with $\xi \in \mathbb{R}^{n_\xi}$ and $s, s' \in \mathbb{R}^{n_s}$. When $n_\xi < n_s$, i.e. the latent noise dimension is smaller than the state dimension, the map $\xi \mapsto f_t(s, a, \xi)$ produces a transition model that is generally singular w.r.t. Lebesgue measure, making the importance ratio $p_t(\cdot | s, a)/q_t^s$ not well-defined. Kernel smoothing [Wand and Jones, 1994] addresses this by convolving the target transition density with a kernel density in $\mathcal{S}$, obtaining a density that is nondegenerate w.r.t. Lebesgue measure:

$$p_{t,\tau_t}(x | s, a) \triangleq \int k_{t,\tau_t}(x | f_t(s, a, \xi)) p_{\xi,t}(d\xi | s, a), \quad (3)$$

where $k_{t,\tau_t}(x | y)$ is a smoothing density on $\mathcal{S}$ with bandwidth parameter $\tau_t > 0$, measurable in y , that is locally concentrated around y as $\tau_t \rightarrow 0$. We refer to smoothed SNIS backups as those that use the SNIS formula (1.A) with p_{t,τ_t} in place of p_t in the importance ratio.

We can control the incurred bias by tuning the bandwidth τ_t , as shown in the following proposition.

Proposition 1 (Smoothing bias). *Assume $G_t^*(s, a, \cdot) : \mathcal{S} \rightarrow \mathbb{R}$ is measurable for $(s, a) \in \mathcal{S} \times \mathcal{A}$, and β_t^s -Hölder with constant L_t^s , i.e. $|G_t^*(s, a, x) - G_t^*(s, a, y)| \leq L_t^s d(x, y)^{\beta_t^s}$ for all $x, y \in \mathcal{S}$. Then:*

$$|\mathbb{E}_{s' \sim p_t(\cdot | s, a)}[G_t^*(s, a, s')] - \mathbb{E}_{x \sim p_{t,\tau_t}(\cdot | s, a)}[G_t^*(s, a, x)]| \leq L_t^s m_{\beta_t^s, t}(\tau_t), \quad (4)$$

$$\text{where} \quad m_{\beta_t^s, t}(\tau_t) \triangleq \sup_y \int_{\mathcal{S}} d(x, y)^{\beta_t^s} k_{t,\tau_t}(x | y) dx. \quad (5)$$

The proof is in Appendix A.1. The integral (3) is often unavailable in closed form, but can be approximated by various methods. In our implementations, we evaluate the linearized approximation around a chosen noise point $\bar{\xi}$ (often the mean of $p_{\xi,t}$):

$$\tilde{p}_{t,\tau_t}^{\text{lin}}(x | s, a) \triangleq \int k_{t,\tau_t}(x | f_t(s, a, \bar{\xi}) + J_{\xi,t}^{s,a}(\xi - \bar{\xi})) p_{\xi,t}(d\xi | s, a), \quad J_{\xi,t}^{s,a} \triangleq D_\xi f_t(s, a, \bar{\xi}). \quad (6)$$

When both $p_{\xi,t} \sim \mathcal{N}(\mu_{\xi,t}, \Sigma_{\xi,t})$ and $k_{t,\tau_t} \sim \mathcal{N}(0, \Sigma_{k,t})$ are Gaussian, this approximation is Gaussian with mean $f_t(s, a, \bar{\xi}) + J_{\xi,t}^{s,a}(\mu_{\xi,t} - \bar{\xi})$ and covariance $J_{\xi,t}^{s,a} \Sigma_{\xi,t} (J_{\xi,t}^{s,a})^\top + \Sigma_{k,t}$, which yields a closed-form density evaluation. Other approximations such as sigma-point transformations [Julier et al., 2004] or Monte Carlo density estimates [Owen, 2013] may be used in various settings.

4 GSS Theoretical Analysis

4.1 Settings and Assumptions

We first isolate the action-side approximation introduced by replacing the full action space by nodewise sampled action sets.

Definition 1 (Max-close action sets). For each depth $t < T$ and realized node s_t^i , we say that the action set A_t^i is *max-close* with radius $\Gamma_t \geq 0$ and local failure probability $\delta_t^a \in [0, 1]$ if

$$0 \leq V_t^*(s_t^i) - \max_{a \in A_t^i} Q_t^*(s_t^i, a) \leq \Gamma_t \quad (7)$$

except with probability at most δ_t^a . If $\mathcal{A}$ is finite and $A_t^i = \mathcal{A}$, then (7) holds with $\Gamma_t = 0$ and $\delta_t^a = 0$.

When $\mathcal{A}$ is metric and the action sets A_t^i are sampled from q_t^a , the following assumptions yield explicit Γ_t and δ_t^a .

Assumption 1 (Proposal small-ball coverage). For each depth $t < T$, there is a radius cap $\bar{\varepsilon}_t > 0$ such that for every $\varepsilon \in (0, \bar{\varepsilon}_t]$ and every $s \in \mathcal{S}$, there exists $a^* \in \text{Argmax}_t(s)$ such that $q_t^a(\{a \in \mathcal{A} : d_{\mathcal{A}}(a, a^*) < \varepsilon\} | s) \geq m_t(\varepsilon)$ with $m_t(\varepsilon) > 0$.

Assumption 2 (Action-side Hölder regularity). For each depth $t < T$, there are constants $L_t^a \geq 0$ and $\beta_t^a \in (0, 1]$ such that $|Q_t^*(s, a) - Q_t^*(s, a')| \leq L_t^a d_{\mathcal{A}}(a, a')^{\beta_t^a}$ for every $s \in \mathcal{S}$ and all $a, a' \in \mathcal{A}$.

Proposition 2 (One-step sampled-action loss). Fix $\varepsilon_t \in (0, \bar{\varepsilon}_t]$ and a node index $i \in 1:C_t$, and suppose that the candidate action set $A_t^i = \{a_t^{i,1:K_t}\}$ is sampled conditionally i.i.d. from $q_t^a(\cdot | s_t^i)$. Under Assumption 1 and Assumption 2, with probability at least $1 - (1 - m_t(\varepsilon_t))^{K_t}$, the one-step loss from maximizing only over the sampled actions satisfies

$$0 \leq V_t^*(s_t^i) - \max_{a \in A_t^i} Q_t^*(s_t^i, a) \leq L_t^a \varepsilon_t^{\beta_t^a} \quad (8)$$

The proof is in Appendix A.2. Under Proposition 2, the sampled action sets are max-close in the sense of Definition 1, with $\Gamma_t = L_t^a \varepsilon_t^{\beta_t^a}$ and $\delta_t^a = (1 - m_t(\varepsilon_t))^{K_t}$.

For the following **graph backup** assumptions, we assume that they hold true for all sampled-graph index pairs $i \in 1:C_t$ and $k \in 1:K_t$. We write $V_{t+1}^{*,1:C_{t+1}} \triangleq (V_{t+1}^{*,j}(s_{t+1}^j))_{j \in 1:C_{t+1}}$ for the optimal-value vector at the next layer, and $\hat{V}_{t+1}^{1:C_{t+1}}$ for the realized next-layer value vector. For the **graph backup** of an arbitrary vector $\mathbf{v} \in \mathbb{R}^{C_{t+1}}$, we shorthand $\mathcal{B}_t^{i,k}[\mathbf{v}] \triangleq \mathbf{B}_t(s_t^i, a_t^{i,k}; S_{t+1}, \mathbf{v}, q_t^s)$. For action-value estimates we denote $Q_t^{*,i,k} \triangleq Q_t^*(s_t^i, a_t^{i,k})$, $\hat{Q}_t^{i,k,*} \triangleq \mathcal{B}_t^{i,k}[V_{t+1}^{*,1:C_{t+1}}]$, and $\hat{Q}_t^{i,k} \triangleq \mathcal{B}_t^{i,k}[\hat{V}_{t+1}^{1:C_{t+1}}]$ respectively for the exact optimal action-value at $(s_t^i, a_t^{i,k})$, the graph backup output under exact next-layer optimal values, and the realized recursive backup output.

Assumption 3 (Graph backup bound). For every $i \in 1:C_t$, $k \in 1:K_t$, and $\lambda_t > 0$, the function $\delta_{\text{loc},t}$ bounds the failure probability: $\mathbb{P}(|\hat{Q}_t^{i,k,*} - Q_t^{*,i,k}| \leq \lambda_t) \geq 1 - \delta_{\text{loc},t}(\lambda_t, C_{t+1})$.

Assumption 4 (Graph backup stability). For every $i \in 1:C_t$, $k \in 1:K_t$, $\alpha \geq 0$, and $\mathbf{v} \in \mathbb{R}^{C_{t+1}}$, if $\|\mathbf{v} - V_{t+1}^{*,1:C_{t+1}}\|_{\infty} \leq \alpha$, then $|\mathcal{B}_t^{i,k}[\mathbf{v}] - \hat{Q}_t^{i,k,*}| \leq \gamma\alpha$.

Definition 2 (Controlled graph backup). $\mathbf{B}_t$ is *controlled* if it satisfies Assumptions 3 and 4.

For a controlled **graph backup**, the following bound holds:

Proposition 3 (Controlled graph backup bound). On the event $\|\hat{V}_{t+1}^{1:C_{t+1}} - V_{t+1}^{*,1:C_{t+1}}\|_{\infty} \leq \alpha_{t+1}$, under Assumption 3 and Assumption 4, with probability at least $1 - C_t K_t \delta_{\text{loc},t}(\lambda_t, C_{t+1})$, the recursive one-step graph backup error satisfies

$$|\hat{Q}_t^{i,k} - Q_t^{*,i,k}| \leq \gamma\alpha_{t+1} + \lambda_t \quad \text{for every } i \in 1:C_t \text{ and } k \in 1:K_t. \quad (9)$$

The proof is in Appendix A.3. For densities p and q , define the exponentiated Rényi divergences [Van Erven and Harremos, 2014] $d_{\infty}(p||q) = \text{ess sup}_{x \sim q} p(x)/q(x)$ and $d_2(p||q) \triangleq \mathbb{E}_{x \sim q}[(p(x)/q(x))^2]$. For example, under the following theorem's assumptions with a bounded d_{∞} divergence, the SNIS backup is a controlled graph backup in the sense of Definition 2, with $\lambda_t = \lambda$ and $\delta_{\text{loc},t} = 3 \exp(-N t(\lambda, N)^2)$:

Theorem (SNIS concentration, bounded d_{∞} Lim et al. [2023]). Let $d_{\infty}(p||q) < \infty$. Let $x_1, \dots, x_N \stackrel{\text{i.i.d.}}{\sim} q$ and let f satisfy $\|f\|_{\infty} \leq f_{\max}$. For any $\lambda > 0$ such that $\lambda > f_{\max} d_{\infty}(p||q)/\sqrt{N}$, define $t(\lambda, N) \triangleq \frac{\lambda}{f_{\max} d_{\infty}(p||q)} - \frac{1}{\sqrt{N}}$. Then

$$\mathbb{P}(|\hat{\mu}_{\text{SNIS},N} - \mathbb{E}_{x \sim p}[f(x)]| \leq \lambda) \geq 1 - 3 \exp(-N t(\lambda, N)^2). \quad (10)$$

Appendix A.3 shows a controlled median-of-means (MoM) SNIS **graph backup**, for a bounded d_2 assumption.

4.2 Graph Concentration Inequalities and Implications

Our main graph concentration inequality result turns single-node bounds into graph guarantees. We first combine the per-depth action and backup parameters into error and failure budgets.

Definition 3 (Graph error budgets). Fix a planning depth T , a tail accuracy $\kappa_T^{\text{tail}} \geq 0$ with per-node tail failure probability $\Delta_T^{\text{tail}} \in [0, 1]$, and, for each $t < T$, max-close parameters Γ_t, δ_t^a (Definition 1) and backup parameters $\lambda_t, \delta_{\text{loc},t}$ (Assumption 3). The per-depth error and failure budgets are $\kappa_t \triangleq \lambda_t + \Gamma_t$ and $\Delta_t \triangleq C_t K_t \delta_{\text{loc},t}(\lambda_t, C_{t+1}) + C_t \delta_t^a$, and the propagated value radii are $\alpha_T \triangleq \kappa_T^{\text{tail}}$ and $\alpha_t \triangleq \gamma \alpha_{t+1} + \kappa_t$ for $t = T-1, \dots, 0$.

Theorem 1 (Graph recursive concentration inequality). Fix finite realized candidate actions $a_t^{i,1:K_t}$ and the budgets of Definition 3. Assume that (i) for each terminal node $i \in 1:C_T$, $|\hat{V}_T^i - V_T^{*,i}| \leq \kappa_T^{\text{tail}}$ except with probability at most Δ_T^{tail} , where $V_T^{*,i} \triangleq V_T^*(s_T^i)$; (ii) for each $t < T$, the graph backup satisfies the conclusion of Proposition 3 with parameter λ_t , e.g., because it is controlled (Definition 2); and (iii) for each $t < T$ and $i \in 1:C_t$, the action set A_t^i is max-close with parameters Γ_t and δ_t^a . Then, with probability at least $1 - C_T \Delta_T^{\text{tail}} - \sum_{t=0}^{T-1} \Delta_t$, for every $t < T$:

$$\|\hat{Q}_t^{1:C_t, 1:K_t} - Q_t^{*, 1:C_t, 1:K_t}\|_\infty \leq \gamma \alpha_{t+1} + \lambda_t, \quad \|\hat{V}_t^{1:C_t} - V_t^{*, 1:C_t}\|_\infty \leq \alpha_t. \quad (11)$$

Moreover, for every $a_0^* \in \text{Argmax}_0(s_0)$, it holds that $Q_0^*(s_0, a_0^*) - Q_0^*(s_0, \hat{a}_0) \leq 2\alpha_0$.

