ANA* vs A* Search-Based Planning

Darren Cleeman

INTRODUCTION

Path planning is a fundamental problem in robotics and artificial intelligence. In many robotic systems, a robot must navigate through an environment containing obstacles while moving from a start configuration to a goal configuration. The objective is to compute a collision-free path that minimizes a cost metric such as distance or travel time. Efficient path planning is essential for reliable robot operation, particularly in applications where robots must act autonomously in complex environments.

One of the most widely used algorithms for optimal path planning is A*. When combined with an admissible heuristic, A* is guaranteed to find the optimal path. However, a key limitation of A* is that it must complete the entire search before returning any solution. In environments with large state spaces or complex obstacle layouts, this can result in long computation times. During this period, the robot is unable to act, even though a feasible, but potentially suboptimal, path may exist early in the search process. As a result, standard A* can be poorly suited for time-critical robotic applications.

In many real-world scenarios, waiting for an optimal solution is not practical. Autonomous vehicles, industrial robots, and emergency response systems often operate under strict time constraints, where a delayed plan can lead to unsafe behavior, reduced productivity, or mission failure. In such cases, it is often preferable to obtain a valid, suboptimal solution quickly rather than wait for the optimal solution to become available. This observation motivates the use of anytime planning algorithms.

Anytime algorithms are designed to produce an initial feasible solution as quickly as possible and then continuously improve that solution as additional computation time becomes available. These algorithms can be interrupted at any time and still provide a valid path, allowing a robot to begin executing immediately while refining its plan in the background. Given sufficient time, an anytime algorithm will eventually converge to the optimal solution.

Anytime Nonparametric A* (ANA*) is an anytime variant of A* that addresses the limitations of classical A* without requiring manual parameter tuning. ANA* quickly finds an initial suboptimal solution and then incrementally improves solution quality over time. As the search progresses, ANA* adapts its behavior automatically and is guaranteed to converge to the same optimal solution as A* given enough computation time. This makes ANA* particularly attractive for robotic navigation problems where responsiveness and solution quality must be balanced.

In this project, A* and ANA* are implemented and compared for navigation of the PR2 robot in several maze-based environments. The search is performed on an 8-connected grid, and two admissible heuristics are evaluated: the standard Euclidean distance heuristic and a diagonal distance heuristic that more accurately reflects movement costs in an 8-connected space. Our evaluation compares the algorithms in terms of solution cost and computation time and focuses on the following questions: how does ANA* perform compared to A* in maze navigation, do different heuristics significantly affect performance, does ANA* find solutions faster than A*, and how does solution quality evolve over time under ANA*.

IMPLEMENTATION

High-Level System Architecture

The planning system follows a modular pipeline structure designed to separate environment representation, collision checking, heuristic evaluation, and search logic. Maze environments are loaded as STL mesh models into the PyBullet physics simulator, and the PR2 robot model is initialized at a specified start configuration. Start and goal states are defined in the robot's configuration space (x,y,θ), and planning is performed on an 8-connected grid discretization of the environment.

Each candidate grid state corresponds to a robot base pose, and transitions are defined between neighboring grid cells according to 8-connected motion. For each candidate state considered during search, a collision checking function determines whether the corresponding robot configuration intersects with the maze geometry. This collision function is passed directly into the planner and is invoked whenever a new state is generated.

The planner module operates independently of the environment representation and accepts the start state, goal state, collision checking function, and heuristic function as inputs. Depending on the selected algorithm, either A* or ANA* is executed. A* performs a single search until the optimal solution is found, while ANA* repeatedly refines its solution over time, producing a sequence of improving paths. Planning results are visualized in PyBullet using colored markers to display paths, and performance data is recorded for later analysis.

Collision Detection for STL Maze Environments

The collision checking functionality used in this project differs from the original homework implementation due to the use of STL mesh-based maze environments. The collision function provided in the homework was designed for simple obstacle representations and relied on predefined obstacle lists rather than mesh geometry. To support collision checking against complex maze meshes, a new collision detection function was implemented.

The custom collision function, make_mesh_collision_fn_with_bounds, performs direct mesh-robot collision queries using PyBullet's pairwise collision detection capabilities. The function takes the maze mesh identifier as an input and checks whether the PR2 base at a given configuration intersects with any part of the mesh geometry. This allows accurate collision detection against arbitrarily shaped maze walls defined by STL files.

In addition to mesh collision checks, explicit boundary conditions are enforced to ensure the robot remains within the valid arena limits. Predefined minimum and maximum bounds in the x and y directions are used to prevent the planner from generating states that lie outside the maze. These boundary checks are integrated directly into the collision function.

To improve computational efficiency during search, collision results are cached. Once a grid cell has been checked for collision, its result is stored so that subsequent visits to the same cell do not require repeated collision queries. This caching significantly reduces the number of expensive collision checks, particularly in cluttered environments where the same states may be revisited multiple times during search.

The ANA* Algorithm

ANA* is implemented following the algorithm described by van den Berg et al. [1], and is integrated into the same planning framework used for A*. Like A*, ANA* maintains an OPEN list of frontier states and tracks the cost-to-come g(s) for each state. However, ANA* differs in how states are prioritized for expansion and how solutions are generated over time.

the function $e(s) = \frac{G - h(g_s(s))}{s}$, where G is the cost of the$f($ best$s) =$ solution$g(s) +$ found$h(s)$ so far, g(s)

is the cost from the

Instead of using the standard A* evaluation function , ANA* prioritizes states using

start state to state sss, and h(s) is the heuristic estimate from sss to the goal. Initially, G is set to infinity, reflecting the fact that no solution has yet been found.

The algorithm begins by inserting the start state into the OPEN list with its corresponding e(s) value. ANA* repeatedly expands the state in OPEN with the maximum e(s), favoring states that have the greatest potential to improve the current best solution. When a goal state is reached, the solution cost G is updated and the corresponding path is recorded as the current best solution.

