ICLR 2020 Spotlight Presentation – Full Version

Learning to Plan in High Dimensions via Neural Exploration-Exploitation Trees

Binghong Chen[1], Bo Dai[2], Qinjie Lin[3], Guo Ye[3], Han Liu[3], Le Song[1,4]

[1]Georgia Institute of Technology, [2]Google Research, Brain Team,
[3]Northwestern University, [4]Ant Financial

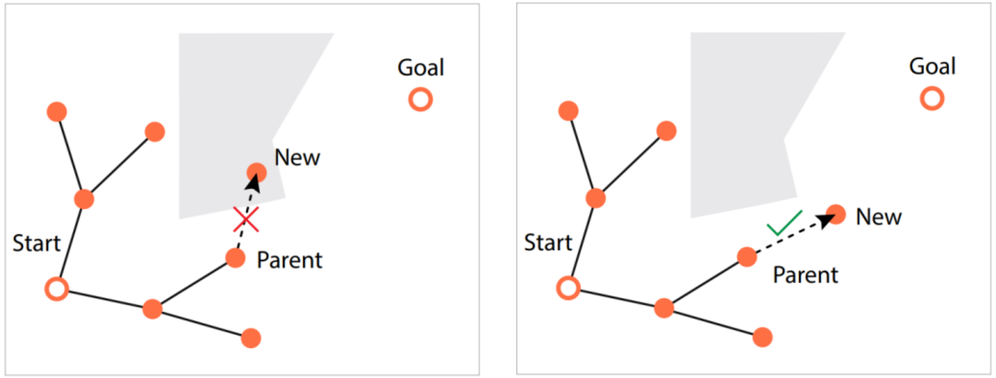
Task: Robot Arm Fetching

Problem: planning a robot arm to fetch a book on a shelf

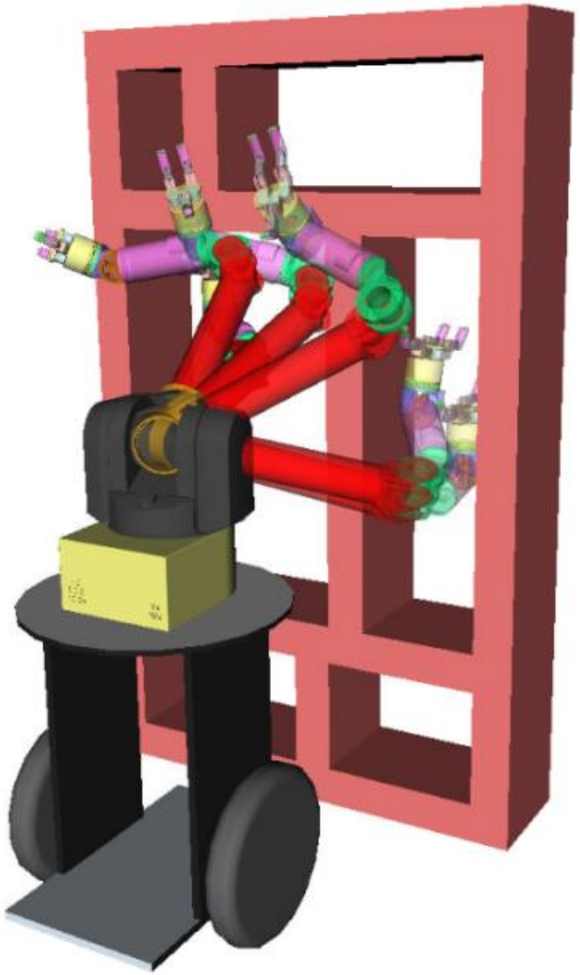
- Given $(state_{start}, state_{goal}, map_{obs})$
- Find the shortest collision-free path

Traditional planners: based on sampling the state space

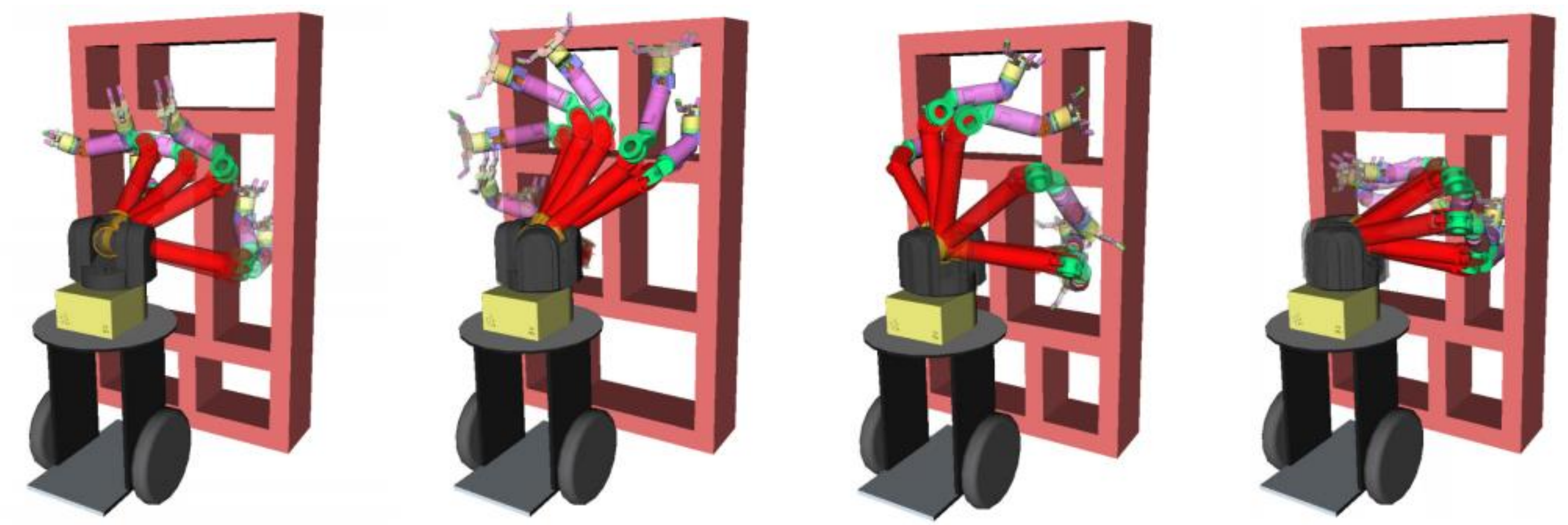
- Construct a tree to search for a path
- Use random samples from a uniform distribution to grow the tree



