Motion-Planning HW2

Nir Manor 305229627, Ortal Cohen 308524875

07/03/2024

Distribution of points in high-dimensional spaces

2.1. In a 2-dimensional unit ball (namely, a disk), how much of the volume is located at most 0.1 units from the surface?

Answer: 20%.

2.2. In a 9-dimensional unit ball, how much of the volume is located at most 0.1 units from the surface?

Answer: 60%.

2.3. Define the fraction of the volume that is ϵ distance from the surface of a d-dimensional unit ball as $\mu_d(\epsilon)$.

Answer: The volume of a d-dimensional ball of radius R is:

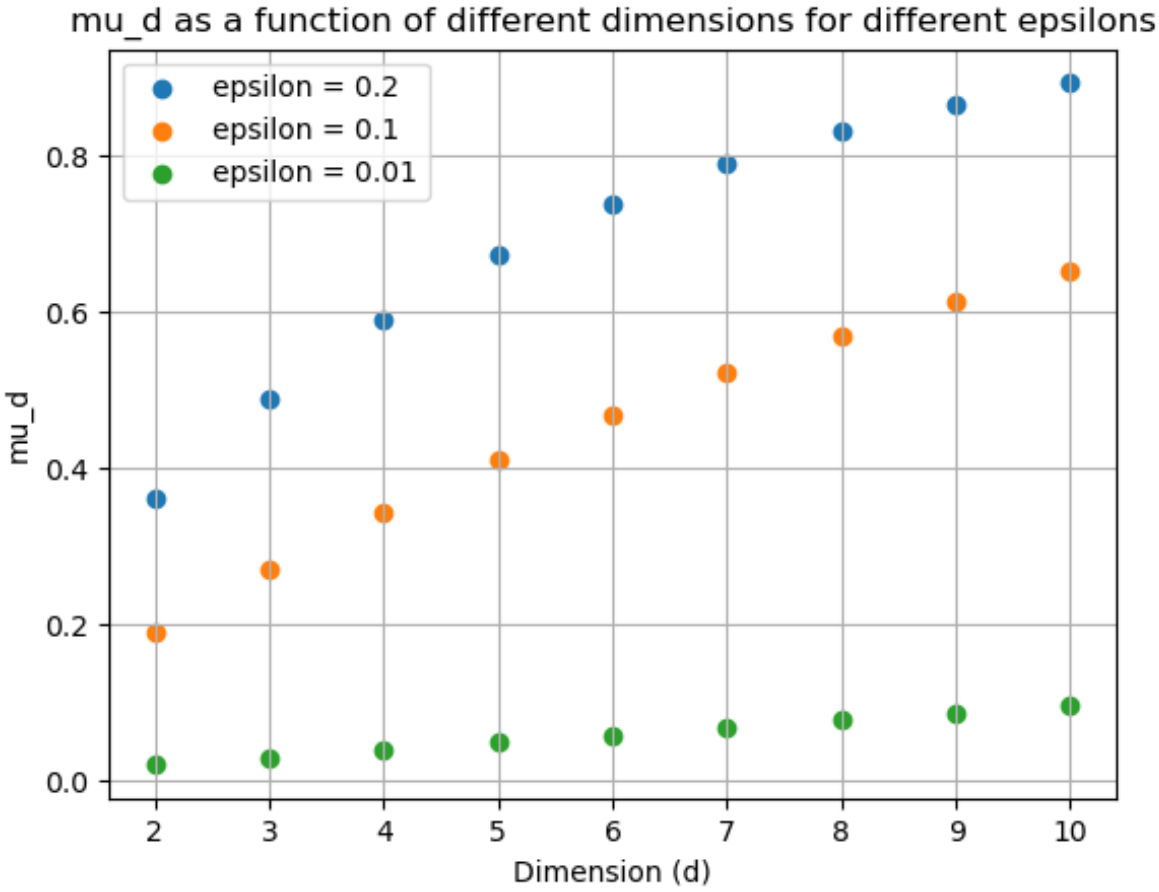
$$V_d(R) = \frac{\pi^{\frac{d}{2}}}{\Gamma(\frac{d}{2}+1)} R^d \text{ where } \Gamma(n) = (n-1)! \quad \forall \mathbb{N}$$

The unit ball is a ball of radius 1 ($R = 1$) and therefore the volume of a d-dimensional unit ball is:

$$V_d(1) = \frac{\pi^{\frac{d}{2}}}{\Gamma(\frac{d}{2}+1)}$$

We will define the fraction of the volume that is ϵ distance from the surface of a d-dimensional unit ball as $\mu_d(\epsilon)$:

$$\mu_d(\epsilon) = \frac{V_d(1) - V_d(1-\epsilon)}{V_d(1)} = \frac{\frac{\pi^{\frac{d}{2}}}{\Gamma(\frac{d}{2}+1)} - \frac{\pi^{\frac{d}{2}}}{\Gamma(\frac{d}{2}+1)}(1-\epsilon)^d}{\frac{\pi^{\frac{d}{2}}}{\Gamma(\frac{d}{2}+1)}} = 1 - (1-\epsilon)^d$$



Reducing the connection radius required for a sampling-based algorithm can improve the algorithm's efficiency by focusing on local exploration and reducing the computational time. However, this come at the cost of optimality, as the algorithm may not explore the entire space thoroughly enough to guarantee convergence towards the optimal solution because by reducing the connection radius, the algorithm may prioritize exploiting known regions over exploration.

Tethered robots

3.1. Describe the structure of a path from a start to a target configuration (robot location + tether description).

Answer:

Based on the sections provided, the structure of a path from a start to target configuration involves the following components:

- Robot Location: Represented as a point (x, y) in the 2D space amidst polygonal obstacles.

- Tether Description: Describes the configuration of the tether relative to the robot's position. This includes the length and shape of the tether, ensuring it remains taut and does not exceed a maximum length L from the base point. Each configuration is represented by a tuple (v, w, c) in the homotopy-augmented graph G_h, where v is a vertex of the graph G, w defines a homotopy class (h-signature), and c is the cost (Euclidean shortest path) to reach the point from the base in the homotopy class defined by w.

Therefore, the path structure consists of a sequence of robot locations and corresponding tether descriptions, ensuring adherence to tether constraints while navigating obstacles. The h-signature denotes the homotopy class, capturing different configurations of the tether.

3.2. Suggest an efficient way to encode the tether's description. Note that the robot can be at the same location but with completely different tether descriptions (e.g., circling once or twice around an obstacle) and what we need to define is the homotopy class of the tether.

Answer:
To efficiently encode the tether's description, we utilize the concept of homotopy classes and h-signatures. This h-signature was learned in class and uniquely identifies the homotopy class of the tether, capturing different configurations, such as circling around obstacles or following curved paths. Each h-signature is a sequence of letters derived from the intersections of the tether with vertical rays extended from obstacle points. By constructing the reduced signature, we obtain the h-signature, a homotopy invariant that uniquely identifies the tether's homotopy class. Therefore, encoding the tether's description involves representing these h-signatures efficiently, allowing for effective path planning while considering tether constraints.

3.3. The homotopy-augmented graph G_h of a graph $G = (V, E)$ encodes for each vertex of G, all homotopy classes that can be used to reach the vertex using a tether of length L. Describe an approach to compute the homotopy-augmented graph of a given graph using a Dijkstra-like algorithm. Describe the nodes, how they are extended and when the algorithm terminates.

Answer:
To compute the homotopy-augmented graph G_h of a given graph $G = (V, E)$, an approach utilizing a Dijkstra-like algorithm can be employed. Here's an outline of the

approach:

1. Nodes Description:

    - Each node in G_h represents a vertex of the original graph G along with all homotopy classes that can be used to reach that vertex using a tether of length L.
    - Nodes are represented as tuples (v, w, c), where v is a vertex of G, w defines an h-signature representing a homotopy class, and c is the cost (Euclidean shortest path) to reach the vertex v from the base in the homotopy class defined by w.

2. Nodes Extension:

    - The algorithm starts by constructing the reduced visibility graph $G^{vis} = (V^{vis}, E^{vis})$ over the set of obstacles, including the base point p_b as an additional vertex.
    - Using the reduced visibility graph G^{vis}, the homotopy-augmented visibility graph G_h^{vis} can be computed, as described in question 2. In class, we learned that curves with the same homotopy class can be merged into an edge in the visibility graph.
    - Then, it constructs a data structure to efficiently compute the h-signature of a line segment, which is crucial for identifying homotopy classes.
    - Using this data structure, the algorithm extends nodes by considering all possible homotopy classes that can be used to reach each vertex of G from the base point p_b using a tether of length L.

3. Algorithm Termination:

    - The algorithm terminates when all nodes of cost less than L have been exposed.
    - Each vertex is considered by the Dijkstra-like algorithm, and for each adjacent vertex, a node is added to G_h with an updated h-signature and cost if the following conditions are met: (i) the cost is less than or equal to L, and (ii) there is no existing node for the same vertex and cost.

3.4. Now we will use the notion of a homotopy-augmented graph to efficiently answer queries solving the motion-planning problem for a point tethered robot. A query is given in the form of two points p_s, p_t and the h-invariant w_s describing the tethered placement at p_s. In order to compute a path (if one exists) between p_s and p_t with an original

tether placement defined by w_s, we need to traverse the homotopy-augmented graph G_h^{vis} (the homotopy-augmented graph of the visibility graph).

Answer:

To efficiently solve the motion planning problem for a point tethered robot using the homotopy-augmented graph G_h^{vis}, we need to consider the following:

1. New Vertices Addition:
    - We need to add new vertices to G_h^{vis} to account for the start point p_s and the target point p_t.
    - For p_s, we add a vertex corresponding to the point p_s along with all possible h-signatures that can be used to reach p_s from the base point p_b using the given tether configuration.
    - Similarly, for p_t, we add a vertex corresponding to the point p_t along with all possible h-signatures that can be used to reach p_t from the base point p_b.
2. New Edges Addition:
    - After adding the vertices for p_s and p_t, we need to add new edges to G_h^{vis} to connect these vertices with the existing vertices in the graph.
    - For p_s, we add edges between the newly introduced vertex corresponding to p_s and existing vertices in G_h^{vis} that are reachable from p_s. These edges represent feasible paths from p_s to other points in the graph while adhering to tether constraints.
    - Similarly, for p_t, we add edges between the newly introduced vertex corresponding to p_t and existing vertices in G_h^{vis} that are reachable from p_t.
3. Efficient Use of the Newly-Constructed Graph:
    - Once the graph G_h^{vis} is constructed with added vertices and edges for p_s and p_t, we can efficiently solve our motion planning problem using graph search algorithms like A*.
    - We start the search from the vertex corresponding to the start point p_s and traverse the graph to find the shortest path to the target point p_t, considering the tether constraints encoded in the h-signatures.
    - By leveraging the precomputed homotopy-augmented graph G_h^{vis}, we can efficiently explore feasible paths for the tethered robot motion planning problem, ensuring optimal solutions while adhering to the tether constraints.

In summary, by adding appropriate vertices and edges to the homotopy-augmented graph G_h^{vis} and utilizing graph search algorithms, we can efficiently solve queries for the motion-planning problem of a point tethered robot.

