ME570 Assignment 4

Zili Wang
zw2445@bu.edu

June 9, 2023

Contents

1 Notes

1.1 Structure: grid

Fields:

i) xx , yy : location information

ii) grid.F : logical array where F(i,j) is true if there is a cell (no collision) at (xx(i) , yy(j)) location

1.2 Structure: graphVector

Fields:

i) neighbors (dim. [NNeighborsx1]): vertex indices abjacent to the current one

ii) neighborsCost (dim. [NNeighborsx1]): cost to move from current vertex to the adjacent vertices in neighbors

iii) g (dim. [1x1]): cost from the starting location along the path through the backpointer

iv) backpointer (dim. [1x1]): index of the previous vertex in the current path from the starting location

v) x (dim. [2x1]): the physical (x,y) coordinate of the vertex

A graph is defined by x , neighbors , and neighborsCost . g and backpointer are used for the graph search algorithm.

1.3 Data Structure: priority queue

i) priority_prepare.m : create an empty structure array for the queue

ii) priority_insert.m : add an element to the queue

iii) priority_minExtract.m : extract the element with minimum cost from the queue

iv) priority_isMember.m : check whether an element with a given key is in the queue or not

2 Question 1.3 Optional

Figure 1 shows the comparison of provided graphVector_solved and the corresponding results from the coded program with an input as graphVector. Figure 2 shows the comparison of provided graphVectorMedium_solved and the corresponding results from the coded program with an input as graphVectorMedium.
The provided data are in graph_testData.mat, which is used in graph_testData_plot.m. And the comparisons are done in graph_search_test.m. The start position is in red cross while the goal position is in red diamond. The backpointers are in blue arrow, and the path provided from $A*$ algorithm is in magenta dashed line.

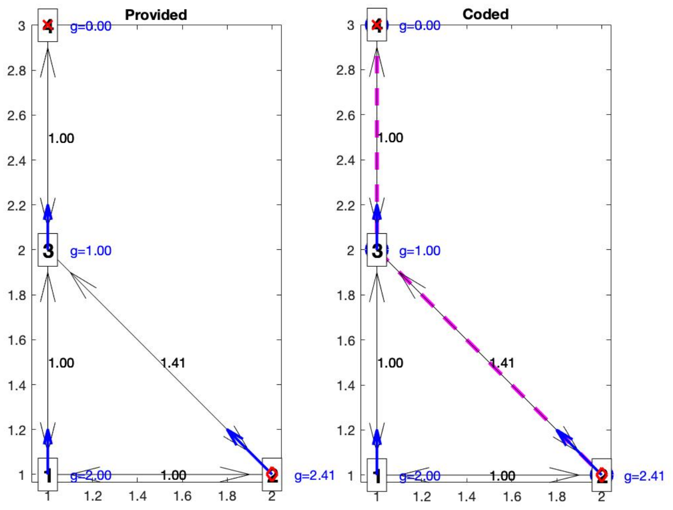


Figure 1: Comparison with Provided graphVector

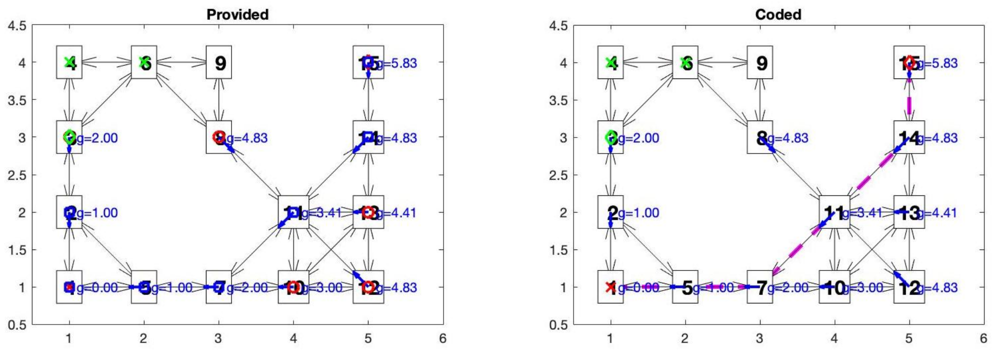


Figure 2: Comparison with Provided graphVectorMedium

Figure 3 shows how the algorithm works in a customized world. By introducing the priority queue, it also shows how the $A*$ planner guarantee the optimal path without exploring all the nodes.

