Monte Carlo Tree Diffusion for System 2 Planning

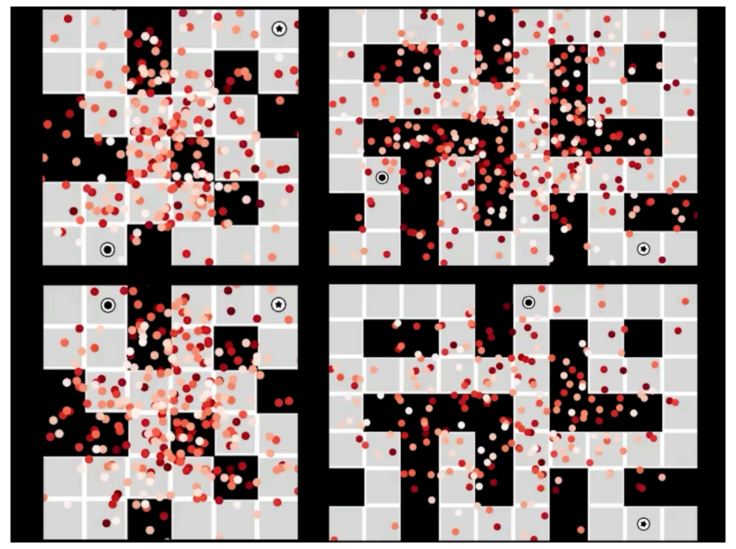
ICML 2025 (Spotlight), paper / webpage

Jaesik Yoon (KAIST & SAP), Hyeonseo Cho (KAIST), Doojin Baek (KAIST), Yoshua Bengio (Mila) and Sungjin Ahn (KAIST & NYU)

Motivation

How could we combine the strengths of Diffusion-based Planner and MCTS?

Diffusion Planner	**Monte Carlo Tree Search (MCTS)**
☺ Good to learn long-term planning	☺ Good for inference-time scaling
☺ Good for sparse reward setting	☺ Good for reasoning task
☹ Limited inference-time scaling	☹ Limited on high-dimensional space
☹ Lack of exploration strategy	☹ Lack of global consistency