Motion Planning: Search and Sampling

A* Implementation

Answer:

$\epsilon = 1.0$

- Total iterations: 43727
- Total expanded nodes: 43727
- Total cost: 349.10
- Update operations on open nodes: 25475
- Update operations on close nodes: 0

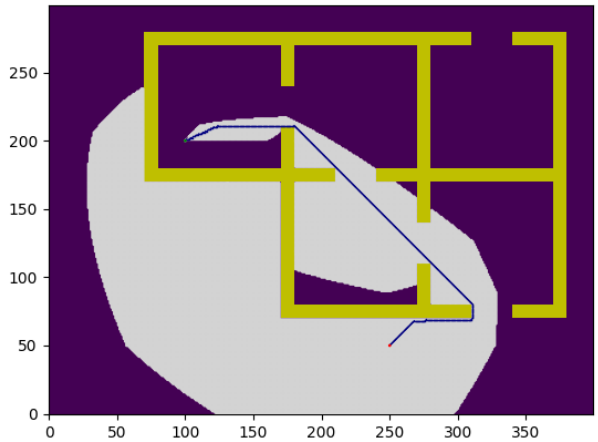


Figure 1: Visualization of A* with $\epsilon = 1.0$

$\epsilon = 10.0$

- Total iterations: 407035
- Total expanded nodes: 407035

- Total cost: 355.55
- Update operations on open nodes: 162471
- Update operations on close nodes: 356715

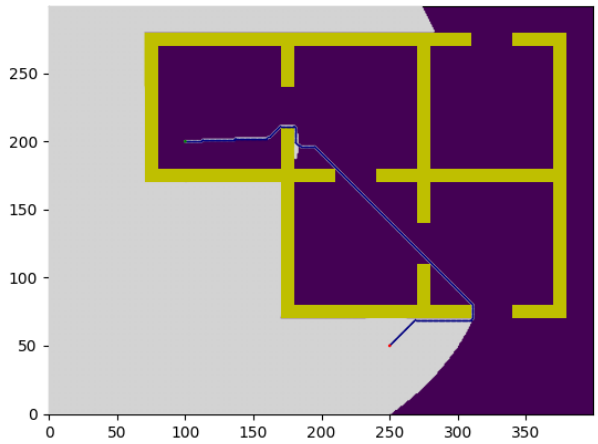


Figure 2: Visualization of A* with $\epsilon = 10.0$

$\epsilon = 20.0$

- Total iterations: 475899
- Total expanded nodes: 50700
- Total cost: 354.96
- Update operations on open nodes: 178370
- Update operations on close nodes: 425199

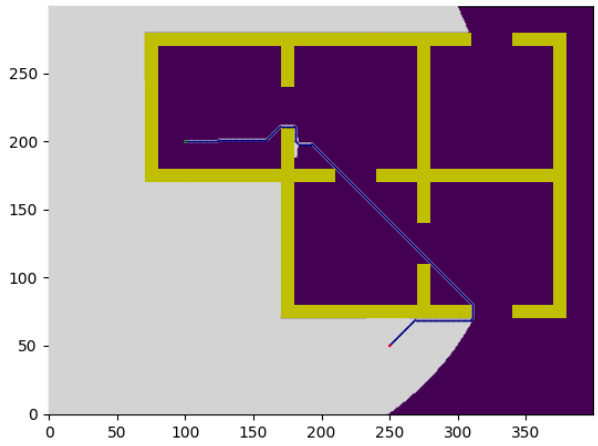


Figure 3: Visualization of A* with $\epsilon = 20.0$

Summary Increasing ϵ causes the algorithm to rely more on the heuristic function, aiming to speed up the search process by prioritizing nodes closer to the target. However, this can lead to a "greedy" behavior where the algorithm tries to get closer to the target without adequately considering obstacles in the space. As a result, the algorithm might overlook potential paths and struggle to find the solution efficiently, as seen in the cases of $\epsilon = 10.0$ and $\epsilon = 20.0$. Therefore, the choice of ϵ should be carefully considered based on the problem's complexity and the desired balance between exploration and exploitation.

RRT and RRT* Implementation

4.4.1. Bias the sampling to pick the goal with 5%, 20% probability. Report the performance (cost, time) and include figures showing the final state of the tree for both values.

Answer:

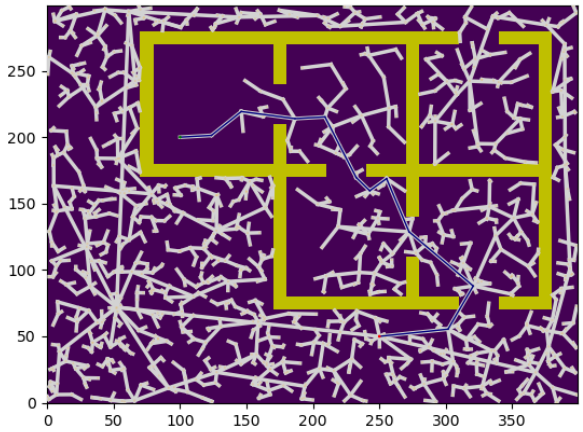
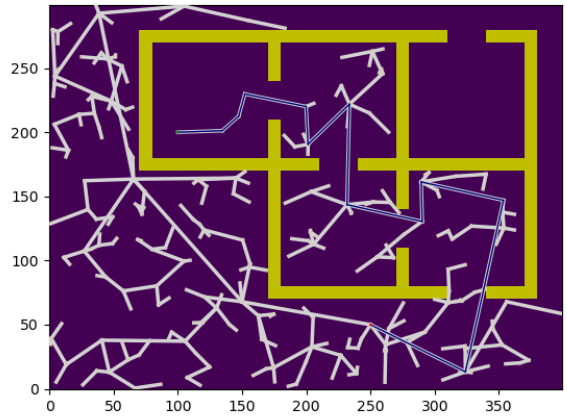
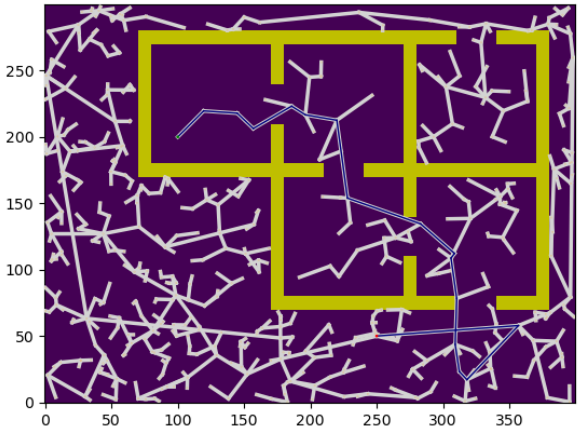
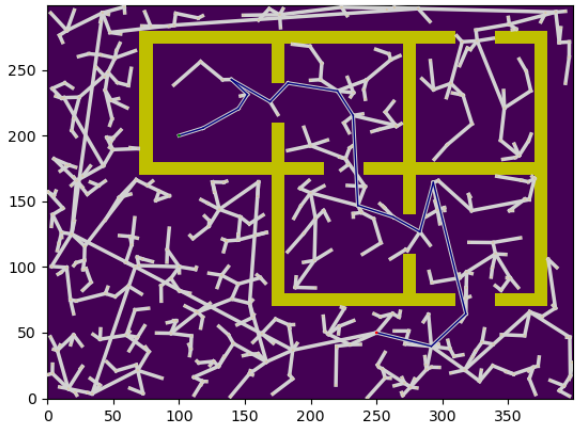
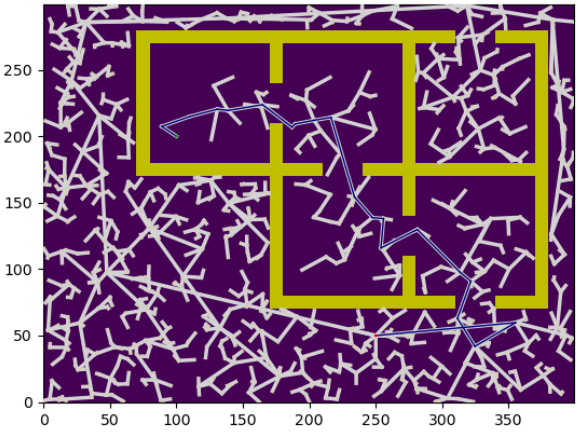
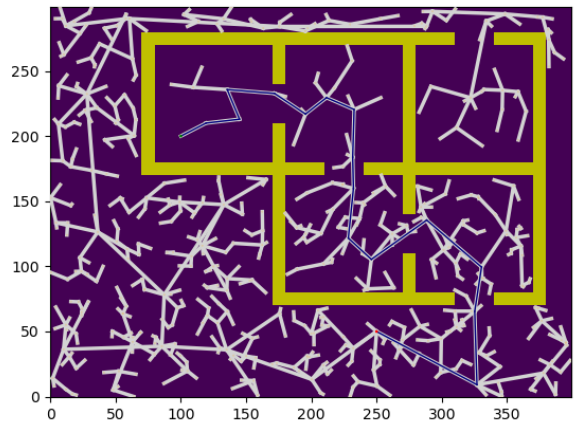
Case 1: Sampling is biased to pick the goal with 5% probability

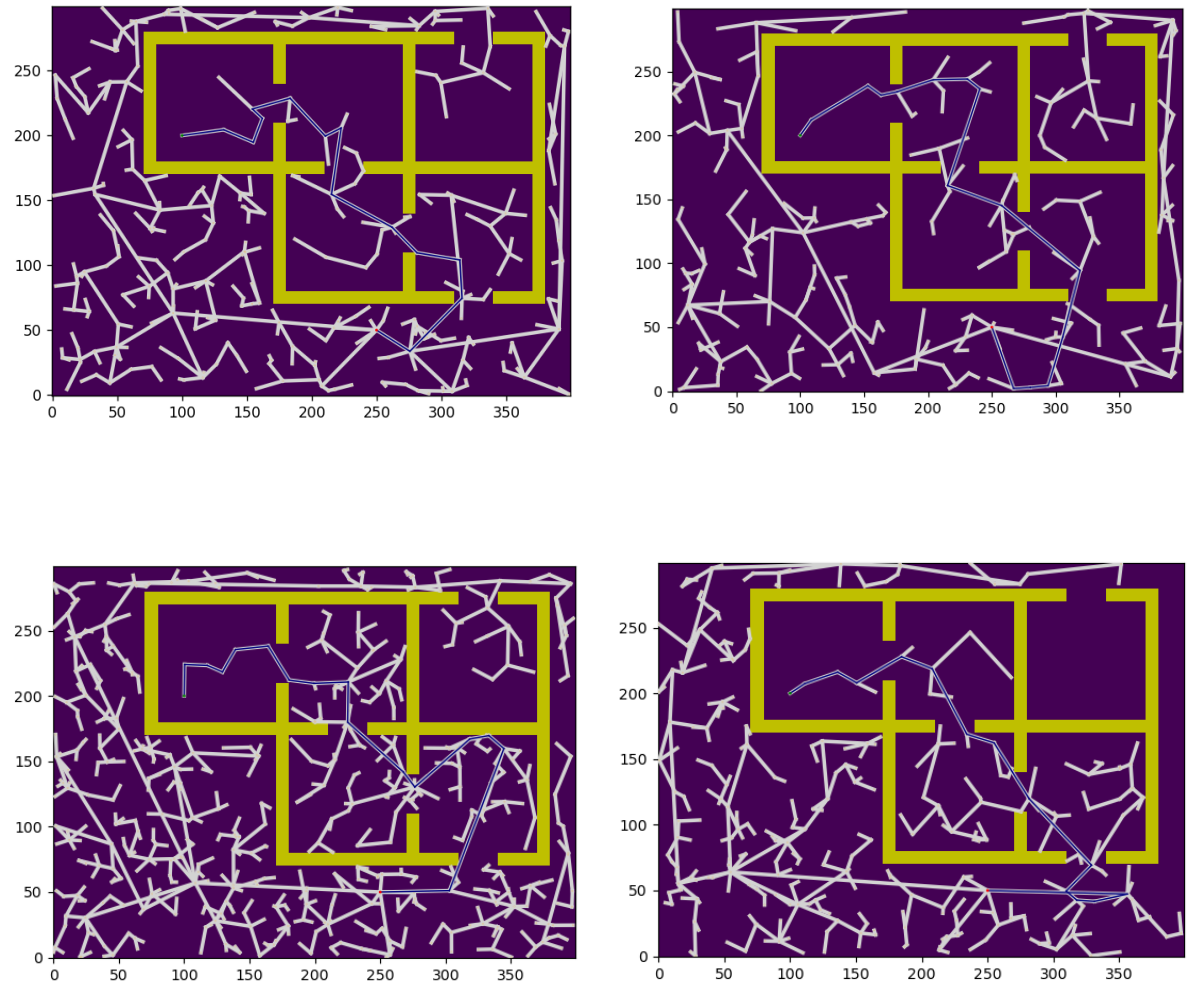
Average cost after 10 runs: 528.853

Average time after 10 runs: 3.024

Costs for 10 runs: [393.35, 646.85, 553.65, 533.97, 546.01, 588.86, 454.51, 536.99, 532.56, 501.78]

Times for 10 runs: [10.38, 0.7, 2.47, 2.2, 6.7, 2.51, 1.11, 0.61, 2.59, 0.97]