After each solution is found, the OPEN list is pruned to remove states that cannot possibly lead to an cost than the current best. The remaining$g$ states in G have their e(s) values updated to reflect the new improved solution. Any state satisfying $(s) + h(s) \geq$ OPEN is removed, as it cannot yield a path with lower value of G, and the search continues. This process repeats until the OPEN list is empty or a predefined timeout is reached.

As G decreases over time, the algorithm naturally becomes less greedy and increasingly focuses on refining the solution. Given sufficient computation time, ANA* is guaranteed to converge to the same optimal solution as A*, while providing valid intermediate solutions throughout the search process.

Heuristic Functions

Two admissible heuristic functions are used in this project: a Euclidean distance heuristic and a diagonal distance heuristic. Admissibility is required to guarantee that A* returns an optimal solution and that ANA* converges to the optimal solution over time.

The Euclidean heuristic computes the straight-line distance between the current state and the goal: $h(s)$ $= (x_{goal} \sqrt{\quad} x_s)^2 + (y_{goal} - y_s)^2$. This heuristic provides a simple and computationally efficient

estimate of the remaining cost and serves as a baseline heuristic. Because straight-line distance is always a lower bound on the true path cost, the Euclidean heuristic is admissible. The heuristic does not account for the robot's orientation θ, as translational distance dominates the navigation cost in the grid-based planning formulation.

The Diagonal heuristic [2] is specifically designed for 8-connected grid movement, where cardinal moves have unit cost and diagonal moves have cost . Let and . The

$\sqrt{\ }$

heuristic is defined as $h(s) = max(dx, dy) + 2(2 - dx1)min = \left| x(_{goal}dx, -dyx).{}_{s} \right|$ This expressiondy $= \left| y_{goal} \right.$ corresponds$- y_{s}\left|\right.$ to

taking as many diagonal steps as possible, followed by cardinal steps to cover the remaining distance. In an obstacle-free 8-connected grid, this heuristic yields the exact optimal path cost. As a result, it provides a tighter estimate than the Euclidean heuristic while remaining admissible.

Both heuristics are evaluated to study how heuristic informativeness affects search behavior, solution quality, and convergence speed for both A* and ANA*. The diagonal heuristic is expected to reduce node expansions by providing a closer approximation to the true cost-to-go, particularly in environments where diagonal motion is frequently used.

RESULTS

Experimental Setup

To evaluate the behavior of A* and ANA* across a range of navigation challenges, experiments were conducted in six maze-based environments of varying structure and complexity. All mazes are represented as 3D STL mesh models and loaded into the PyBullet simulator. Each maze includes predefined start and goal configurations, with the start position shown as a blue sphere and the goal position as a green sphere, and is designed to highlight different aspects of search behavior, heuristic guidance, and anytime performance. Using multiple environments ensures that comparisons are not biased by a single problem instance and allows evaluation under diverse planning conditions.

The first environment is a winding corridor maze consisting of a single narrow path with many turns. This maze forces the robot to follow a long, constrained route that deviates significantly from the straight-line path to the goal. It is used to examine planner behavior when only one feasible path exists. In this setting, A* must fully explore the corridor to guarantee optimality, while ANA* is expected to find a feasible solution quickly by greedily following the corridor. Because no alternative routes exist, little improvement is expected after the initial solution is found.

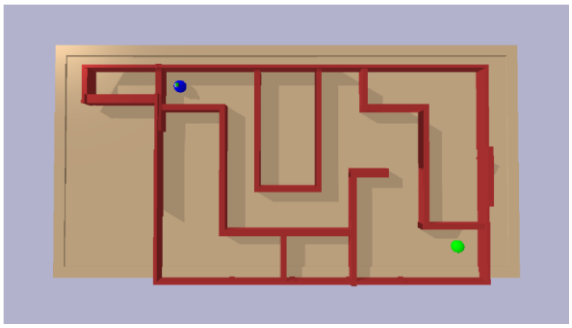

Figure 1: Maze 1 Layout

The second environment is a two-path maze containing two competing routes between the start and goal: one path that is shorter but more winding, and another that is longer but more direct. This maze is designed to test planner behavior when multiple valid routes exist and the optimal choice is not immediately obvious. A* must explore both paths sufficiently to prove optimality, while ANA* may return an early solution along one path and later improve it by discovering the better alternative. This environment highlights the anytime nature of ANA*, as well as the influence of heuristic guidance on early solution selection.

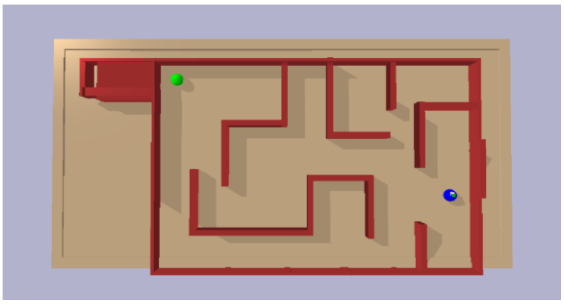

Figure 2: Maze 2 Layout

The third environment is a large obstacle maze, characterized by large central blocks that force the robot to take long detours around obstacles. Several paths exist with similar total cost, differing only slightly in length. This maze tests the planners' ability to distinguish between near-optimal solutions and examines how small cost differences affect search behavior. In this environment, heuristic accuracy becomes particularly important, as the planners must identify subtle improvements among competing routes.