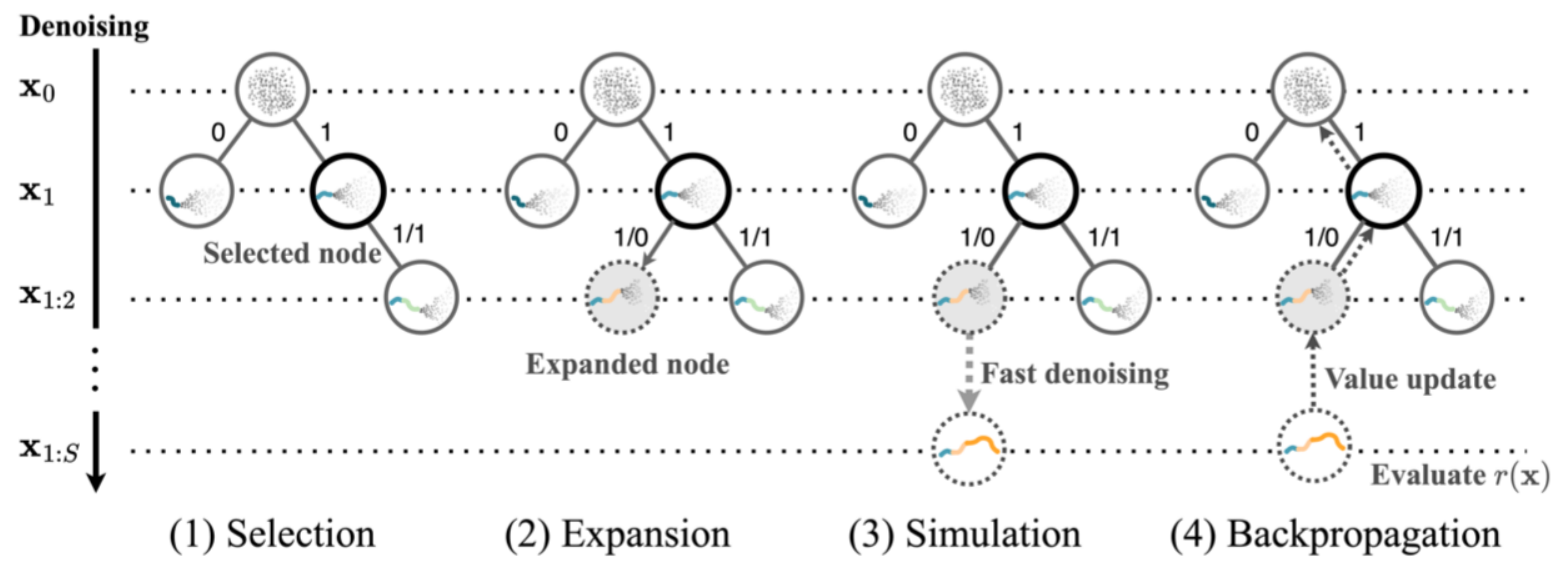


Chen, Boyuan, et al. "Diffusion forcing: Next-token prediction meets full-sequence diffusion." *Advances in Neural Information Processing Systems* 37 (2024): 24081-24125.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. Mastering the game of go with deep neural networks and tree search. nature, 529(7587):484–489, 2016.

Proposed Method

Monte Carlo Tree Diffusion (MCTD) Overview



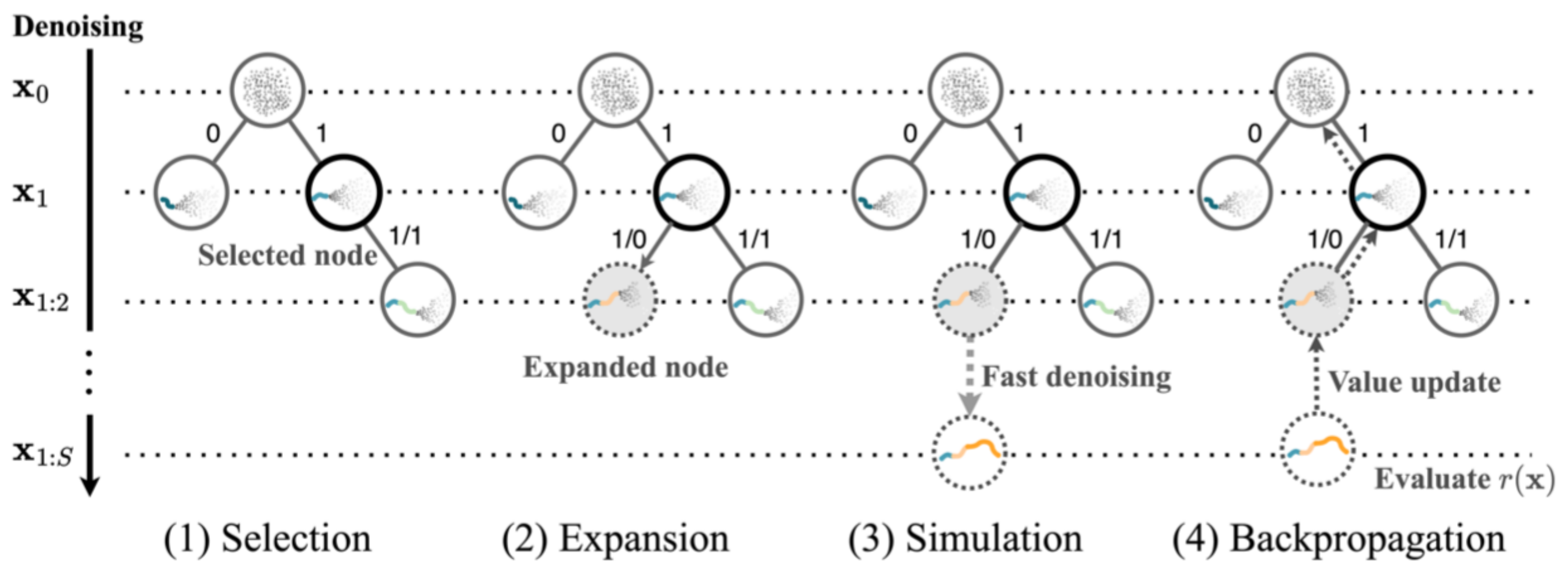
Monte Carlo Tree Diffusion (MCTD) is designed to integrate the diffusion-based trajectory generation with the interactive search capabilities of MCTS for more efficient and scalable planning.

