A CODE AVAILABILITY

The Gen-POMCP implementation is available here: https://anonymous.4open.science/r/GenPlan-FBCD/README.md

B PERFORMANCE BOUNDS FOR STRUCTURE-BASED PLANNING

In structured environments Gen-POMCP can explore the environment faster than Naive-POMCP (using fewer rollouts and in less time) by taking advantage of limited resources. However, it simplifies the planning problem by entirely exploring each fragment it enters before moving to the next. This heuristic can result in longer overall paths taken to search the environments. It is reasonable to ask by how much the global Naive-POMCP can actually improve on the path length taken by Gen-POMCP (and specifically the Structure-Based Planner).

Below give a sense of this answer by sketching a proof that considers the limit in which each planner fully optimizes its respective objective: Naive-POMCP follows the Bayes-optimal plan in each fragment and Gen-POMCP follows the Bayes-optimal global policy for the maze. We bound the cost difference according to expected and worst-case cost in steps (the latter is more analytically tractable).

Expected and worst-case The expected number of steps it takes for a policy to explore a maze is the average over the length of path this policy takes to reach uniformly sampled exit locations. The worst-case number of steps is the largest number of steps that the policy could take for some exit position. This is bounded below by the number of steps required to fully explore the maze.

Lemma 1. *There exists a fragment of size $n \times n$ which takes $O(n^2)$ steps to search in expectation, and to explore fully.*

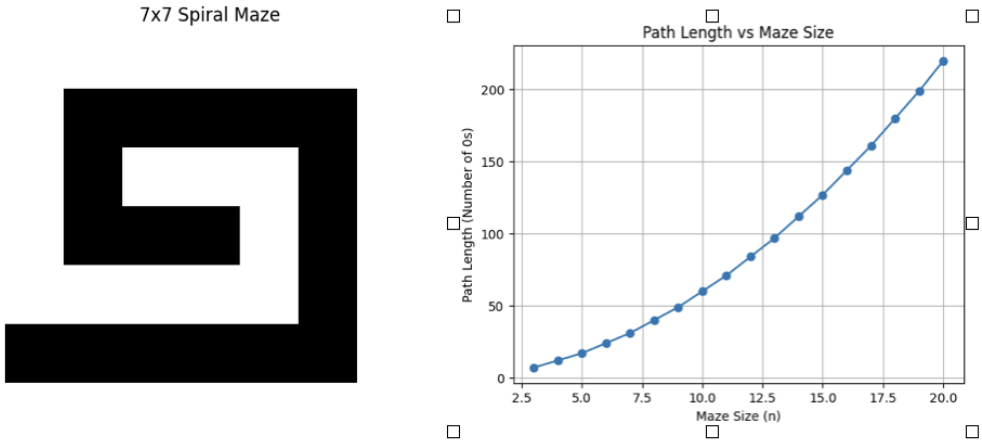

Figure 5: Consider a maze with a spiral wall - the white cells indicate traversable floor, and black indicate intraversable wall. Simulating these environments shows that the maximal length of path (white cells) in the environment grows as $\approx \frac{1}{2}n^2$

.

Proof. Consider a fragment with the maximum spiral path (e.g. Figure 5). The length of this path scales quadratically with n. In particular, following a spiral path takes a series of four legs at each depth, and the length of every other leg reduces by two (one for the wall and one for the path itself).

This yields

$$n + \sum_{i=0}^{\lfloor n/2 \rfloor - 1} 2(n - 2i - 1) = \frac{n}{2} 2n - 4\frac{(n/2)(n/2+1)}{2} + O(n) \\ = \frac{1}{2}n^2 + O(n) \tag{3}$$

□

Theorem 2. *In the an $n \times n$ maze, the expected number of steps taken by SBP may exceed the expected number of steps of an optimal policy by $\Omega(n^2)$.*

Proof. Build a fragment by adjoining an empty room and a spiral by a single door at a corner. Now connect the two fragments by adding a door between the empty rooms in the opposite corner. Assume the size of the empty rooms is such that the optimal algorithm can find the exit with probability $1/2$ by checking each empty room, but the SBP algorithm must explore entirely the first fragment that it enters. With probability $3/4$, the exit is not in the first empty room, so it must explore the spiral, which takes time $\Omega(n^2)$ to fully explore by Lemma 1. The spiral also must be exited, so around n^2 steps are spent when the exit is in the other empty room (in this case the optimal planner finds immediately by checking each room). Since the optimal planner takes only a constant number of steps to check each empty room, and then behaves identically to the SBP, the expected cost when the exit is in any other location is asymptotically the same, so the expected cost difference is roughly $\frac{1}{4}n^2 = \Omega(n^2)$. □