Sample inefficient



Daily Task: Robot Arm Fetching

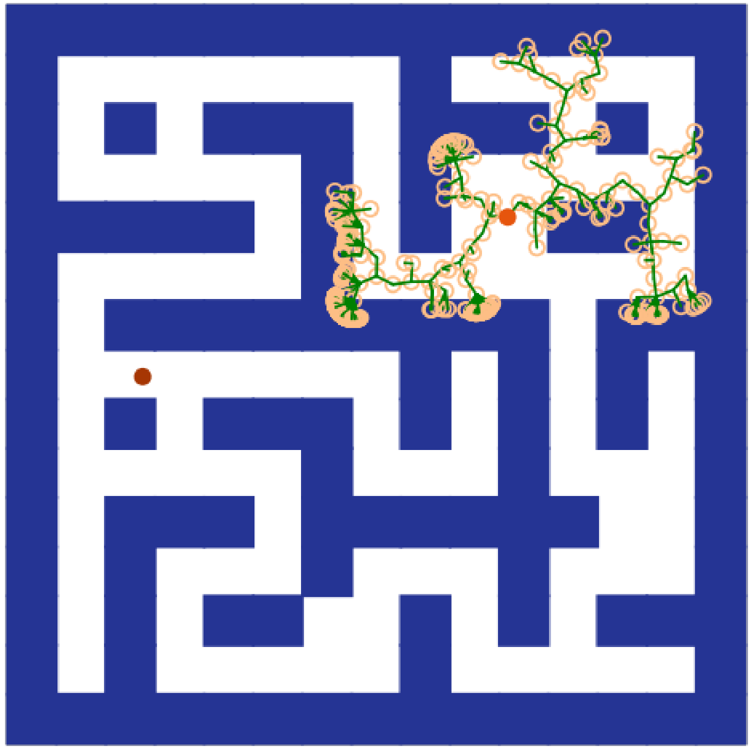


Similar problems are solved again and again with $(state_{start}, state_{goal}, map_{obs}) \sim Dist$

Can we learn from past planning experience to build a smarter sampling-based planner than simply uniform random sampling?

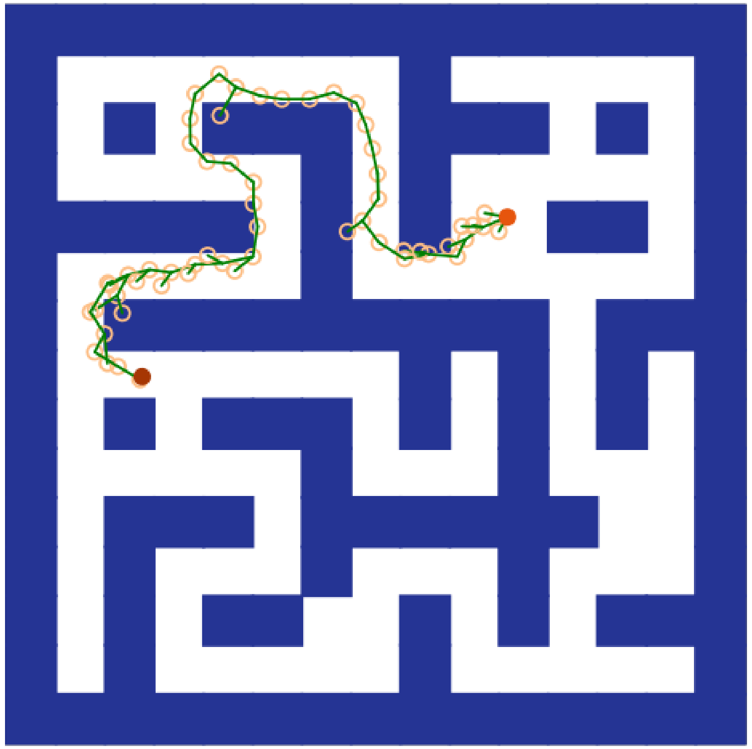
Yes! Learned planner → improve sample efficiency → solve problems in less time.