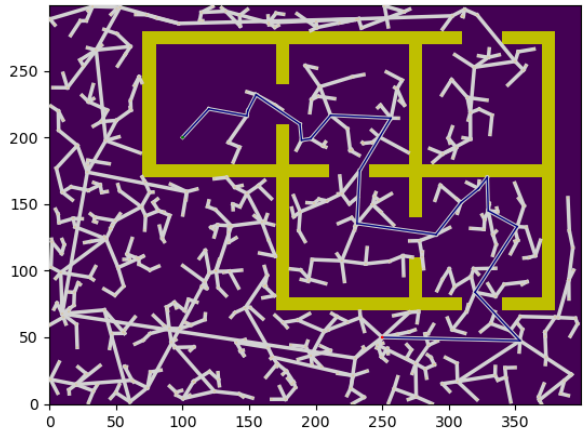
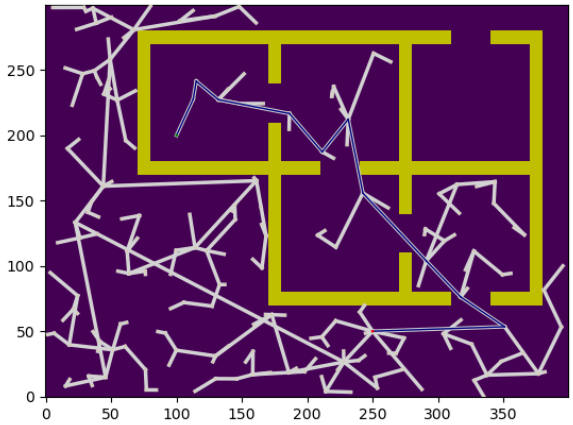
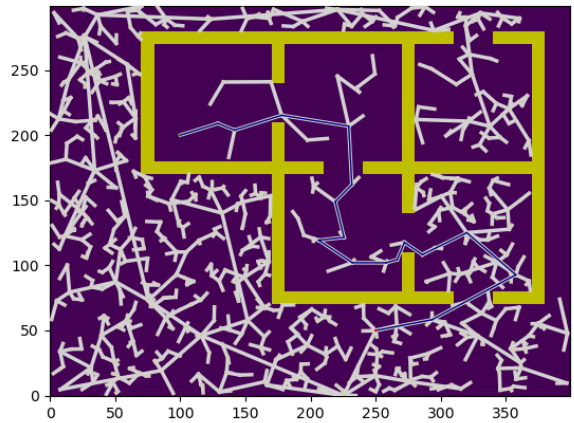
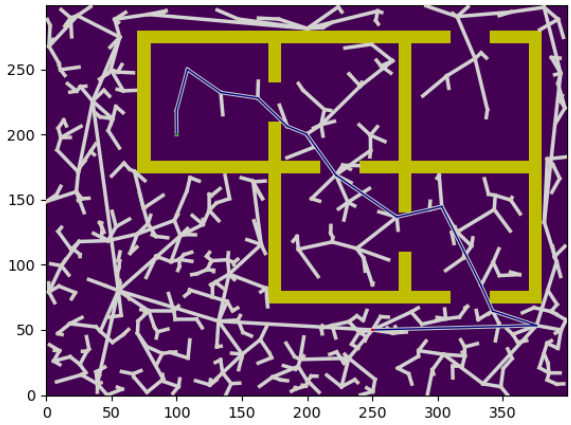
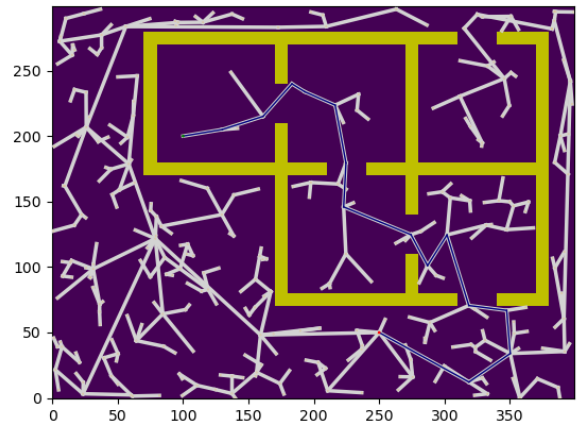
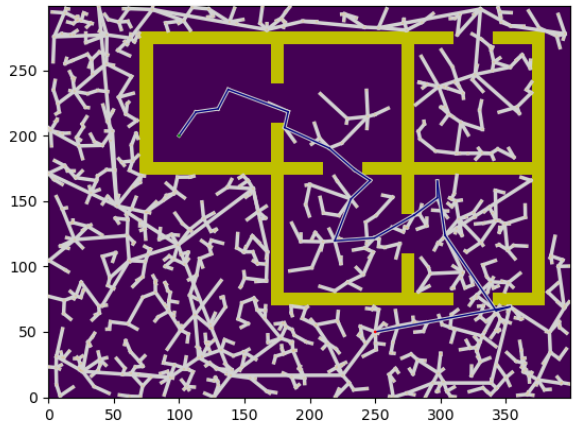
Case 2: Sampling is biased to pick the goal with 20% probability

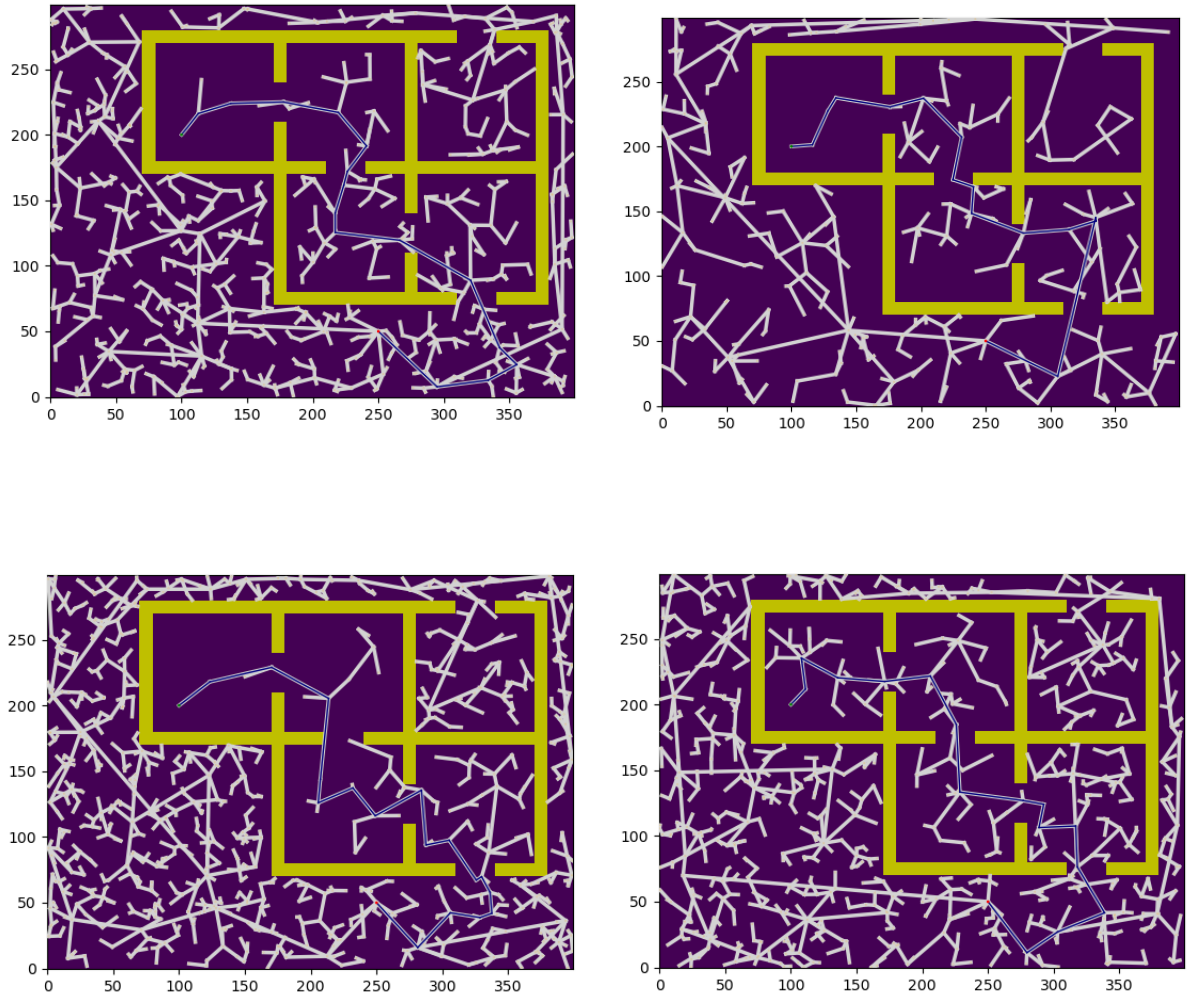
Average cost after 10 runs: 554.87

Average time after 10 runs: 4.091

Costs for 10 runs: [668.15, 501.19, 546.62, 539.41, 557.75, 577.56, 546.58, 529.41, 548.46, 533.57]

Times for 10 runs: [3.22, 0.58, 6.48, 2.81, 1.07, 7.58, 4.44, 0.92, 8.99, 4.82]



4.4.2. We chose $\mu = 10$.

Case 1.1: Sampling is biased to pick the goal with 5% probability with E1

Like in 4.4.1.

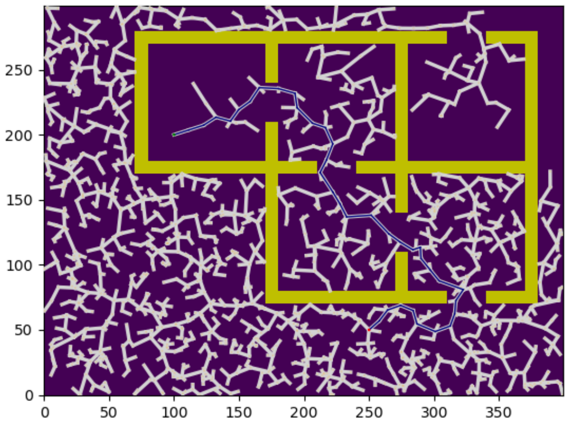
Case 1.2: Sampling is biased to pick the goal with 5% probability with E2

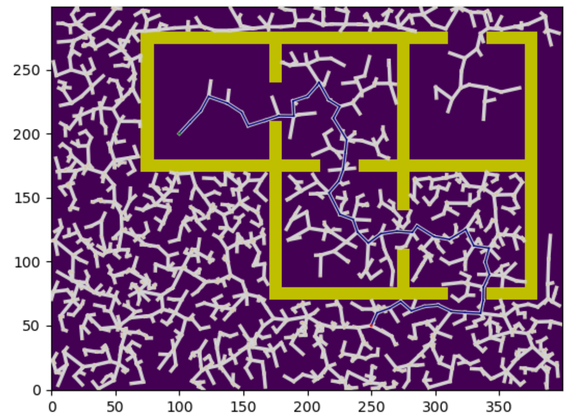
Average cost after 10 runs: 506.108

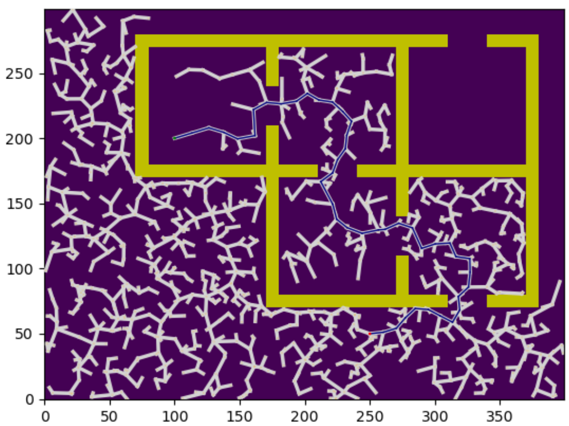
Average time after 10 runs: 9.790000000000001

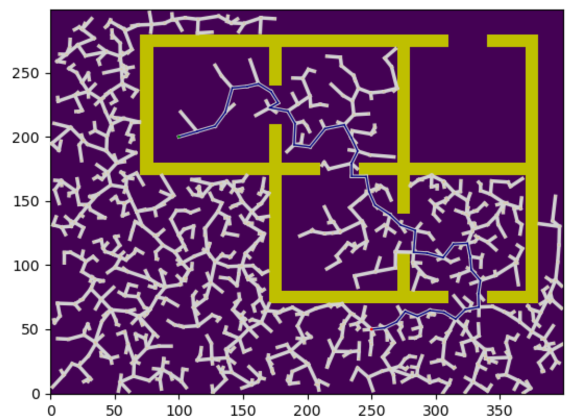
Costs for 10 runs: [453.28, 569.86, 508.62, 530.08, 515.48, 487.49, 581.89, 445.2, 492.58, 476.6]

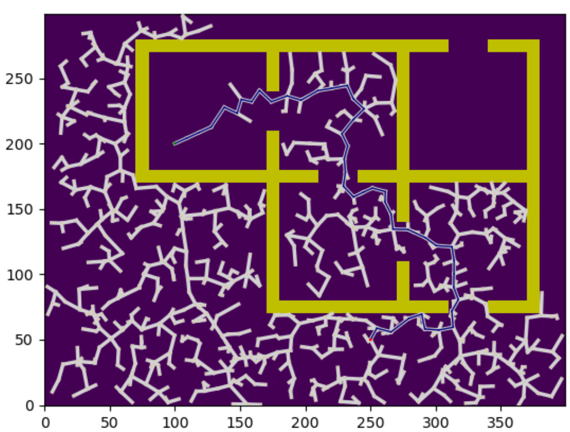
Times for 10 runs: [14.26, 15.9, 8.49, 8.45, 5.68, 10.44, 10.67, 13.13, 7.32, 3.56]