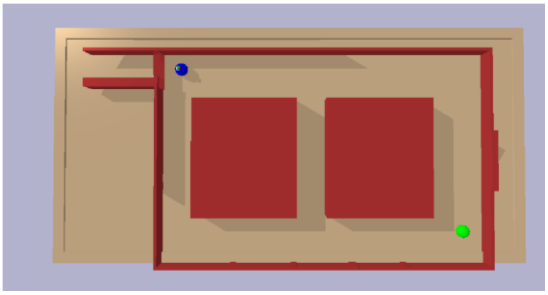

Figure 3: Maze 3 Layout

The fourth environment is a cluster maze containing many small obstacles scattered throughout an open area. This configuration results in a high branching factor, with numerous feasible paths weaving between obstacles. The cluster maze is used to evaluate planner performance in highly cluttered environments where many similar-looking partial paths exist. In such cases, heuristics tend to underestimate true costs due to frequent obstacle avoidance, making it a useful setting for analyzing how ANA* refines suboptimal solutions over time.

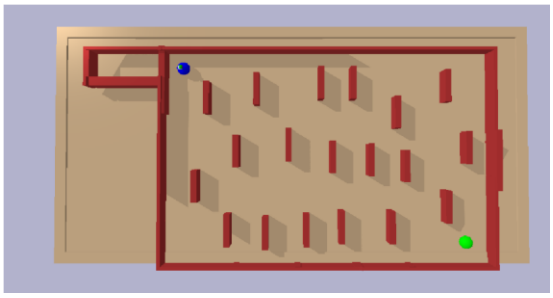

Figure 4: Maze 4 Layout

The fifth environment is a bottleneck maze, consisting of a largely open area with a narrow corridor that serves as the only valid passage to the goal. Although the open space appears promising from a heuristic perspective, all feasible solutions must pass through the narrow corridor. This maze tests the planners' ability to discover and commit to a critical chokepoint. It also examines how misleading heuristic guidance toward open regions affects early exploration before the bottleneck is identified.

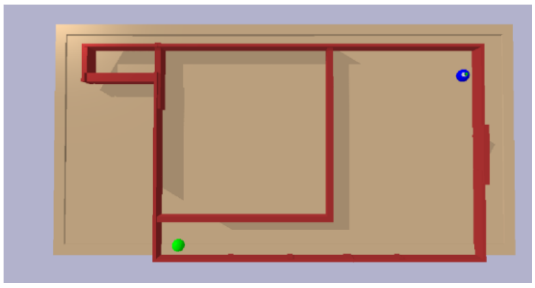

Figure 5: Maze 5 Layout

The sixth environment is a two-path maze with a spiral trap, combining multiple challenges in a single scenario. Two long, simple routes connect the start and goal, while a small spiral structure between them acts as a deceptive region that appears heuristically promising but leads to inefficient exploration. This maze is designed to test how each planner handles heuristic traps and whether ANA* can recover from initially poor exploration choices by refining its solution over time.

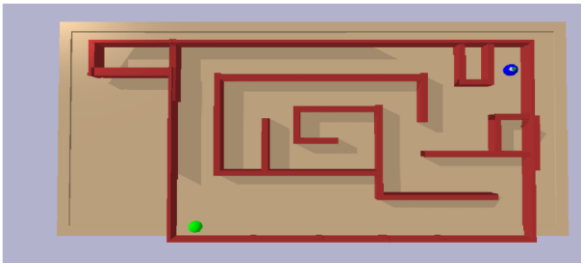

Figure 6: Maze 6 Layout

For each maze environment, experiments are performed using both the Euclidean and Diagonal heuristics. For each heuristic, A* is executed until the optimal solution is found, while ANA* is executed with a fixed timeout of 120 seconds to allow sufficient time for convergence. Performance metrics collected include total path cost, planning time, number of nodes expanded, and, for ANA*, all intermediate solutions with their associated time and cost values. Paths generated by each algorithm are visualized as dotted trails in the mazes, with A* having blue dots and ANA* having green dots (different from the blue

and green spheres used to indicate start and goal respectively). Lastly, results are recorded to a CSV file to generate a table of the collected metrics, as well as being used to create cost-versus-time plots.

Results by Maze Maze 1: Winding Corridor

Using the Euclidean heuristic, A* produced a solution with cost 43.5615 in 6.5017 seconds while expanding 9,501 nodes. With the Diagonal heuristic, A* produced a slightly lower-cost path of 43.3958 and expanded fewer nodes (8,798), indicating that the tighter heuristic improved pruning efficiency. However, the Diagonal heuristic took longer wall-clock time (11.3475 seconds versus 6.5017 seconds). This highlights a trade-off in this maze: although the Diagonal heuristic reduced the number of expanded nodes, its additional computation overhead outweighed the savings from pruning, making it practically slower in this particular environment.

For ANA*, the Euclidean heuristic yielded a solution of cost 43.7472 in 10.3965 seconds with 15,998 nodes expanded, while the Diagonal heuristic yielded a solution of cost 43.8443 in 21.9710 seconds with 16,090 nodes expanded. In both heuristic cases, ANA* found only a single solution and exhibited no iterative improvement. This matches the expected behavior for a single-path maze: because the corridor structure strongly constrains the robot to essentially one feasible route, there is little opportunity for ANA* to discover meaningfully better alternatives after its first solution. ANA*'s final costs were also slightly higher than A*'s (on the order of ~0.4%), reinforcing that ANA* did not provide a quality advantage here.

Notably, the Diagonal heuristic performed worse than Euclidean for ANA* in both time and cost in this maze. Because node counts were nearly identical (~16,000), the additional time is best explained as pure overhead rather than improved search efficiency. This outcome is also consistent with the fact that ANA* prioritizes states using $e(s) = \frac{G - h(g_S(s))}{}$ rather than the additive $f(s) = g(s) + h(s)$ used by A*. In a highly constrained single-path maze, changing the ordering of expansions provides little benefit because both searches must traverse essentially the same reachable region, so a more expensive heuristic can increase runtime without reducing the effective search space.