Sampling from Uniform v.s. Learned Dist



Uniform Distribution

Failed, 500 samples

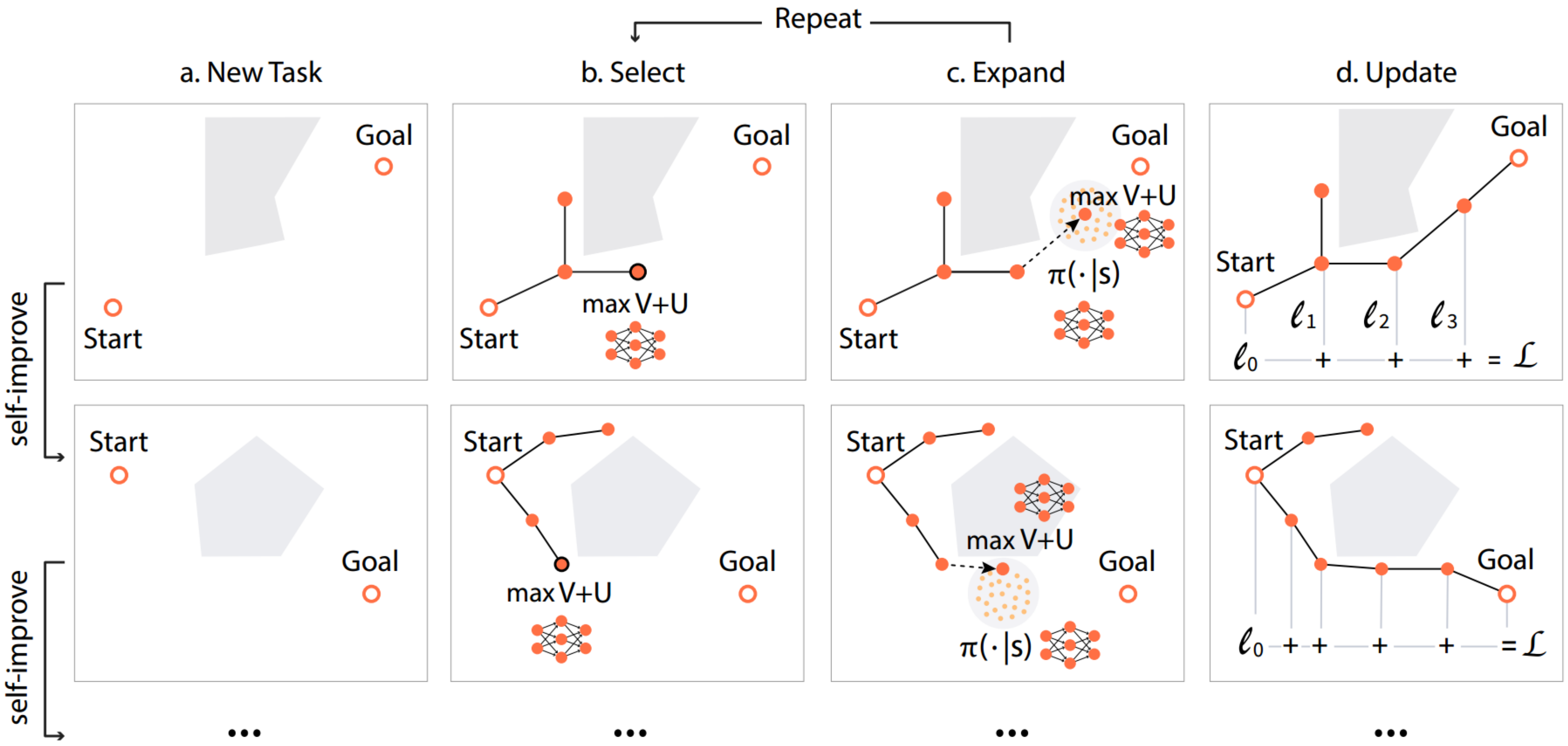


Learned $q_\theta(s|tree, s_{goal}, map_{obs})$

Success, <60 samples

Different sampling distributions for growing a search tree on the same problem.

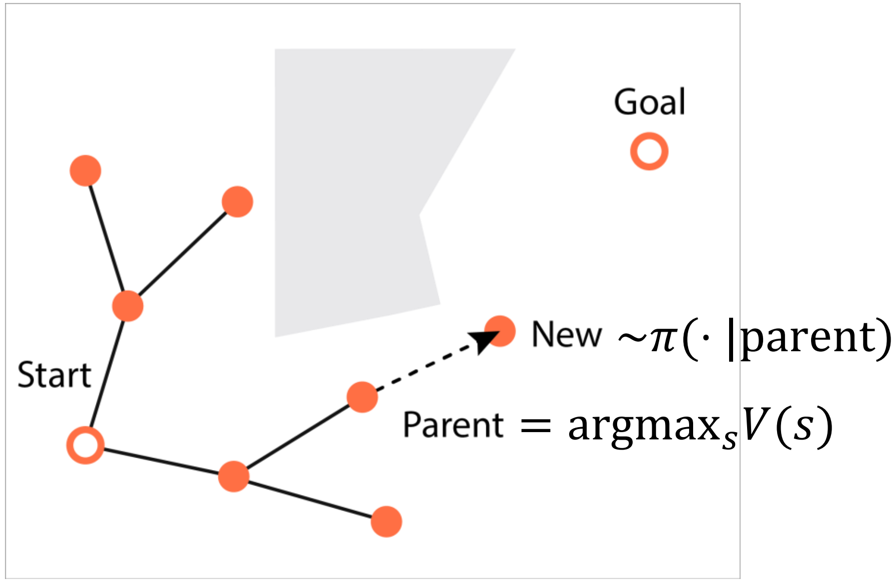
Neural Exploration-Exploitation Trees (NEXT)



- Learn a search bias from past planning experience and uses it to guide future planning.
- Balance exploration and exploitation to escape local minima.
- Improve itself as it plans.

Learning to Plan by Learning to Sample

1. Define a sampling policy $q_\theta(s|tree, s_{goal}, map_{obs})$.
2. Learn θ from past planning experience.



$q_\theta(s|tree, s_{goal}, map_{obs})$:

1. $s_{parent} = \text{argmax}_{s\in tree} V_\theta(s|s_{goal}, map_{obs})$;
2. $s_{new} \sim \pi_\theta(\cdot\,|s_{parent}, s_{goal}, map_{obs})$.