The proof is given in Appendix A.4, and the proof strategy is similar to that of Lim et al. [2023].

We now show how Propositions 1 and 2 apply to Theorem 1.

Corollary 4.1 (Sampled-action graph budgets). In Definition 1, the fully enumerated case $A_t^i = \mathcal{A}$ gives $\Gamma_t = 0$ and $\delta_t^a = 0$, thus $\kappa_t = \lambda_t$ and $\Delta_t = C_t |\mathcal{A}| \delta_{\text{loc},t}(\lambda_t, C_{t+1})$. If instead $A_t^i = \{a_t^{i,1:K_t}\}$ is sampled as in Proposition 2, then $\Gamma_t = L_t^a \varepsilon_t^{\beta_t^a}$ and $\delta_t^a = (1 - m_t(\varepsilon_t))^{K_t}$, and thus $\kappa_t = \lambda_t + L_t^a \varepsilon_t^{\beta_t^a}$ and $\Delta_t = C_t K_t \delta_{\text{loc},t}(\lambda_t, C_{t+1}) + C_t (1 - m_t(\varepsilon_t))^{K_t}$.

Corollary 4.2 (Smoothed-transition graph budgets). Suppose the exact transition density is replaced by the smoothed density $p_{t,\tau_t}(\cdot | s, a)$ and $G_t^*(s, a, \cdot)$ satisfies the assumptions of Proposition 1. Then Proposition 3 holds for the smoothed SNIS backup with the same $\delta_{\text{loc},t}$ and with λ_t replaced by $\lambda_t + L_t^s m_{\beta_t^s, t}(\tau_t)$.

Finally, we show some of the implications that follow by assuming bounded $d_\infty(p_t(\cdot | s, a) \| q_t^s)$ on GSS concentration inequalities with SNIS backups.

Corollary 4.3 (Finite-action, absolutely continuous transition sample complexity). Assume $\mathcal{A}$ is finite and fully enumerated, the terminal estimate is exact, and the exact full-rank SNIS backup is used. Assume also that $\sup_{t < T, s, a} d_\infty(p_t(\cdot | s, a) \| q_t^s) \leq d_\infty^{\text{max}}$. Then, for every fixed $\epsilon > 0$ and root failure probability $\delta \in (0, 1)$, choosing a common layer width $C = \tilde{O}(R_{\text{max}}^2 (d_\infty^{\text{max}})^2 T^4 / \epsilon^2)$ suffices to make the returned root action $\hat{a}_0$ satisfy

$$\Pr(Q_0^*(s_0, a_0^*) - Q_0^*(s_0, \hat{a}_0) \leq \epsilon) \geq 1 - \delta \quad \text{for every } a_0^* \in \underset{0}{\text{Argmax}}(s_0). \quad (12)$$

Thus, the number of simulator queries is $N_{\text{GSS}} = TC = \tilde{O}(R_{\text{max}}^2 (d_\infty^{\text{max}})^2 T^5 / \epsilon^2)$, and the number of backup operations is $O(|\mathcal{A}| TC^2) = \tilde{O}(|\mathcal{A}| R_{\text{max}}^4 (d_\infty^{\text{max}})^4 T^9 / \epsilon^4)$.

Remark 1 (Comparison with Sparse Sampling). Corollary 4.3 gives $N_{\text{GSS}} = \tilde{O}(R_{\text{max}}^2 (d_\infty^{\text{max}})^2 T^5 / \epsilon^2)$. If $d_\infty^{\text{max}} = O(T^n)$ for $n \in \mathbb{N}$, then N_{GSS} remains polynomial in T , whereas the Sparse Sampling budget N_{SS} required for target accuracy ϵ is exponential in T [Kearns et al., 2002]. See Appendix A.7, which also gives a simple example with $n = n_s$.

Corollary 4.4 (Joint schedule for low-rank sampled-action GSS). Assume the hypotheses of Corollaries 4.1 and 4.2. Suppose, in addition, that the terminal estimate is exact, or that its tail accuracy and failure budgets can be made small enough after the final width C_T is fixed. Assume also that $L_t^s m_{\beta_t^s, t}(\tau_t) \rightarrow 0$ as $\tau_t \downarrow 0$, and that the smoothed backup can be controlled, uniformly over realized node-action pairs, to any prescribed local accuracy $\lambda_t > 0$ and local failure level $\rho_t \in (0, 1)$ by increasing the state-side width C_{t+1} . Then for every $\varepsilon_{\text{root}} > 0$ and $\delta \in (0, 1)$, one can choose bandwidths τ_t , action tolerances ε_t , action budgets K_t , and state-side widths C_{t+1} so that $2\alpha_0 \leq \varepsilon_{\text{root}}$ and $C_T \Delta_T^{\text{tail}} + \sum_{t=0}^{T-1} \Delta_t \leq \delta$. Consequently, $\Pr(Q_0^*(s_0, a_0^*) - Q_0^*(s_0, \hat{a}_0) \leq \varepsilon_{\text{root}}) \geq 1 - \delta$ for every $a_0^* \in \text{Argmax}_0(s_0)$.

5 Experiments and Results

We evaluate GSS in three continuous-control settings chosen to stress different parts of the method: controlled rollout mismatch, singular non-linear dynamics, and smoothed backups in high-dimensional state spaces. Full configurations, including action mixtures, state proposals, smoothing bandwidths, and DPW/VPW parameters, are in Appendix B.

Implementation. GSS is implemented in JAX [Bradbury et al., 2018] with fixed-shape batched graph construction and runs on a GPU. In Rotating DDI and Lunar Lander, we compare it to the tree-based planners DPW [Couëtoux et al., 2011] and VPW [Lim et al., 2021], which use the same rollout policy as GSS and run on a CPU with the MCTS.jl solvers of POMDPs.jl [Egorov et al., 2017] in Julia. A tree search expands one node at a time, each expansion depending on the statistics of the previous ones, which suits a sequential CPU implementation, whereas GSS issues the transitions and backups of each layer as large batched operations, which a GPU executes in parallel. As a result, GSS draws far more samples per decision in a given planning time: counting every sampled state, including those inside rollouts, GSS draws more than six times as many as DPW and VPW runs with equal or longer planning times (Appendix B.3). Our time comparisons are therefore end-to-end comparisons of each method in an implementation suited to its computational structure.

The implementation is guided by the analysis of Section 4. Every **action proposal** includes a uniform component, consistent with Assumption 1, and every **graph backup** is an SNIS backup, exact in Rotating DDI and smoothed in Lunar Lander and Reacher, whose normalized nonnegative weights give the stability of Assumption 4. The implementation also uses heuristics outside the analyzed algorithm: **state proposals** are fitted to the sampled transitions of the current layer and, in Rotating DDI and Lunar Lander, restricted to the components with the highest rollout scores; smoothed densities use the linearization (6); **tail values** are single-sample returns of π_{rollout} ; and in Rotating DDI and Lunar Lander, the backup includes a direct successor sample.

Direct successor sample. For each pair $(s_t^i, a_t^{i,k})$, this heuristic draws $\tilde{s}_{t+1}^{i,k} \sim p_t(\cdot \mid s_t^i, a_t^{i,k})$ and values it by $\hat{G}_t^{i,k,0} \triangleq r_t(s_t^i, a_t^{i,k}, \tilde{s}_{t+1}^{i,k}) + \gamma \hat{R}_{t+1}(\tilde{s}_{t+1}^{i,k})$, where $\hat{R}_{t+1}$ is a single-sample return of π_{rollout} . Since $\tilde{s}_{t+1}^{i,k}$ is drawn from the transition model rather than from q_t^s , it enters the backup with unit weight:

$$\hat{Q}_{t,\text{DS}}^{i,k} = \frac{\hat{G}_t^{i,k,0} + \sum_{j=1}^{C_{t+1}} \rho_t^{i,k}(j) \hat{G}_t^{i,k,j}}{1 + \eta_\rho}, \quad (13)$$

with $\rho_t^{i,k}$ and η_ρ as in the SNIS backup (1.A), or its smoothed version in Lunar Lander. The direct sample has weight $1/(1 + \eta_\rho)$, and η_ρ has expectation C_{t+1} when q_t^s covers the support of the density in the ratios, $p_t(\cdot \mid s_t^i, a_t^{i,k})$ or its smoothed version. When few graph states fall where this density has mass, η_ρ is small and the estimate approaches the rollout-based target $\hat{G}_t^{i,k,0}$; when η_ρ is near its expectation, the direct sample has weight about $1/(C_{t+1} + 1)$. This parallels MCTS, which initializes the value of a new node with a rollout and refines it as its subtree grows.

Rotating 4D-Double-Integrator (Rotating DDI). This domain evaluates GSS under controlled rollout mismatch, with exact transition densities and SNIS backups. Extending the benchmark of Mansley et al. [2011], the state is $s = (x, v) \in \mathbb{R}^8$ and the acceleration action is $a \in [-2, 2]^4$. However, an externally rotating frame introduces $a_\alpha = a - 2\alpha Jv + \alpha^2 x$, where J rotates coordinate pairs (1, 2) and (3, 4) by 90° . The transition density is the full-rank Gaussian $p_t(s' \mid s, a) = \mathcal{N}(s'; \mu_\alpha(s, a), \Sigma)$, where $\mu_\alpha(s, a)$ is given by Euler numerical integration over the decision interval $\Delta t = 0.2$, with diagonal isotropic covariance $\Sigma = 0.02^2 I_8$. Episodes last at most 50 steps.

We compare against the clipped LQR feedback baseline $a_{\text{LQR}}(x, v) = -(x - x_{\text{goal}}) - \sqrt{2}v$ projected to $[-2, 2]^4$, against DPW and VPW with per-decision time budgets from 0.01 to 1.0 seconds, and also report a full-state LQG/Riccati reference that is analytically optimal when $\alpha = 0$ and actions are unbounded. Thus $\alpha = 0$ is the matched-guide setting: the LQR rollout/proposal follows the nominal double-integrator controller, and the LQG/Riccati reference is analytically optimal. As the external rotation α increases, the same rollout/proposal becomes increasingly mismatched, so the sweep tests GSS under degrading rollout and proposal quality. GSS uses an LQR-guided **action proposal**; as **state proposal**, the equally weighted mixture of the transition moments of the top-0.95 fraction of state-action components, ranked by their direct-sample targets $\hat{G}_t^{i,k,0}$; the LQR rollout policy for **tail values**; and the exact SNIS **graph backup** with the direct successor sample (13). GSS plans with

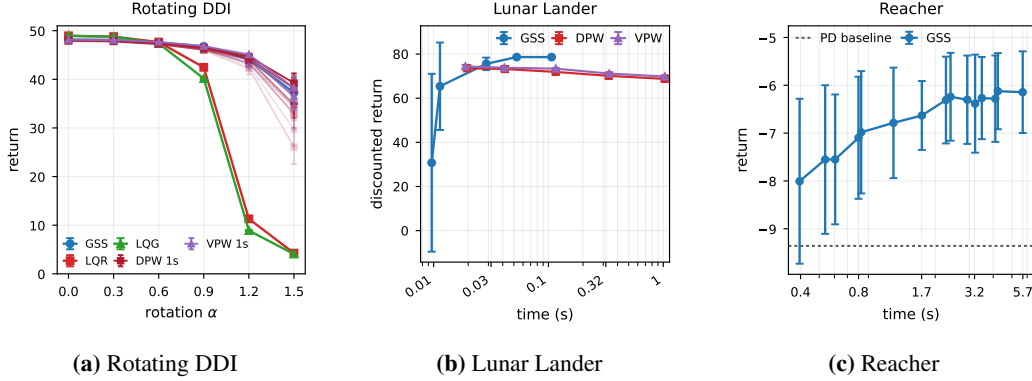


Figure 2: Error bars show ± 2 standard errors. We report mean planner performance for displayed domains, over discounted returns for Lunar Lander, or over undiscounted returns for the other domains. In Rotating DDI, the x-axis is the rotation parameter α , which controls the mismatch of the rollout guide and the difficulty of the problem. DPW/VPW curves span wall-clock budgets 0.01, 0.0316, 0.1, 0.316, and 1.0 seconds, with lower-budget curves faded and the 1.0 second curves emphasized; GSS mean runtimes are at most 0.7. In Lunar Lander and Reacher, the x-axis is mean planner runtime per planning session, averaged over the closed-loop execution in the environment.

horizon 50 and rollouts capped at 50 steps from the current decision, with action budget 64 and state width 512; its mean planning time is about 0.2 seconds per decision at $\alpha = 0$ and 0.6–0.7 seconds for $\alpha > 0$.

The results are shown in Figure 2a. As α increases, LQR and LQG degrade sharply from $\alpha = 0.9$ onward, while GSS maintains relatively consistent performance despite having a degraded rollout guide. For all tested time budgets, GSS achieves better or comparable performance to DPW/VPW. The tree-based planners show a clear degradation pattern as α increases, achieving competitive performance with GSS in high-rotation regimes only for the highest time budgets. The full results table is in Appendix 2.