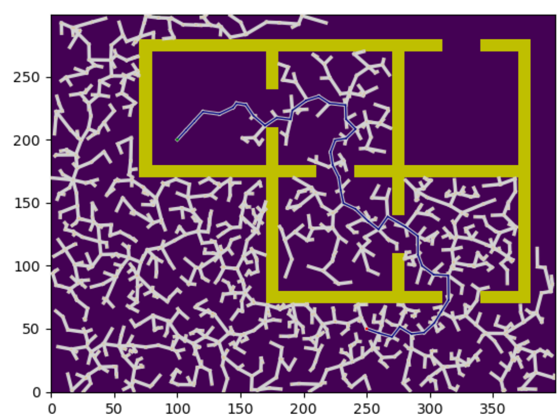

0 50 100 150 200 250
0 50 100 150 200 250 300 350

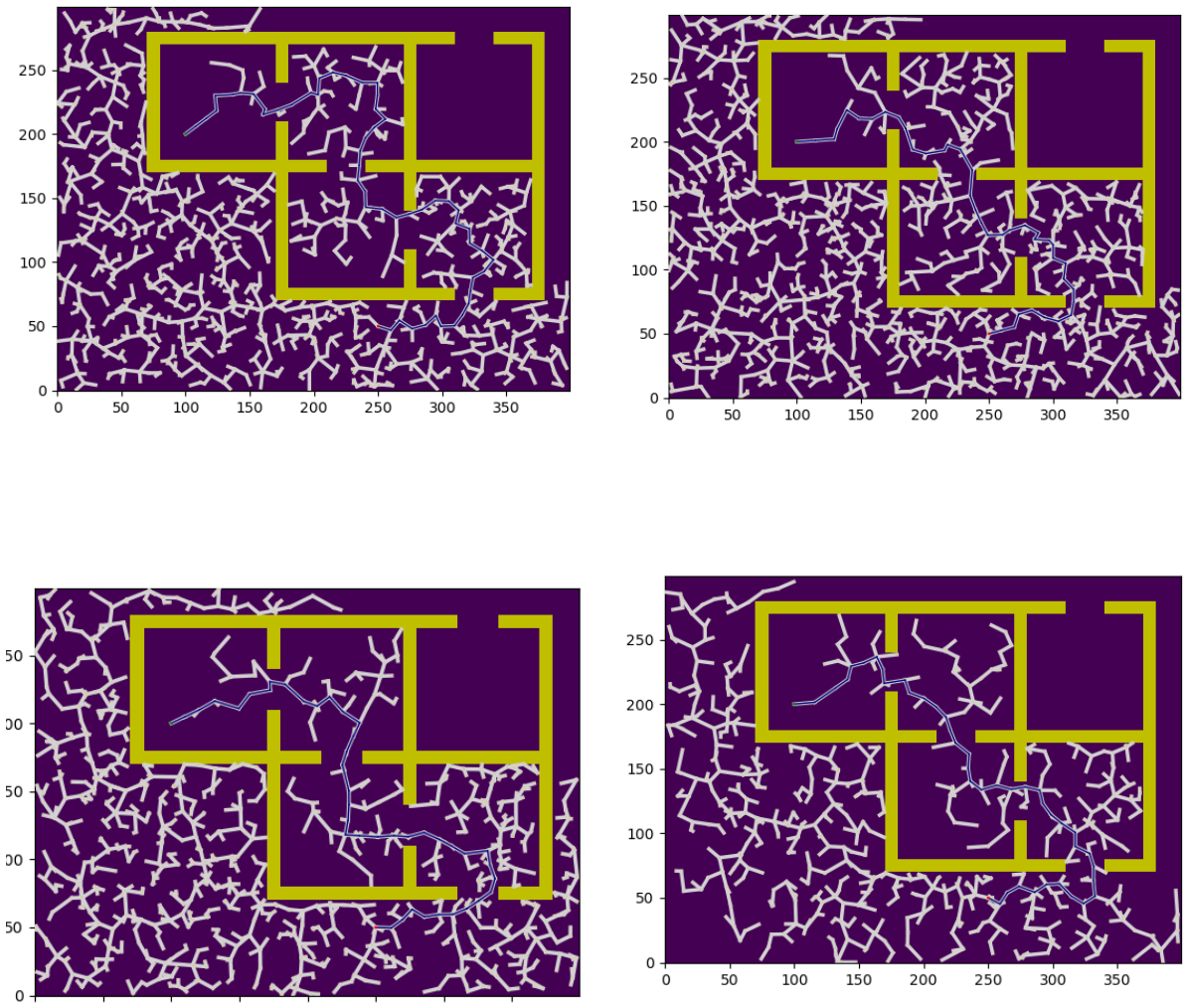

0 50 100 150 200 250
0 50 100 150 200 250 300 350


0 50 100 150 200 250
0 50 100 150 200 250 300 350


0 50 100 150 200 250
0 50 100 150 200 250 300 350


0 50 100 150 200 250
0 50 100 150 200 250 300 350


0 50 100 150 200 250
0 50 100 150 200 250 300 350



Case 2.1: Sampling is biased to pick the goal with 5% probability with E1

Like in 4.4.1.

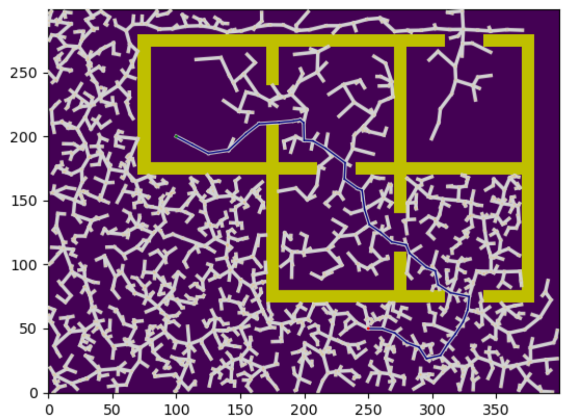
Case 2.2: Sampling is biased to pick the goal with 20% probability with E2

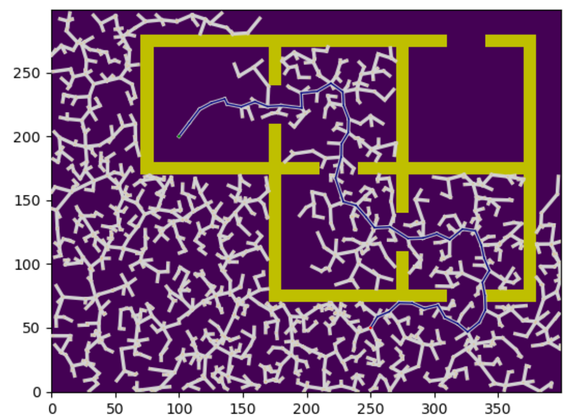
Average cost after 10 runs: 507.088

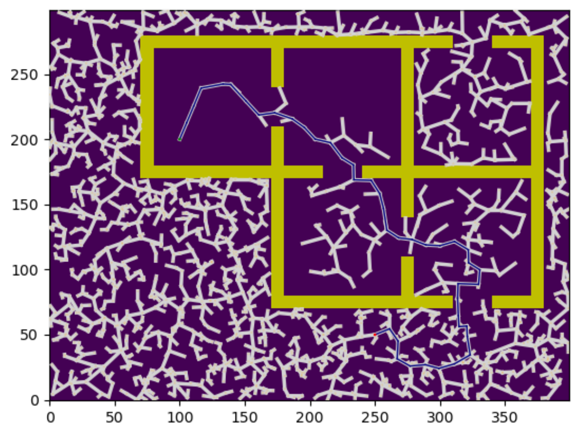
Average time after 10 runs: 15.642

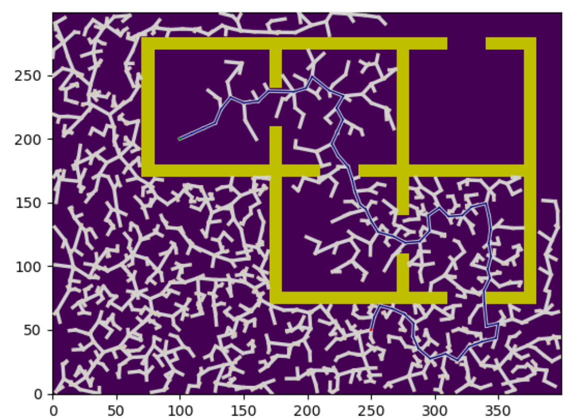
Costs for 10 runs: [440.05, 553.29, 523.73, 654.45, 492.9, 422.78, 492.12, 437.3, 505.13, 549.13]

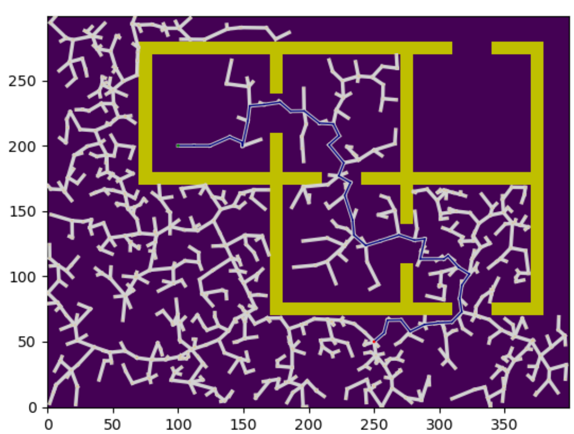
Times for 10 runs: [23.92, 16.74, 26.91, 17.46, 7.03, 8.68, 8.16, 14.12, 26.89, 6.51]