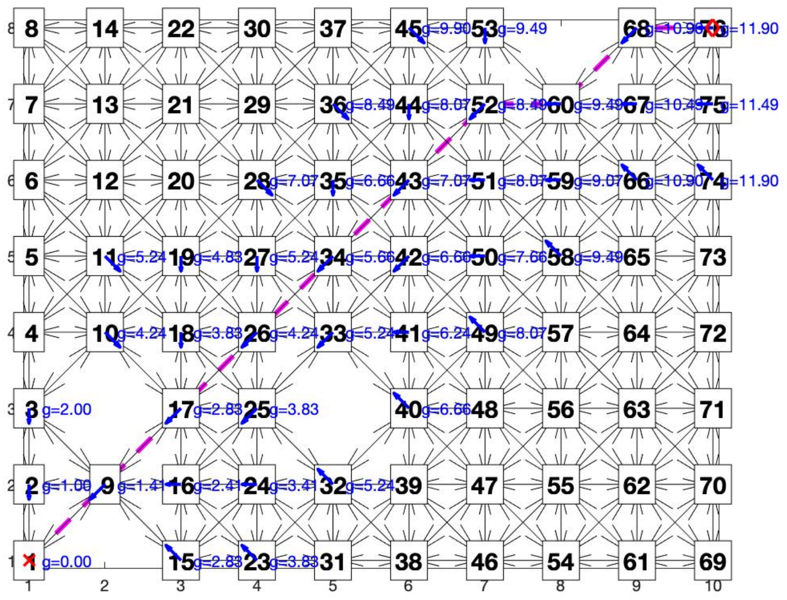


Figure 3: $A*$ from Custom World

3 Question 2.1

Figure 4, 5, 6, and 7 show the result of discretization of the sphere world, from sphereworld_freeSpace_graph_test.m .

- With a low number of cells (NCells=8, Figure 4), the obstacles are easily fused together.
- With a right number of cells (NCells=20,27, Figure 5,6), the topology of world can be identified while it is not too fine.
- With a high number of cells (NCells=80, Figure 7), the graph is too fine to be necessary, and will bring lots of computation.