Theorem 3. *The number of steps to fully explore a maze is $O(n^2)$.*

Proof. Consider a v-vertex connected graph. The maximum width (roughly achievable by the spiral) is v, leading to a naive bound of $O(v^2) = O(n)$. This can be improved to $O(v)$ by running a depth-first search. Since there are 4 movement directions the degree of this graph is 4 meaning the maximum number of backtracks *to* a vertex is 3, which immediately gives $4v$. However, in a depth first search there is only one backtrack *from* each vertex is 1, which leads to an easy inductive proof that the bound is $O(2v - 1)$ regardless of degree, yielding $2n^2 - 1 = O(n^2)$. Note that further improvements should be possible by considering the number of walls required to induce the worst-case topology. □

This implies that the Bayes-optimal policy has $O(n^2)$ expected cost (since its *expected* cost must be at least as good as the *expected* cost of exhaustive search), regardless of the maze. Together, Lemma 1 and Theorem 3 demonstrate that the SBP heuristic does not damage the (asymptotic) expected cost in the worst maze.

Theorem 4. *Assume that an $n \times n$ maze is fragmented in such a way that any time a fragment is entered, it can be fully explored before exiting, into c^2 square $(n/c) \times (n/2)$ fragments. The asymptotic expected cost is $\Theta(n^2)$ in the worst such maze for the modular optimal and globally optimal policies.*

Proof. First, consider the global optimal policy. The additional requirements placed on the maze cannot make the $O(n^2)$ bound in Theorem 3 worse, and we can get a matching lower bound by simply adjoining multiple spiral examples as in Lemma 1 and adding doors between them.

Now consider the modular optimal policy. It is clear that the globally optimal policy has an expected cost as least as low as the modular optimal policy (even in their respective worst mazes), by definition, so the $\Omega(n^2)$ lower bound automatically carries over to the modular optimal policy. We assumed that the modular optimal policy takes the Bayes-optimal paths between fragments. This must be at least as good as the following strategy: mimic the global optimal policy, but any time a new fragment is entered, first explore it completely and return to the entrance. By Theorem 3, each such "extra" exploration detour takes at most $2(\frac{n}{c})^2 - 1$ steps, and the return takes at most $(\frac{n}{c})^2$ steps. The total is $3(\frac{n}{c})^2 - 1$. There are exactly c^2 such detours, for $4n^2 - c^2 = \Theta(n^2)$ extra steps. The global optimal policy also takes $\Theta(n^2)$ steps. □

Therefore, in the worst case the modular algorithm is inferior by at least a constant factor of the total search time in expectation. Examining the proof of Theorem 4 yields a factor of 2.5 over our upper bound in Theorem 3, but presumably this can be improved substantially since a lot of exploration is being redone after the detours.

Improving expected cost upper bounds Substantial improvements to the worst-case cost bound in Theorem 3 are easy to obtain when the proof is applied to expected cost by e.g. noting that the depth-first search visits at least one new cell every two steps, meaning that there is clearly at least a $1/4$ chance of finding the exit after n^2 steps, or by noting that the true number of "vertices" is reduced by walls. These improvements seem to apply equally to the modular and global optimal policies, and probably do not affect our constants much.

For worst-case cost, the situation is similar. However, the worst-case cost analysis simplifies significantly with the additional assumption that transitions between fragments are negligible (say, if they all branch off from a central room). This observation is trivial but worth stating explicitly:

Theorem 5. *When the cost to transition between fragments is negligible, each has one entrance, and there is no line-of-sight across fragments, the modular algorithm has the same worst-case step count as the optimal algorithm.*

Proof. In the worst case, the optimal algorithm must explore each fragment, and since there is only one entrance to each fragment it is not possible to gain any advantage by exiting a fragment before it has been fully explored. □