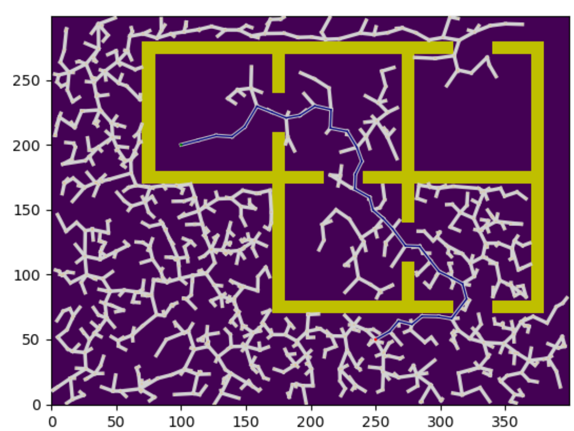

0 50 100 150 200 250
0 50 100 150 200 250 300 350

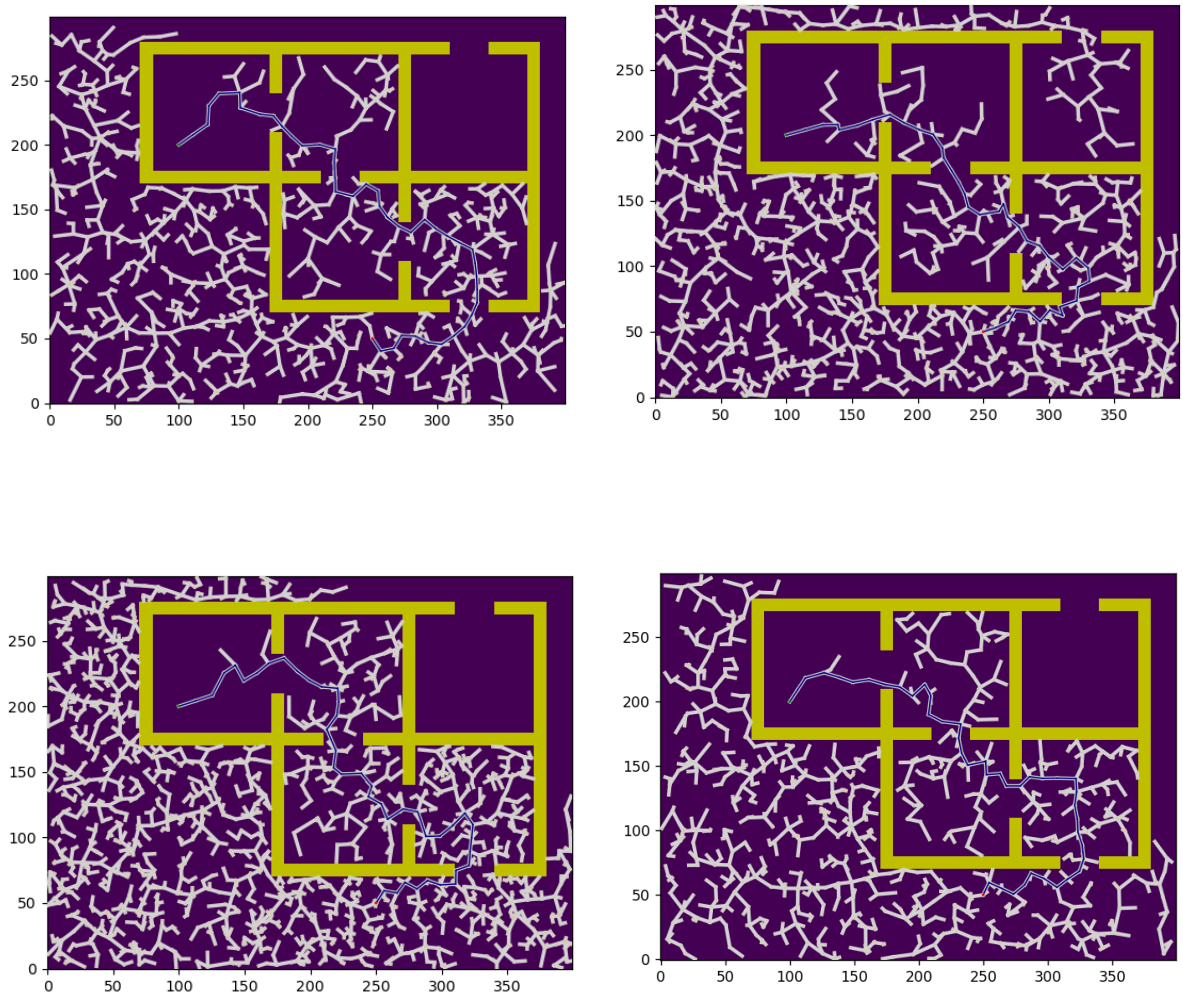

0 50 100 150 200 250
0 50 100 150 200 250 300 350


0 50 100 150 200 250
0 50 100 150 200 250 300 350

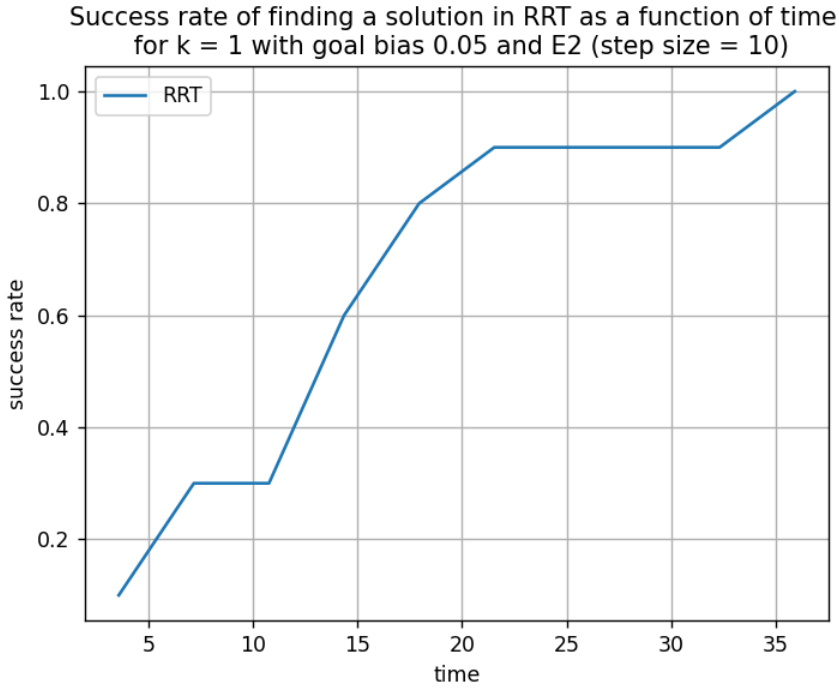

0 50 100 150 200 250
0 50 100 150 200 250 300 350

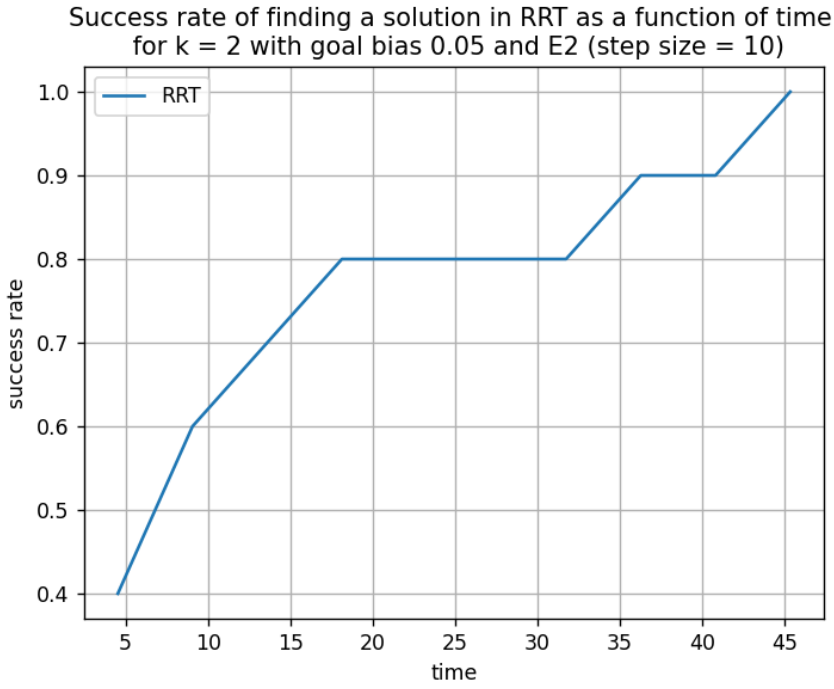

0 50 100 150 200 250
0 50 100 150 200 250 300 350

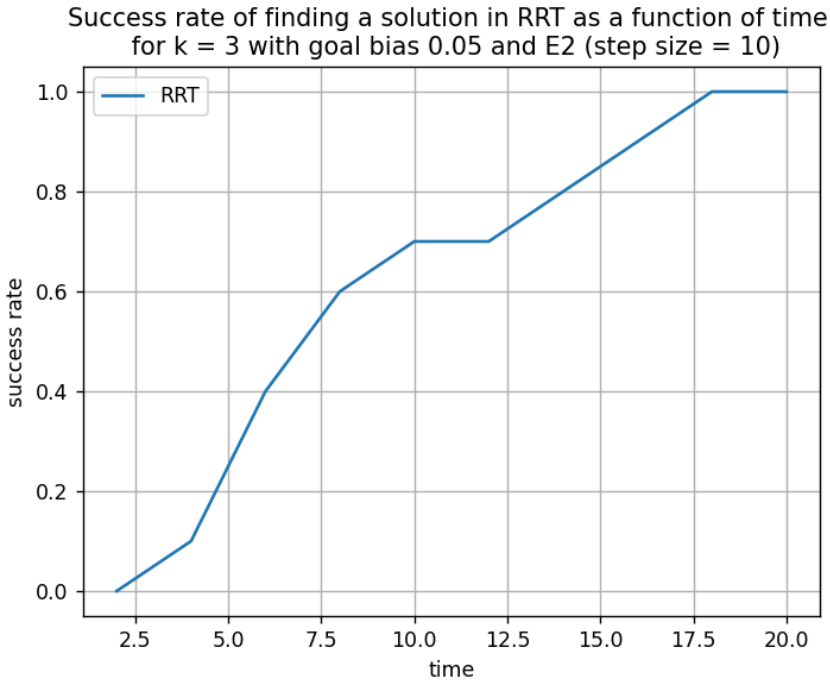

0 50 100 150 200 250
0 50 100 150 200 250 300 350

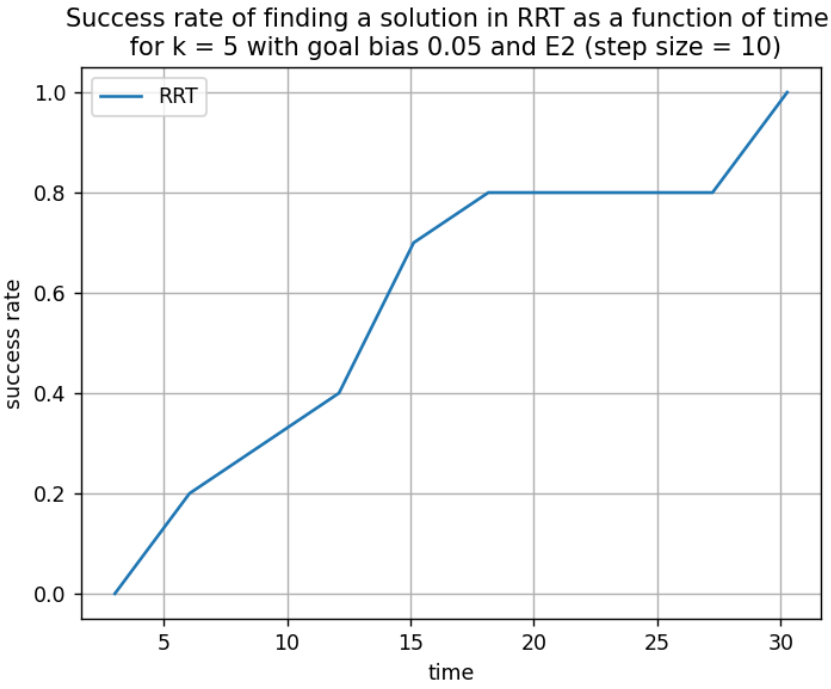


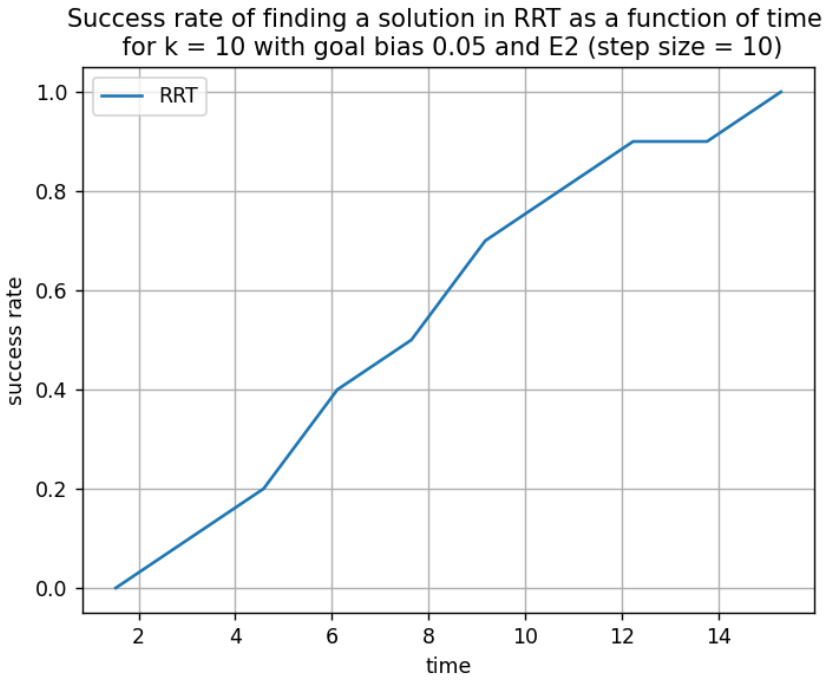
4.4.3. **Plots of success rate of RRT:**

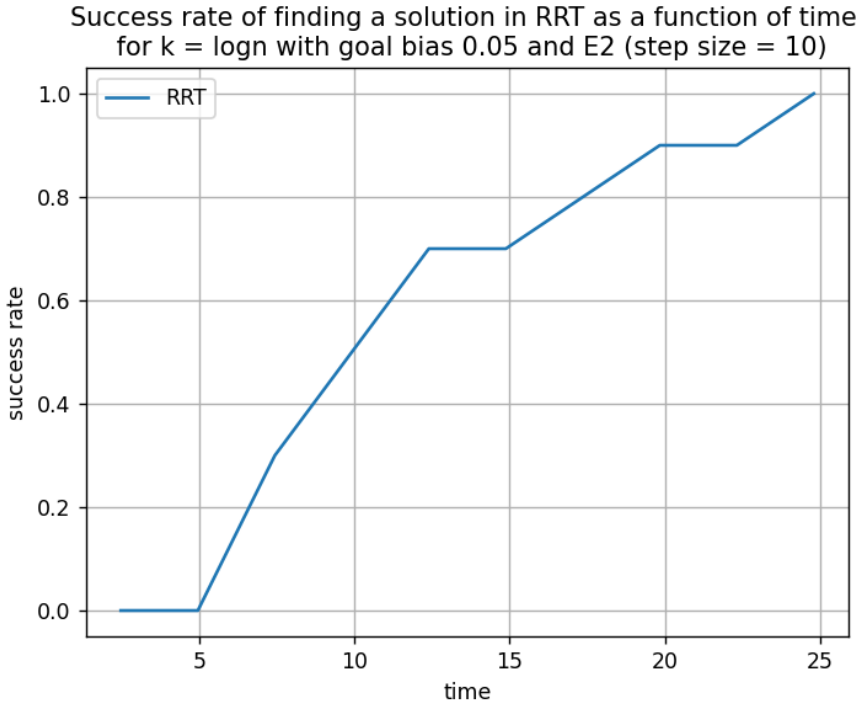

Success rate of finding a solution in RRT as a function of time
for k = 1 with goal bias 0.05 and E2 (step size = 10)
RRT
success rate
time
1.0
0.8
0.6
0.4
0.2
5
10
15
20
25
30
35


Success rate of finding a solution in RRT as a function of time
for k = 2 with goal bias 0.05 and E2 (step size = 10)
RRT
success rate
time
1.0
0.9
0.8
0.7
0.6
0.5
0.4
5
10
15
20
25
30
35
40
45