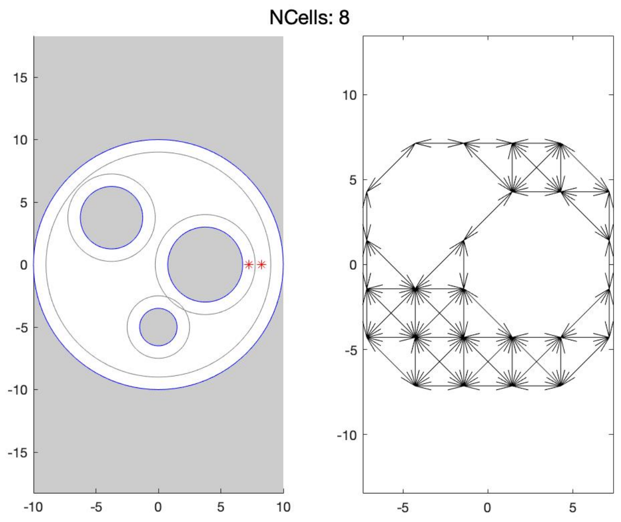


Figure 4: Low NCells

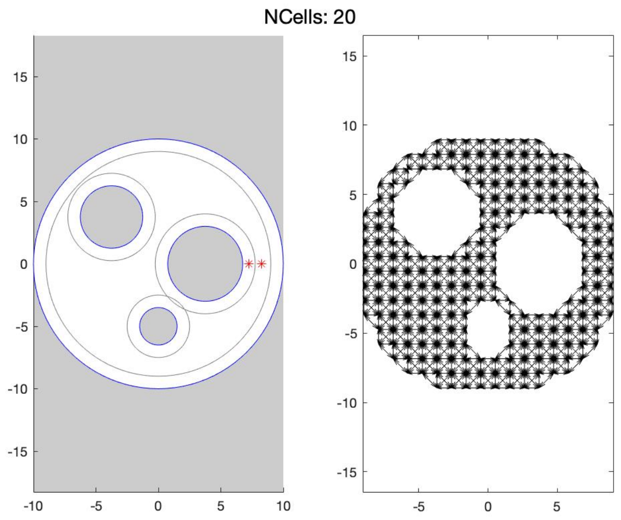


Figure 5: Right NCells 1

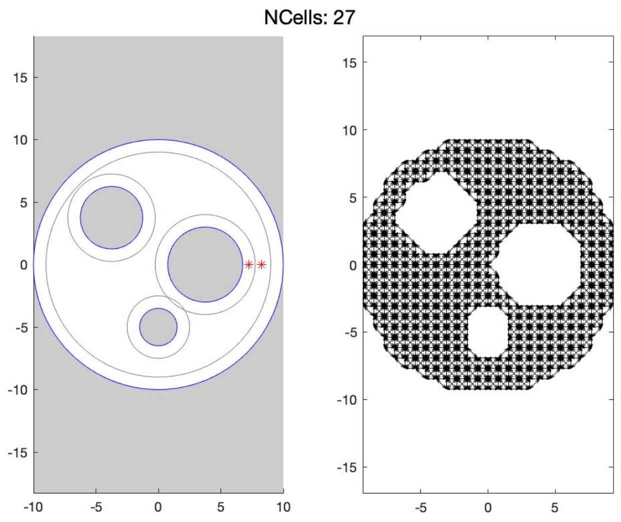


Figure 6: Right NCells 2

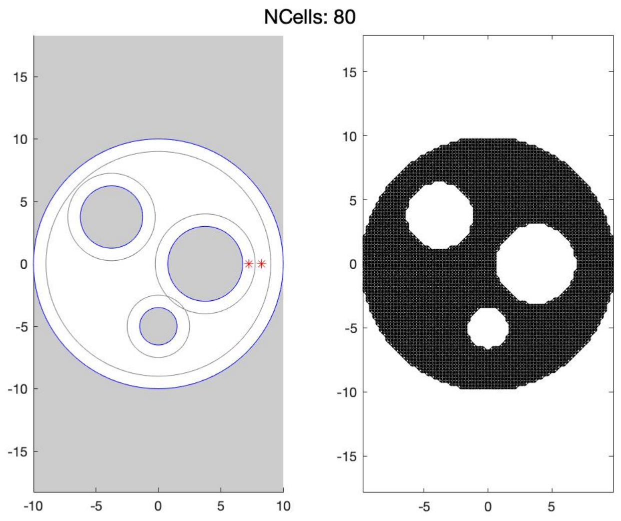


Figure 7: High NCells

4 Question 2.2

Figure 8, 9, 10, and 11 show the result of $A*$ motion planning under a discretization of the sphere world, from sphereworld_search.m . The path from different starting points are plotted in green, cyan, red, magenta, and blue color respectively. And the graph vertices the agents traverse are plotted in black dots.

- With a low number of cells (NCells=8, Figure 8), the obstacles are easily fused together. Therefore, the agent can either collide with the obstacle or miss the shortest path.

- With a right number of cells (NCells=20,27, Figure 9,10), the topology of world can be identified while it is not too fine. The agents can successfully avoid the obstacles while reaching the goal location. Here, Figure 9 and 10 shows different optimal trajectory. While Figure 10 is consistent with the result of finer discretization Figure 11.
- With a high number of cells (NCells=80, Figure 11), the graph is too fine to be necessary, and will bring lots of computation. The trajectory it gives is similar to the one with coarser discretization. While it mimics the continuous world better, it requires higher computation.