C EXPERIMENTAL ENVIRONMENTS

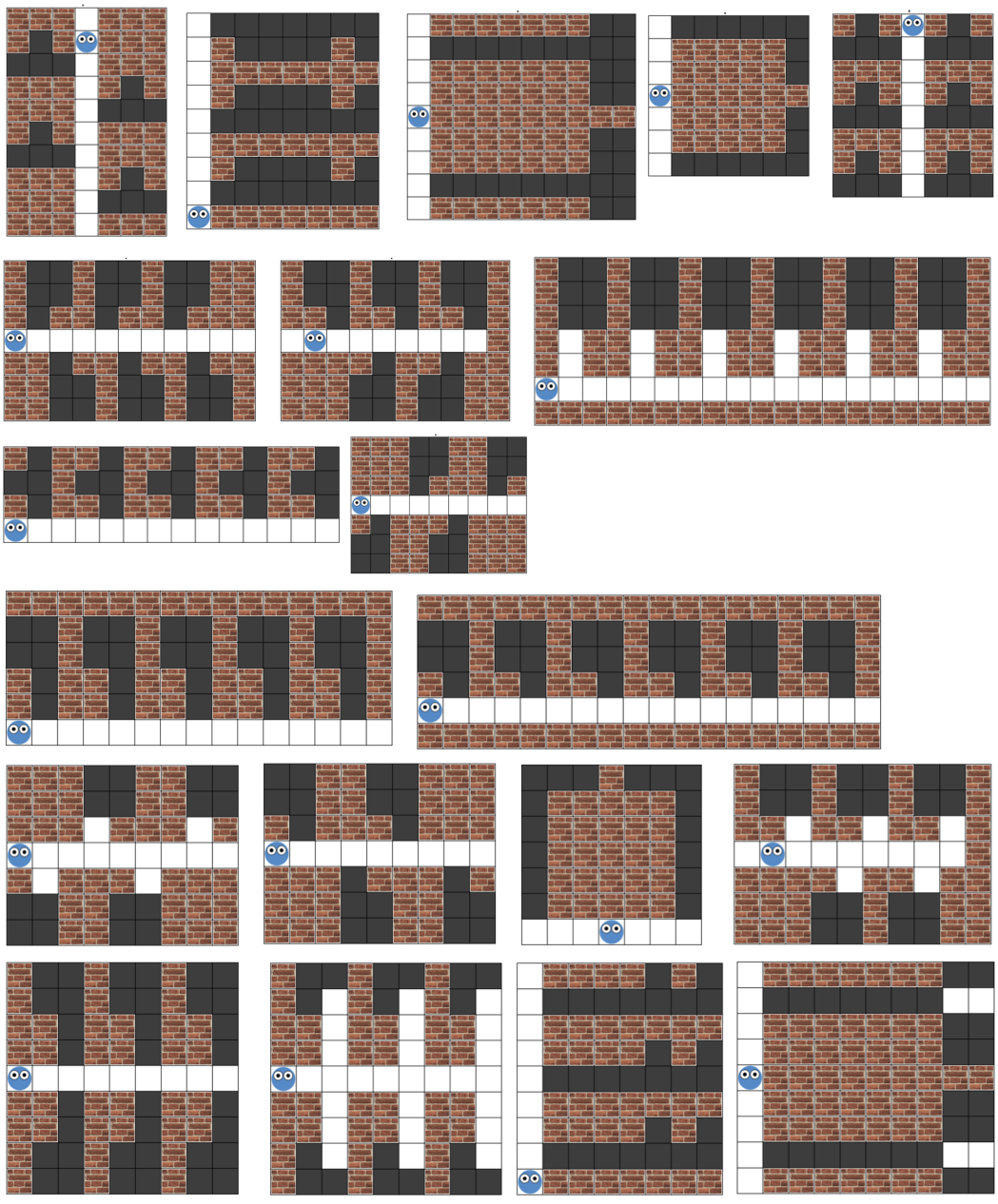

Figure 6: Environments used in Behavioral Experiment 1.