Lunar Lander. This domain evaluates GSS with singular, non-linear dynamics. The continuous Lunar Lander benchmark of Mern et al. [2021] features a 6D spacecraft attempting to land safely from an initially random position. The state is $(x, z, \theta, v_x, v_z, \omega) \in \mathbb{R}^6$, containing horizontal position, altitude, attitude, and their respective velocities, and the action $a \in \mathcal{A} \subseteq \mathbb{R}^3$ consists of lateral thrust, main thrust, and thruster offset. The transition model is a deterministic transition function perturbed by 3D additive Gaussian noise, thus it is singular in $\mathbb{R}^6$; its generative equations are given in (14) in Appendix B.5. Episodes last at most 24 steps.

We compare against DPW and VPW with per-decision time budgets from 0.01 to 1.0 seconds; all three planners use the same brake-near-ground guide and rollout policy. GSS samples its **action proposal** from multiple sources: the guide’s deterministic action, perturbed rollout actions, randomly sampled actions, and a few expert-chosen actions. Its **state proposal** is a diagonal Gaussian fitted to the direct successor samples of the top-0.8 fraction of state-action components, ranked by $\hat{G}_t^{i,k,0}$; it uses the brake-near-ground rollout policy for **tail values**; and its **graph backup** is SNIS with the single-point smoothed density (6) and the direct successor sample (13). GSS plans with horizon 12 and rollouts capped at 24 steps from the current decision; we chose five graph shapes, with action budgets from 10 to 50 and state widths from 384 to 2048, whose mean planning times are at most about 0.1 seconds per decision.

The results are shown in Figure 2b, where each planner is placed at its measured mean planning time. GSS reaches the high-discounted-return regime from action budget 24 and state width 1024 and remains stable as the action and state budgets grow. The tree planners are strongest at short plotted budgets, but their plotted mean performances decline as time budgets increase (see Appendix B.5).

Reacher. This domain evaluates smoothed backups in a high-dimensional state space with deterministic transitions. We implement Reacher-v5 [Towers et al., 2024] in JAX [Bradbury et al., 2018] using the MJX interface to MuJoCo [Todorov et al., 2012]. The 10D state contains joint-angle cosines and

sines, target coordinates, arm angular velocities, and fingertip-target displacement; the actions are two clipped actuator torques. The transition dynamics are deterministic. Episodes last at most 50 steps.

We compare against a deterministic proportional-derivative (PD) controller. GSS uses this controller in the *action proposal*, with one controller action and the remaining action budget split between noisy-controller and uniform samples. Its *state proposal* is a transition-moment mixture without top-component pruning; it uses the controller as the rollout policy for *tail values*; and its *graph backup* is SNIS with the single-point smoothed density (6). GSS plans with horizon 20 and rollouts capped at 50 steps from the current decision; we vary the state width from 256 to 2048 and the action budget from 24 to 64, resulting in mean planning times from 0.4 to 5.7 seconds per decision.

The results are shown in Figure 2c. Every tested GSS budget improves the local controller, and increasing the state width and action budget generally improves performance with diminishing returns. This shows that GSS can improve baseline policies and scale effectively in a high-dimensional, deterministic-transition domain.

6 Conclusion

We present Graph Sparse Sampling, an alternative to tree-based online planning, by sampling shared successor state layers, evaluating many candidate actions against them, and using generalized graph backups to propagate values through a fixed-shape planning graph. GSS changes the bottleneck from branching a search tree, to designing effective state and action proposals to focus on relevant portions of the action and state spaces, yet with enough overlap to support reliable reuse. Our analysis makes this tradeoff explicit, and we show explicit finite-sample performance guarantees in cases of density-ratio overlap, sampled action coverage, and bounded smoothing-bias. We obtain performance guarantees with polynomial dependence on the planning horizon in regimes where the shared state layers remain representative, and we discuss how to jointly schedule the state and action budgets and smoothing bandwidths to achieve any desired root accuracy and failure probability in low-rank, sampled-action settings. Empirically, the same design supports large GPU batches, and improves or matches MCTS-based baselines across continuous-control problems with stochastic and deterministic transition cases, challenging horizons, and varying domain dimensions.

Limitations. GSS is proposal-limited: poor action or state proposals can make the graph spend most of its computation in the wrong regions. Our theoretical analysis is also limited in scope. Theorem 1 applies to any controlled graph backup, but we establish this property only for SNIS backups and their smoothed and median-of-means variants; other graph backups, such as the KDE, nearest-neighbor, and learned backups (1.B)–(1.D), require their own analysis and may admit different guarantees. The polynomial dependence on the planning horizon holds under the stated overlap, regularity, and coverage conditions, and weakens when proposals mismatch the transitions so that $d_{\infty}^{\max}$ is large. As for our experiments, they do not isolate horizon scaling; they measure the usefulness of GSS in practice, including heuristics beyond the analyzed algorithm.

Future work. It is natural to extend GSS into multiple passes, where proposals are adapted based on earlier value estimates, or use learned approximations as part of GSS components. We plan to explore these directions, along with tightening our performance guarantees, adapting to anytime settings, and analyzing our methods in partially observable domains.

Acknowledgments

This work was supported by the Israel Ministry of Innovation, Science and Technology.

References

David Auger, Adrien Couetoux, and Olivier Teytaud. Continuous upper confidence trees with polynomial exploration–consistency. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2013, Prague, Czech Republic, September 23–27, 2013, Proceedings, Part I 13*, pages 194–209. Springer, 2013.

- M. Barenboim and V. Indelman. Online POMDP planning with anytime deterministic optimality guarantees. *Artificial Intelligence*, 350:104442, 2026. doi: <https://doi.org/10.1016/j.artint.2025.104442>.
- Denis Belomestny, Maxim Kaledin, and John Schoenmakers. Semitractability of optimal stopping problems via a weighted stochastic mesh algorithm. *Mathematical Finance*, 30(4):1591–1616, 2020.
- Denis Belomestny, John Schoenmakers, and Veronika Zorina. Weighted mesh algorithms for general markov decision processes: Convergence and tractability. *Journal of Complexity*, 88:101932, 2025.
- James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Yash Katariya, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL <http://github.com/jax-ml/jax>.
- Mark Broadie, Paul Glasserman, et al. A stochastic mesh method for pricing high-dimensional american options. *Journal of Computational Finance*, 7:35–72, 2004.
- Cameron B Browne, Edward Powley, Daniel Whitehouse, Simon M Lucas, Peter I Cowling, Philipp Rohlfschagen, Stephen Tavenor, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of Monte Carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1):1–43, 2012.
- Adrien Couëtoux, Jean-Baptiste Hoock, Nataliya Sokolovska, Olivier Teytaud, and Nicolas Bonnard. Continuous upper confidence trees. In *International conference on learning and intelligent optimization*, pages 433–445. Springer, 2011.
- Hai Dang Dau. *Sequential Bayesian Computation*. PhD thesis, Institut Polytechnique de Paris, September 2022. URL <https://theses.hal.science/tel-03848268>.
- Maxim Egorov, Zachary N Sunberg, Edward Balaban, Tim A Wheeler, Jayesh K Gupta, and Mykel J Kochenderfer. POMDPs.jl: A framework for sequential decision making under uncertainty. *The Journal of Machine Learning Research*, 18(1):831–835, 2017.
- Simon J. Julier, Jeffrey, and K. Uhlmann. Unscented filtering and nonlinear estimation. In *Proceedings of the IEEE*, pages 401–422, 2004.
- Michael Kearns, Yishay Mansour, and Andrew Y Ng. A sparse sampling algorithm for near-optimal planning in large Markov decision processes. *Machine learning*, 49(2):193–208, 2002.
- Beomjoon Kim, Kyungjae Lee, Sungbin Lim, Leslie Kaelbling, and Tomás Lozano-Pérez. Monte Carlo tree search in continuous spaces using Voronoi optimistic optimization with regret bounds. In *AAAI Conf. on Artificial Intelligence*, volume 34, pages 9916–9924, 2020.
- Teun Kloek and Herman K Van Dijk. Bayesian estimates of equation system parameters: an application of integration by Monte Carlo. *Econometrica: Journal of the Econometric Society*, pages 1–19, 1978.
- Levente Kocsis and Csaba Szepesvári. Bandit based Monte-Carlo planning. In *European conference on machine learning*, pages 282–293. Springer, 2006.
- Kalle Kujanpää, Amin Babadi, Yi Zhao, Juho Kannala, Alexander Ilin, and Joni Pajarinen. Continuous Monte Carlo graph search. In *Intl. Conf. on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 1047–1056, 2024.
- Jongmin Lee, Wonseok Jeon, Geon-Hyeong Kim, and Kee-Eung Kim. Monte-Carlo tree search in continuous action spaces with value gradients. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 4561–4568, 2020.
- Edouard Leurent and Odalric-Ambrym Maillard. Monte-Carlo graph search: the value of merging similar states. In Sinno Jialin Pan and Masashi Sugiyama, editors, *Asian Conference on Machine Learning (ACML 2020)*, pages 577 – 592, Bangkok, Thailand, November 18-20 2020.

- Idan Lev-Yehudi, Michael Novitsky, Moran Barenboim, Ron Benchetrit, and Vadim Indelman. Value gradients with action adaptive search trees in continuous (po)mdps. 2025.
- Michael H Lim, Claire J Tomlin, and Zachary N Sunberg. Voronoi progressive widening: efficient online solvers for continuous state, action, and observation POMDPs. In *2021 60th IEEE conference on decision and control (CDC)*, pages 4493–4500. IEEE, 2021.
- Michael H Lim, Tyler J Becker, Mykel J Kochenderfer, Claire J Tomlin, and Zachary N Sunberg. Optimality guarantees for particle belief approximation of POMDPs. *Journal of Artificial Intelligence Research*, 77:1591–1636, 2023.
- Chris Mansley, Ari Weinstein, and Michael Littman. Sample-based planning for continuous action markov decision processes. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 21, pages 335–338, 2011.
- Weichao Mao, Kaiqing Zhang, Qiaomin Xie, and Tamer Basar. Poly-hoot: Monte-carlo planning in continuous space mdps with non-asymptotic analysis. volume 33, pages 4549–4559, 2020.
- John Mern, Anil Yildiz, Zachary Sunberg, Tapan Mukerji, and Mykel J Kochenderfer. Bayesian optimized Monte Carlo planning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 11880–11887, 2021.
- Art B. Owen. *Monte Carlo theory, methods and examples*. <https://artowen.su.domains/mc/>, 2013.
- Christos H. Papadimitriou and John N. Tsitsiklis. The complexity of Markov decision processes. *Mathematics of operations research*, 12(3):441–450, 1987.
- Devavrat Shah, Qiaomin Xie, and Zhi Xu. Nonasymptotic analysis of Monte Carlo Tree Search. *Operations Research*, 70(6):3234–3260, 2022.
- Emanuel Todorov, Tom Erez, and Yuval Tassa. MuJoCo: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012. doi: 10.1109/IROS.2012.6386109.
- Mark Towers, Ariel Kwiatkowski, Jordan Terry, John U. Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Markus Krimmel, Arjun KG, Rodrigo Perez-Vicente, Andrea Pierré, Sander Schulhoff, Jun Jet Tai, Hannah Tan, and Omar G. Younis. Gymnasium: A standard interface for reinforcement learning environments. *arXiv preprint arXiv:2407.17032*, 2024.
- Tim Van Erven and Peter Harremoës. Rényi divergence and kullback-leibler divergence. *IEEE Transactions on Information Theory*, 60(7):3797–3820, 2014.
- Matt P Wand and M Chris Jones. *Kernel smoothing*. CRC press, 1994.
- Grady Williams, Andrew Aldrich, and Evangelos A. Theodorou. Model predictive path integral control: From theory to parallel computation. *Journal of Guidance, Control, and Dynamics*, 40(2): 344–357, 2017. doi: 10.2514/1.G001921.
- Timothy Yee, Viliam Lisý, and Michael Bowling. Monte Carlo Tree Search in continuous action spaces with execution uncertainty. In Subbarao Kambhampati, editor, *IJCAI*, pages 690–697, 2016.