It builds on three key concepts,

1. **Denoising as Tree-Rollout**: Partition the trajectory into subplans and expand them in a tree structure
2. **Guidance Levels as Meta-Actions**: Each node (subplan) can be sampled with different guidance intensities to navigate exploration vs. exploitation
3. **Jumpy Denoising for Simulation**: Rapid partial denoising to quickly evaluate rewards, analogous to MCTS rollout

Proposed Method

MCTD Algorithm Flow

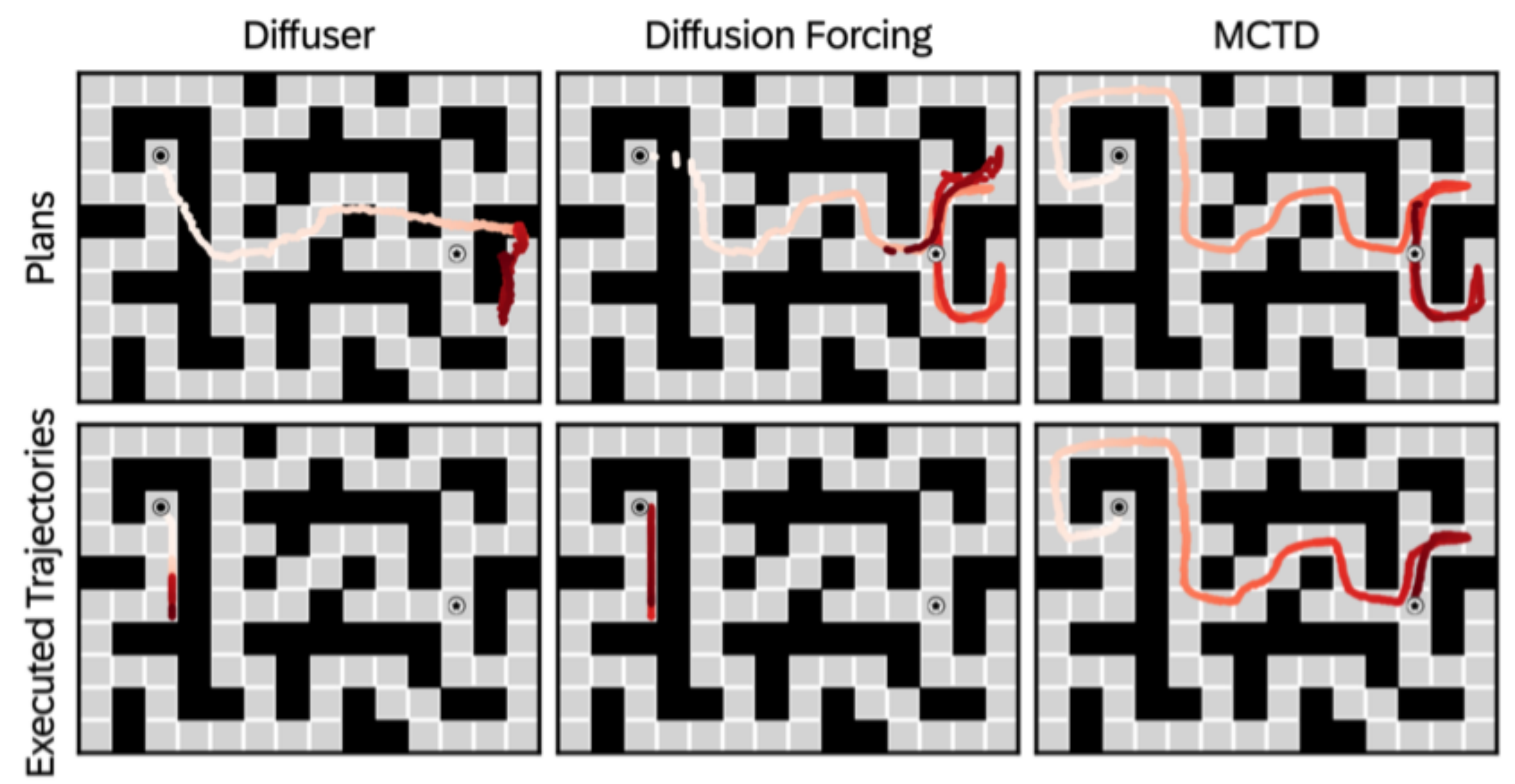


Algorithm 1 Monte Carlo Tree Diffusion

```
procedure MCTD($root$, $iterations$)
    for $i = 1$ to $iterations$ do
        $node \leftarrow root$
        while IsFullyExpanded($node$) and not
                IsLeaf($node$) do
            $node \leftarrow$ BestUCTChild($node$)
        end while
        if IsExpandable($node$) then
            $g_s \leftarrow$ SelectMetaAction($node$)
            $newSubplan \leftarrow$ DenoiseSubplan($node, g_s$)
            $child \leftarrow$ CreateNode($newSubplan$)
            AddChild($node, child$)
            $node \leftarrow child$
        end if
        $partial \leftarrow$ GetPartialTrajectory($node$)
        $remaining \leftarrow$ FastDenoising($partial$)
        $fullPlan \leftarrow (partial, remaining)$
        $reward \leftarrow$ EvaluatePlan($fullPlan$)
        while $node \neq null$ do
            $node.visitCount \leftarrow node.visitCount + 1$
            $node.value \leftarrow node.value + reward$
            UpdateMetaActionSchedule(
                $node, reward$)
            $node \leftarrow node.parent$
        end while
    end for
    return BestChild($root$)
end procedure
```