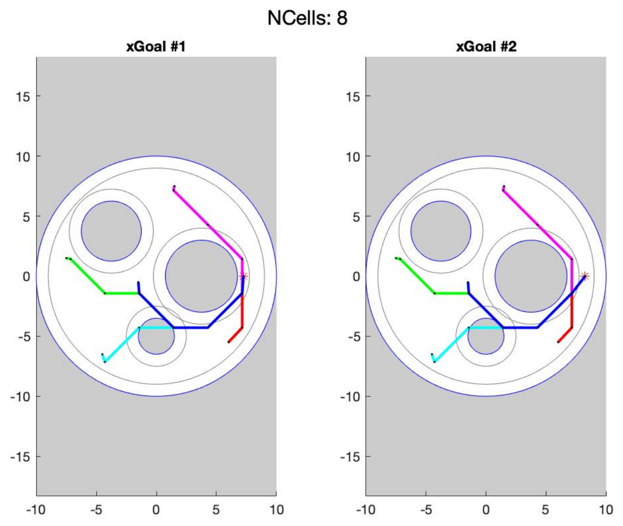


Figure 8: $A*$ with Low NCells

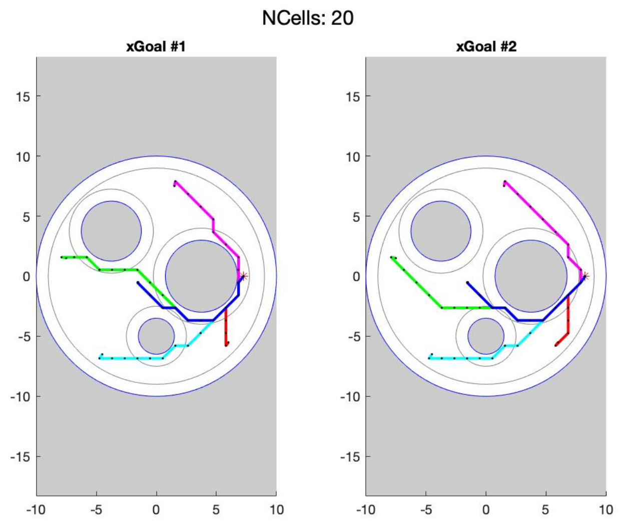


Figure 9: $A*$ with Right NCells 1

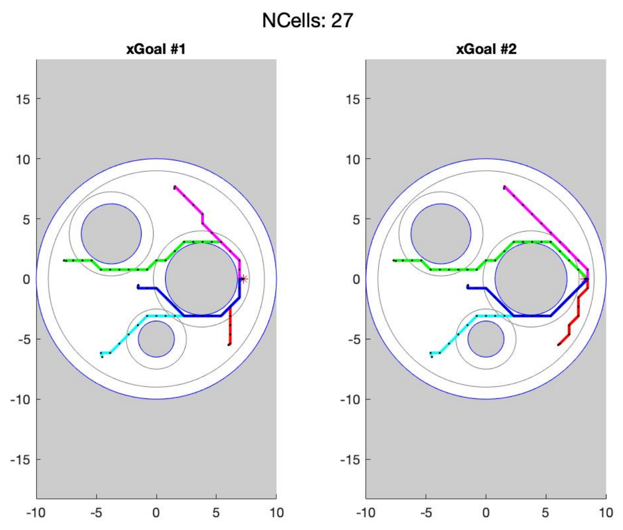


Figure 10: $A*$ with Right NCells 2

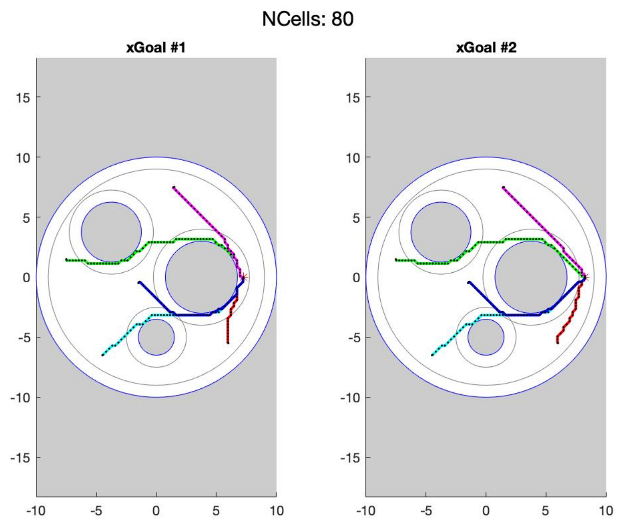


Figure 11: $A*$ with High NCells

5 Question 2.3

$A*$ planner behaves differently with different selection of NCells, which gives a different discretization of the sphere world.

- With a low number of cells, the obstacles are easily fused together. Therefore, the agent can either collide with the obstacles or miss the shortest path.
- With a right number of cells, the topology of world can be identified while it is not too fine. The agents can successfully avoid the obstacles while reaching the goal

location. A wide range of choices can give a good topology, however, a fine tuning is needed to find a good NCells that gives a consistent path with the one with finer discretization.

- With a high number of cells, the graph is too fine to be necessary, and will bring lots of computation. The trajectory it gives is similar to the one with coarser discretization. While it mimics the continuous world better, it has more requires higher computation.

6 Question 2.4

$A*$ and potential planner are different in the following aspects:

- $A*$ is a grid-based search planner, it requires environment modelling. With a regular grid, it can provide optimal trajectory. However, the number of points on the graph grows exponentially on the configuration space dimensions. Also, when there is a change in the configuration space (e.g. displacement of an obstacle), the search algorithm has to re-plan. A good selection of grid size setting is also important.

- Potential planner does not require a environment modelling to do real-time plannning. It is based on attractive potential function to reach the goal location and repulsive potential function to avoid the obstacles. And it is based on gradient descent on the potential map to move. It requires less computation. However, it is possible to be trapped into the local minima or find a non-optimal path. Again, a good selection of weight on repulsion and step size is important.