A Proofs and Derivations

A.1 Proof of Proposition 1

Proof. Fix $(s, a) \in \mathcal{S} \times \mathcal{A}$. Let Y have transition law $p_t(\cdot \mid s, a)$ and, conditionally on $Y = y$, let X have density $k_{t,\tau_t}(\cdot \mid y)$. By the definition of the smoothed density in (3), the marginal law of X has density $p_{t,\tau_t}(\cdot \mid s, a)$. Therefore,

$$\begin{aligned} & |\mathbb{E}_{s' \sim p_t(\cdot \mid s, a)}[G_t^*(s, a, s')] - \mathbb{E}_{x \sim p_{t,\tau_t}(\cdot \mid s, a)}[G_t^*(s, a, x)]| \\ &= \left| \int_{\mathcal{S}} \int_{\mathcal{S}} (G_t^*(s, a, y) - G_t^*(s, a, x)) k_{t,\tau_t}(x \mid y) dx p_t(dy \mid s, a) \right| \\ &\leq \int_{\mathcal{S}} \int_{\mathcal{S}} |G_t^*(s, a, y) - G_t^*(s, a, x)| k_{t,\tau_t}(x \mid y) dx p_t(dy \mid s, a) \\ &\leq L_t^s \int_{\mathcal{S}} \int_{\mathcal{S}} d(x, y)^{\beta_t^s} k_{t,\tau_t}(x \mid y) dx p_t(dy \mid s, a) \\ &\leq L_t^s \sup_y \int_{\mathcal{S}} d(x, y)^{\beta_t^s} k_{t,\tau_t}(x \mid y) dx = L_t^s m_{\beta_t^s, t}(\tau_t). \end{aligned}$$

The first inequality is the triangle inequality, the second is the assumed successor-state Hölder regularity of $G_t^*(s, a, \cdot)$, and the last display is exactly the definition of $m_{\beta_t^s, t}(\tau_t)$. $\square$

A.2 Proof of Proposition 2

Proof. Fix $t < T$, $\varepsilon_t \in (0, \bar{\varepsilon}_t]$, and a realized node s_t^i . By Assumption 1 with $\varepsilon = \varepsilon_t$ and $s = s_t^i$, there exists $a^* \in \text{Argmax}_t(s_t^i)$ whose ε_t -neighborhood has probability at least $m_t(\varepsilon_t)$ under $q_t^a(\cdot \mid s_t^i)$. Conditionally on s_t^i , each candidate action therefore lands in this neighborhood with probability at least $m_t(\varepsilon_t)$. The K_t candidates in $A_t^i = \{a_t^{i,1:K_t}\}$ are conditionally independent, hence all of them miss this neighborhood with probability at most $(1 - m_t(\varepsilon_t))^{K_t}$.

On the complementary event, choose $a_i \in A_t^i$ with $d_{\mathcal{A}}(a_i, a^*) < \varepsilon_t$. Since $Q_t^*(s_t^i, a^*) = V_t^*(s_t^i)$, Assumption 2 gives

$$V_t^*(s_t^i) - Q_t^*(s_t^i, a_i) = Q_t^*(s_t^i, a^*) - Q_t^*(s_t^i, a_i) \leq L_t^a \varepsilon_t^{\beta_t^a}.$$

Maximizing over all actions in A_t^i can only improve on a_i , so

$$0 \leq V_t^*(s_t^i) - \max_{a \in A_t^i} Q_t^*(s_t^i, a) \leq L_t^a \varepsilon_t^{\beta_t^a}.$$

This is (8). $\square$

A.3 Additional Graph-Backup Details

Proof of Proposition 3. Work on the event

$$\|\hat{V}_{t+1}^{1:C_{t+1}} - V_{t+1}^{*,1:C_{t+1}}\|_{\infty} \leq \alpha_{t+1}.$$

For a fixed pair $i \in 1:C_t$ and $k \in 1:K_t$, Assumption 4 gives

$$|\hat{Q}_t^{i,k} - \hat{Q}_t^{i,k,*}| = |\mathcal{B}_t^{i,k}[\hat{V}_{t+1}^{1:C_{t+1}}] - \mathcal{B}_t^{i,k}[V_{t+1}^{*,1:C_{t+1}}]| \leq \gamma \alpha_{t+1}.$$

Assumption 3 gives

$$|\hat{Q}_t^{i,k,*} - Q_t^{*,i,k}| \leq \lambda_t$$

except with probability at most $\delta_{\text{loc},t}(\lambda_t, C_{t+1})$. On the intersection of these two events, the triangle inequality yields

$$|\hat{Q}_t^{i,k} - Q_t^{*,i,k}| \leq \gamma \alpha_{t+1} + \lambda_t.$$

Union-bounding the local concentration failures over the $C_t K_t$ realized node-action pairs gives, with probability at least $1 - C_t K_t \delta_{\text{loc},t}(\lambda_t, C_{t+1})$, the pointwise bound

$$|\hat{Q}_t^{i,k} - Q_t^{*,i,k}| \leq \gamma \alpha_{t+1} + \lambda_t \quad \text{for every } i \in 1:C_t \text{ and } k \in 1:K_t.$$

Equivalently,

$$\|\hat{Q}_t^{1:C_t, 1:K_t} - Q_t^{*,1:C_t, 1:K_t}\|_{\infty} \leq \gamma \alpha_{t+1} + \lambda_t.$$

$\square$

For probability laws $P \ll Q$, write $\omega(y) \triangleq dP/dQ(y)$ and define

$$d_\infty(P\|Q) \triangleq \operatorname{ess\,sup}_{Y \sim Q} \omega(Y), \quad d_2(P\|Q) \triangleq \mathbb{E}_{Y \sim Q} [\omega(Y)^2].$$

Definition 4 (SNIS and MoM-SNIS). Given samples $Y_{1:N} \sim Q$, weights $\omega_n = \omega(Y_n)$, and values $f(Y_n)$, SNIS returns

$$\hat{\mu}_{\text{SNIS},N}(f) \triangleq \frac{\sum_{n=1}^N \omega_n f(Y_n)}{\sum_{n=1}^N \omega_n}.$$

For MoM-SNIS, partition $\{1, \dots, N\}$ into nonempty blocks $B_1, \dots, B_B$, form

$$Z_b(f) \triangleq \frac{\sum_{n \in B_b} \omega_n f(Y_n)}{\sum_{n \in B_b} \omega_n}, \quad b = 1, \dots, B,$$

and return

$$\hat{\mu}_{\text{MoM-SNIS},N}(f) \triangleq \operatorname{median}\{Z_1(f), \dots, Z_B(f)\}.$$

Ordinary SNIS backups satisfy Assumption 4 because their normalized weights are nonnegative and sum to one, so changing the child-value vector by at most α changes the continuation part of the backup by at most $\gamma\alpha$. The same stability argument holds for median-of-blocks normalized backups: each blockwise normalized average is γ -Lipschitz in child-value sup norm, and the median is nonexpansive in sup norm.

The following source theorem is a bounded- d_2 analogue for MoM-SNIS. It supplies one possible choice of the local function $\delta_{\text{loc},t}$ in Assumption 3. Denote

$$\sigma_{\text{CS}}^2 \triangleq d_2(P\|Q) - 1, \quad \sigma_{\text{IS}}^2 \triangleq \mathbb{E}_{x \sim Q} [\omega(x)^2 (f(x) - \mathbb{E}_{z \sim P}[f(z)])^2].$$

Theorem (MoM-SNIS one-sided concentration, bounded d_2 Dau [2022]). Fix $\delta \in (0, 1)$ and assume $\sigma_{\text{IS}}^2 < \infty$. If

$$N \geq 8 (\max\{32\sigma_{\text{CS}}^2, 1\}) \log(1/\delta),$$

then the MoM-SNIS estimator of Definition 4 satisfies

$$\mathbb{P}\left(\hat{\mu}_{\text{MoM-SNIS},N}(f) - \mathbb{E}_{x \sim P}[f(x)] \geq 16 \sigma_{\text{IS}} \sqrt{\frac{2 \log(1/\delta)}{N}}\right) \leq \delta.$$

Applying Theorem to f and to $-f$ gives the two-sided statement: for a target local failure level $\rho \in (0, 1)$, if

$$N \geq 8 (\max\{32\sigma_{\text{CS}}^2, 1\}) \log(2/\rho),$$

then

$$\mathbb{P}\left(|\hat{\mu}_{\text{MoM-SNIS},N}(f) - \mathbb{E}_{x \sim P}[f(x)]| \leq 16 \sigma_{\text{IS}} \sqrt{\frac{2 \log(2/\rho)}{N}}\right) \geq 1 - \rho.$$

If $\|f\|_\infty \leq f_{\max}$, then $\sigma_{\text{IS}}^2 \leq 4f_{\max}^2 d_2(P\|Q)$. Indeed, since $|f(Y) - \mathbb{E}_{x \sim P}[f(x)]| \leq 2f_{\max}$,

$$\sigma_{\text{IS}}^2 = \mathbb{E}_{Y \sim Q} [\omega(Y)^2 (f(Y) - \mathbb{E}_{x \sim P}[f(x)])^2] \leq 4f_{\max}^2 \mathbb{E}_{Y \sim Q} [\omega(Y)^2] = 4f_{\max}^2 d_2(P\|Q).$$

We now translate this local estimator statement into the graph-backup notation of Assumption 3. Fix a depth $t < T$ and a realized node-action pair $(s_t^i, a_t^{i,k})$. Let

$$P_t^{i,k} \triangleq p_t(\cdot \mid s_t^i, a_t^{i,k}), \quad Q_t \triangleq q_t^s, \quad N \triangleq C_{t+1},$$

and apply the MoM-SNIS backup to the exact Bellman target

$$f_t^{i,k}(x) \triangleq G_t^*(s_t^i, a_t^{i,k}, x) = r_t(s_t^i, a_t^{i,k}, x) + \gamma V_{t+1}^*(x).$$

Assume that, uniformly over the realized node-action pairs at depth t ,

$$\|f_t^{i,k}\|_\infty \leq f_{\max,t}, \quad d_2(P_t^{i,k}\|Q_t) \leq D_{2,t} < \infty.$$

Then for every such pair,

$$\sigma_{\text{CS},t}^2 = d_2(P_t^{i,k}\|Q_t) - 1 \leq D_{2,t} - 1, \quad \sigma_{\text{IS},t}^2 \leq 4f_{\max,t}^2 D_{2,t}.$$

Consequently, for any local failure level $\rho_t \in (0, 1)$ such that

$$C_{t+1} \geq 8 \max\{32(D_{2,t} - 1), 1\} \log(2/\rho_t),$$

the exact-value MoM-SNIS graph backup satisfies

$$\mathbb{P}\left(|\hat{Q}_t^{i,k,*} - Q_t^{*,i,k}| \leq 32f_{\max,t} \sqrt{\frac{2D_{2,t} \log(2/\rho_t)}{C_{t+1}}}\right) \geq 1 - \rho_t.$$

Thus Assumption 3 holds for the MoM-SNIS backup at depth t with the explicit choice

$$\lambda_t = 32f_{\max,t} \sqrt{\frac{2D_{2,t} \log(2/\rho_t)}{C_{t+1}}}, \quad \delta_{\text{loc},t}(\lambda_t, C_{t+1}) = \rho_t,$$

provided the displayed lower bound on C_{t+1} holds. Equivalently, a uniform depthwise d_2 overlap bound supplies the static, exact-next-value concentration term in the same role that the bounded- d_∞ theorem plays in the main text. Together with the normalized-weight stability argument above, this makes the MoM-SNIS graph backup controlled in the sense of Definition 2, with the propagated recursive bound then given by Proposition 3.

A.4 Proof of Theorem 1

Proof. Use the main-text shorthand

$$V_t^{*,i} \triangleq V_t^*(s_t^i), \quad Q_t^{*,i,k} \triangleq Q_t^*(s_t^i, a_t^{i,k}).$$

Let κ_t , Δ_t , and α_t be the budgets of Definition 3. For each depth t , define the value event

$$\mathcal{V}_t \triangleq \left\{ |\hat{V}_t^i - V_t^{*,i}| \leq \alpha_t \quad \text{for every } i \in 1:C_t \right\}.$$

For $t < T$, define the simultaneous backup event

$$\mathcal{Q}_t \triangleq \left\{ |\hat{Q}_t^{i,k} - Q_t^{*,i,k}| \leq \gamma\alpha_{t+1} + \lambda_t \quad \text{for every } i \in 1:C_t, k \in 1:K_t \right\},$$

and the simultaneous action-coverage event

$$\mathcal{A}_t \triangleq \left\{ 0 \leq V_t^{*,i} - \max_{k \in 1:K_t} Q_t^{*,i,k} \leq \Gamma_t \quad \text{for every } i \in 1:C_t \right\}.$$

Assumptions (i)–(iii) of the theorem give

$$\begin{aligned} \mathbb{P}(\mathcal{V}_T^c) &\leq \sum_{i \in 1:C_T} \mathbb{P}\left(|\hat{V}_T^i - V_T^{*,i}| > \kappa_T^{\text{tail}}\right) \leq C_T \Delta_T^{\text{tail}}, \\ \mathbb{P}(\mathcal{Q}_t^c \cap \mathcal{V}_{t+1}) &\leq C_t K_t \delta_{\text{loc},t}(\lambda_t, C_{t+1}), \\ \mathbb{P}(\mathcal{A}_t^c) &\leq C_t \delta_t^a. \end{aligned}$$

The first inequality is assumption (i) with a union bound over terminal nodes. The second inequality is assumption (ii), i.e., the conclusion of Proposition 3 on the event $\mathcal{V}_{t+1}$. The third inequality is the union bound over the C_t max-close action-set failures of assumption (iii).