Pure exploitation policy

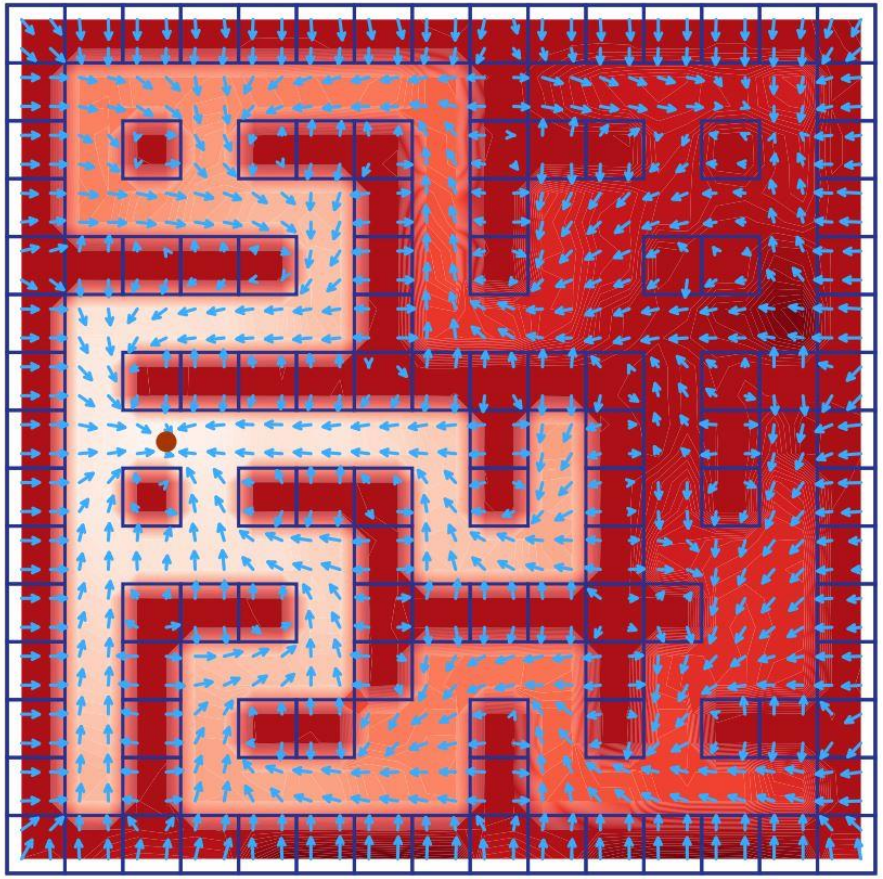
Learning Sampling Bias by Imitation

- The learned $\pi_\theta(\cdot \,|s_{parent}, s_{goal}, map_{obs})$ is pointing to the goal and avoiding the obstacles.
- The learned $V_\theta(s|s_{goal}, map_{obs})$ (heatmap) decreases as it moves away from the goal.

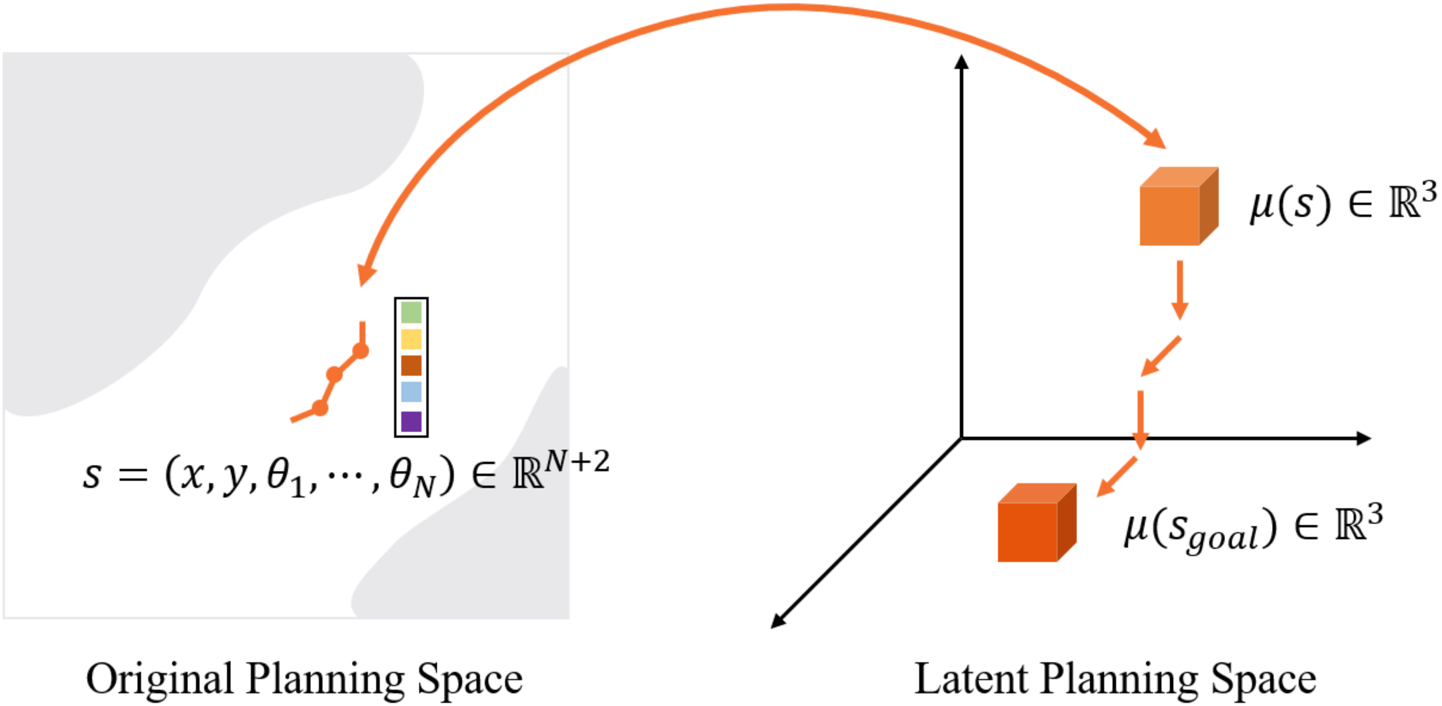
We learn π_θ and V_θ by imitating the shortest solutions from previous planning problems.

Minimize:

$$\sum_{i=1}^{n}\left(-\sum_{j=1}^{m-1}\log \pi_\theta(s_{j+1}|s_j) + \sum_{j=1}^{m}\left(V_\theta(s_j) - v_j\right)^2\right)$$