7 Question 3.1

Figure 12 gives the plot of results twolink_freeSpaceGraph.m, which has been saved to twolink_freeSpaceGraph_graph.mat.

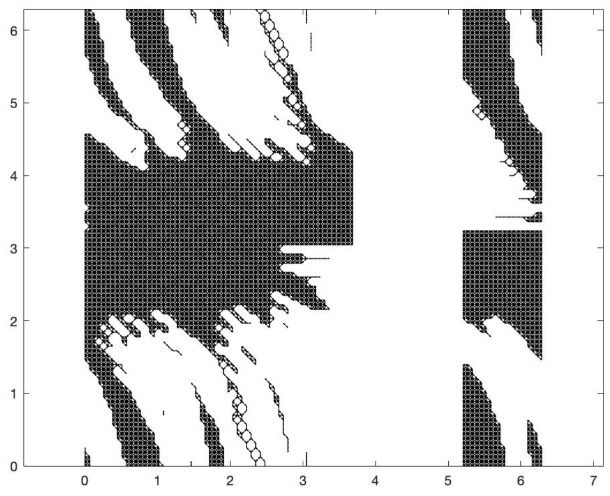
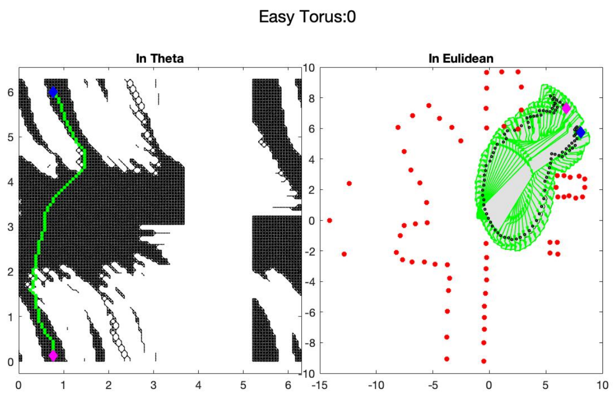


Figure 12: Twolink Graph

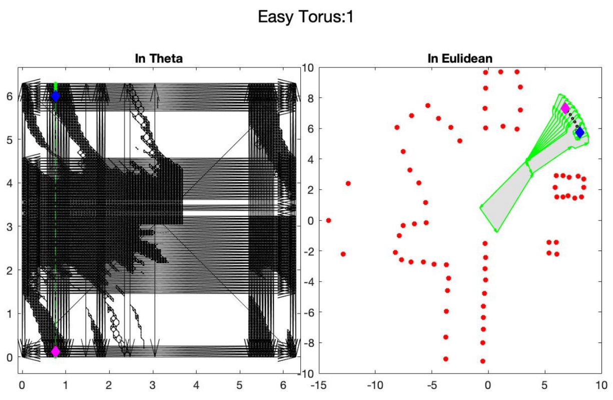
8 Question 3.2

Figure 13, 14, and 15 are the results from twolink_search_test.m. While for each figure, the sub-figure (a) is from setting flagTorus to false (run twolink_freeSpaceGraph('torus',flagTorus) and graph_search_startGoal()), the sub-figure (b) is from setting flagTorus to true (run twolink_freeSpaceGraph('torus',flagTorus) and twolink_search_startGoal()).

In each sub-figure, the left plot is the $A*$ result in two-link configuration space, and the right plot is in the Euclidean space. The magenta diamond is the start location, and blue diamond is the goal location. In the left plot, the vertices the agent traverse are in green dots, while green dashed dotted line is used to connect two vertices. In the right plot, the obstacles in twolink_testData.mat are in red thick cross, the two-link manipulator has the green edges, and the end effectors are in black hollow circles.