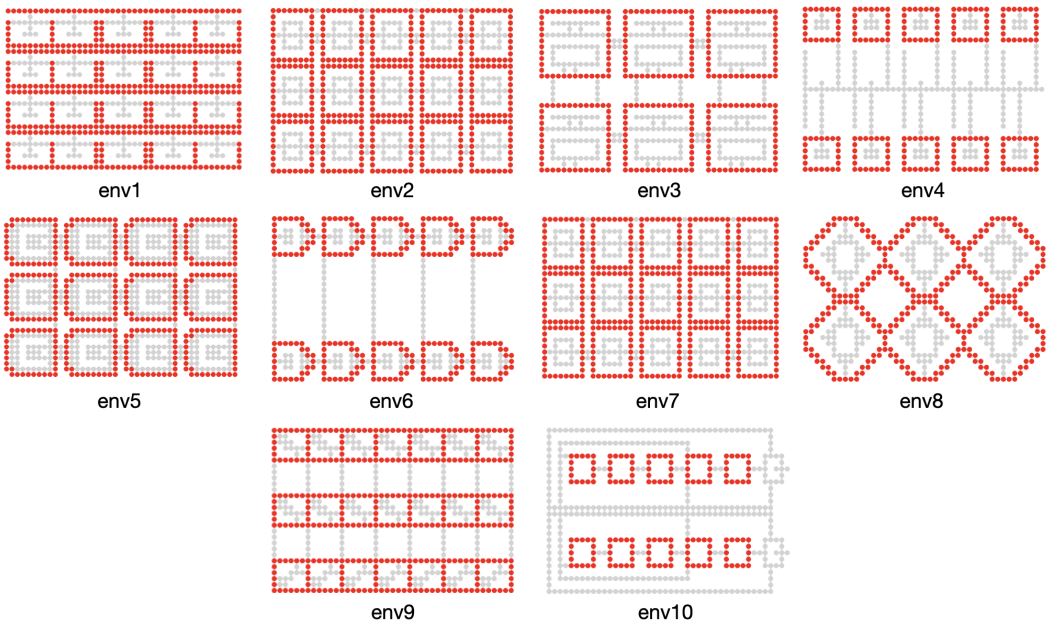

Figure 7: Environments used in Simulation Experiment 2.

D ADDITIONAL RESULTS - SIMULATION EXPERIMENT

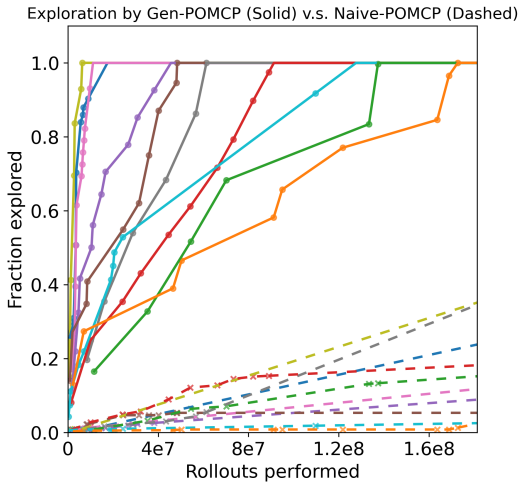

Figure 8: The fractions of each environment searched by Gen-POMCP and Naive-POMCP given identical computational budget. Gen-POMCP requires fewer rollouts and saves computing costs. Each environment is shown in a different color (see also Figure 4.)

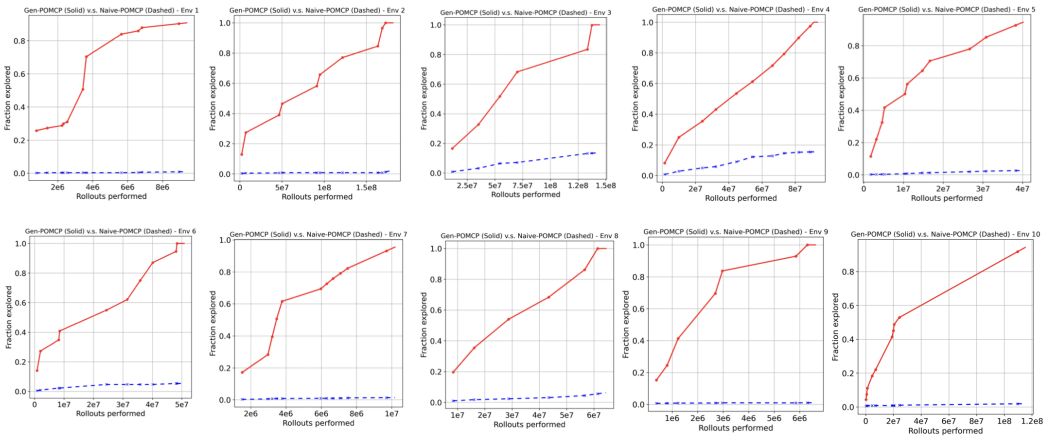

Figure 9: The fractions of each environment searched by Gen-POMCP and Naive-POMCP given identical computational budget. In each individual environment Gen-POMCP requires fewer rollouts and saves computing costs.)