Experimental Results

Long-Horizon Maze



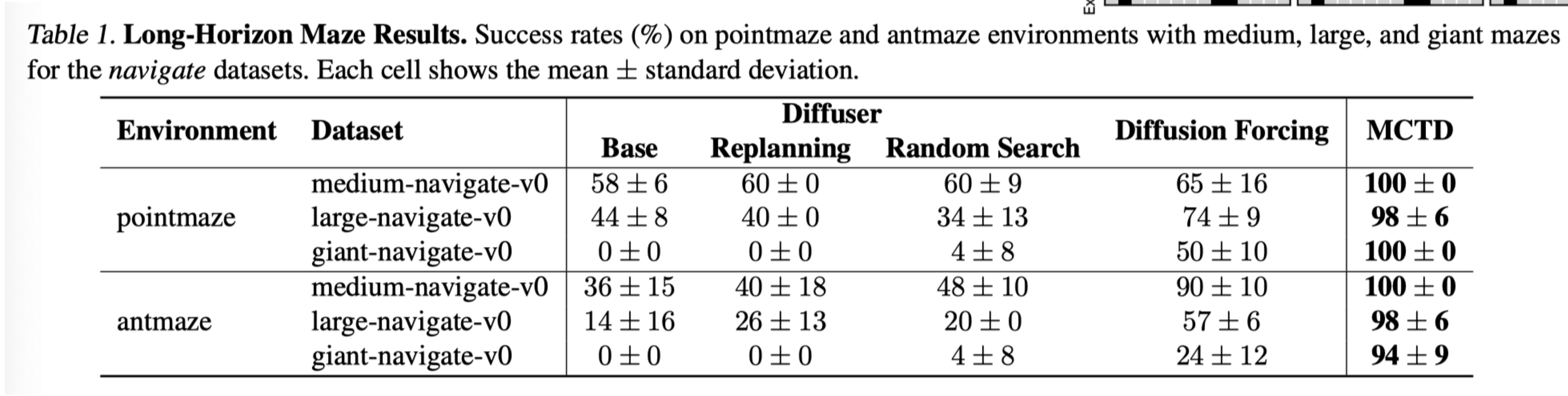


Table 1. **Long-Horizon Maze Results.** Success rates (%) on pointmaze and antmaze environments with medium, large, and giant mazes for the *navigate* datasets. Each cell shows the mean ± standard deviation.

