We thank the reviewers for their valuable feedback. Please see the following for the response, and we will make corresponding modifications in the revised paper.

R2 and R3 had questions about the **time complexity** of our method. The only additional cost of our method over standard RL is running graph search, which must be done once per evaluation episode. On an 8-core desktop with 32GB of RAM, it takes 65 seconds to estimate the $\mathcal{O}(|V|^2)$ pairwise distances between the 1000 observations in the replay buffer. As noted in Appendix A, this computation can be amortized across many goal-reaching tasks. Given that pairwise distances, it takes 0.029 seconds to compute the shortest path (using Dijkstra's Algorithm, which runs in $\mathcal{O}(|V|^2)$). In terms of **sample complexity**, we found that SoRB required more samples to converge than SPTM, likely because dynamic programming (the optimization problem for SoRB) is more challenging than supervised learning (the optimization problem for SPTM). Lastly, we agree with R2 that the construction of "good" replay buffers is an important problem to explore.

Reviewer 1:

- $d_{s \to g} > \text{MaxDist}$ – For most distant goals, the shortest path does not go through the waypoint, so the distance directly to the goal is almost always shorter. However, always conditioning on the goal would be problematic, because the goal-conditioned policy often fails to reach distant goals directly (See Fig. 6). We will clarify this in Section 2.3.

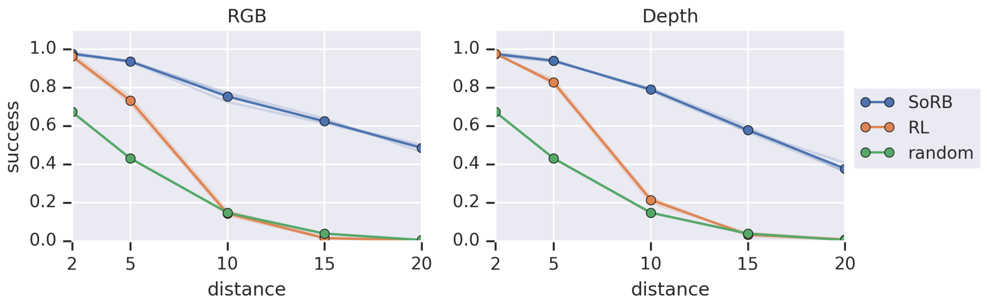
Reviewer 2:

- **Replay buffer**: For the generalization experiments, we initially seeded the replay buffer with 1,000 observations taken from random states and actions in the new environment. We will clarify this in Alg. 1.
- **Where should SoRB excel?** We expect SoRB to outperform existing RL approaches on long-horizon tasks, especially those with high-dimensional inputs. The Atari games fit that mould, as does StarCraft II. Many of the Mujoco tasks in the OpenAI Gym don't require long-horizon reasoning, so we expect little benefit from using SoRB for environments like Hopper and HalfCheetah. We will add this discussion to Section 6.
- **Comparison with state representations** – We agree that learning compact state representations might boost the performance of the underlying goal-conditioned agent. While we found Value Iteration Networks (which learn a representation based on the values of states) performed poorly (see Fig. 6), it is plausible that other methods (e.g., [14]) would work better. We will include a comparison to [14] for the final version.

Reviewer 3:

- **RL and Planning, PRM-RL** – We discussed some prior work that combines RL and planning, including PRM-RL, on L165 - 167. The key difference between PRM-RL and SoRB is graph construction. Whereas SoRB simply uses the value function to predict the distance between two states, PRM-RL requires evaluating the policy in the real-world to determine this distance.[12, Section III.B]
- **Why does Q-learning fail to generalize?** – While, in theory, pure Q-learning should be able to solve any task, in practice the optimization problem is quite challenging, and only succeeds in learning to reach nearby goals. Our method generalizes better than Q-learning because, while both are solving a dynamic programming problem at the high-level, planning provides an exactly solution, while Q-learning gives an approximate solution.
- We will include a comparison to a hierarchical RL method (e.g., [36]) in the final version.

We ran an additional **generalization experiment**. The only difference from Fig. 9 in Section 5.5 is that we now normalize input images to be in [0, 1] (i.e., divide RGB by 255). We found that normalization substantially improved the generalization results for RGB images, reaching almost 80% of goals 10 steps away, while goal-conditioned RL reaches less than 20% of these goals. Transparent lines correspond to average success rate across 22 held-out houses for each of three random seeds.


RGB
Depth
success
distance
SoRB
RL
random