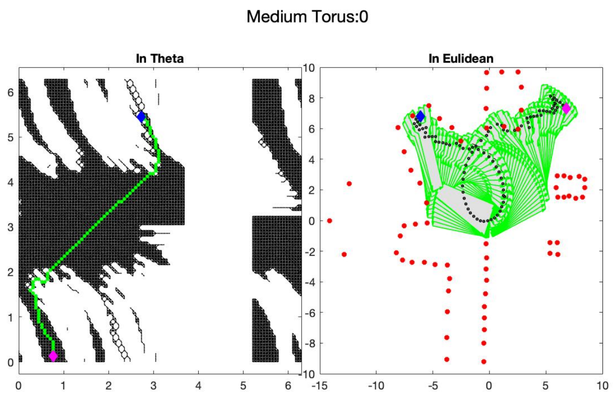


(a) without Torus

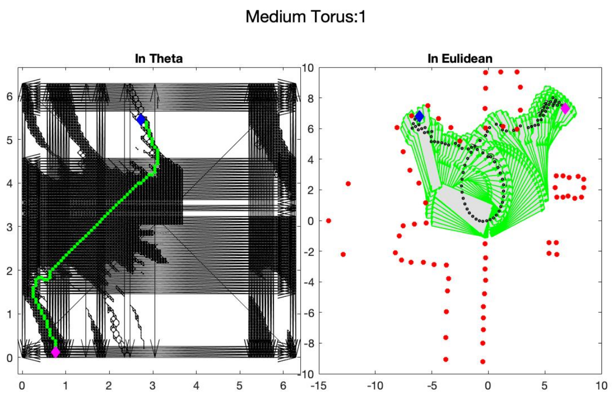


(b) with Torus

Figure 13: Easy Configuration

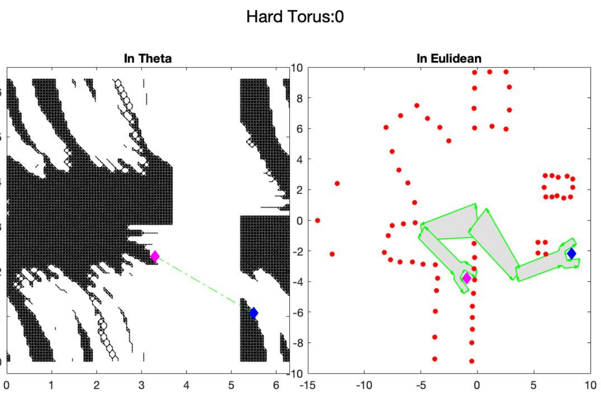


(a) without Torus

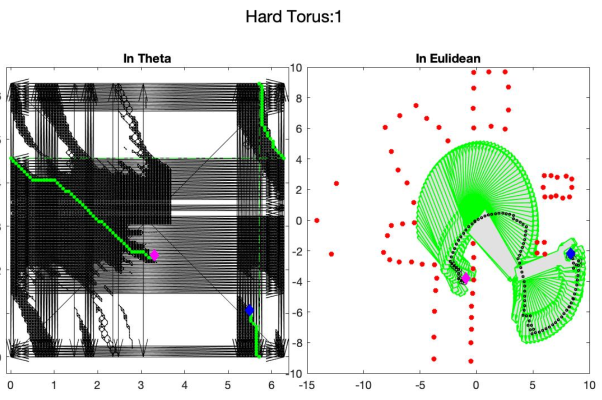


(b) with Torus

Figure 14: Medium Configuration



(a) without Torus



(b) with Torus

Figure 15: Hard Configuration

9 Question 3.3

Figure 16 is the torus charts from Homework 2. Three regions in (a) (b) (c) can cover the entire flat representation of torus as in (d). For the square region, the opposite sides are glued together. Therefore, there is no discontinuity at 2π as in (a) (open).