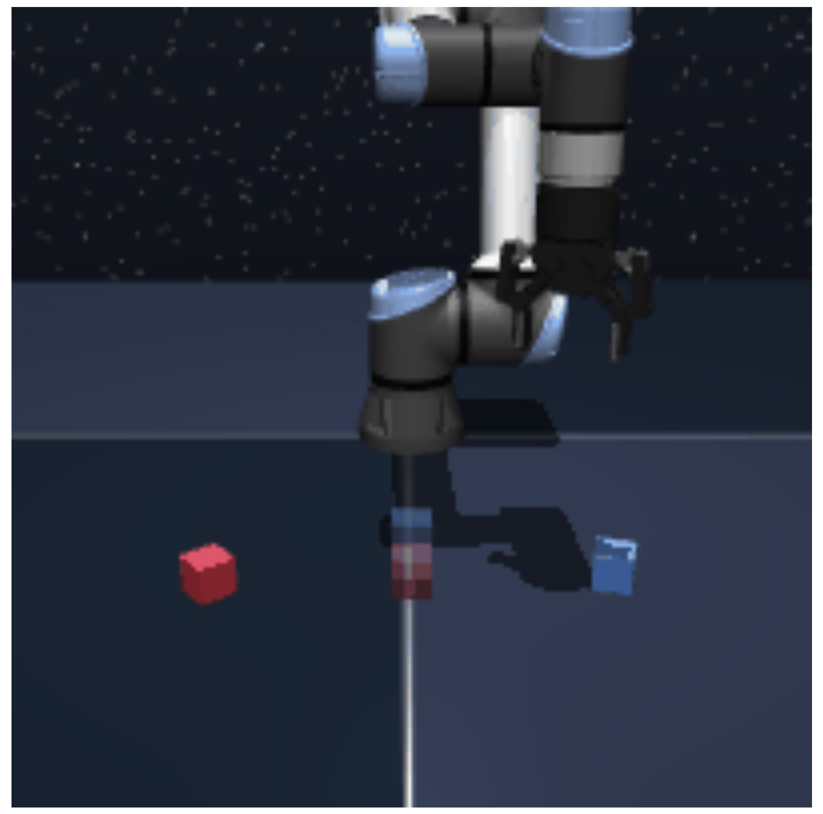
Environment	Dataset	Diffuser Base	Diffuser Replanning	Diffuser Random Search	Diffusion Forcing	MCTD
pointmaze	medium-navigate-v0	58 ± 6	60 ± 0	60 ± 9	65 ± 16	$\mathbf{100 \pm 0}$
	large-navigate-v0	44 ± 8	40 ± 0	34 ± 13	74 ± 9	$\mathbf{98 \pm 6}$
	giant-navigate-v0	0 ± 0	0 ± 0	4 ± 8	50 ± 10	$\mathbf{100 \pm 0}$
antmaze	medium-navigate-v0	36 ± 15	40 ± 18	48 ± 10	90 ± 10	$\mathbf{100 \pm 0}$
	large-navigate-v0	14 ± 16	26 ± 13	20 ± 0	57 ± 6	$\mathbf{98 \pm 6}$
	giant-navigate-v0	0 ± 0	0 ± 0	4 ± 8	24 ± 12	$\mathbf{94 \pm 9}$

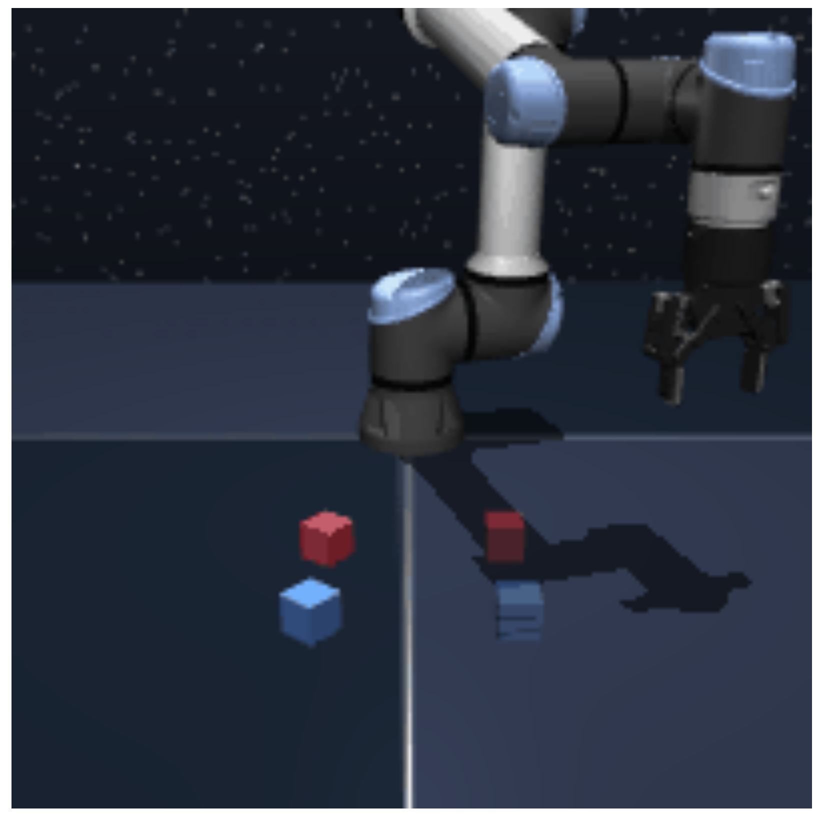
Two different types of tasks are evaluated on three different sized maps.

MCTD consistently surpasses other methods by a large margin.