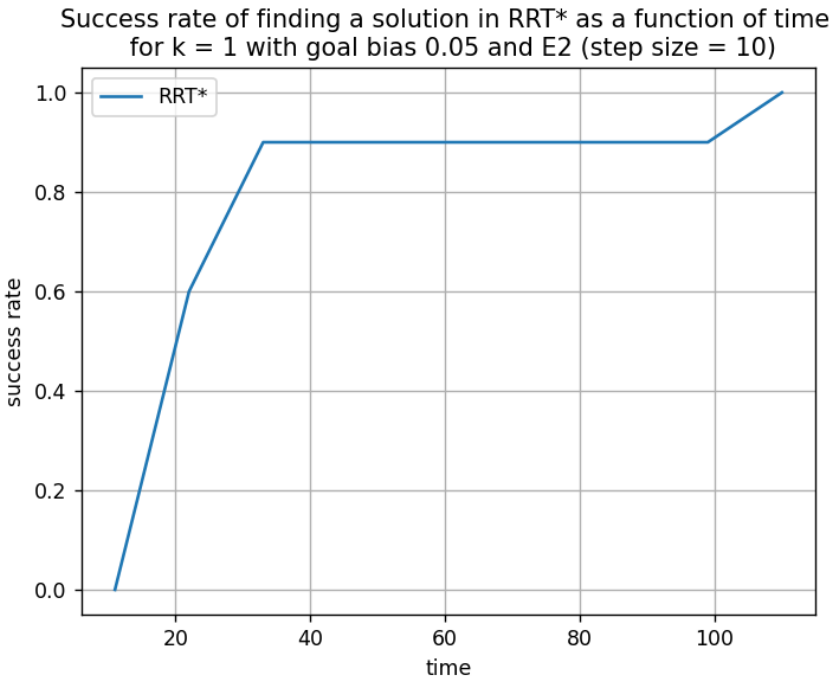

Success rate of finding a solution in RRT as a function of time
for k = 3 with goal bias 0.05 and E2 (step size = 10)
RRT
success rate
time
0.0
0.2
0.4
0.6
0.8
1.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0

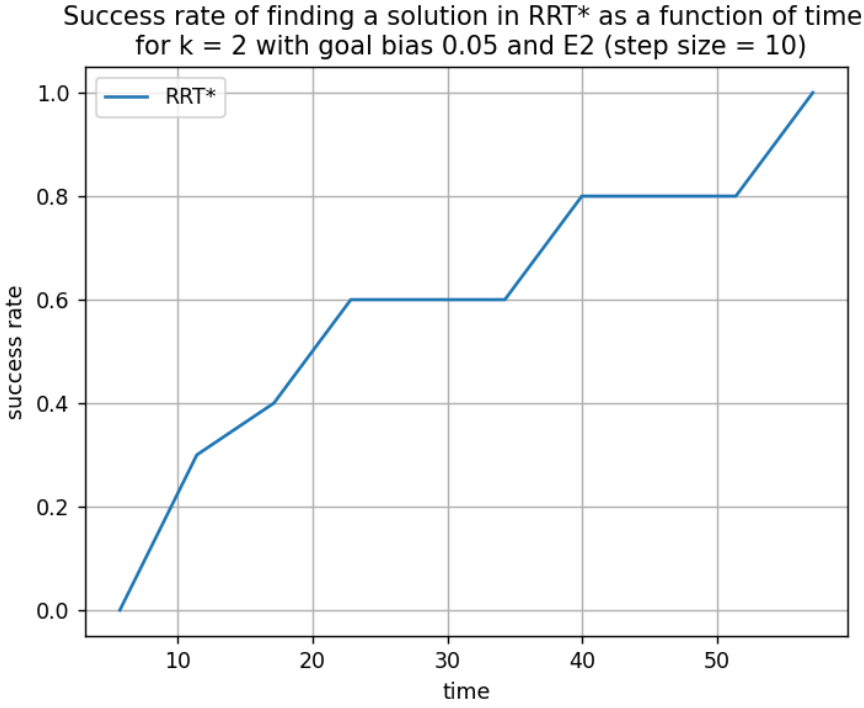

Success rate of finding a solution in RRT as a function of time
for k = 5 with goal bias 0.05 and E2 (step size = 10)
RRT
success rate
time
0.0
0.2
0.4
0.6
0.8
1.0
5
10
15
20
25
30

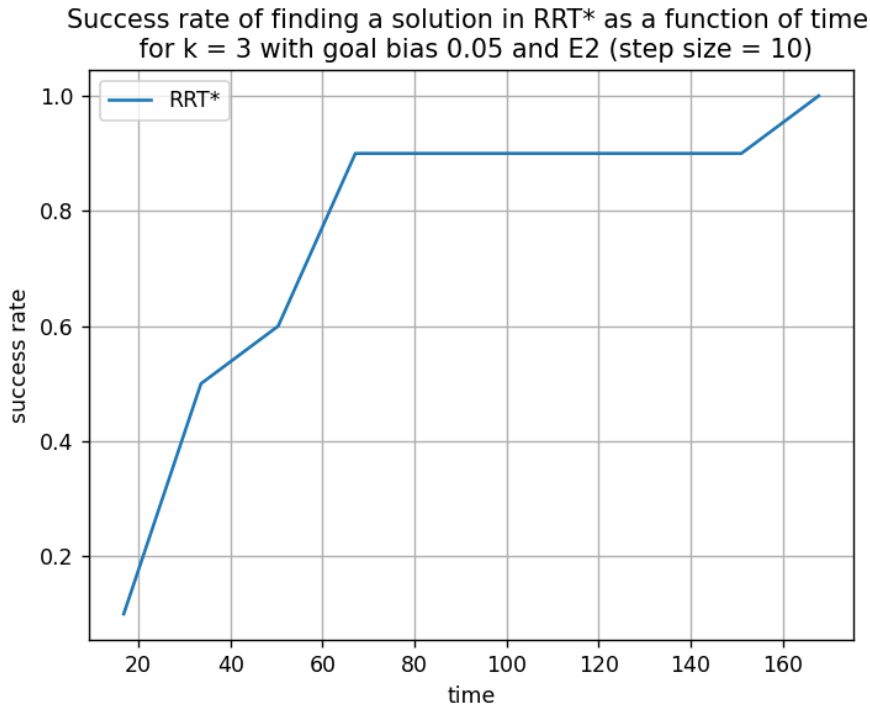


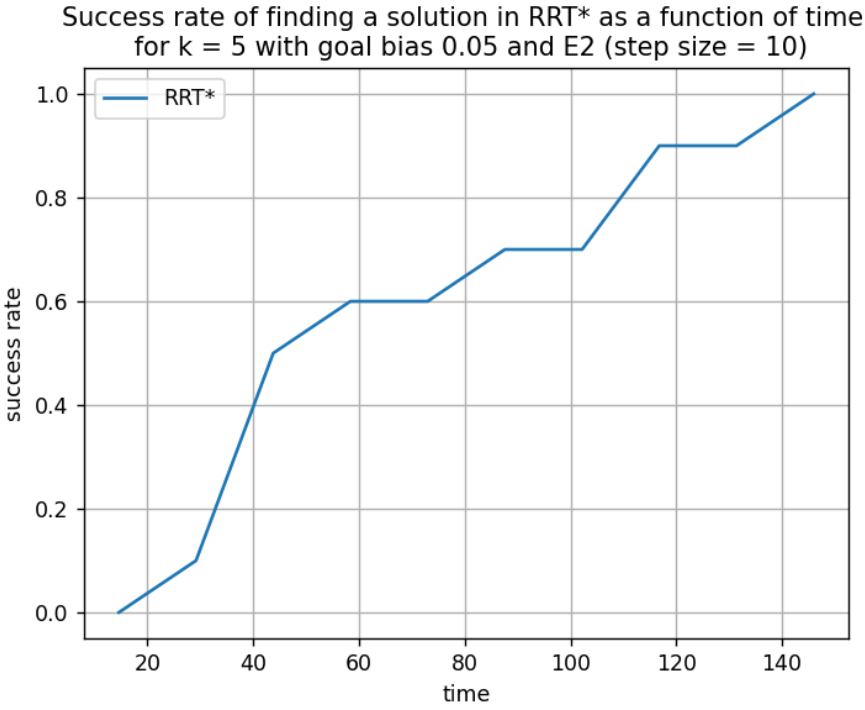


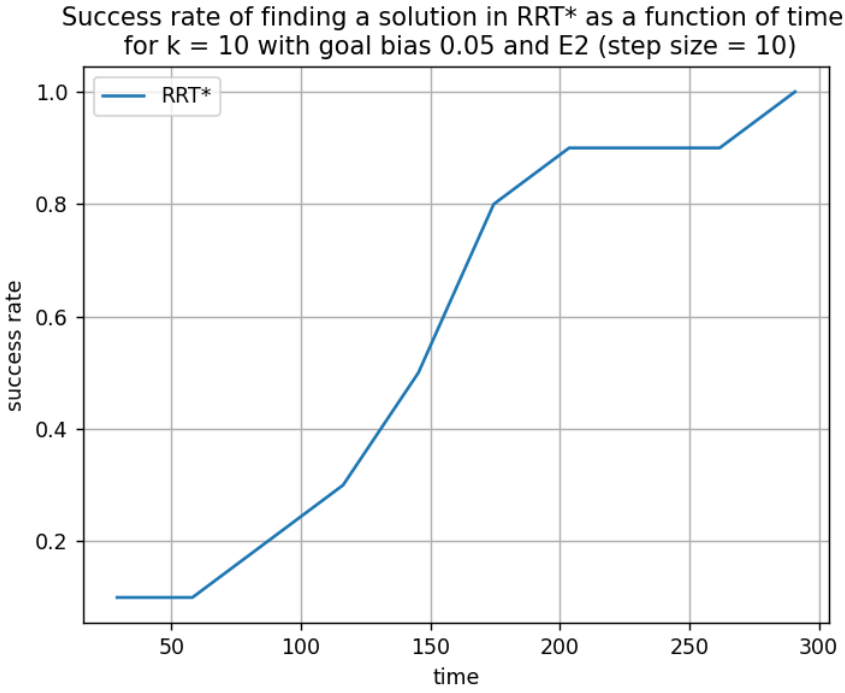
Plots of success rate of RRT*:

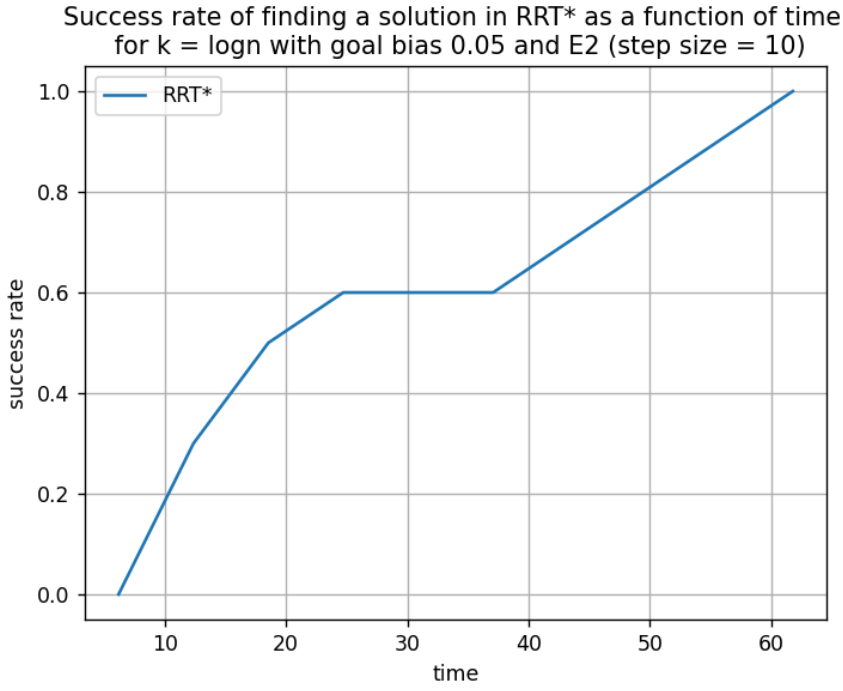

Success rate of finding a solution in RRT* as a function of time
for k = 1 with goal bias 0.05 and E2 (step size = 10)
RRT*
success rate
time
0.0
0.2
0.4
0.6
0.8
1.0
20
40
60
80
100


Success rate of finding a solution in RRT* as a function of time
for k = 2 with goal bias 0.05 and E2 (step size = 10)
RRT*
success rate
time
0.0
0.2
0.4
0.6
0.8
1.0
10
20
30
40
50