Overall, A* outperformed ANA* in both time and cost in Maze 1, and the "anytime" benefit did not materialize. Visually, the paths produced by A* and ANA* were nearly identical for both heuristics, which is expected given that the single winding corridor constrains both planners to the same route regardless of heuristic choice.

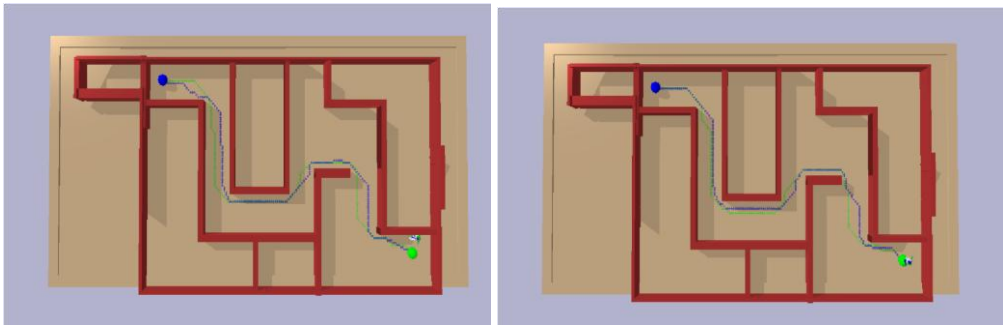

Figure 7: Maze 1 paths - Euclidean (left) and Diagonal (right)

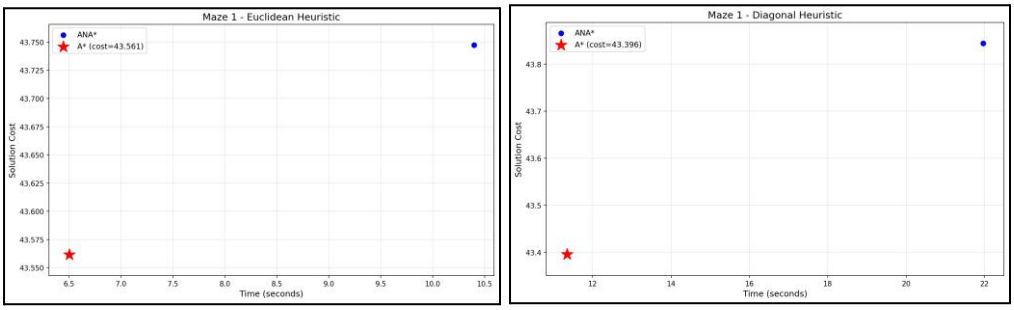

Figure 8: Maze 1 cost vs. time graphs - Euclidean (left) and Diagonal (right)

Maze 2: Two-Path Maze

For Maze 2, A* found the same optimal cost under both heuristics (37.6502). With the Euclidean heuristic, A* required 31.1733 seconds and expanded 25,577 nodes, while the Diagonal heuristic reduced both the runtime (29.3233 seconds) and the number of expanded nodes (23,366). Unlike Maze 1, the tighter Diagonal heuristic provided a clear practical benefit here: in a more complex maze with more competing expansions, the improved pruning was substantial enough that the heuristic overhead was worthwhile.

For ANA*, the Euclidean heuristic produced a solution of cost 38.4301 in 44.2724 seconds with 35,318 nodes expanded, while the Diagonal heuristic produced a slightly lower cost of 38.1472 in 46.7025 seconds with 35,002 nodes expanded. Similar to Maze 1, ANA* found only one solution under both heuristics and showed no iterative improvement. Despite the existence of two viable routes in this environment, ANA* did not return an initial solution and later improve it by switching to a better alternative. Instead, it converged to a single path whose cost remained about ~2% higher than A*'s optimal result.

Comparing heuristics for ANA* in this maze, the differences were minor. Diagonal produced slightly better solution cost but took longer, and node expansions were nearly identical (~35,000). This again suggests that the Diagonal heuristic's tighter estimate did not translate into strong pruning benefits for ANA* in the same way it did for A*. The key outcome in Maze 2 is that A* outperformed ANA* in both time and cost across both heuristics. Visually, the paths were nearly identical between algorithms and heuristics, with both selecting the middle winding route rather than the longer bottom route. This indicates that the optimal route was sufficiently dominant that both planners committed to the same overall strategy, leaving little room for ANA* to demonstrate its anytime refinement advantage.

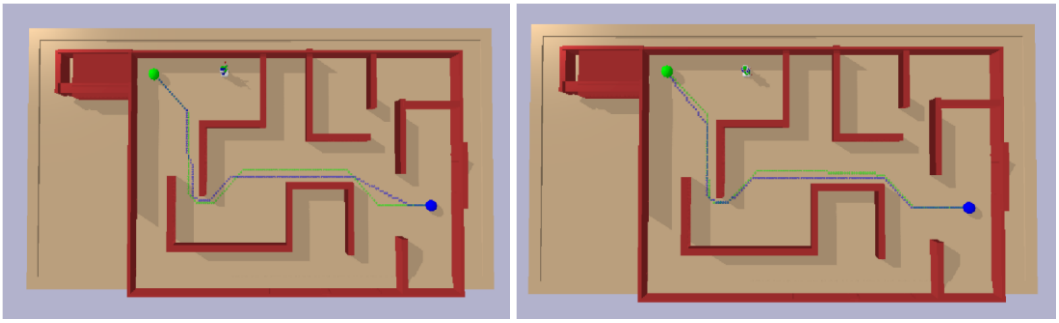

Figure 9: Maze 2 paths - Euclidean (left) and Diagonal (right)