Experimental Results

Robot Arm Manipulation





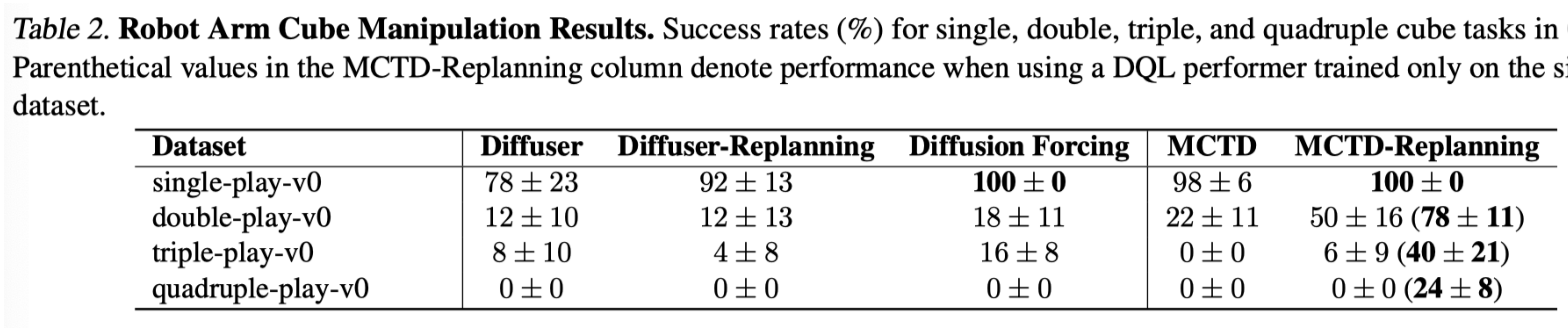
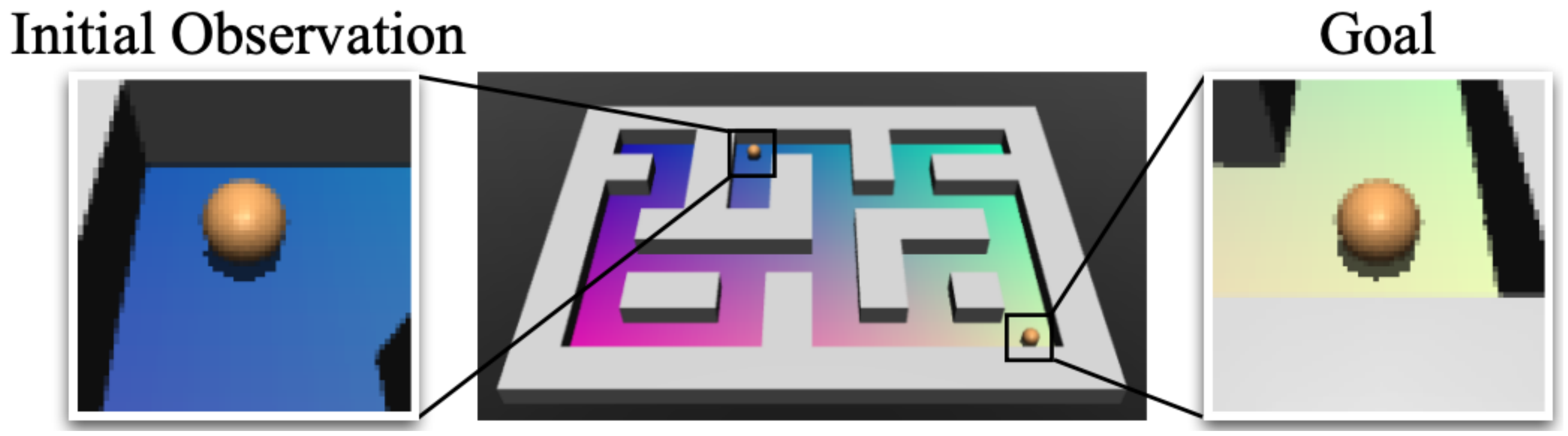


Table 2. **Robot Arm Cube Manipulation Results.** Success rates (%) for single, double, triple, and quadruple cube tasks in OGBench. Parenthetical values in the MCTD-Replanning column denote performance when using a DQL performer trained only on the single-cube dataset.

Dataset	Diffuser	Diffuser-Replanning	Diffusion Forcing	MCTD	MCTD-Replanning
single-play-v0	78 ± 23	92 ± 13	$\mathbf{100 \pm 0}$	98 ± 6	$\mathbf{100 \pm 0}$
double-play-v0	12 ± 10	12 ± 13	18 ± 11	22 ± 11	50 ± 16 ($\mathbf{78 \pm 11}$)
triple-play-v0	8 ± 10	4 ± 8	16 ± 8	0 ± 0	6 ± 9 ($\mathbf{40 \pm 21}$)
quadruple-play-v0	0 ± 0	0 ± 0	0 ± 0	0 ± 0	0 ± 0 ($\mathbf{24 \pm 8}$)

