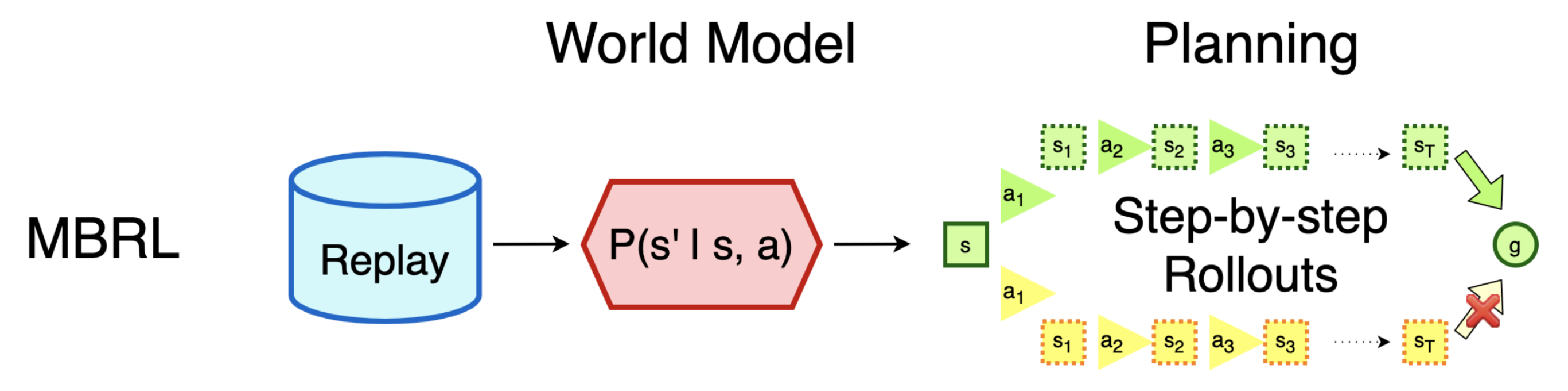
# World Model as a Graph: Learning Latent Landmarks for Planning

Lunjun Zhang[1,2], Ge Yang[3], Bradly Stadie[4]

1. University of Toronto, 2. Vector Institute, 3. MIT, 4. TTIC

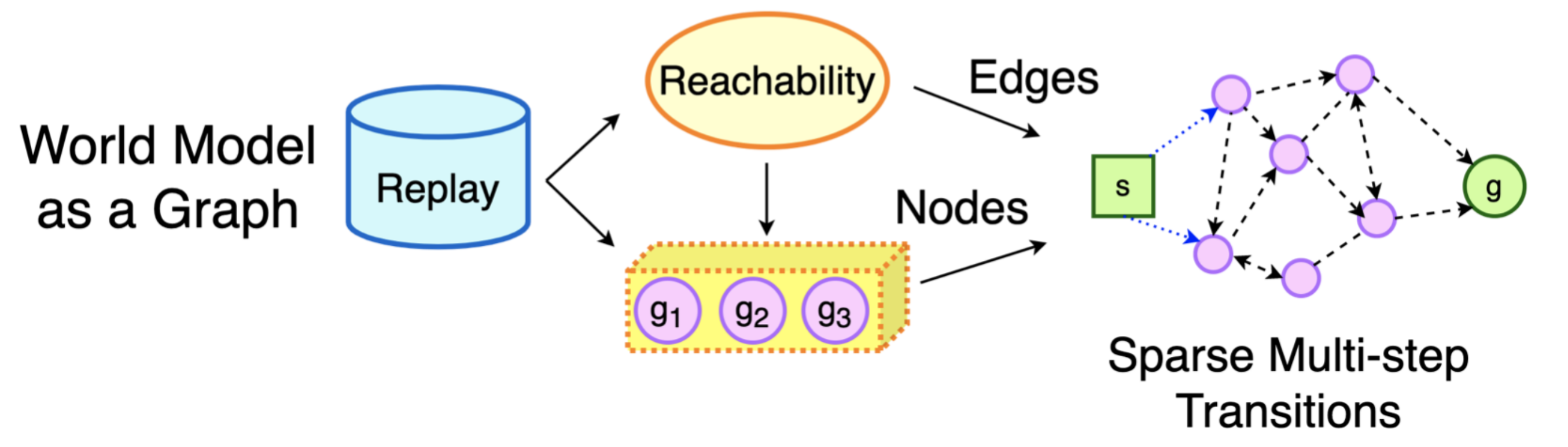
## Why does MBRL struggle?