We next prove the deterministic induction step. Assume $\mathcal{V}_{t+1} \cap \mathcal{Q}_t \cap \mathcal{A}_t$ holds. Set

$$R_t^Q \triangleq \gamma\alpha_{t+1} + \lambda_t, \quad R_t \triangleq R_t^Q + \Gamma_t = \gamma\alpha_{t+1} + \kappa_t = \alpha_t.$$

For every node index $i \in 1:C_t$,

$$\hat{V}_t^i = \max_{k \in 1:K_t} \hat{Q}_t^{i,k} \leq \max_{k \in 1:K_t} Q_t^{*,i,k} + R_t^Q \leq V_t^{*,i} + R_t.$$

For the lower deviation,

$$\begin{aligned} V_t^{*,i} - \hat{V}_t^i &\leq V_t^{*,i} - \max_{k \in 1:K_t} Q_t^{*,i,k} + \max_{k \in 1:K_t} Q_t^{*,i,k} - \max_{k \in 1:K_t} \hat{Q}_t^{i,k} \\ &\leq \Gamma_t + \max_{k \in 1:K_t} |\hat{Q}_t^{i,k} - Q_t^{*,i,k}| \leq \Gamma_t + R_t^Q = R_t. \end{aligned}$$

Thus $\mathcal{V}_t$ holds. Equivalently,

$$\|\hat{V}_t^{1:C_t} - V_t^{*,1:C_t}\|_\infty \leq \alpha_t.$$

The event $\mathcal{Q}_t$ itself is exactly the stated pointwise candidate-backup bound at depth t . Equivalently,

$$\|\hat{Q}_t^{1:C_t, 1:K_t} - Q_t^{*,1:C_t, 1:K_t}\|_\infty \leq \gamma\alpha_{t+1} + \lambda_t.$$

Now define the total bad event

$$\mathcal{F} \triangleq \mathcal{V}_T^c \cup \bigcup_{t=0}^{T-1} (\mathcal{Q}_t^c \cap \mathcal{V}_{t+1}) \cup \bigcup_{t=0}^{T-1} \mathcal{A}_t^c.$$

By the union bound and Definition 3,

$$\mathbb{P}(\mathcal{F}) \leq C_T \Delta_T^{\text{tail}} + \sum_{t=0}^{T-1} [C_t K_t \delta_{\text{loc},t}(\lambda_t, C_{t+1}) + C_t \delta_t^a] = C_T \Delta_T^{\text{tail}} + \sum_{t=0}^{T-1} \Delta_t.$$

On $\mathcal{F}^c$, the terminal event $\mathcal{V}_T$ holds. If $\mathcal{V}_{t+1}$ holds, then $\mathcal{Q}_t$ must hold because $\mathcal{Q}_t^c \cap \mathcal{V}_{t+1}$ has been excluded, and $\mathcal{A}_t$ also holds. The deterministic induction step therefore gives $\mathcal{V}_t$. Backward induction proves that every $\mathcal{V}_t$ and every $\mathcal{Q}_t$ hold simultaneously on $\mathcal{F}^c$. Thus, for every $t < T$, the pointwise value and candidate-backup inequalities hold for every $i \in 1:C_t$ and $k \in 1:K_t$. Equivalently, the two displayed max-norm inequalities in the theorem hold with probability at least $1 - C_T \Delta_T^{\text{tail}} - \sum_{t=0}^{T-1} \Delta_t$.

It remains to prove the root-action consequence on the same event. Let $\hat{k}_0 \in \arg \max_{k \in 1:K_0} \hat{Q}_0^{1,k}$, so that $\hat{a}_0 = a_0^{1,\hat{k}_0}$ and $\hat{V}_0^1 = \hat{Q}_0^{1,\hat{k}_0}$. For any $a_0^* \in \text{Argmax}_0(s_0)$,

$$\begin{aligned} Q_0^*(s_0, a_0^*) - Q_0^*(s_0, \hat{a}_0) &= V_0^{*,1} - \hat{V}_0^1 + \hat{Q}_0^{1,\hat{k}_0} - Q_0^{*,1,\hat{k}_0} \\ &\leq \alpha_0 + \gamma\alpha_1 + \lambda_0 \leq 2\alpha_0, \end{aligned}$$

because $\gamma\alpha_1 + \lambda_0 \leq \gamma\alpha_1 + \lambda_0 + \Gamma_0 = \alpha_0$. $\square$

A.5 Proof of Corollary 4.1

Proof. If $\mathcal{A}$ is finite and $A_t^i = \mathcal{A}$ for every node, then

$$\max_{a \in A_t^i} Q_t^*(s_t^i, a) = \max_{a \in \mathcal{A}} Q_t^*(s_t^i, a) = V_t^*(s_t^i)$$

deterministically. Thus Definition 1 holds with $\Gamma_t = 0$ and $\delta_t^a = 0$. Substituting these parameters and $K_t = |\mathcal{A}|$ into Theorem 1 gives $\kappa_t = \lambda_t$ and $\Delta_t = C_t |\mathcal{A}| \delta_{\text{loc},t}(\lambda_t, C_{t+1})$.

For sampled action sets, Proposition 2 states that each nodewise action set is max-close with

$$\Gamma_t = L_t^a \varepsilon_t^{\beta_t^a}, \quad \delta_t^a = (1 - m_t(\varepsilon_t))^{K_t}.$$

Substituting these parameters into Theorem 1 gives

$$\kappa_t = \lambda_t + L_t^a \varepsilon_t^{\beta_t^a}, \quad \Delta_t = C_t K_t \delta_{\text{loc},t}(\lambda_t, C_{t+1}) + C_t (1 - m_t(\varepsilon_t))^{K_t}.$$

$\square$

A.6 Proof of Corollary 4.2

Proof. Fix a depth $t < T$ and a realized node-action pair $(s_t^i, a_t^{i,k})$. For this proof, the local concentration function $\delta_{\text{loc},t}$ is applied to the SNIS estimator whose target density is the smoothed density $p_{t,\tau_t}(\cdot | s_t^i, a_t^{i,k})$ and whose proposal is q_t^s . Thus, with the exact next-layer optimal values inserted into the smoothed SNIS backup, the local concentration event is

$$|\hat{Q}_{t,\tau_t}^{i,k,\star} - \mathbb{E}_{x \sim p_{t,\tau_t}(\cdot | s_t^i, a_t^{i,k})}[G_t^*(s_t^i, a_t^{i,k}, x)]| \leq \lambda_t,$$

except with probability at most $\delta_{\text{loc},t}(\lambda_t, C_{t+1})$. Proposition 1 gives the deterministic comparison

$$|\mathbb{E}_{x \sim p_{t,\tau_t}(\cdot | s_t^i, a_t^{i,k})}[G_t^*(s_t^i, a_t^{i,k}, x)] - Q_t^*(s_t^i, a_t^{i,k})| \leq L_t^s m_{\beta_t^s, t}(\tau_t).$$

Combining the two displays yields the static graph-backup error

$$|\hat{Q}_{t,\tau_t}^{i,k,*} - Q_t^{*,i,k}| \leq \lambda_t + L_t^s m_{\beta_t^s,t}(\tau_t)$$

with the same local failure probability. Assumption 4 then adds the usual propagated term $\gamma\alpha_{t+1}$ when $\|\hat{V}_{t+1}^{1:C_{t+1}} - V_{t+1}^{*,1:C_{t+1}}\|_\infty \leq \alpha_{t+1}$. The proof of Proposition 3, with λ_t replaced by $\lambda_t + L_t^s m_{\beta_t^s,t}(\tau_t)$, gives the corollary. $\square$

A.7 Proof of Corollary 4.3

Proof. In the finite-action fully enumerated case, Corollary 4.1 gives $\Gamma_t = 0$, $\delta_t^a = 0$, and $K_t = |\mathcal{A}|$. With an exact terminal estimate, $\kappa_T^{\text{tail}} = \Delta_T^{\text{tail}} = 0$. Set $C_0 = 1$ and choose a common successor-layer width $C_t = C$ for $1 \leq t \leq T$ and a common local radius $\lambda_t = \lambda$. The recursion in Definition 3 gives

$$\alpha_0 = \sum_{t=0}^{T-1} \gamma^t \lambda \leq T\lambda,$$

because $\gamma \leq 1$ in the finite-horizon setting. Thus $\lambda = \epsilon/(2T)$ ensures $2\alpha_0 \leq \epsilon$.

We next choose C so that the graph failure probability is at most δ . The bounded reward assumption implies, for every $t < T$,

$$\|V_{t+1}^*\|_\infty \leq (T - t - 1)R_{\max}.$$

Therefore the single-sample Bellman target satisfies

$$\|G_t^*(s, a, \cdot)\|_\infty \leq R_{\max} + \gamma(T - t - 1)R_{\max} \leq TR_{\max}, \quad B_T \triangleq TR_{\max}.$$

By the uniform overlap assumption, $d_\infty(p_t(\cdot | s, a) \| q_t^s) \leq d_\infty^{\max}$. Applying Theorem with $f_{\max} = B_T$, $N = C$, and $d_\infty(p \| q) \leq d_\infty^{\max}$ gives

$$\delta_{\text{loc},t}(\lambda, C) \leq 3 \exp\left(-C \left(\frac{\lambda}{B_T d_\infty^{\max}} - \frac{1}{\sqrt{C}}\right)^2\right),$$

provided $\lambda > B_T d_\infty^{\max} / \sqrt{C}$. Define

$$A_\lambda \triangleq \left(\frac{B_T d_\infty^{\max}}{\lambda}\right)^2, \quad \ell_\lambda \triangleq \max\left\{1, \log\left(\frac{12eT|\mathcal{A}|A_\lambda}{\delta}\right)\right\}.$$

Choose

$$C \geq 8A_\lambda \ell_\lambda.$$

Then $C \geq 4A_\lambda$, so the positivity condition in Theorem holds and

$$\frac{\lambda}{B_T d_\infty^{\max}} - \frac{1}{\sqrt{C}} = \frac{1}{\sqrt{A_\lambda}} - \frac{1}{\sqrt{C}} \geq \frac{1}{2\sqrt{A_\lambda}}.$$

Consequently, uniformly over depths and realized node-action pairs,

$$\delta_{\text{loc},t}(\lambda, C) \leq 3 \exp\left(-\frac{C}{4A_\lambda}\right).$$

It remains to check the graph-level union bound. Since $C_0 = 1$ and $C_t = C$ for $t \geq 1$, $\sum_{t=0}^{T-1} C_t \leq TC$ for $C \geq 1$. Theorem 1 and Corollary 4.1 therefore require

$$\sum_{t=0}^{T-1} C_t |\mathcal{A}| \delta_{\text{loc},t}(\lambda, C) \leq 3TC|\mathcal{A}| \exp\left(-\frac{C}{4A_\lambda}\right).$$

Let $x = C/(4A_\lambda)$ and $M = 12eT|\mathcal{A}|A_\lambda/\delta$. The choice of C gives $x \geq 2 \max\{1, \log M\}$. For such x , we have $xe^{-x} \leq e/M$: if $M \leq e$, then $x \geq 2$ gives $xe^{-x} \leq 2e^{-2} \leq e/M$, while if $M > e$, then xe^{-x} is decreasing for $x \geq 1$ and $2 \log(M)/M^2 \leq e/M$. Hence

$$3TC|\mathcal{A}| \exp(-C/(4A_\lambda)) = 12T|\mathcal{A}|A_\lambda xe^{-x} \leq \delta.$$

Thus the total failure probability in Theorem 1 is at most δ .

Substituting $\lambda = \epsilon/(2T)$ and $B_T = TR_{\max}$ gives

$$A_\lambda = \frac{4R_{\max}^2(d_\infty^{\max})^2T^4}{\epsilon^2}.$$

Hence the explicit width choice above is

$$C \geq \frac{32R_{\max}^2(d_\infty^{\max})^2T^4}{\epsilon^2} \max \left\{ 1, \log \left(\frac{48eT|\mathcal{A}|R_{\max}^2(d_\infty^{\max})^2T^4}{\delta\epsilon^2} \right) \right\}.$$

Equivalently,

$$C = \tilde{O} \left(\frac{R_{\max}^2(d_\infty^{\max})^2T^4}{\epsilon^2} \right),$$

where the logarithm is the one displayed above. Theorem 1 then gives the stated root-action bound. Since the graph uses T shared successor layers of width C , the simulator-query count scales as $N_{\text{GSS}} = TC$. For the exact dense SNIS backup, count one backup edge operation for one parent-action-child contribution in the sum (1.A). At depth t , there are $C_t|\mathcal{A}|$ enumerated parent-action pairs, and each pair is evaluated against all C_{t+1} child states. Hence the total number of backup operations is

$$\sum_{t=0}^{T-1} C_t|\mathcal{A}|C_{t+1} = |\mathcal{A}|(C + (T-1)C^2) \leq |\mathcal{A}|TC^2,$$

where $C_0 = 1$, $C_t = C$ for $1 \leq t \leq T$, and $C \geq 1$. Substituting the width bound gives

$$O(|\mathcal{A}|TC^2) = \tilde{O} \left(\frac{|\mathcal{A}|R_{\max}^4(d_\infty^{\max})^4T^9}{\epsilon^4} \right).$$

Thus, if $d_\infty^{\max} = O(T^n)$ for fixed $n \in \mathbb{N}$, N_{GSS} scales polynomially in T . $\square$

Example of polynomial overlap growth. The proof above uses the overlap bound only at realized node-action pairs. As a deliberately simple example in which $d_\infty^{\max}$ grows polynomially in T , suppose that each transition moves the state by at most $\ell > 0$, so that $p_t(\cdot | s, a)$ is supported in the ball $\mathbb{B}(s, \ell) \subseteq \mathbb{R}^{n_s}$, and that the transition densities are bounded, $p_t(x | s, a) \leq \bar{p}$ for all t, s, a, x . Let each state proposal be uniform on the ball of states reachable from the root, $q_t^s \triangleq \text{Unif}(\mathbb{B}(s_0, (t+1)\ell))$. Then every node at depth t lies in $\mathbb{B}(s_0, t\ell)$, the support of $p_t(\cdot | s_t^i, a)$ lies in $\mathbb{B}(s_0, (t+1)\ell)$, and for every $t < T$,

$$d_\infty(p_t(\cdot | s_t^i, a) \| q_t^s) \leq \bar{p} \text{Vol}(\mathbb{B}(s_0, (t+1)\ell)) = \bar{p} \omega_{n_s} ((t+1)\ell)^{n_s} \leq \bar{p} \omega_{n_s} \ell^{n_s} T^{n_s},$$

where ω_{n_s} is the volume of the unit ball in $\mathbb{R}^{n_s}$. Hence $d_\infty^{\max} = O(T^{n_s})$, and the width and query bounds above become $C = \tilde{O}(T^{4+2n_s}/\epsilon^2)$ and $N_{\text{GSS}} = \tilde{O}(T^{5+2n_s}/\epsilon^2)$, which are polynomial in T for a fixed state dimension. This proposal ignores all problem structure and differs from the proposals used in our experiments.