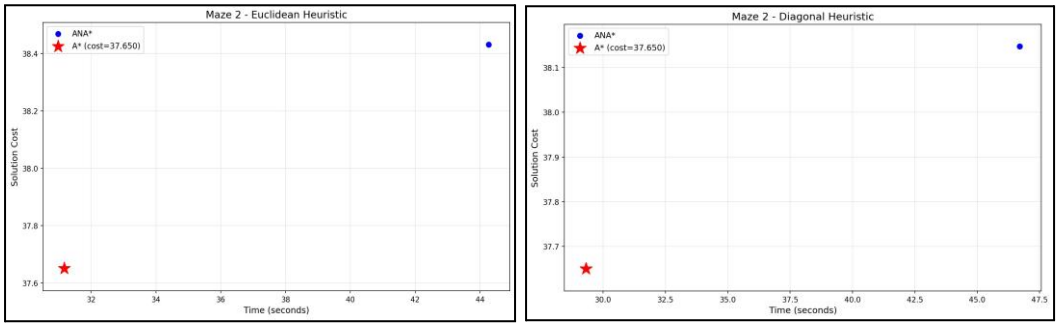

Figure 10: Maze 2 cost vs. time graphs - Euclidean (left) and Diagonal (right)

Maze 3: Open Environment with Central Blocks

In Maze 3, A* with the Euclidean heuristic achieved cost 39.2904 in 21.3886 seconds while expanding 14,531 nodes. With the Diagonal heuristic, A* expanded fewer nodes (12,582) and ran faster (17.9349 seconds), but it produced a slightly worse path cost of 39.4075. In this open environment, multiple routes around the central obstacles are similar in cost, and the tighter Diagonal heuristic appears to have biased the search toward a route that was marginally suboptimal in total cost, even while reducing expansions and runtime.

For ANA*, both heuristics converged to the same cost of 39.2904 and again found only a single solution in each run. With Euclidean, ANA* required 39.9187 seconds and expanded 24,983 nodes; with Diagonal, it required 32.2714 seconds and expanded 24,479 nodes. This is the first maze where the Diagonal heuristic clearly improved ANA* runtime, suggesting that in a more open layout, the interaction between a tighter h(s) and ANA*'s greedy e(s)-based prioritization can sometimes be beneficial.

A key comparison in Maze 3 is that A* remained faster than ANA* under both heuristics. However, Maze 3 is also the first case where ANA* outperformed A* in solution quality under the Diagonal heuristic: ANA* Diagonal achieved cost 39.2904, whereas A* Diagonal returned 39.4075. While ANA* still did not demonstrate iterative improvement over multiple solutions, its broader search behavior in this open environment enabled it to find a better route than A* did under the same heuristic. Visually, the paths produced by both heuristics were still largely similar, which is consistent with the maze structure: clear obstacle boundaries guide planners into comparable routes around the central blocks, even when small cost differences exist.

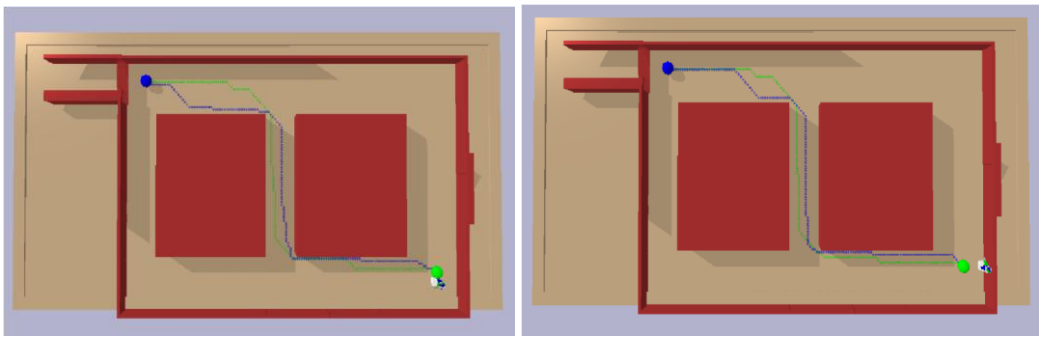

Figure 11: Maze 3 paths - Euclidean (left) and Diagonal (right)

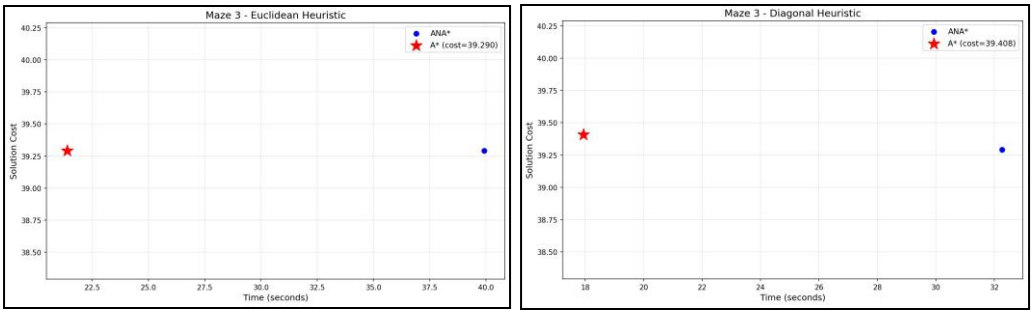

Figure 12: Maze 3 cost vs. time graphs - Euclidean (left) and Diagonal (right)

Maze 4: Cluster Maze with Scattered Obstacles

For Maze 4, A* produced the same optimal cost under both heuristics (35.0183), but performance differed significantly. With Euclidean, A* took 22.6781 seconds and expanded 15,285 nodes. With Diagonal, A* was dramatically faster at 13.9572 seconds and expanded only 8,801 nodes. This cluster environment benefits strongly from the Diagonal heuristic's tighter bound because scattered obstacles create many local routing decisions and many near-equivalent partial paths; the tighter heuristic reduces ambiguity and improves pruning, leading to large reductions in both node expansions and runtime.