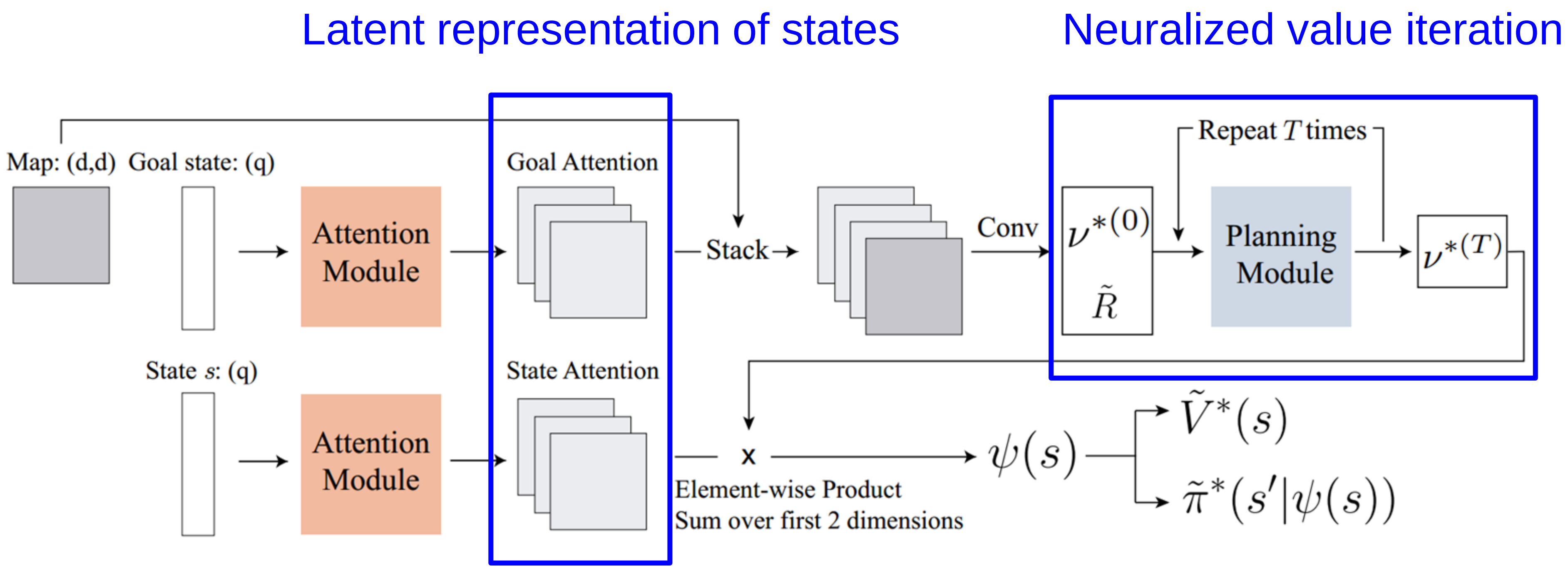


Value Iteration as Inductive Bias for π/V



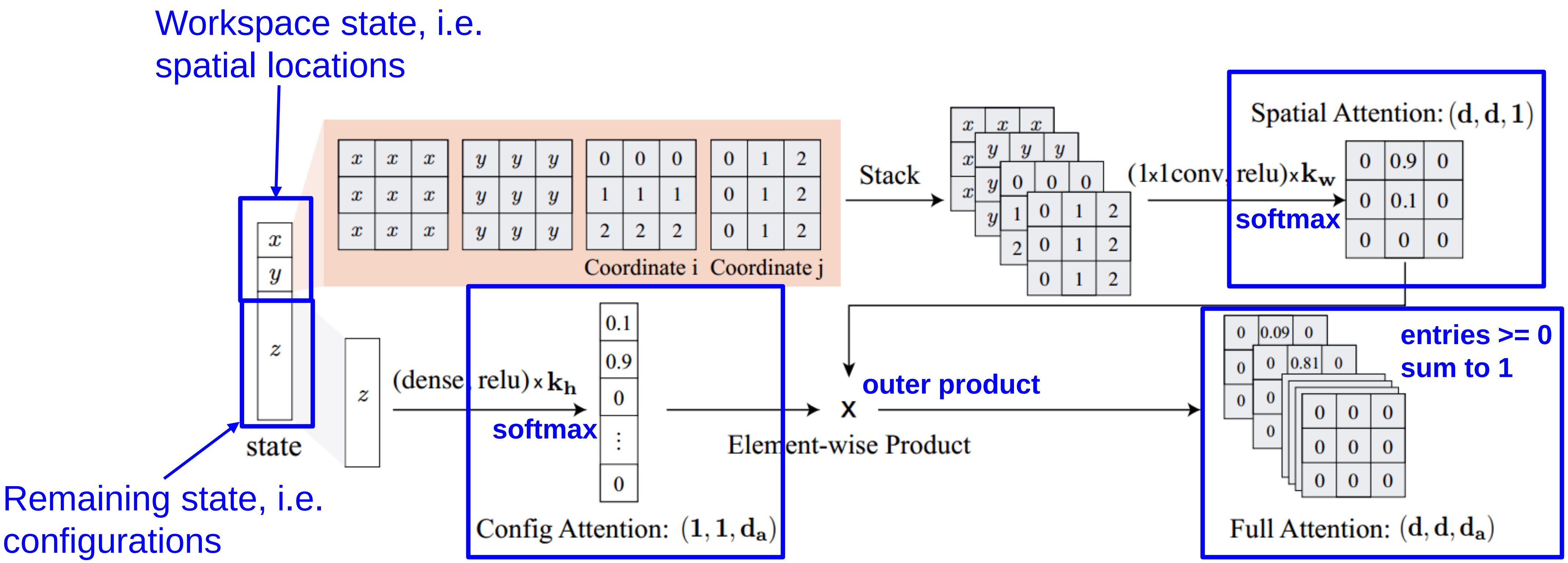
Intuition: first embedding the state and problem into a discrete latent representation via an attention-based module, then the neuralized value iteration (planning module) is performed to extract features for defining V_θ and π_θ.