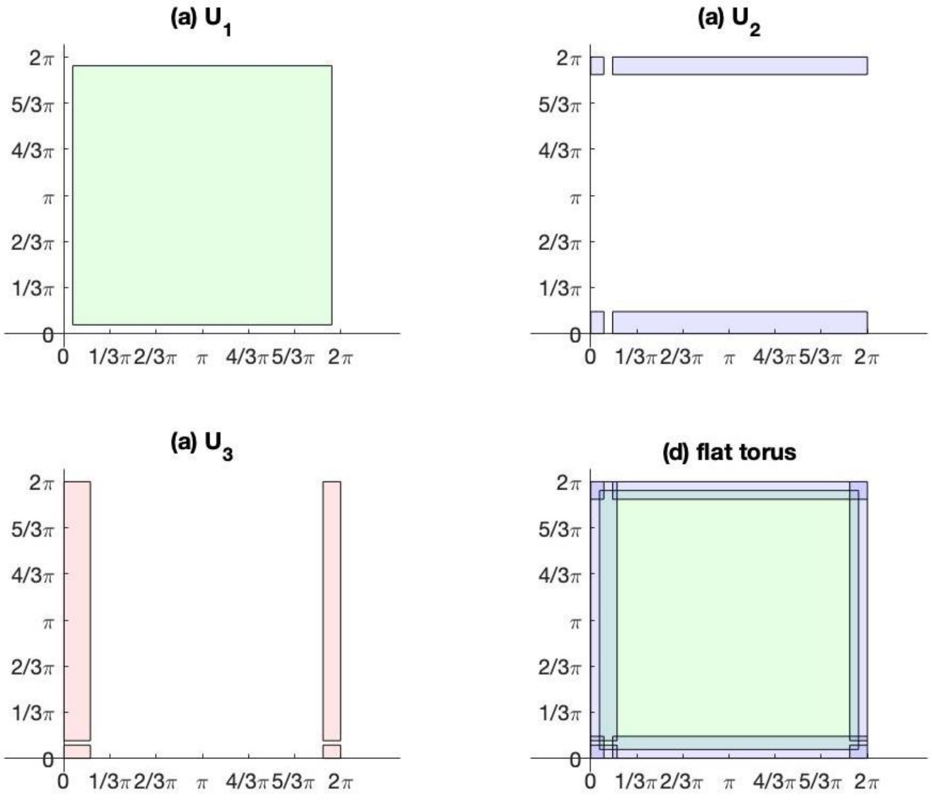


Figure 16: Torus Charts

In the *Easy* case as in Figure 13. Without completing the charts, Figure 13a has a discontinuous configuration space as U_1 in Figure 16a. So there is an unwinding phenomenon. The agent can only span mostly of the range in one dimension to reach the goal location. Completing the charts gives a flat torus representation ($\mathbb{R}^2 \rightarrow \mathbb{S}^1 \times \mathbb{S}^1$), Figure 13b has a continuous configuration space as Figure 16d. Therefore, the agent can go directly through the glued region, and only the second link is needed to move. Noting that on the torus, the rightmost vertex is connected with its leftmost neighbors and the bottom vertex is connected with its neighbors on the top.

10 Question 3.4

There are three configurations:

- Easy Configuration ($thetaStart = [0.76; 0.12], thetaEnd = [0.76; 6.00]$, Figure 13): the configuration space with 'torus' is continuous, so the agent moves straightly to the goal from the glued region. The configuration space without 'torus' id discontinuous, so the agent spans in a large range to reach the goal location.
- Medium Configuration ($thetaStart = [0.76; 0.12], thetaEnd = [2.72; 5.45]$, Figure 14): with and without 'torus' give the same result.
- Hard Configuration ($thetaStart = [3.30; 2.34], thetaEnd = [5.49; 1.07]$, Figure 15): without 'torus', the planner fails to find a path to the goal location. With 'torus', the planner can successfully find a path.

As shown in Figure 13, 14, and 15 , the planner can traverse just next to the obstacles. Since it can be very close to the obstacles, in practice, it is easier to collide. In order to avoid collisions, we can

i) Making the obstacles larger than its real size before discretization.

ii) Using a repulsive potential function to generate a repulsion to the obstacles.

11 Question 3.2 Optional

Figure 17 shows the graph plots for different start and goal locations using different methods:

- BFS: it is based on $f(n) = g(n)$ to update, where $g(n)$ is the cost of the path from node n to the start vertex
- Greedy: it is based on $f(n) = h(n)$ to update, where $h(n)$ is the heuristic (Euclidean distance) from node n to the goal vertex