**Learned transition quickly diverges from reality when the planning horizon increases.**

- In Model-Based RL (MBRL), a single-step forward dynamics function is learned.
- For planning, it uses step-by-step virtual rollouts with the world model.
- The model error compounds as the planning horizon increases.
- Robotics makes planning more difficult because the transition function is non-deterministic and the action space is continuous.
- Long-horizon action-by-action rollouts are infeasible if a robot takes an action every 100ms.

## What kind of planning is more plausible?



**Graph-structured world models can facilitate temporally extended planning in RL.**

- Humans are able to plan days or months ahead, by planning on a more abstract level that operates on a more temporally extended scale.
- We seek to build the type of planning that would analyze the structure of a problem in the large, and decompose it into interrelated sub-problems.
- Our method plans for subgoals to reach rather than actions to take, and learns the world as a graph of sparse multi-step transitions.
- On the graph, nodes are learned in a structured latent space, and the edges are the reachability estimates (distilled from $Q$-function).
- Our implementation of $L^3P$ can be found at: `https://github.com/LunjunZhang/world-model-as-a-graph`

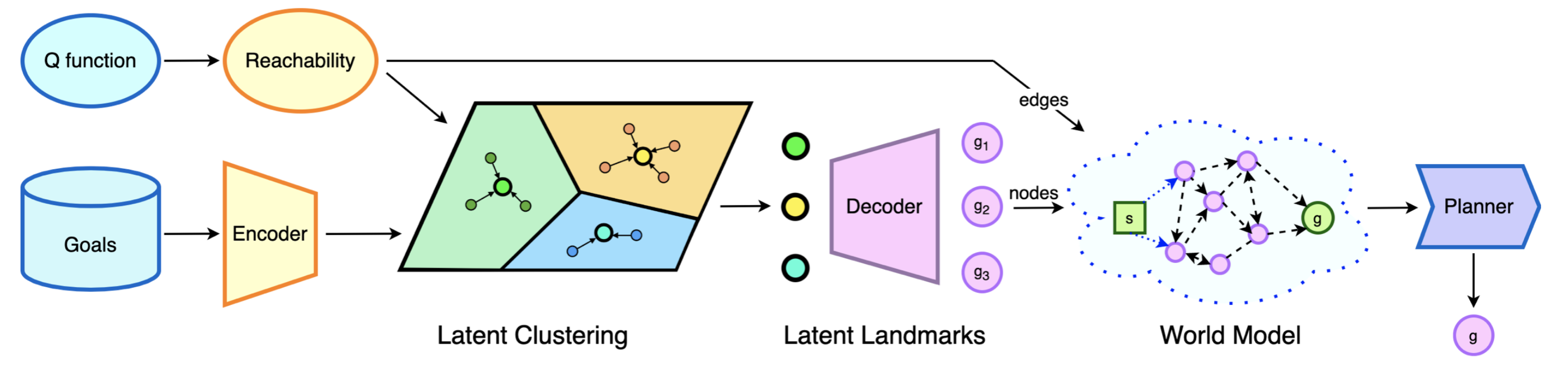
## Major differences in $L^3P$ from prior works

- In $L^3P$, nodes are learned rather than heuristically selected.
- $L^3P$ better leverages temporal abstraction in online planning.