Neural Architecture for π/V



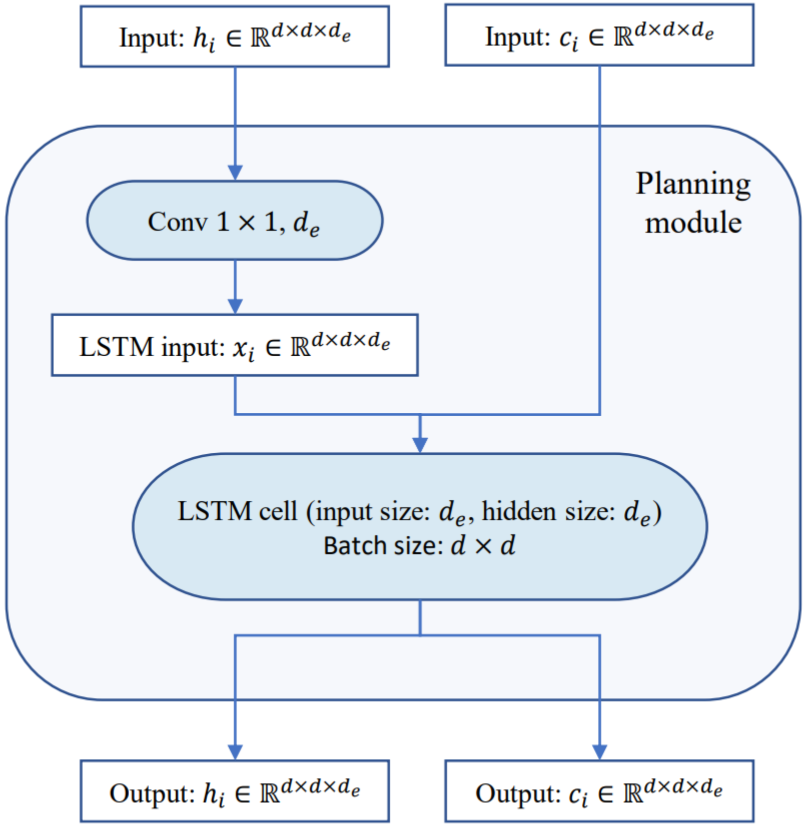
Overall model architecture.

Attention-based State Embedding



Attention module. For illustration purpose, assume workspace is 2d and map shape is 3x3.

Neuralized Value Iteration

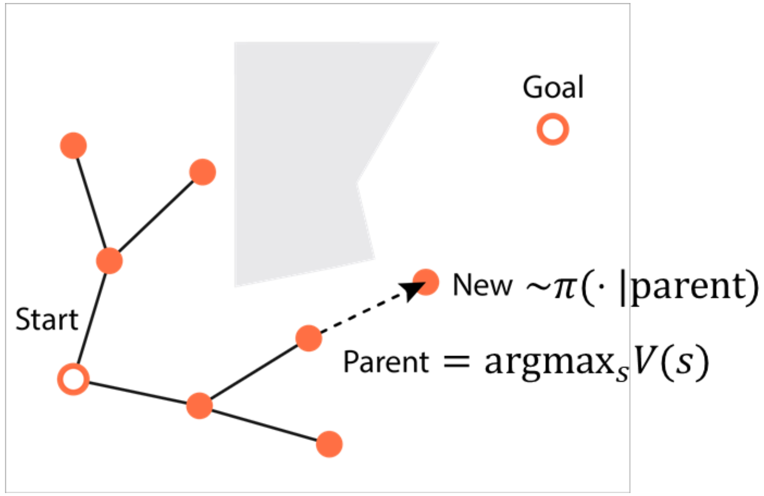


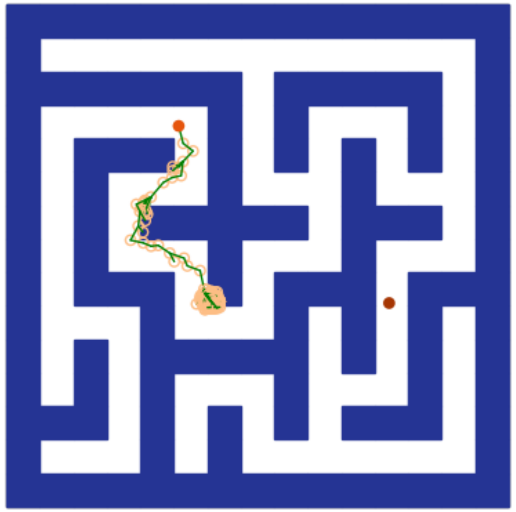
Planning module. Inspired by Value Iteration Networks*.

*Tamar, Aviv, et al. "Value iteration networks." *Advances in Neural Information Processing Systems*. 2016.