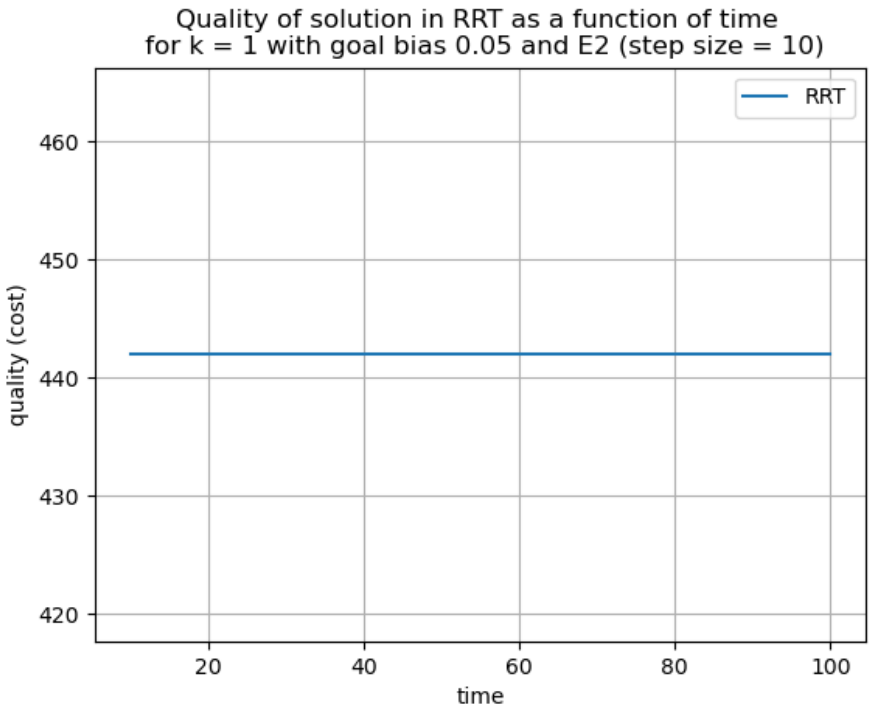

Success rate of finding a solution in RRT* as a function of time
for k = 3 with goal bias 0.05 and E2 (step size = 10)
RRT*
success rate
time
1.0
0.8
0.6
0.4
0.2
20
40
60
80
100
120
140
160

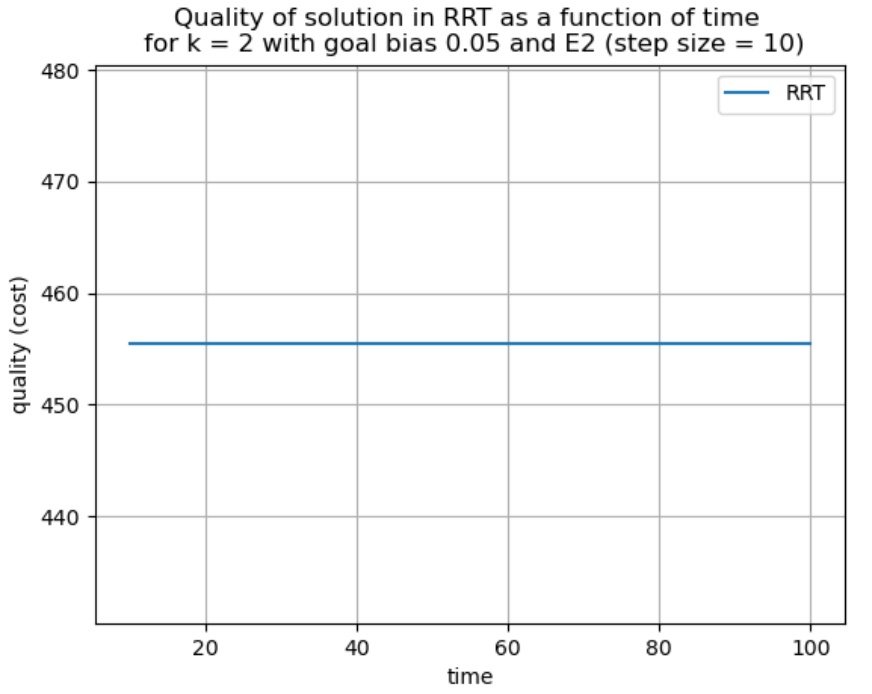

Success rate of finding a solution in RRT* as a function of time
for k = 5 with goal bias 0.05 and E2 (step size = 10)
RRT*
success rate
time
1.0
0.8
0.6
0.4
0.2
0.0
20
40
60
80
100
120
140

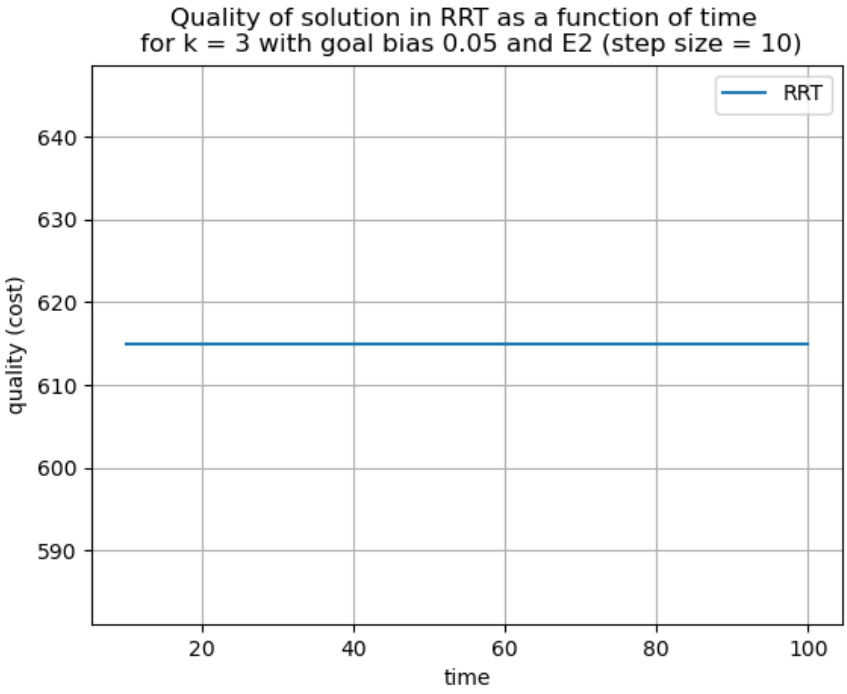


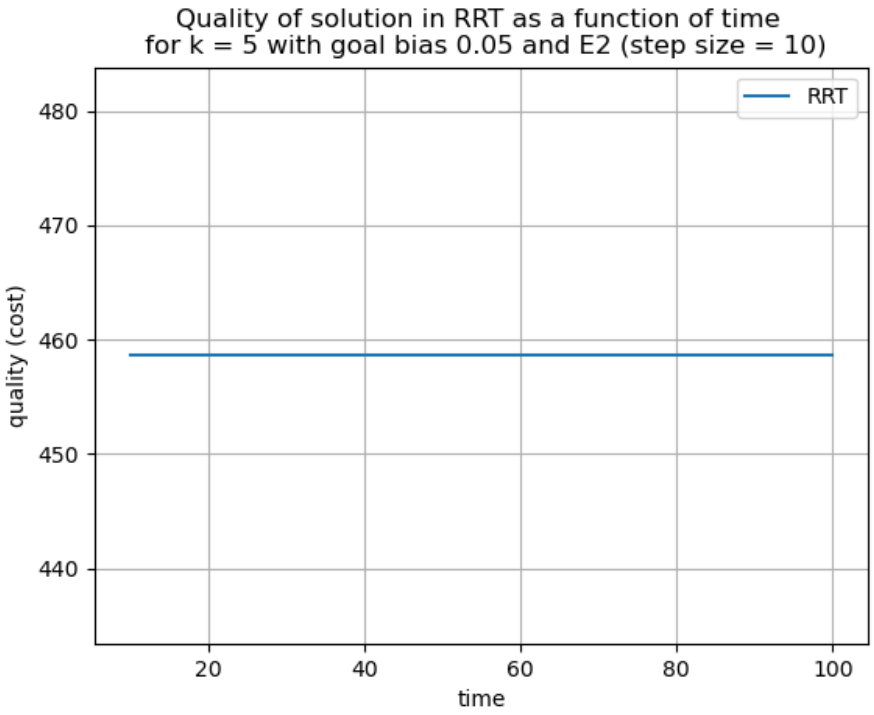


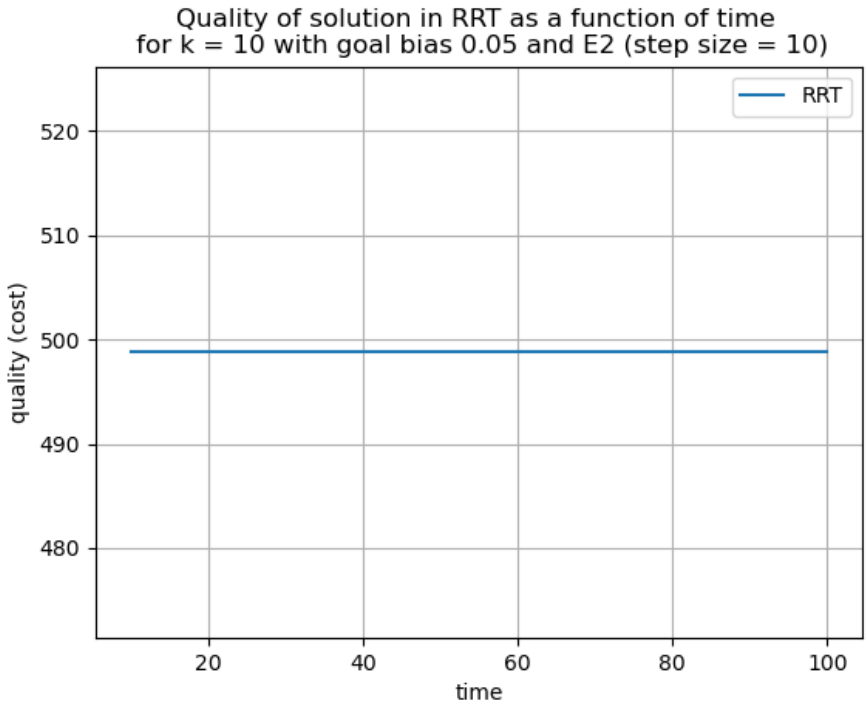
Plots of quality of solutions of RRT:

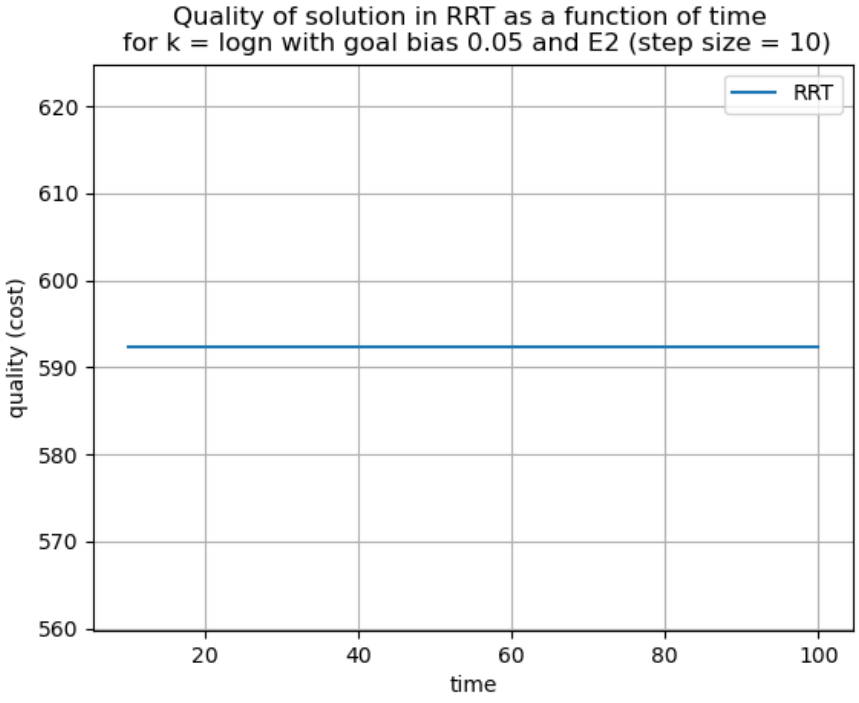

Quality of solution in RRT as a function of time
for k = 1 with goal bias 0.05 and E2 (step size = 10)
RRT
460
450
440
430
420
quality (cost)
20
40
60
80
100
time


Quality of solution in RRT as a function of time
for k = 2 with goal bias 0.05 and E2 (step size = 10)
RRT
480
470
460
450
440
quality (cost)
20
40
60
80
100
time