We use the following planner-to-policy conversion, adapted to our notation from Kearns et al. [2002, Lemma 5].

Theorem A.1 (Planner-to-policy conversion). *Let a planning subroutine be invoked at states encountered by the induced policy. For the discounted infinite-horizon setting, suppose that for every state s , the returned action $\hat{a}(s)$ satisfies*

$$\mathbb{P}(Q^*(s, a^*) - Q^*(s, \hat{a}(s)) \leq \beta) \geq 1 - \rho, \quad a^* \in \arg \max_{a \in \mathcal{A}} Q^*(s, a).$$

If $\|V^\|_\infty \leq V_{\max}$, then the induced randomized policy π satisfies*

$$V^*(s) - V^\pi(s) \leq \frac{\beta + 2\rho V_{\max}}{1 - \gamma} \quad \text{for every } s.$$

For a finite horizon of length T , if the same condition holds at every state and time with Q_t^ in place of Q^* and $\|Q_t^*\|_\infty \leq TR_{\max}$, then the induced time-dependent policy satisfies*

$$V_0^*(s) - V_0^\pi(s) \leq T(\beta + 2\rho TR_{\max}).$$

Additional policy-value scalings. The main text states the root-action consequence because it follows directly from Theorem 1. For completeness, we record how the same local-width calculation changes when the planner is invoked repeatedly to induce a policy. The finite-horizon part of Theorem A.1 applies once the planner has a per-call action-value gap bound β and per-call failure probability ρ . Let $\bar{\beta} \triangleq \beta + 2\rho TR_{\max}$. The theorem gives the policy-value loss bound

$$V_0^*(s) - V_0^\pi(s) \leq T\bar{\beta} = T(\beta + 2\rho TR_{\max}).$$

Here Theorem 1 gives $\beta = 2\alpha_0 \leq 2T\lambda$ for one planner call. To make the accumulated deterministic gap at most $\epsilon/2$, choose $2T\lambda \leq \epsilon/(2T)$, equivalently $\lambda = \Theta(\epsilon/T^2)$. To make the accumulated failure contribution at most $\epsilon/2$, take $\rho = O(\epsilon/(R_{\max}T^2))$. Substituting this λ into the explicit width calculation above, with δ replaced by the per-call budget ρ , yields

$$C = \tilde{O}\left(\frac{R_{\max}^2(d_{\infty}^{\max})^2T^6}{\epsilon^2}\right), \quad N_{\text{GSS}} = \tilde{O}\left(\frac{R_{\max}^2(d_{\infty}^{\max})^2T^7}{\epsilon^2}\right).$$

For discounted infinite-horizon policy value, let $V_{\max} = R_{\max}/(1-\gamma)$ and truncate the discounted problem at $T = \tilde{O}(1/(1-\gamma))$ so that the tail satisfies $\gamma^T V_{\max} = O(\epsilon(1-\gamma))$. The local envelope becomes $B_\gamma \leq R_{\max} + \gamma V_{\max} = V_{\max}$. Choosing $\lambda = \Theta(\epsilon(1-\gamma)^2)$ gives $\alpha_0 \leq \lambda/(1-\gamma) = O(\epsilon(1-\gamma))$. The discounted part of Theorem A.1 then requires a per-call failure budget $\rho = O(\epsilon(1-\gamma)/V_{\max})$ so that $(2\alpha_0 + 2\gamma^T V_{\max} + 2\rho V_{\max})/(1-\gamma) = O(\epsilon)$. Substituting $B_\gamma = O(R_{\max}/(1-\gamma))$ and $\lambda = \Theta(\epsilon(1-\gamma)^2)$ into $A_\lambda = (B_\gamma d_{\infty}^{\max}/\lambda)^2$ gives

$$C = \tilde{O}\left(\frac{R_{\max}^2(d_{\infty}^{\max})^2}{\epsilon^2(1-\gamma)^6}\right).$$

Multiplying by the truncation depth $T = \tilde{O}(1/(1-\gamma))$ yields

$$N_{\text{GSS}} = \tilde{O}\left(\frac{R_{\max}^2(d_{\infty}^{\max})^2}{\epsilon^2(1-\gamma)^7}\right).$$

A.8 Proof of Corollary 4.4

Proof. Choose strictly positive deterministic budgets η_t^{est} , η_t^{moll} , and η_t^{act} so that the recursion

$$\bar{\alpha}_T = \kappa_T^{\text{tail}}, \quad \bar{\alpha}_t = \gamma\bar{\alpha}_{t+1} + \eta_t^{\text{est}} + \eta_t^{\text{moll}} + \eta_t^{\text{act}}, \quad t = T-1, \dots, 0,$$

satisfies $2\bar{\alpha}_0 \leq \varepsilon_{\text{root}}$. Equivalently,

$$\bar{\alpha}_0 = \gamma^T \kappa_T^{\text{tail}} + \sum_{t=0}^{T-1} \gamma^t (\eta_t^{\text{est}} + \eta_t^{\text{moll}} + \eta_t^{\text{act}}) \leq \varepsilon_{\text{root}}/2.$$

Such budgets exist whenever the terminal estimate is exact, or more generally once the terminal tail accuracy has been made small enough to leave positive slack. Choose positive failure allocations ζ_T^{tail} , ζ_t^{est} , and ζ_t^{act} such that

$$\zeta_T^{\text{tail}} + \sum_{t=0}^{T-1} (\zeta_t^{\text{est}} + \zeta_t^{\text{act}}) \leq \delta.$$

In the non-exact terminal case, the terminal estimator is understood to be chosen after the final width C_T is fixed so that $C_T \Delta_T^{\text{tail}} \leq \zeta_T^{\text{tail}}$ for the selected κ_T^{tail} . In the exact terminal case, take $\kappa_T^{\text{tail}} = \Delta_T^{\text{tail}} = 0$.

Construct the schedule forward in t , starting from $C_0 = 1$. At depth $t < T$, assume C_t has already been fixed. First choose τ_t so that

$$L_t^s m_{\beta_t^s, t}(\tau_t) \leq \eta_t^{\text{moll}},$$

which is possible by the assumed vanishing of the smoothing bias. Next choose $\varepsilon_t \in (0, \bar{\varepsilon}_t]$ so that

$$L_t^a \varepsilon_t^{\beta_t^a} \leq \eta_t^{\text{act}}.$$

For instance, when $L_t^a > 0$ it is enough to take $\varepsilon_t \leq \min\{\bar{\varepsilon}_t, (\eta_t^{\text{act}}/L_t^a)^{1/\beta_t^a}\}$; when $L_t^a = 0$, any $\varepsilon_t \in (0, \bar{\varepsilon}_t]$ suffices. Since Assumption 1 gives $m_t(\varepsilon_t) > 0$, choose

$$K_t \geq \left\lceil \frac{\log(C_t/\zeta_t^{\text{act}})}{m_t(\varepsilon_t)} \right\rceil.$$

Then, using $1 - x \leq e^{-x}$ for $x \geq 0$,

$$\begin{aligned} K_t m_t(\varepsilon_t) &\geq \log(C_t / \zeta_t^{\text{act}}), \\ C_t (1 - m_t(\varepsilon_t))^{K_t} &\leq C_t \exp(-K_t m_t(\varepsilon_t)) \leq C_t \exp(-\log(C_t / \zeta_t^{\text{act}})) = \zeta_t^{\text{act}}. \end{aligned}$$

Set

$$\rho_t^{\text{est}} \triangleq \frac{\zeta_t^{\text{est}}}{C_t K_t}.$$

Finally, use the assumed uniform controllability of the smoothed backup to choose C_{t+1} and the local SNIS radius λ_t so that

$$\lambda_t \leq \eta_t^{\text{est}}, \quad \delta_{\text{loc},t}(\lambda_t, C_{t+1}) \leq \rho_t^{\text{est}},$$

or equivalently $C_t K_t \delta_{\text{loc},t}(\lambda_t, C_{t+1}) \leq \zeta_t^{\text{est}}$. The same construction can use a common nonroot state width under a bounded- d_∞ condition for the fixed smoothed targets. Suppose that, uniformly over realized node-action pairs,

$$d_\infty(p_{t,\tau_t}(\cdot \mid s_t^i, a_t^{i,k}) \parallel q_t^s) \leq D_{\infty,t}$$

and that the corresponding Bellman targets are bounded by $f_{\text{max},t}$. Choose fixed local accuracy levels $0 < \lambda_t \leq \eta_t^{\text{est}}$ and, for a candidate common width C , define

$$C_t(C) \triangleq \begin{cases} 1, & t = 0, \\ C, & t \geq 1, \end{cases} \quad K_t(C) \triangleq \left\lceil \frac{\log(C_t(C) / \zeta_t^{\text{act}})}{m_t(\varepsilon_t)} \right\rceil, \quad \rho_t^{\text{est}}(C) \triangleq \frac{\zeta_t^{\text{est}}}{C_t(C) K_t(C)}.$$

Let

$$A_t \triangleq \left(\frac{f_{\text{max},t} D_{\infty,t}}{\lambda_t} \right)^2.$$

Choose a common integer C satisfying

$$C \geq \max_{t < T} 4A_t \log \left(\frac{3}{\rho_t^{\text{est}}(C)} \right).$$

Such a C exists because $K_t(C)$ grows logarithmically in C , so the right-hand side grows only logarithmically in C , while the left-hand side grows linearly. Set the actual widths and action budgets by $C_t = C_t(C)$ and $K_t = K_t(C)$. Since $\rho_t^{\text{est}}(C) < 1$, the same lower bound gives $C \geq 4A_t$ and hence

$$\frac{\lambda_t}{f_{\text{max},t} D_{\infty,t}} - \frac{1}{\sqrt{C}} = \frac{1}{\sqrt{A_t}} - \frac{1}{\sqrt{C}} \geq \frac{1}{2\sqrt{A_t}}.$$

Thus the positivity condition in the bounded- d_∞ SNIS concentration bound (10) holds, and for $C_{t+1} = C$,

$$\delta_{\text{loc},t}(\lambda_t, C) \leq 3 \exp \left(-C \left(\frac{\lambda_t}{f_{\text{max},t} D_{\infty,t}} - \frac{1}{\sqrt{C}} \right)^2 \right)$$

Therefore

$$\delta_{\text{loc},t}(\lambda_t, C) \leq 3 \exp \left(-\frac{C}{4A_t} \right) \leq \rho_t^{\text{est}}(C),$$

and hence $C_t(C) K_t(C) \delta_{\text{loc},t}(\lambda_t, C) \leq \zeta_t^{\text{est}}$ at every depth.

With these choices, Corollaries 4.1 and 4.2 give

$$\kappa_t \leq \eta_t^{\text{est}} + \eta_t^{\text{moll}} + \eta_t^{\text{act}}, \quad \Delta_t \leq \zeta_t^{\text{est}} + \zeta_t^{\text{act}}.$$

Therefore the recursion defining α_t in Definition 3 gives $\alpha_t \leq \bar{\alpha}_t$ for every depth, while the total failure probability is at most δ . The root-action conclusion of Theorem 1 then yields

$$Q_0^*(s_0, a_0^*) - Q_0^*(s_0, \hat{a}_0) \leq 2\alpha_0 \leq 2\bar{\alpha}_0 \leq \varepsilon_{\text{root}}$$

with probability at least $1 - \delta$, for every $a_0^* \in \text{Argmax}_0(s_0)$. $\square$

B Experimental Details

All experiments use the fixed-shape JAX GSS planner described in Section 5. The JAX GSS experiments were run on a GPU machine equipped with an NVIDIA GeForce RTX 2080 Ti with 11 GB VRAM. The Julia DPW/VPW baselines were run on a CPU machine with an AMD Ryzen 7 3700X 8-core/16-thread processor at 3.60 GHz. We record action-mixture counts in the order primitive/rollout/noisy/uniform. The primitive component is a domain-specific list of fixed actions, the rollout component repeats the guide-policy action, the noisy component adds Gaussian perturbations scaled by the action range, and the uniform component samples from the action box.

B.1 GSS Configuration

We use three state-proposal settings across the experiments. In Rotating DDI, parent-action components are scored by a sampled one-step transition target plus rollout tail, the top fraction is retained, and successor states are sampled from the retained transition-moment mixture. Lunar Lander uses the same top-fraction scoring, but the next-state proposal is one diagonal Gaussian fitted to the sampled one-step successors. Reacher uses an unpruned transition-moment mixture, without top-fraction pruning. In Rotating DDI and Lunar Lander, the SNIS backup also includes the direct successor sample (13), whose sampled transition and rollout tail also provide the scores used for top-fraction pruning.