- $A*$: it is based on $f(n) = g(n) + h(n)$ to update

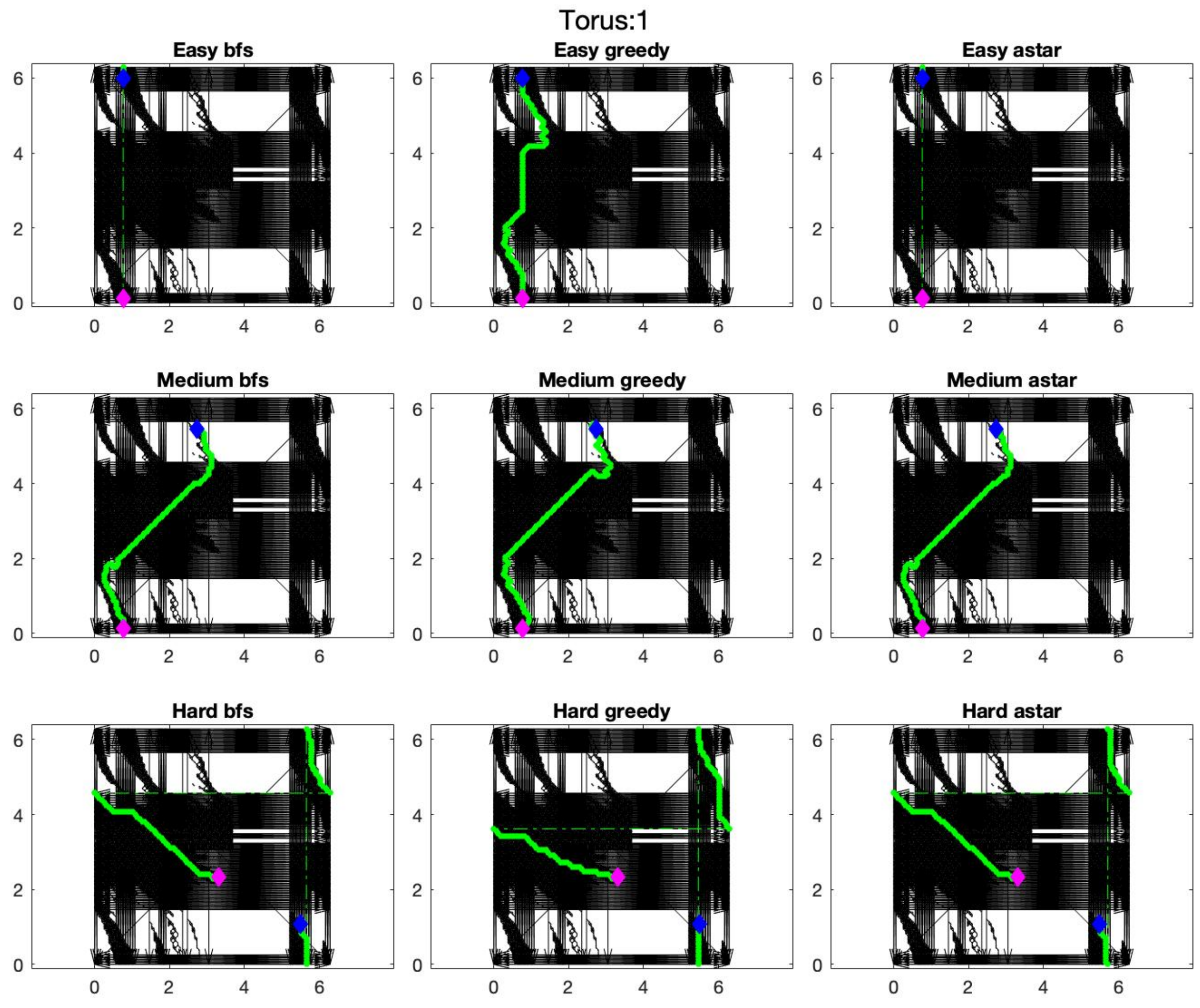


Figure 17: Graph Plot for Different Methods

Below is the diary saved in twolink_search_method.txt and run from twolink_search_method_test.m. The greedy method always requires a shortest computation time, while BFS is easier to consume higher computation time. It is known that greedy method is not necessarily giving the optimal path although it is the fastest. While BFS and $A*$ is guaranteed to give the optimal path, $A*$ inherits BFS's property of thoroughness and greedy's property of fastness.

The animations in https://cs.stanford.edu/people/abisee/tutorial/astar.html demonstrate well the above three search algorithms.

```
flagTorus: 1
Case: Easy
Method: bfs Computation time:1.1069
Method: greedy Computation time:0.38493
Method: astar Computation time:0.52718
Case: Medium
Method: bfs Computation time:1.1294
Method: greedy Computation time:0.31688
Method: astar Computation time:0.93308
Case: Hard
Method: bfs Computation time:1.238
Method: greedy Computation time:1.1574
Method: astar Computation time:1.1989
```

12 Question 3.3 Optional

Instead of analysing neighbors and costs of every vertex beforehand, lazy evaluation initializes an empty neighbor field in the graphVector and update the neighbors only when expanding the lists.

Below is the diary saved in twolink_graph_search.txt and run from twolink_graph_search_test.m.

```
flagTorus: 1
Case: Easy
Method: bfs Computation time:1.0761
Method: greedy Computation time:0.16282
Method: astar Computation time:0.34557
Case: Medium
Method: bfs Computation time:1.0862
Method: greedy Computation time:0.074495
Method: astar Computation time:0.84874
Case: Hard
Method: bfs Computation time:1.2757
Method: greedy Computation time:1.1644
Method: astar Computation time:1.2686
```

It is assumed that using lazy evaluations could consume less computation time, however, here only the easy and medium configuration cases give a less computation time.

13 Question 4

- Code and Debug: 6 hours
- Report: 2 hours
- Total: 8 hours
- Simple Part: Some implementations are based on previous homework
- Difficult Part: it will be somehow difficult if some parts of the previous homework is forgotten. Also, the last optional question needs some analytical thinking.