For ANA*, Euclidean produced cost 35.2526 in 44.3578 seconds with 38,534 nodes expanded, while Diagonal produced cost 35.0183 in 45.2349 seconds with 37,995 nodes expanded. ANA* again found only a single solution under both heuristics. However, in this maze the tighter Diagonal heuristic improved ANA* solution quality by matching A*'s optimal cost, whereas Euclidean remained slightly suboptimal. Despite this, ANA* runtime remained much higher than A*'s, and the difference in node expansions between Euclidean and Diagonal for ANA* was small, reinforcing that ANA* did not realize strong pruning benefits from the tighter heuristic.

Overall, A* substantially outperformed ANA* in runtime, while Diagonal showed its strongest advantage for A* in this scattered-obstacle environment. Maze 4 also produced the first visually significant divergence between A* and ANA* when using Euclidean: the Euclidean paths differed notably in the middle section of the maze. This can be explained by Euclidean's looser bound making many routes through the obstacle field appear similarly promising; as a result, A*'s f(s)-based ordering and ANA*'s e(s)-based ordering can favor different corridors through the cluster. Under the Diagonal heuristic, paths were nearly identical, consistent with the tighter estimate reducing ambiguity and guiding both planners to the same optimal route.

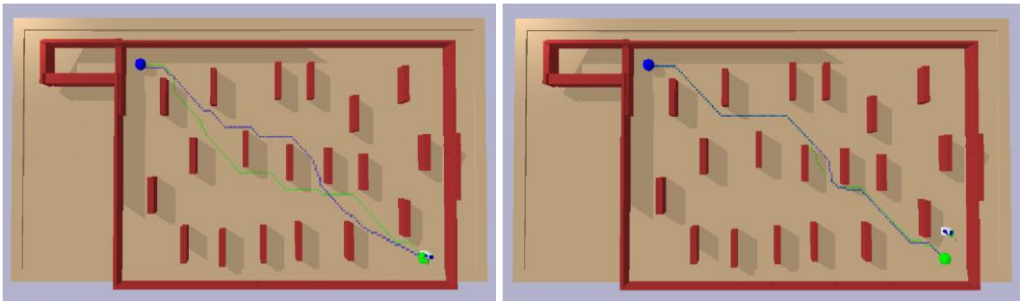

Figure 13: Maze 4 paths - Euclidean (left) and Diagonal (right)

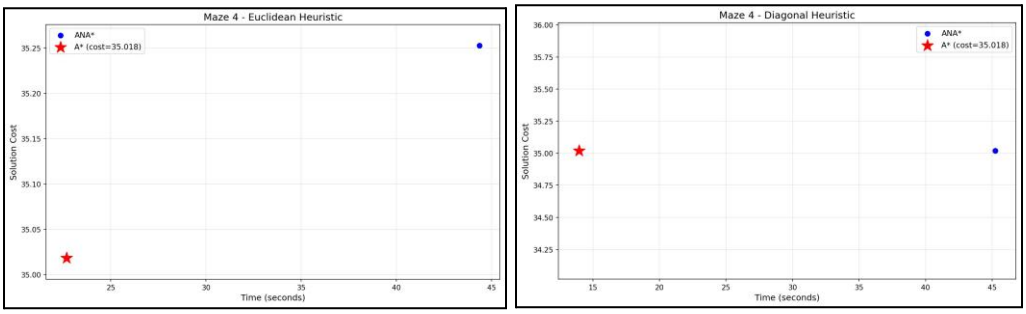

Figure 14: Maze 4 cost vs. time graphs - Euclidean (left) and Diagonal (right)

Maze 5: Bottleneck Maze with Narrow Corridor

In Maze 5, A* with the Euclidean heuristic produced cost 36.0727 in 16.6635 seconds with 15,046 nodes expanded. With the Diagonal heuristic, A* produced a lower cost of 35.9556, ran faster at 12.9025 seconds, and expanded fewer nodes (9,759). This indicates that the tighter Diagonal heuristic helped A* identify the optimal approach to the bottleneck, while the looser Euclidean heuristic led A* to a slightly suboptimal approach despite still finding a feasible path.

For ANA*, both heuristics converged to the same optimal cost of 35.9556 and again found only one solution per run. With Euclidean, ANA* achieved cost 35.9556 in 24.4900 seconds with 22,643 nodes expanded; with Diagonal, it achieved the same cost in 22.7837 seconds with 22,299 nodes expanded. Node counts were nearly identical, which is consistent with a bottleneck maze constraining the effective search space once the narrow corridor is discovered. The primary difference is that ANA* found the optimal route even with the Euclidean heuristic, whereas A* with Euclidean remained slightly worse.

Maze 5 is therefore the second maze where ANA* outperformed A* in solution quality under Euclidean (35.9556 versus 36.0727). Under Diagonal, both algorithms converged to the same optimal cost. Visually, paths were nearly identical for both heuristics, though both showed slight divergence in the open region before entering the bottleneck. This can be explained by the fact that the open space offers many near-equivalent approaches to the corridor entrance, allowing f(s) and e(s) priorities to select slightly different trajectories initially; once the bottleneck corridor is reached, the path becomes effectively constrained and solutions converge.