Experimental Results

Visual PointMaze



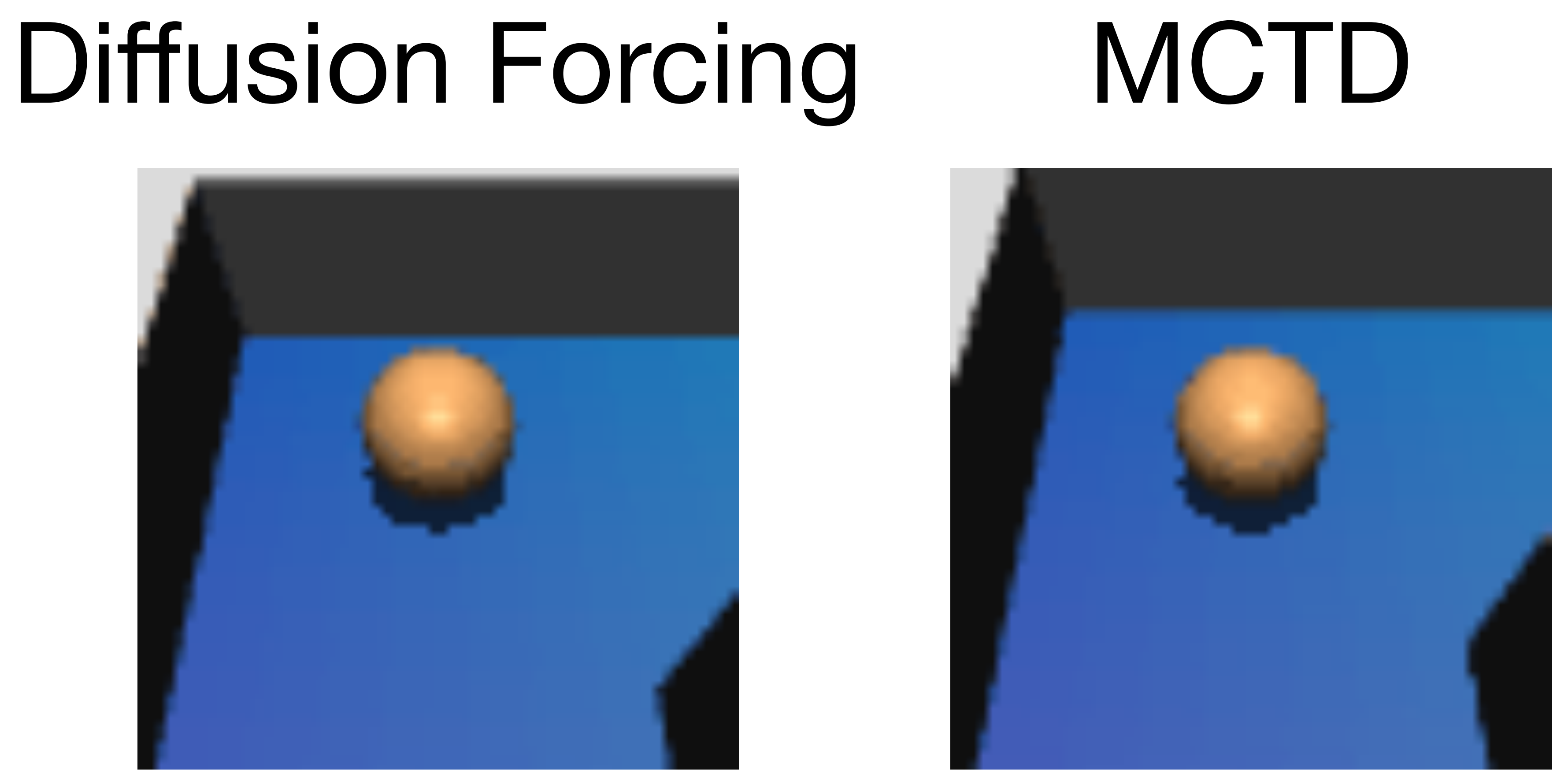


Table 3. **Visual Pointmaze Results.** Success rates (%) on partially observable, image-based mazes of medium and large sizes.