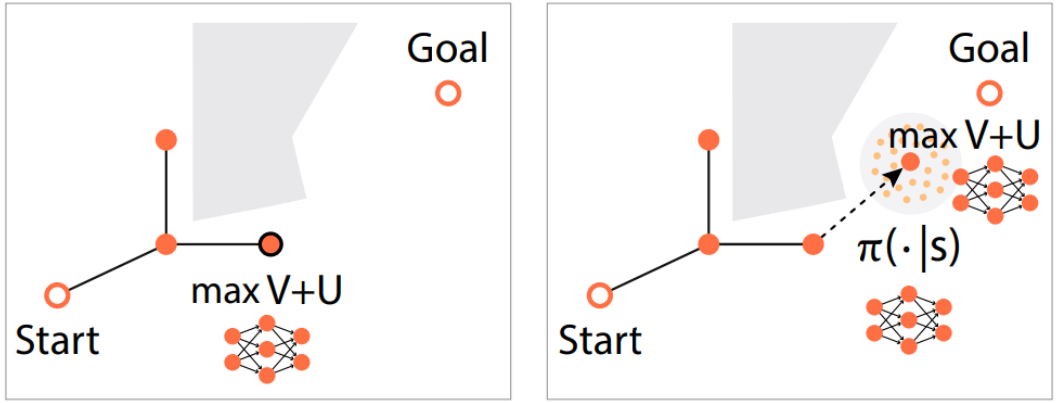
Escaping the Local Minima

Pure exploitation policy

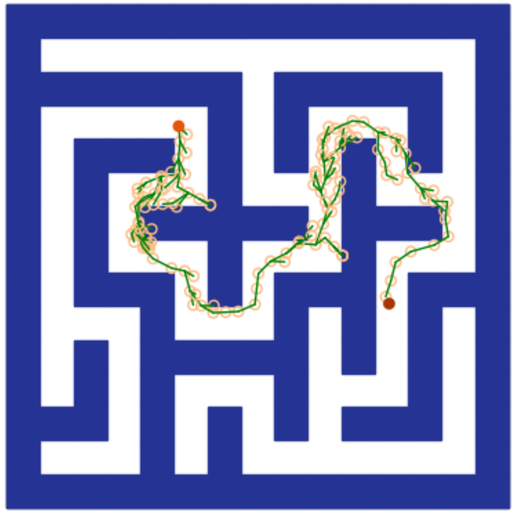




Exploration-exploitation policy



Variance U estimates the local density



AlphaGo-Zero-style Learning

Algorithm 3: Meta Self-Improving Learning

Initialize dataset $\mathcal{D}_0$;
for epoch $n \leftarrow 1$ **to** N **do**
  Sample a planning task U;
  $\mathcal{T} \leftarrow \mathtt{TSA}(U)$ with $\epsilon \sim \mathcal{U}nif[0,1]$, and $\epsilon \cdot \mathtt{RRT} :: \mathtt{Expand} + (1-\epsilon) \cdot \mathtt{NEXT} :: \mathtt{Expand}$;
  Postprocessing $\mathcal{T}$ with $\mathtt{RRT}^* :: \mathtt{Postprocess}$;
  $\mathcal{D}_n \leftarrow \mathcal{D}_{n-1} \cup \{(\mathcal{T}, U)\}$;
  for $j \leftarrow 0$ **to** L **do**
    Sample $(\mathcal{T}_j, U_j)$ from $\mathcal{D}_n$;
    Reconstruct optimal path $\{s^i\}_{i=1}^m$ and the cost of paths based on $\mathcal{T}_j$;
    Update parameters $W \leftarrow W - \eta \nabla_W \ell(\tilde{V}^*, \tilde{\pi}^*; \mathcal{T}_j, U_j)$;
  Anneal $\epsilon = \alpha\epsilon$, $\alpha \in (0,1)$;

Anneal ϵ

Use ϵ-exploration to improve the result

AlphaGo Zero
MCTS as a policy improvement operator.

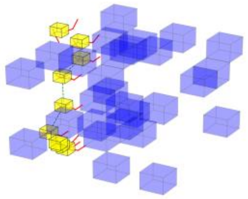
NEXT (Meta Self-Improving Learning)
Exploration with random sampling as a policy improvement operator!