## Prior works on RL + graph search

Eysenbach et al, "Search on the Replay Buffer: Bridging Planning and Reinforcement Learning," NeurIPS 2019.
Huang et al, "Mapping State Space using Landmarks for Universal Goal Reaching," NeurIPS 2019.
Emmons et al, "Sparse Graphical Memory for Robust Planning," NeurIPS 2020.

## Overview of Learning Latent Landmarks for Planning



**Q-learning $\rightarrow$ Reachability $\rightarrow$ Constrained Latent Space $\rightarrow$ Latent Clustering $\rightarrow$ Graph Search**

- Learning reachability estimates jointly with Q-learning
- Learning a reachability constrained latent space
- Learning latent landmarks via clustering in the latent space
- Graph search on the world model for online planning

## Are goal-conditioned Q-functions reachability estimators?

- In goal-conditioned RL, the agent receives a reward of 0 when it reaches the goal and a reward of -1 every single step before reaching the goal.
- Thus, the Q-function is estimating the number of steps it takes the agent to reach the goal from the current state after an action is taken.
- We still need to consider the discount factor $\gamma$.

$$Q(s,a,g) = \sum_{t=0}^{D(s,a,g)-1} \gamma^t \cdot (-1) + \sum_{t=D(s,a,g)}^{T-1} \gamma^t \cdot 0 = -\frac{1-\gamma^{D(s,a,g)}}{1-\gamma}$$

We also want to marginalize over the actions and directly estimate the number of steps, $V$, it takes the policy to go from one goal to another.

$$\min_V \mathbb{E}_{\tau\sim B, t\sim\{0\cdots T-1\}, k\sim\{1\cdots T-t\}} \Big( D\big(s_t, a_t, g_{t+k}\big) - V\big(g_{t+1}, g_{t+k}\big)\Big)^2$$

Functions $D$ and $V$ provide the edges of the world model graph.

## How to learn the nodes on the graph?

**We aim to learn a set of nodes sufficiently scattered across the state space in terms of the agent's reachability.**

- Prior RL + Graph Search methods either sub-sample or heuristically select from the replay to obtain the nodes. Can the nodes be learned as well?
- We use an auto-encoder architecture with an additional loss: the distance between two latent codes is regressed towards the corresponding reachability.
- Our key insight: if we do clustering in a reachability-constrained latent space, then goals that are easily reachable from one another will be grouped together to form landmarks in the latent space.
- The nodes on the graph are the centroids in the latent space.

$$\mathcal{L} = \Big(\big\|f_E(g_1) - f_E(g_2)\big\|_2^2 - \frac{1}{2}\big(V(g_1,g_2) + V(g_2,g_1)\big)\Big)^2 + \lambda_{rec}\big\|f_D(f_E(g)) - g\big\|_2^2 - \lambda_{cluster}\cdot\mathbb{E}_{q(\mathbf{c}\mid z)}\Big[\log p(z\mid\mathbf{c})\Big] + \lambda_{KL}\cdot D_{KL}\Big(q(\mathbf{c}\mid z)\parallel p(\mathbf{c})\Big)$$