Dataset	Diffuser	Diffuser-Replanning	Diffusion Forcing	MCTD	MCTD-Replanning
medium-navigate-v0	8 ± 13	8 ± 10	66 ± 32	82 ± 18	$\mathbf{90 \pm 9}$
large-navigate-v0	0 ± 0	0 ± 0	8 ± 12	0 ± 0	$\mathbf{20 \pm 21}$

Experimental Results

Inference-Time Compute Scalability & Time Complexity

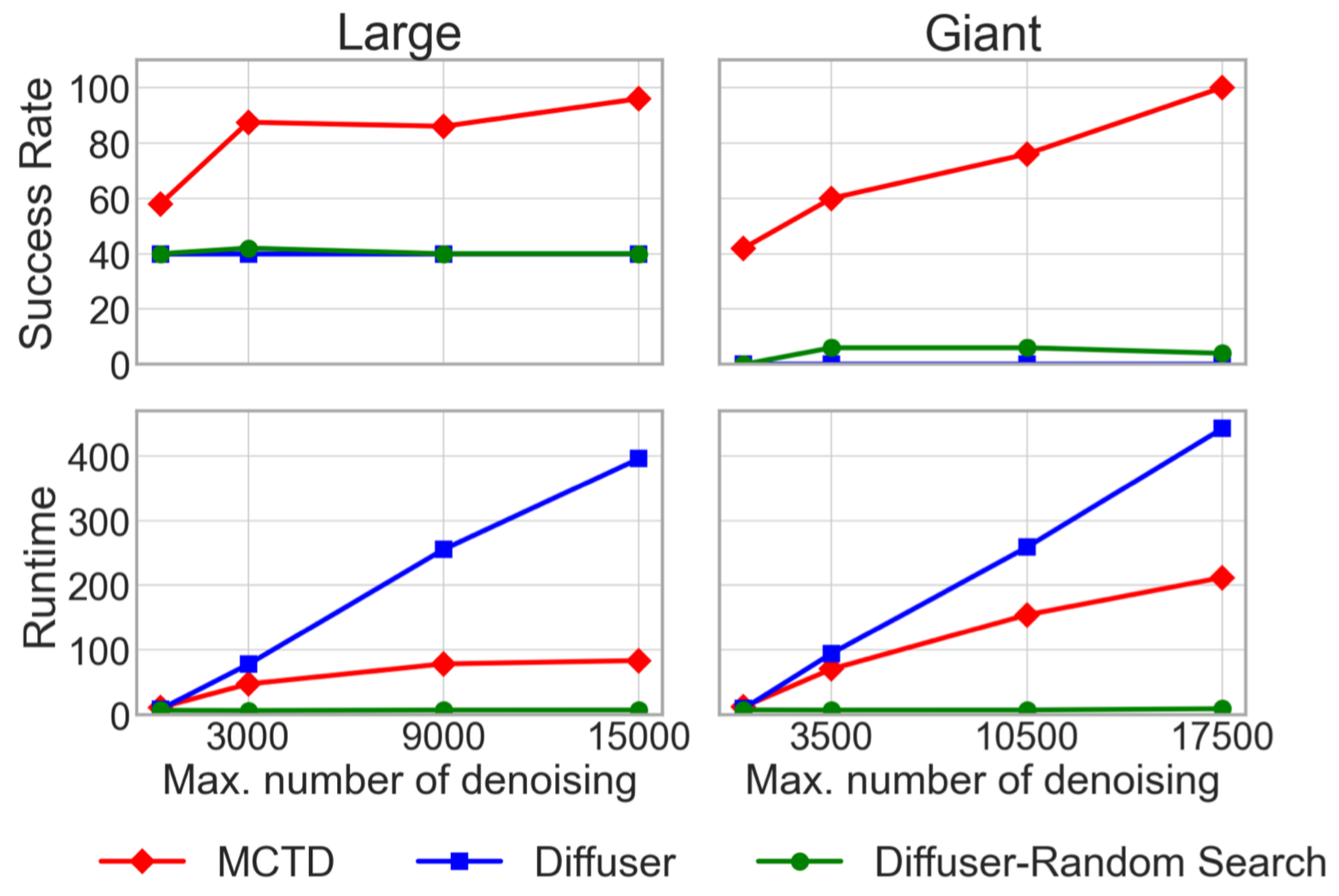


Figure 6. Success rates (%) and wall-clock runtime (Sec.) as the maximum number of denoising steps (test-time budget) increases for large and giant pointmaze tasks

Conclusion

We introduced MCTD, combining diffusion model with MCTS to exploit inference-time compute scalability with the balance between exploration and exploitation.

It shows better performance than previous methods for long-horizontal complex planning tasks which require “system 2” thinking.

However, the tree search can be expensive because it requires sequential search.

Future directions can be the improvement on the search overhead and expanding to discrete domains or multi-modal reasoning tasks.

Thank you!

If you are interested in our paper, please visit our poster session!

Contact: jaesik.yoon@kaist.ac.kr