Graph backups use either an exact-moment density, when the target transition law is the domain’s diagonal Gaussian transition, or a single-point density, when the target transition law is the single-point linearization of the smoothed transition density obtained by adding a positive diagonal bandwidth τ to the simulator transition moments. All GSS runs use rollout tail values; rollout tails use the same rollout policy as the action proposal.

B.2 DPW and VPW Baselines

The DPW and VPW baselines use the double progressive widening solver of MCTS.jl with parameters

$$(c_{\text{uct}}, k_a, \alpha_a, k_s, \alpha_s) = (60.0, 1.45, 0.60, 0.1, 0.30).$$

The rollout value estimator simulates the domain’s rollout policy under the domain transition. For DPW, the first action added at a node is the guide-policy action, and later widened actions are sampled uniformly from the action space. For VPW, the first action is chosen in the same way, but later widened actions are generated by VOO. In the DDI VPW runs, VOO uses global exploration probability 0.9 and a diagonal Gaussian local sampler with standard deviation 0.5 in each action dimension. In the Lunar VPW implementation, the local covariance is $\text{diag}(0.2^2, 0.5^2, 0.05^2)$.

B.3 Sample Counts

We count the samples each planner draws per decision: every state drawn from the transition model or from a state proposal, including the states visited by rollouts. GSS draws C_t states from the state proposal at each layer $t \geq 1$. For each state-action pair, it also draws a direct successor sample and rolls it out, and it rolls out each leaf state. Its rollouts run as fixed-length batches, so we count each rollout at its full capped length. DPW and VPW draw at most one new successor per search iteration and then roll out to the search depth, which is the episode cap (35 steps in Lunar Lander and 50 in Rotating DDI). Their runs record only the number of iterations, so we report the number of iterations times the search depth, an upper bound on their samples. Table 5 reports the Lunar Lander counts and Table 1 the Rotating DDI counts. Even against DPW and VPW runs with equal or longer planning times, GSS draws more than six times as many samples per decision.

Table 1: Rotating DDI samples per decision for each rotation α . For GSS, the budget is the action budget K and state width C ; for DPW and VPW, it is the per-decision time budget. The GSS count does not depend on α and is averaged over the decisions of a 50-step episode, during which its planning horizon shrinks with the remaining steps; DPW and VPW counts are upper bounds.

Planner	Budget	α					
		0.0	0.3	0.6	0.9	1.2	1.5
GSS	$K=64, C=512$	13.66M	13.66M	13.66M	13.66M	13.66M	13.66M
DPW	0.01s	$\leq 0.02\text{M}$	$\leq 0.01\text{M}$	$\leq 0.01\text{M}$	$\leq 0.01\text{M}$	$\leq 0.01\text{M}$	$\leq 0.07\text{M}$
DPW	0.0316s	$\leq 0.04\text{M}$	$\leq 0.03\text{M}$	$\leq 0.03\text{M}$	$\leq 0.03\text{M}$	$\leq 0.03\text{M}$	$\leq 0.12\text{M}$
DPW	0.1s	$\leq 0.13\text{M}$	$\leq 0.09\text{M}$	$\leq 0.09\text{M}$	$\leq 0.09\text{M}$	$\leq 0.10\text{M}$	$\leq 0.28\text{M}$
DPW	0.316s	$\leq 0.35\text{M}$	$\leq 0.25\text{M}$	$\leq 0.26\text{M}$	$\leq 0.25\text{M}$	$\leq 0.30\text{M}$	$\leq 0.59\text{M}$
DPW	1.0s	$\leq 0.90\text{M}$	$\leq 0.72\text{M}$	$\leq 0.72\text{M}$	$\leq 0.72\text{M}$	$\leq 0.83\text{M}$	$\leq 1.46\text{M}$
VPW	0.01s	$\leq 0.02\text{M}$	$\leq 0.01\text{M}$	$\leq 0.01\text{M}$	$\leq 0.01\text{M}$	$\leq 0.01\text{M}$	$\leq 0.05\text{M}$
VPW	0.0316s	$\leq 0.04\text{M}$	$\leq 0.03\text{M}$	$\leq 0.03\text{M}$	$\leq 0.03\text{M}$	$\leq 0.03\text{M}$	$\leq 0.09\text{M}$
VPW	0.1s	$\leq 0.12\text{M}$	$\leq 0.09\text{M}$	$\leq 0.09\text{M}$	$\leq 0.09\text{M}$	$\leq 0.10\text{M}$	$\leq 0.21\text{M}$
VPW	0.316s	$\leq 0.34\text{M}$	$\leq 0.25\text{M}$	$\leq 0.25\text{M}$	$\leq 0.25\text{M}$	$\leq 0.29\text{M}$	$\leq 0.56\text{M}$
VPW	1.0s	$\leq 0.88\text{M}$	$\leq 0.71\text{M}$	$\leq 0.71\text{M}$	$\leq 0.71\text{M}$	$\leq 0.80\text{M}$	$\leq 1.35\text{M}$

B.4 Rotating 4D Double-Integrator

The DDI benchmark uses the 4D double-integrator domain with $D = 4$. States are ordered as $s = (x, v)$ with $x, v \in \mathbb{R}^4$, and actions are accelerations $a \in [-2, 2]^4$. The initial state is $(\mathbf{1}_4, \mathbf{0}_4)$ and the goal is the origin. The rotating-frame experiments use

$$\Delta t = 0.2, \quad \sigma_x = \sigma_v = 0.02, \quad \gamma = 0.99.$$

We also set the drag substep count to 32 and enable the rotating-frame coupling. The implementation uses one Euler step when drag and nonlinear coupling are inactive, and 32 Euler substeps when the rotating-frame coupling is active; in this sweep, $\alpha = 0$ is the single-step nominal DDI case and nonzero α uses the 32-substep rotating-frame integration. Within each microstep,

$$x^+ = x + \Delta t_{\text{sub}} v, \quad v^+ = v + \Delta t_{\text{sub}} a_\alpha,$$

where

$$a_\alpha = a - 2\alpha Jv + \alpha^2 x$$

and J rotates each adjacent coordinate pair by 90° . The sampled transition then adds independent Gaussian noise with standard deviation 0.02 to each position and velocity coordinate. Episodes terminate when any coordinate leaves $[-2, 2]$ or after 50 steps. The reward uses the current state and clipped action,

$$r(s, a, s') = 1 - \frac{\|x\|_2^2 + \|a\|_2^2}{8D},$$

with no velocity cost.

The rotating-frame GSS sweep uses $\alpha \in \{0.0, 0.3, 0.6, 0.9, 1.2, 1.5\}$, 100 seeds, a 50-step planner horizon, and a 50-step rollout cap. GSS uses action budget 64, state width 512, action mixture 0/1/31/32 (one LQR guide action, 31 noisy-LQR actions, and 32 uniform actions), and an LQR guide and rollout policy. Graph backups use exact-moment SNIS densities with rollout tail values, and the state proposal is the top-fraction transition-moment mixture described above, with top fraction 0.95, fitted from transition moments and without proposal inflation, with direct successor samples (13) at all depths. The companion Julia DPW/VPW rotating-frame runs use the same DDI dynamics, 100 seeds, the same alpha grid, and per-action wall-clock budgets in seconds

$$0.01, 0.0316227766, 0.1, 0.3162277660, 1.0.$$

Tables 2, 3, and 4 report the returns, planning times, and early terminations of every planner in Figure 2a, with one row per planner and budget and one column per rotation α .

Table 2: Rotating DDI mean undiscounted return ± 2 standard errors over 100 seeds. For GSS, the budget is the action budget K and state width C ; for DPW and VPW, it is the per-decision time budget.

Planner	Budget	α					
		0.0	0.3	0.6	0.9	1.2	1.5
LQR	–	48.939 $\pm$ 0.006	48.783 $\pm$ 0.010	47.666 $\pm$ 0.031	42.465 $\pm$ 0.128	11.307 $\pm$ 0.201	4.255 $\pm$ 0.054
LQG	–	48.955 $\pm$ 0.007	48.757 $\pm$ 0.011	47.385 $\pm$ 0.039	40.162 $\pm$ 0.167	8.900 $\pm$ 0.138	4.071 $\pm$ 0.024
GSS	$K=64, C=512$	48.194 $\pm$ 0.035	48.085 $\pm$ 0.034	47.586 $\pm$ 0.045	46.796 $\pm$ 0.049	44.138 $\pm$ 0.839	37.258 $\pm$ 2.286
DPW	0.01s	48.172 $\pm$ 0.036	48.148 $\pm$ 0.037	47.362 $\pm$ 0.058	45.944 $\pm$ 0.076	42.266 $\pm$ 1.250	26.072 $\pm$ 3.492
DPW	0.0316s	48.006 $\pm$ 0.043	47.885 $\pm$ 0.042	47.215 $\pm$ 0.048	46.090 $\pm$ 0.064	43.345 $\pm$ 0.767	32.929 $\pm$ 2.960
DPW	0.1s	47.850 $\pm$ 0.045	47.750 $\pm$ 0.039	47.199 $\pm$ 0.051	46.118 $\pm$ 0.074	44.038 $\pm$ 0.081	34.781 $\pm$ 2.764
DPW	0.316s	47.992 $\pm$ 0.039	47.823 $\pm$ 0.038	47.354 $\pm$ 0.041	46.360 $\pm$ 0.053	44.391 $\pm$ 0.084	38.134 $\pm$ 2.151
DPW	1.0s	47.998 $\pm$ 0.032	47.920 $\pm$ 0.033	47.509 $\pm$ 0.035	46.546 $\pm$ 0.044	44.694 $\pm$ 0.074	39.242 $\pm$ 1.970
VPW	0.01s	48.113 $\pm$ 0.034	48.009 $\pm$ 0.031	47.333 $\pm$ 0.046	46.047 $\pm$ 0.082	42.892 $\pm$ 1.006	29.664 $\pm$ 3.290
VPW	0.0316s	47.972 $\pm$ 0.031	47.905 $\pm$ 0.031	47.299 $\pm$ 0.043	46.241 $\pm$ 0.049	43.343 $\pm$ 1.055	34.332 $\pm$ 2.798
VPW	0.1s	47.939 $\pm$ 0.029	47.836 $\pm$ 0.028	47.348 $\pm$ 0.038	46.390 $\pm$ 0.051	44.478 $\pm$ 0.071	36.641 $\pm$ 2.519
VPW	0.316s	48.081 $\pm$ 0.025	47.956 $\pm$ 0.023	47.557 $\pm$ 0.030	46.649 $\pm$ 0.042	44.761 $\pm$ 0.067	36.387 $\pm$ 2.640
VPW	1.0s	48.119 $\pm$ 0.023	48.064 $\pm$ 0.022	47.694 $\pm$ 0.027	46.825 $\pm$ 0.034	45.109 $\pm$ 0.055	38.262 $\pm$ 2.390

Table 3 reports the measured planning times. The GSS time depends on α because the transition model is evaluated with one Euler step at $\alpha = 0$ and with 32 Euler substeps for $\alpha > 0$.

Table 3: Rotating DDI mean planning time per decision in seconds. Budgets are as in Table 2.

Planner	Budget	α					
		0.0	0.3	0.6	0.9	1.2	1.5
GSS	$K=64, C=512$	0.189	0.579	0.579	0.579	0.586	0.655
DPW	0.01s	0.041	0.026	0.027	0.027	0.028	0.032
DPW	0.0316s	0.047	0.041	0.041	0.041	0.041	0.042
DPW	0.1s	0.135	0.118	0.120	0.118	0.118	0.118
DPW	0.316s	0.405	0.362	0.364	0.364	0.363	0.358
DPW	1.0s	1.073	1.049	1.051	1.053	1.053	1.048
VPW	0.01s	0.033	0.027	0.027	0.027	0.027	0.029
VPW	0.0316s	0.047	0.041	0.041	0.041	0.041	0.043
VPW	0.1s	0.129	0.118	0.119	0.119	0.117	0.119
VPW	0.316s	0.402	0.364	0.363	0.363	0.364	0.360
VPW	1.0s	1.071	1.055	1.052	1.051	1.053	1.050

Episodes end early only by leaving the state box. No planner leaves it for $\alpha \leq 0.9$; at $\alpha \geq 1.2$, LQR and LQG leave it in every episode.

Table 4: Rotating DDI episodes, out of 100, that end early by leaving the state box. Budgets are as in Table 2.