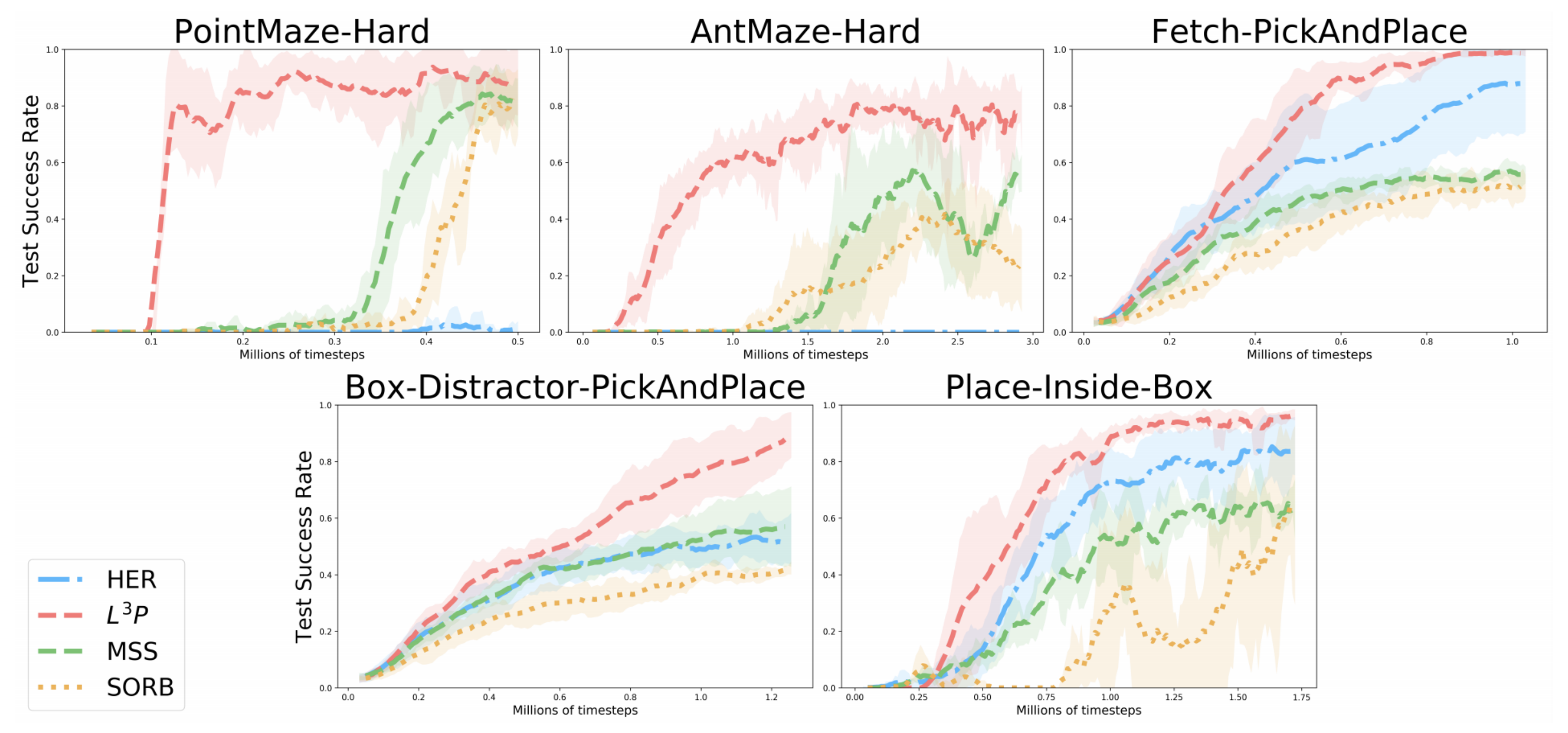
where $p(\mathbf{c})$ is set to be a uniform prior, and $q(\mathbf{c}\mid z)$ is calculated using Bayes Rule.

## Online Planning in $L^3P$

$L^3P$ leverages temporal abstraction to facilitate making progress towards the long-horizon goal.