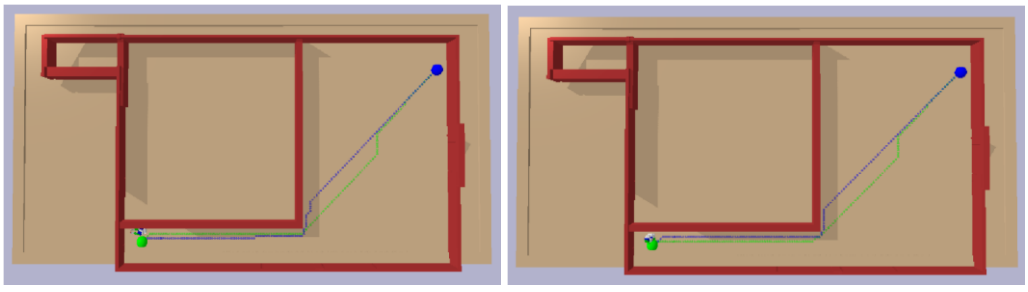

Figure 15: Maze 5 paths - Euclidean (left) and Diagonal (right)

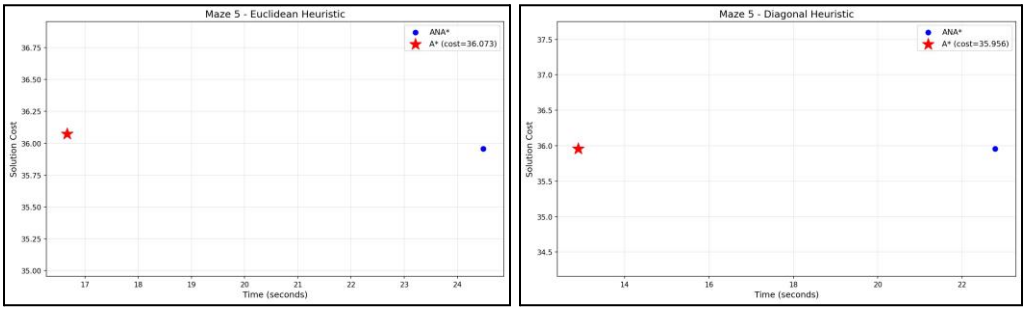

Figure 16: Maze 5 cost vs. time graphs - Euclidean (left) and Diagonal (right)

Maze 6: Two-Path Maze with Spiral Trap

For Maze 6, A* with Euclidean produced cost 45.5389 in 25.7717 seconds with 19,874 nodes expanded. With the Diagonal heuristic, A* achieved a better cost of 44.9732 with slightly improved runtime (25.0290 seconds) and fewer nodes expanded (18,504). In this environment, the tighter heuristic helped A* optimize among multiple route options while still successfully avoiding the spiral trap region.

For ANA*, Euclidean produced cost 45.7531 in 42.5883 seconds with 32,426 nodes expanded, while Diagonal produced cost 46.5531 in 44.1547 seconds with 31,629 nodes expanded. ANA* again found only a single solution in both runs, and in this maze both ANA* solutions were suboptimal relative to A*. Importantly, this is the first maze where the Diagonal heuristic hurt ANA*'s solution quality: Diagonal produced a worse cost than Euclidean (46.5531 versus 45.7531). A plausible explanation is that because ANA* uses e(s) where h(s) appears as a denominator, tightening h(s) can significantly change expansion order and may guide the initial greedy phase toward a suboptimal route in complex layouts, even though it benefits A*'s additive f(s) evaluation.

The overall comparison for Maze 6 shows A* outperforming ANA* in both time and cost, with the largest cost gap occurring under Diagonal: A* achieved 44.9732 whereas ANA* produced 46.5531 (about 3.5% worse). Both algorithms avoided the spiral trap. This can be explained by the trap's structure: the spiral increases g(s) significantly without providing a corresponding reduction in h(s), which deprioritizes the region under both f(s) and e(s)-based prioritization. Visually, paths were nearly identical across heuristics within each algorithm, with both preferring the upper route and avoiding the spiral region, which is consistent with the relatively similar costs observed within each algorithm.

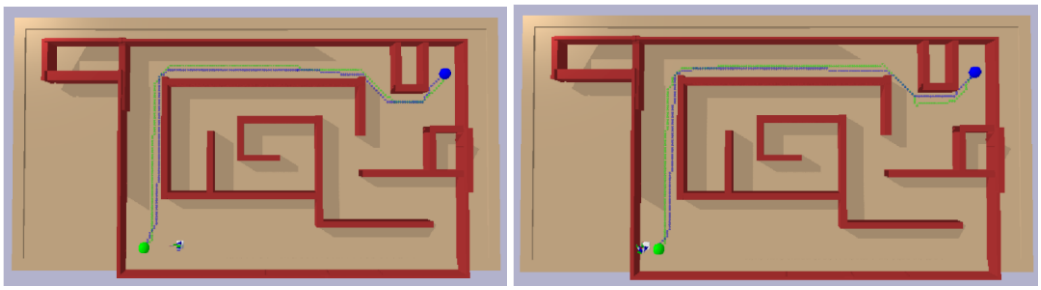

Figure 17: Maze 6 paths - Euclidean (left) and Diagonal (right)

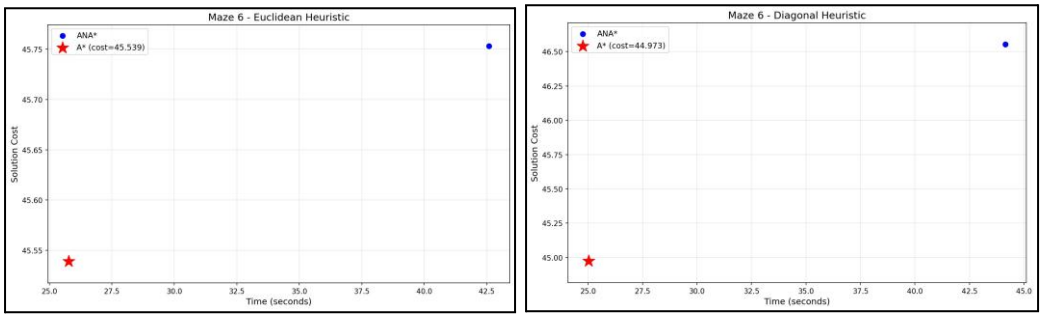

Figure 18: Maze 6 cost vs. time graphs - Euclidean (left) and Diagonal (right)