Planner	Budget	α					
		0.0	0.3	0.6	0.9	1.2	1.5
LQR	–	0	0	0	0	100	100
LQG	–	0	0	0	0	100	100
GSS	$K=64, C=512$	0	0	0	0	1	11
DPW	0.01s	0	0	0	0	3	40
DPW	0.0316s	0	0	0	0	1	22
DPW	0.1s	0	0	0	0	0	18
DPW	0.316s	0	0	0	0	0	10
DPW	1.0s	0	0	0	0	0	8
VPW	0.01s	0	0	0	0	2	31
VPW	0.0316s	0	0	0	0	2	19
VPW	0.1s	0	0	0	0	0	14
VPW	0.316s	0	0	0	0	0	16
VPW	1.0s	0	0	0	0	0	12

B.5 Lunar Lander

The Lunar Lander domain follows the benchmark used by Mern et al. [2021] and Lim et al. [2021]. The state is $s = (x, z, \theta, v_x, v_z, \omega)$. The action is (F_x, T, δ) with

$$F_x \in [-5, 5], \quad T \in [0, 15], \quad \delta \in [-1, 1].$$

The parameters are mass $m = 1$, inertia $I = 10$, gravity 9.0, and $\Delta t = 0.4$. The transition computes

$$\begin{aligned}
 f_x &= \cos(\theta)F_x - \sin(\theta)T, & f_z &= \cos(\theta)T + \sin(\theta)F_x, \\
 \tau &= -\delta F_x, & \dot{\omega} &= \tau/I, \\
 v'_x &= v_x + \Delta t f_x/m + \epsilon_1 0.1, & v'_z &= v_z + \Delta t (f_z/m - 9.0) + \epsilon_2 0.1, \\
 \omega' &= \omega + \Delta t \dot{\omega} + \epsilon_3 0.01, & x' &= x + \Delta t v_x, \\
 z' &= z + \Delta t v_z, & \theta' &= \theta + \Delta t \omega,
 \end{aligned} \tag{14}$$

where $\epsilon_1, \epsilon_2, \epsilon_3$ are independent standard normal variables. The transition is therefore low-rank in the six-dimensional state, and GSS uses the single-point smoothed density. The default initial mean is $(0, 50, 0, 0, -10, 0)$ with diagonal initialization standard deviations

$$\sqrt{(0.1, 0.1, 0.01, 0.1, 0.1, 0.01)}.$$

Termination occurs when $|x| \geq 15$, $|\theta| \geq 0.5$, or $z \leq 1$. The reward is

$$r(s, a, s') = \begin{cases} -1000, & |x'| \geq 15 \text{ or } |\theta'| \geq 0.5, \\ 100 - |x'| - (v'_z)^2, & z' \leq 1, \\ -1, & \text{otherwise.} \end{cases}$$

The discount is $\gamma = 0.99$.

The Lunar Lander experiments use 100 seeds and a brake-near-ground policy for both guide and rollout. Graph backups use single-point smoothed densities with rollout tail values and smoothing bandwidth $(0.02, 0.02, 0.005, 0.02, 0.02, 0.002)$. The state proposal is the top-fraction transition-moment Gaussian described above, fitted to the sampled one-step successors, with top fraction 0.8 and inflated by a factor 1.1, and the backup includes the direct successor sample (13). The planner horizon is 12, the rollout cap and closed-loop cap are 24, and the GSS action and state sweep uses the following configurations:

action budget	state width	primitive/rollout/noisy/uniform
10	384	2/1/5/2
16	512	4/1/6/5
24	1024	6/1/8/9
40	1536	6/1/16/17
50	2048	6/1/21/22

The Julia Lunar tree curves use the same brake-near-ground guide and rollout policy, 100 seeds, a 35-step closed-loop cap, which does not affect the comparison since every DPW and VPW episode lands within 24 steps, and per-action budgets

0.0001, 0.000316228, 0.001, 0.00316228, 0.01, 0.0316228, 0.1, 0.316228, 1.0

seconds. DPW and VPW generate actions as described above: the guide-policy action first, then uniform (DPW) or VOO (VPW) samples. Table 5 reports the Lunar planner-performance rows corresponding to the plotted comparison.

Table 5: Lunar Lander planner performance from Figure 2b over 100 seeds. For GSS, the budget is the action budget K and state width C ; for DPW and VPW, it is the per-decision time budget. Time is the measured mean planning time per decision. Return entries are discounted mean episode return ± 2 standard errors. Landed counts the episodes that end by landing, and Samples is the number of states drawn per decision (Appendix B.3).

Planner	Budget	Time (s)	Return	Landed	Samples
GSS	$K=10, C=384$	0.0096	30.763 ± 40.288	96	0.77M
GSS	$K=16, C=512$	0.0114	65.407 ± 19.808	99	1.63M
GSS	$K=24, C=1024$	0.0287	75.570 ± 2.814	100	4.89M
GSS	$K=40, C=1536$	0.0529	78.641 ± 0.216	100	12.20M
GSS	$K=50, C=2048$	0.1065	78.618 ± 0.360	100	20.33M
DPW	0.01s	0.0196	73.582 ± 0.415	100	$\leq 0.11M$
DPW	0.0316s	0.0418	73.219 ± 0.513	100	$\leq 0.31M$
DPW	0.1s	0.1157	71.908 ± 0.632	100	$\leq 0.79M$
DPW	0.316s	0.3347	70.125 ± 0.454	100	$\leq 1.79M$
DPW	1.0s	1.0300	68.747 ± 0.502	100	$\leq 3.87M$
VPW	0.01s	0.0189	74.402 ± 0.707	100	$\leq 0.10M$
VPW	0.0316s	0.0414	73.795 ± 0.592	100	$\leq 0.28M$
VPW	0.1s	0.1159	73.391 ± 0.322	100	$\leq 0.71M$
VPW	0.316s	0.3385	71.161 ± 0.658	100	$\leq 1.68M$
VPW	1.0s	1.0315	69.835 ± 0.537	100	$\leq 3.66M$

Tree-planner returns across budgets. The DPW and VPW returns in Table 5 decrease as the budget grows beyond 0.01 seconds, while every DPW and VPW episode lands, so the decline reflects landing quality, namely touchdown position, touchdown speed, and episode length, rather than crashes. Progressive-widening planners do not guarantee monotone improvement at finite budgets, since a larger budget changes the value estimates and therefore the recommended action. The rollout-policy action is the first action expanded at every node, and we hypothesize that as the budget grows, the search departs further from the rollout policy before its tree is large enough to evaluate the alternative actions reliably.

B.6 Reacher

The Reacher benchmark is implemented in JAX [Bradbury et al., 2018] using the MJX interface to MuJoCo [Todorov et al., 2012], with a Gymnasium Reacher-v5-style model and observation/action convention [Towers et al., 2024]. The MJX model has $n_q = 4$, $n_v = 4$, and $n_u = 2$. The planner state is the 10D observation

$$s = (\cos q_0, \cos q_1, \sin q_0, \sin q_1, g_x, g_y, \dot{q}_0, \dot{q}_1, d_x, d_y),$$

where $g = (g_x, g_y)$ is the target coordinate and d is the fingertip-target displacement. Actions are 2D torques clipped to the XML actuator control range $[-1, 1]^2$. The transition converts the 10D observation back to MJX state variables, applies the clipped control for frame skip two with $\Delta t = 0.02$, and maps the resulting MJX data back to the same 10D observation. There is no transition noise and no early terminal condition. Initial states sample the two arm angles uniformly from $[-0.1, 0.1]$, arm angular velocities from $[-0.005, 0.005]$, and the target uniformly in a disk of radius 0.2. The reward is

$$r(s, a, s') = -\|d'\|_2 - \|a\|_2^2,$$

and the discount is $\gamma = 0.99$.

The default controller used for the PD baseline, guide policy, and rollout policy is a clipped task-space PD-style controller. From the observation, it recovers $q = (q_0, q_1)$ by applying atan2 to the sine and cosine coordinates, takes $\dot{q} = (\dot{q}_0, \dot{q}_1)$ from the velocity entries, and uses $d = (d_x, d_y)$ as the fingertip-target displacement. With link lengths $\ell_1 = \ell_2 = 0.1$, let

$$J(q) = \begin{pmatrix} -\ell_1 \sin q_0 - \ell_2 \sin(q_0 + q_1) & -\ell_2 \sin(q_0 + q_1) \\ \ell_1 \cos q_0 + \ell_2 \cos(q_0 + q_1) & \ell_2 \cos(q_0 + q_1) \end{pmatrix}$$

be the planar fingertip Jacobian. The implemented controller is

$$a_{\text{PD}}(s) = \text{clip}_{[-1,1]^2}(-12 J(q)^\top d - 0.35 \dot{q}).$$

Thus the proportional term acts on fingertip-target displacement through the Jacobian transpose, and the derivative term damps the two arm angular velocities.

Because the Reacher transition is deterministic in the ambient 10D observation representation, the domain only permits the single-point smoothed density with positive bandwidth. The Reacher GSS sweeps use dense SNIS backups with rollout tail values: graph leaves are valued by a single rollout under the default rollout policy, capped by the 50-step rollout/episode horizon. The state proposal is an unpruned transition-moment mixture fitted from transition moments, without proposal inflation; rollout-based proposal pruning and direct successor samples are not used. The action guide is the default controller, and each action set contains one guide action plus noisy-guide and uniform samples. The bandwidth vector is

$$(0.05, 0.05, 0.05, 0.05, 0.02, 0.02, 0.10, 0.10, 0.03, 0.03).$$

The closed-loop cap and rollout cap are 50 steps, and all Reacher GSS budget-scaling experiments use planner horizon 20 and 20 seeds. The reported sweep varies state width and action budget over

$$\begin{aligned} & (256, 24), (512, 24), (512, 32), (512, 48), (768, 32), \\ & (768, 48), (1024, 48), (1280, 48), (1024, 64), (1536, 48), \\ & (1280, 64), (1792, 48), (1536, 64), (2048, 48), (2048, 64), \end{aligned}$$

where each pair is (state width, action budget). The action mixtures, reported as rollout/noisy/uniform, are 1/3/20 for action budget 24, 1/4/27 for action budget 32, 1/6/41 for action budget 48, and 1/8/55 for action budget 64.

Table 6 reports the Reacher budget-scaling results used in Figure 2c, together with the PD controller, which makes no planner call.

Table 6: Reacher GSS budget-scaling rows and the PD controller. Time is mean planning time per decision in seconds. Return entries are undiscounted 50-step return, and final distance is the fingertip-target distance at the end of the episode, both ± 2 standard errors.

State width	Action budget	Time (s)	Return	Final distance
PD controller	–	–	-9.358 ± 2.464	0.159 ± 0.047
256	24	0.40	-8.006 ± 1.724	0.082 ± 0.019
512	24	0.54	-7.552 ± 1.554	0.081 ± 0.021
512	32	0.60	-7.549 ± 1.358	0.077 ± 0.015
512	48	0.80	-7.097 ± 1.276	0.067 ± 0.016
768	32	0.83	-6.982 ± 1.279	0.063 ± 0.017
768	48	1.21	-6.785 ± 1.156	0.058 ± 0.013
1024	48	1.70	-6.631 ± 0.720	0.060 ± 0.011
1280	48	2.27	-6.308 ± 0.907	0.061 ± 0.020
1024	64	2.40	-6.240 ± 0.919	0.055 ± 0.016
1536	48	2.93	-6.302 ± 0.924	0.054 ± 0.017
1280	64	3.22	-6.384 ± 1.025	0.052 ± 0.016
1792	48	3.49	-6.267 ± 0.860	0.058 ± 0.019
1536	64	4.10	-6.280 ± 0.903	0.054 ± 0.017
2048	48	4.23	-6.123 ± 0.796	0.056 ± 0.019
2048	64	5.69	-6.144 ± 0.855	0.052 ± 0.014

NeurIPS Paper Checklist

1. Claims

Question: Do the main claims made in the abstract and introduction accurately reflect the paper’s contributions and scope?

Answer: [\[Yes\]](#)

Justification: All of our reported results are either proven theoretically, or based on specific simulative experiments which we present in the paper and detail in the Appendix.

Guidelines:

- The answer [\[N/A\]](#) means that the abstract and introduction do not include the claims made in the paper.
- The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [\[No\]](#) or [\[N/A\]](#) answer to this question will not be perceived well by the reviewers.
- The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
- It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. Limitations

Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: [\[Yes\]](#)

Justification: Each theoretical result states its assumptions explicitly, and the Conclusion discusses the limitations of GSS and the scope of the guarantees, and Section 5 describes how the implementation departs from the analyzed algorithm. Appendix B.3 reports sample counts alongside the CPU/GPU time comparisons.

Guidelines:

- The answer [\[N/A\]](#) means that the paper has no limitation while the answer [\[No\]](#) means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate “Limitations” section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren’t acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. Theory assumptions and proofs

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: We rigorously state all assumptions in the paper and provide full theorem statements in the main paper. All proofs are provided in the appendix.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. Experimental result reproducibility

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: We detail the setup and configurations of all experiments in the paper Appendix.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general, releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
 - (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
 - (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
 - (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
 - (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. Open access to data and code

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: Our code and experimental data are publicly available at <https://github.com/IdanLevYehudi/GraphSparseSampling> under the MIT license. The repository documents the software environment, the configurations and commands that run GSS and the DPW/VPW baselines, and the packaged result data from which the paper figures are regenerated.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (<https://neurips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (<https://neurips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. Experimental setting/details

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

Answer: [Yes]

Justification: We explain the principal algorithmic choices in the main paper body, and write the full algorithmic details in the Appendix.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. Experiment statistical significance

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: We report error bars for all experiments and define them as ± 2 standard-error intervals in the figure and table captions.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. Experiments compute resources

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: The compute resources used for our experiments are detailed in the Appendix.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. Code of ethics

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics <https://neurips.cc/public/EthicsGuidelines>?

Answer: [Yes]

Justification: Reviewed the NeurIPS Code of Ethics and confirm that our research conforms to it.

Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. Broader impacts

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [N/A]

Justification: The work is theoretical and does not have direct societal impacts.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.

- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. Safeguards

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: The paper does not release any data or models.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. Licenses for existing assets

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: We cite all the original papers for the code packages and datasets we use, and we respect the licenses of all of them. No new dataset asset is introduced.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: We release our code and experimental data at <https://github.com/IdanLevYehudi/GraphSparseSampling> under the MIT license, documented in the repository.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: This research does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: This paper does not involve research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: The paper's method does not involve LLMs as any important, original, or non-standard components.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.