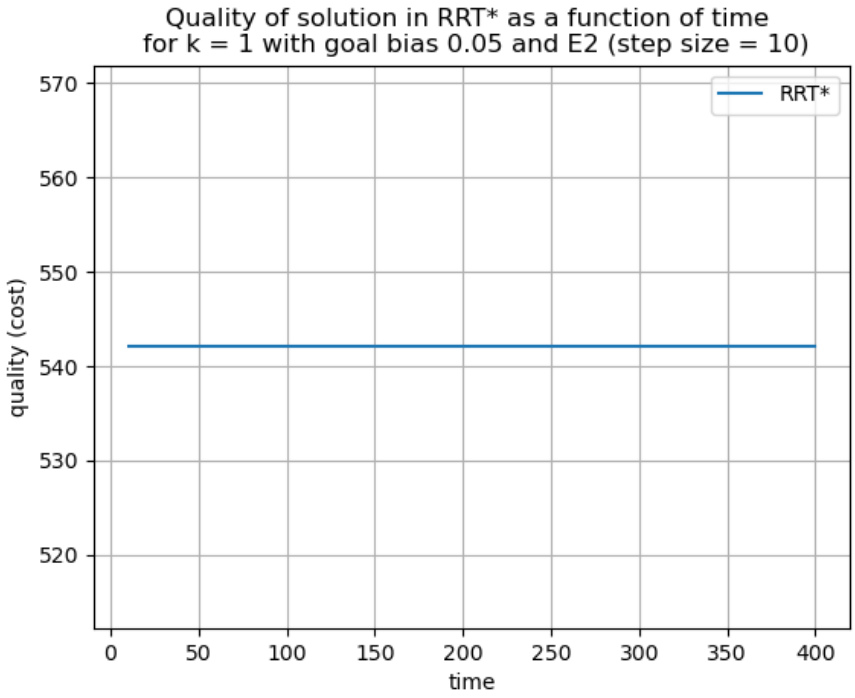

Quality of solution in RRT as a function of time
for k = 3 with goal bias 0.05 and E2 (step size = 10)
RRT
640
630
620
610
600
590
quality (cost)
20
40
60
80
100
time

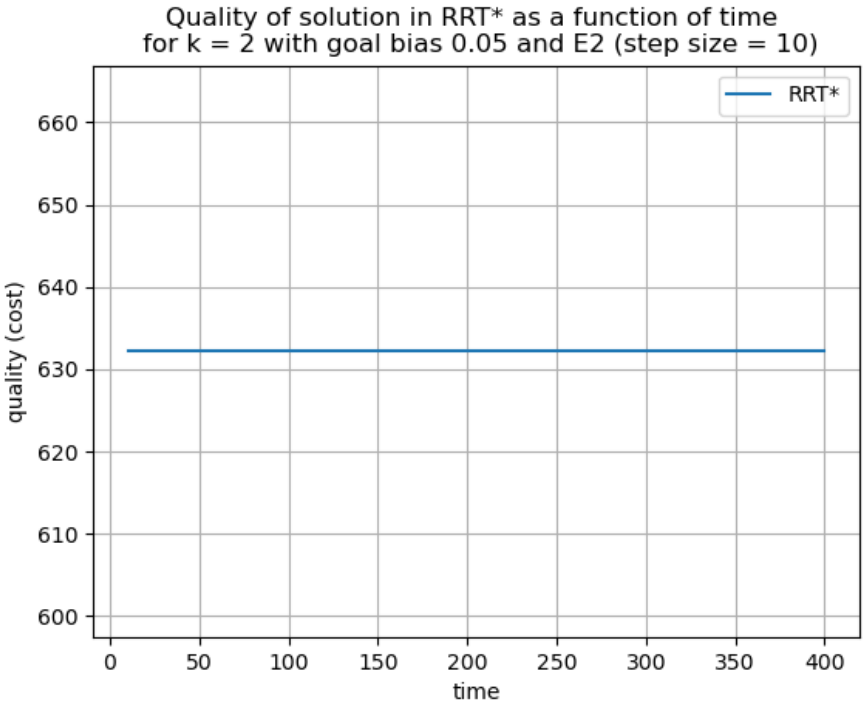

Quality of solution in RRT as a function of time
for k = 5 with goal bias 0.05 and E2 (step size = 10)
RRT
480
470
460
450
440
quality (cost)
20
40
60
80
100
time

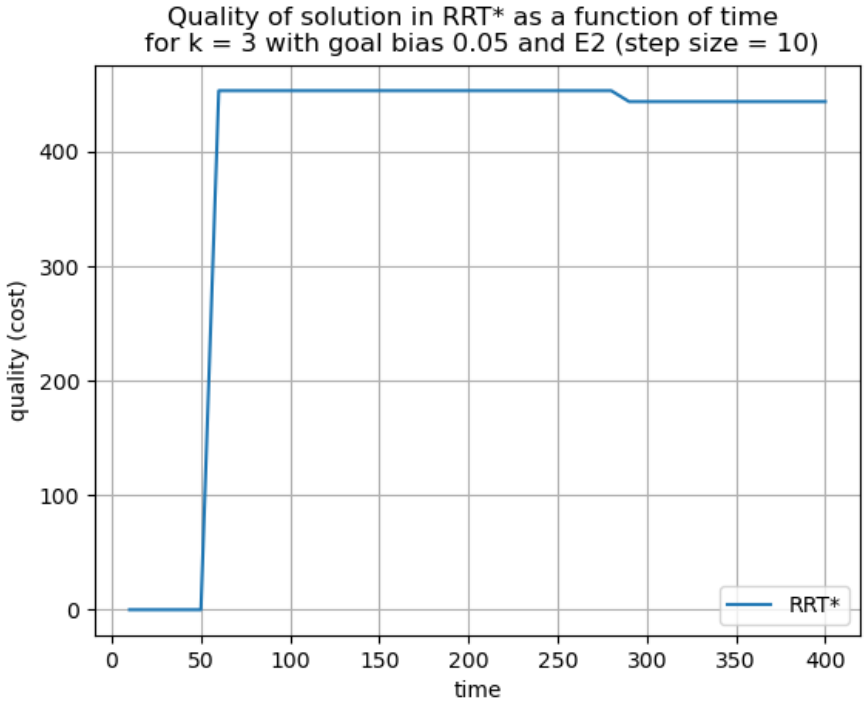


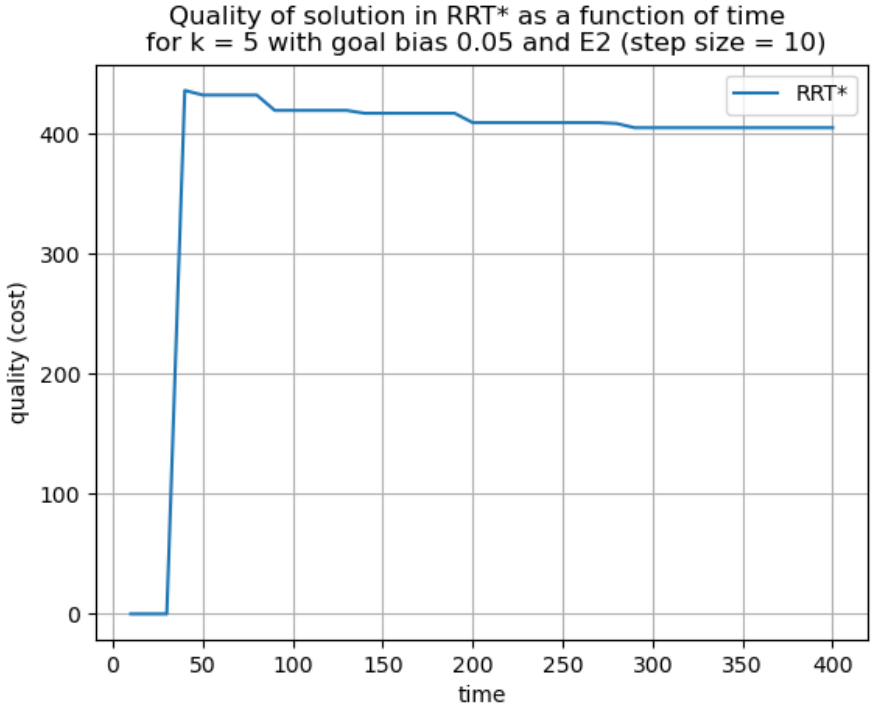


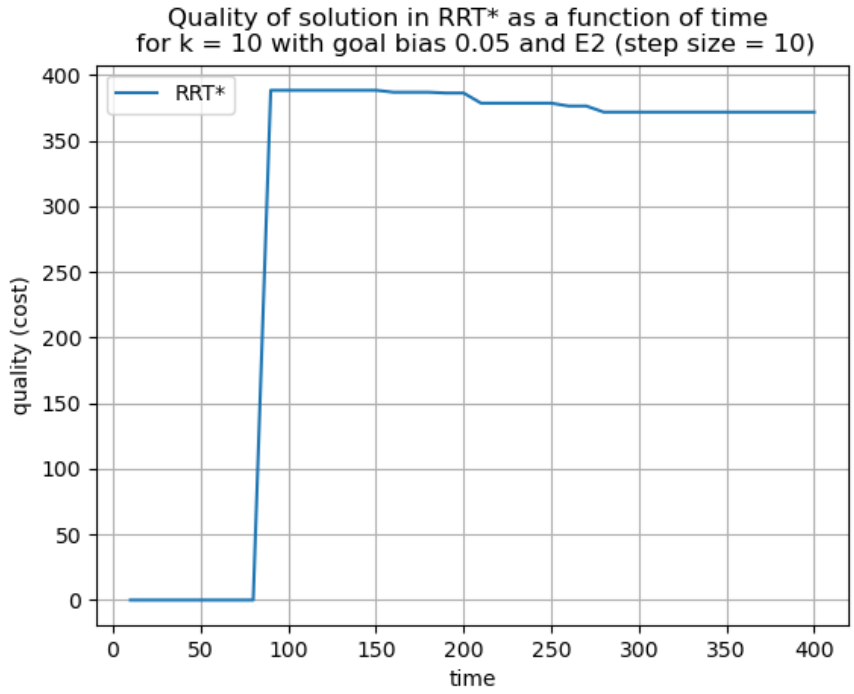
Plots of quality of solutions RRT*:

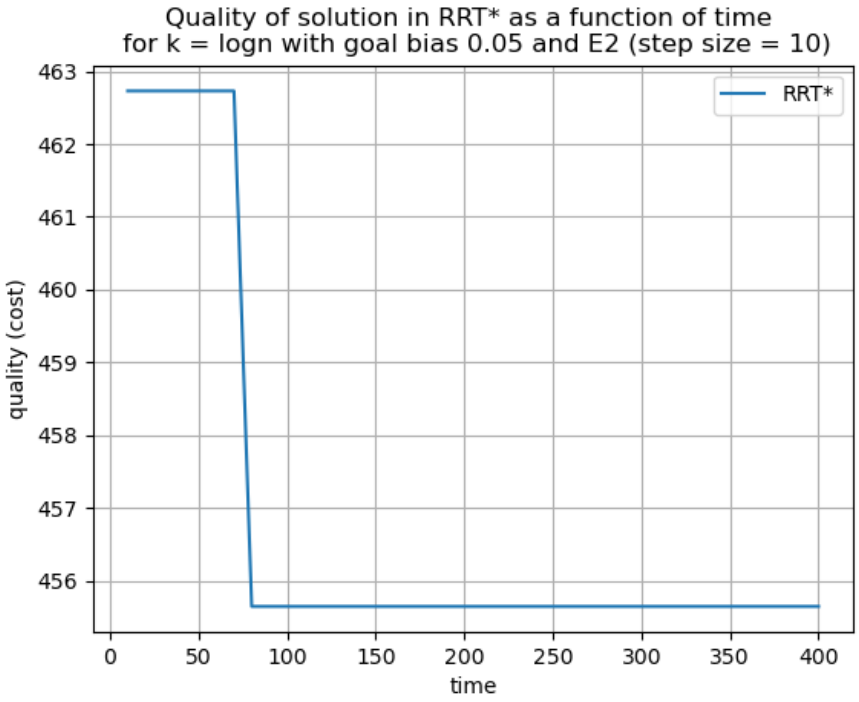

Quality of solution in RRT* as a function of time
for k = 1 with goal bias 0.05 and E2 (step size = 10)
RRT*
quality (cost)
time
570
560
550
540
530
520
0
50
100
150
200
250
300
350
400


Quality of solution in RRT* as a function of time
for k = 2 with goal bias 0.05 and E2 (step size = 10)
RRT*
quality (cost)
time
660
650
640
630
620
610
600
0
50
100
150
200
250
300
350
400


Quality of solution in RRT* as a function of time
for k = 3 with goal bias 0.05 and E2 (step size = 10)
quality (cost)
time
RRT*
0
100
200
300
400
0
50
100
150
200
250
300
350
400


Quality of solution in RRT* as a function of time
for k = 5 with goal bias 0.05 and E2 (step size = 10)
quality (cost)
time
RRT*
0
100
200
300
400
0
50
100
150
200
250
300
350
400





As can be seen from the plots, only for values of $k \geq 3$ the costs of the solutions of RRT* are being reduced a little, meaning the quality is increased. We assume that if we would have run the algorithm more time the cost would have reduce even more. On the other hand, the costs of the solutions of RRT are not being changed.

Note: Where the plots of the RRT* begin with zero it means that the goal has not been reached yet.