Experiments

2D

3D

5D



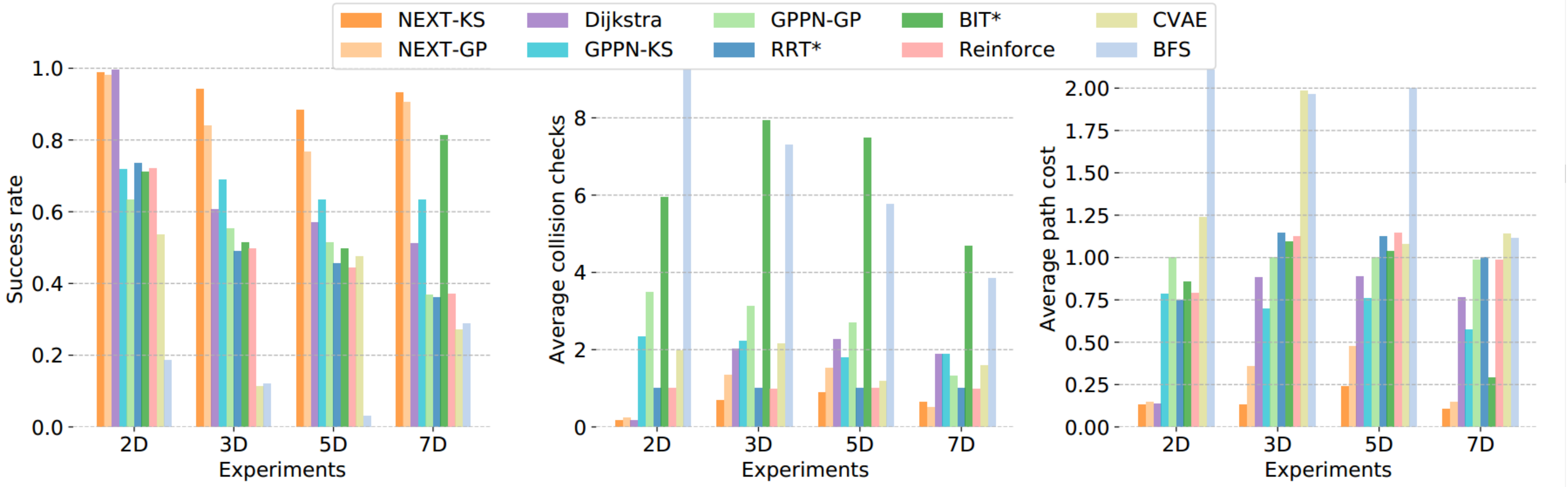
7D

(...) $\times$ 3000

distinct maps
(planning problems)
per environment

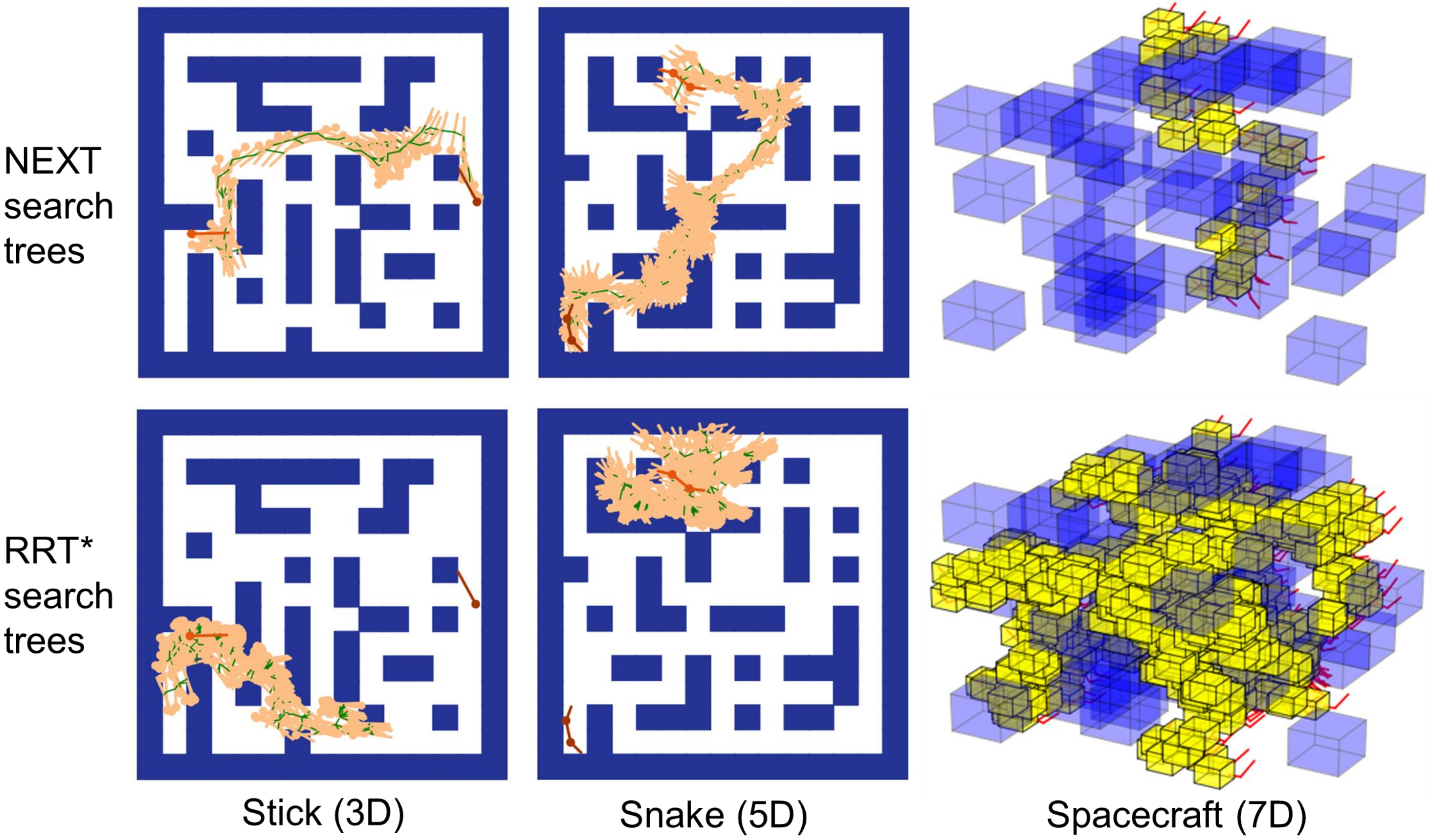
- 2000 Training (SL/RL/MSIL)
- 1000 testing

Experiment Results



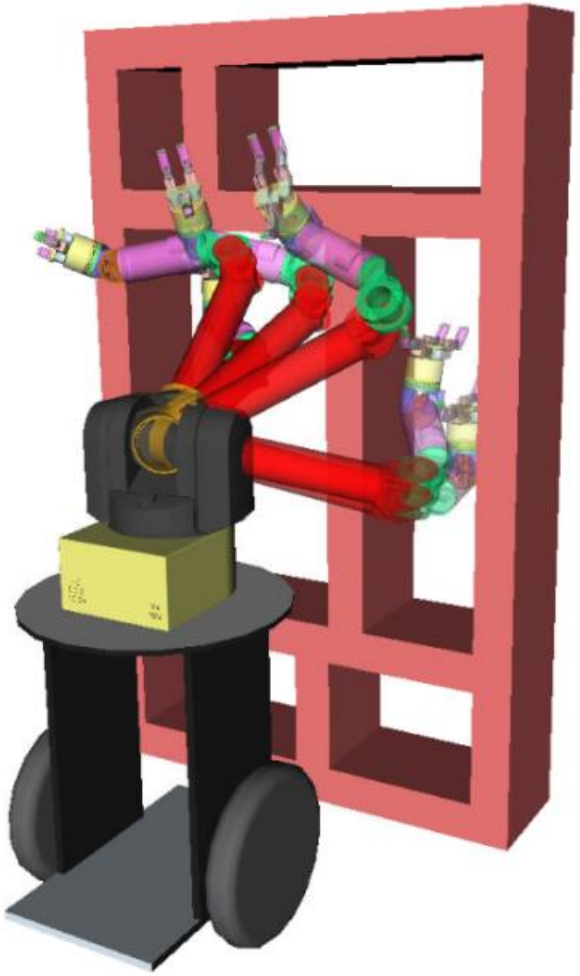
Learning-based							Non-learning		
Proposed method		Ablation studies			Baselines		Ablation	Baselines	
MSIL		Architecture		UCB	RL	SL	Heuristic	Sampling-based	
NEXT-KS	NEXT-GP	GPPN-KS	GPPN-GP	BFS	Reinforce	CVAE	Dijkstra	RRT*	BIT*

Search Tree Comparison



Case Study: Robot Arm Fetching

	5s	10s	15s	20s	25s	30s	35s	40s	45s	50s
NEXT	$\boxed{\textbf{0.579}}$	**0.657**	**0.703**	**0.709**	**0.745**	**0.743**	**0.746**	**0.752**	**0.772**	**0.763**
CVAE	0.354	0.437	0.482	0.509	0.507	0.539	0.553	0.551	0.579	$\boxed{0.580}$
Reinforce	0.160	0.170	0.200	0.150	0.180	0.190	0.175	0.180	0.225	0.175
BIT*	0.226	0.288	0.320	0.365	0.364	0.429	0.425	0.422	0.443	0.475
RRT*	0.135	0.148	0.144	0.136	0.147	0.159	0.152	0.158	0.157	0.165



Success rates of different planners under varying time limits, the higher the better.

Thanks for listening!

For more details, please refer to our paper/full slides/poster:

Paper

Full Slides

Poster