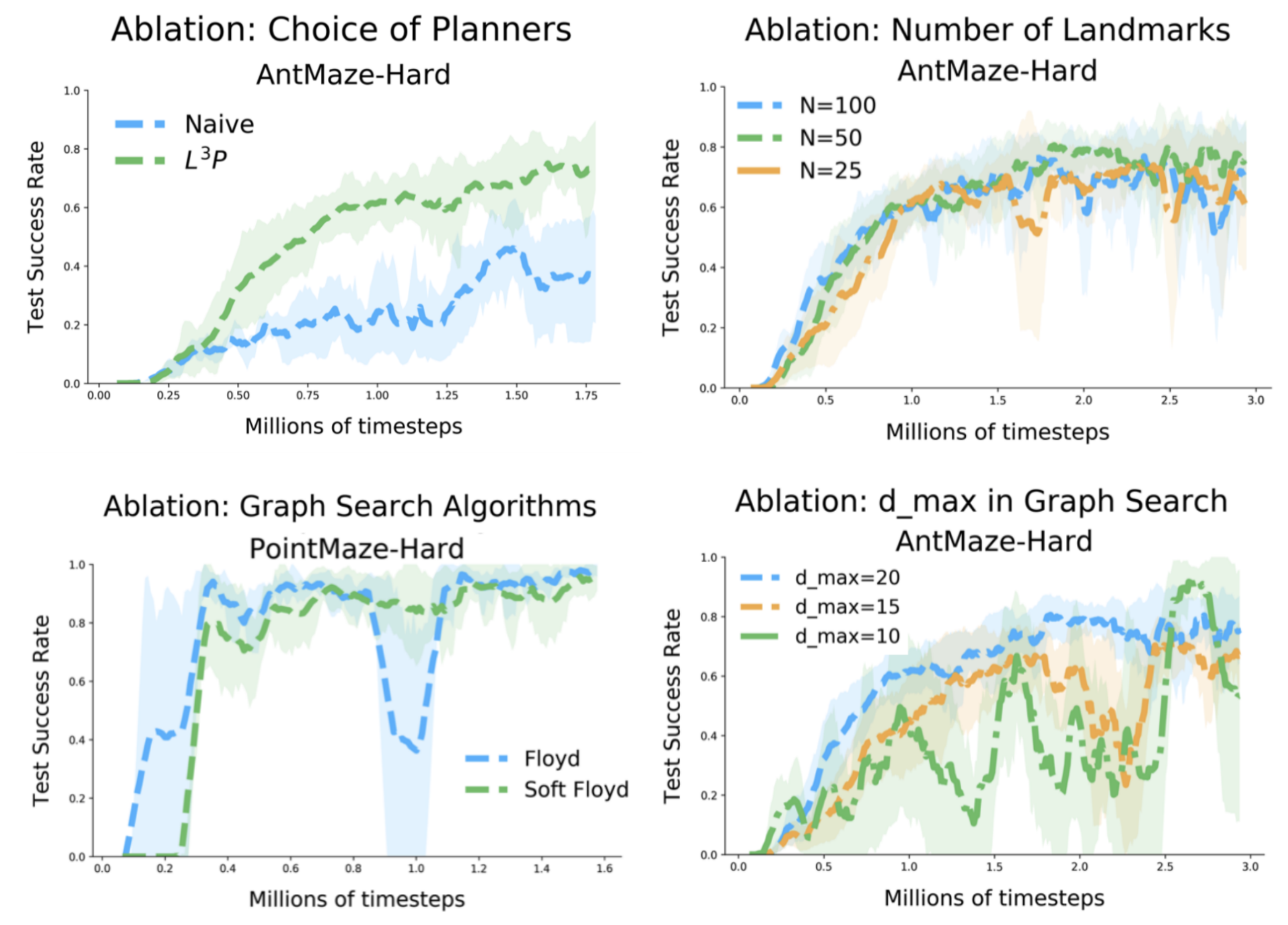
- Crucially, it does not re-plan at every step, but rather uses reachability predictions to decide when to re-plan.
- Each sub-goal is kept fixed for the estimated number of steps to get there.
- To avoid getting stuck (continuing pursuing an unsuccessful sub-goal), it removes the immediate previous goal from the node list when running graph search to propose the next goal.

## Experimental Results



- On the Point maze and Ant maze tasks, the method has to be able to dynamically stitch together simpler goals seen during training to achieve a longer-horizon goal at test time.
- $L^3P$ also works well on robotic manipulation tasks, where prior RL + Graph Search methods fall short on.

## What design choices are important?



- The online planning module in $L^3P$ is important.
- A common trick for RL + Graph Search: once a distance is above a certain threshold, set it to infinity. This trick is found to be important.