Overall Findings

COMPLETE RESULTS SUMMARY

Maze	Heuristic	Algorithm	Cost	Time (s)	Nodes Expanded
maze_1	Euclidean	A*	43.5615	6.5017	9501
maze_1	Euclidean	ANA*	43.7472	10.3965	15998
maze_1	Diagonal	A*	43.3958	11.3475	8798
maze_1	Diagonal	ANA*	43.8443	21.9710	16090
maze_2	Euclidean	A*	37.6502	31.1733	25577
maze_2	Euclidean	ANA*	38.4301	44.2724	35318
maze_2	Diagonal	A*	37.6502	29.3233	23366
maze_2	Diagonal	ANA*	38.1472	46.7025	35002
maze_3	Euclidean	A*	39.2904	21.3886	14531
maze_3	Euclidean	ANA*	39.2904	39.9187	24983
maze_3	Diagonal	A*	39.4075	17.9349	12582
maze_3	Diagonal	ANA*	39.2904	32.2714	24479
maze_4	Euclidean	A*	35.0183	22.6781	15285
maze_4	Euclidean	ANA*	35.2526	44.3578	38534
maze_4	Diagonal	A*	35.0183	13.9572	8801
maze_4	Diagonal	ANA*	35.0183	45.2349	37995
maze_5	Euclidean	A*	36.0727	16.6635	15046
maze_5	Euclidean	ANA*	35.9556	24.4900	22643
maze_5	Diagonal	A*	35.9556	12.9025	9759
maze_5	Diagonal	ANA*	35.9556	22.7837	22299
maze_6	Euclidean	A*	45.5389	25.7717	19874
maze_6	Euclidean	ANA*	45.7531	42.5883	32426
maze_6	Diagonal	A*	44.9732	25.0290	18504
maze_6	Diagonal	ANA*	46.5531	44.1547	31629

Total runs: 24

Table: Summary table of results for all 12 runs

Across all six mazes, A* outperformed ANA* in runtime under both heuristics. With the Euclidean heuristic, A* was faster in all six mazes and was approximately 1.7× faster than ANA* on average. In terms of solution quality, A* achieved better or equal cost in five of the six mazes, with ANA* achieving a better cost only in Maze 5 (35.9556 versus 36.0727). A* also expanded significantly fewer nodes in exploitation, while ANA*'s greedy$f(s)$ e(s)-based= $g(s)$ + priority$h(s)$ expanded many additional states searching for

every maze, indicating that its ordering efficiently balances exploration and

improvements that rarely occurred.

With the Diagonal heuristic, the performance gap widened: A* was again faster in all six mazes and was approximately 2.5× faster than ANA* on average. A* achieved better or equal cost in five of six mazes, with ANA* achieving a better cost only in Maze 3 (39.2904 versus 39.4075). This widening gap is consistent with the fact that Diagonal's tighter estimate amplifies A*'s pruning advantage, whereas ANA* cannot leverage the same benefit because it uses the heuristic differently in its priority function.

The most significant overall finding is that ANA*'s anytime property did not manifest in these experiments. Across all 12 test runs (6 mazes × 2 heuristics), ANA* found only a single solution each time and no iterative improvement was observed. This indicates that, in these maze environments, either (1) the structure strongly constrains the solution to a dominant route, or (2) multiple routes exist but are sufficiently similar, such that once an initial solution is found, there is little opportunity or incentive for ANA* to discover and report improved alternatives. In such conditions, ANA*'s overhead, maintaining the incumbent cost G, computing e(s), and managing pruning and reordering, adds runtime without yielding the expected anytime benefits.

Heuristic effects were more consistent for A* than for ANA*. For A*, the Diagonal heuristic reduced nodes expanded in five of the six mazes and reduced time in four of the six mazes, and it produced equal or better costs in five of the six mazes. The only case where Diagonal hurt A*'s cost was Maze 3, where it returned 39.4075 versus 39.2904 for Euclidean. Overall, Diagonal was clearly beneficial for A* because tighter heuristic estimates reduce unnecessary exploration by decreasing the number of states that appear equally promising under f(s).

For ANA*, the effect of heuristic choice was inconsistent. Runtime split evenly, with Diagonal faster in three mazes and Euclidean faster in three mazes. Solution quality was also mixed: Diagonal produced better cost in two mazes, Euclidean produced better cost in two mazes, and the remaining two were ties. Node counts remained within a similar range regardless of heuristic, suggesting that heuristic tightness typically changed expansion order more than it reduced the total explored search space. This inconsistency is explained by the fact that ANA* uses the heuristic as a denominator in $e(s) = \frac{G - g(s)}{h(s)}$,

meaning that tightening h(s) can reshuffle expansion priorities in ways that are sensitive to maze structure, sometimes helping and sometimes hurting.

Overall, these results indicate that A* is the better algorithm choice for the PR2 navigation tasks studied here. ANA* did not provide an anytime advantage in these environments due to limited path diversity and minimal observable solution refinement. The Diagonal heuristic reliably improved A* performance, while its impact on ANA* was unpredictable and dependent on maze structure.

BIBLIOGRAPHY:

[1] J. van den Berg, R. Shah, A. Huang, and K. Goldberg, "ANA*: Anytime Nonparametric A*," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 25, no. 1, San Francisco, CA, USA, 2011

[2] A. Patel, "Heuristics for Grid-Based Pathfinding," *Red Blob Games*, Stanford University. [Online]. Available: https://theory.stanford.edu/~amitp/GameProgramming/Heuristics.html