QUIZ 1

Hoang-Dung Bui,
George Mason University
Fairfax, USA
hbui20@gmu.edu

I. Building Roadmaps

A. Different Roadmaps

Trapezoidal Decomposition and Visibility Graph are two roadmap's techniques used for low dimension environments. Their complexities depend on the number of obstacle's vertices in the environment, specifically, for Trapezoidal Decomposition: $O(nlogn)$ and for Visibility graph $O(n^2logn)$. Therefore, for environments with less obstacle's vertices, we can use Visibility Graph because it is complete and provides the shortest paths. However, for a complex environment with a lot of obstacles vertices, we should use Trapezoidal Decomposition, because it will bring much less computation's time.

B. Shortest Path Roadmap

In Fig 1, there is no edges connecting the vertices (B,C), (A,D), or (B,D), because the vertices A and B lies in a non-convex area. If we start from C and D, then go to A, and D, it will terminate the search. This is because there is no connection from A and B to any other vertices. We also explain why there is not any path going through A or B, then connect to C and D. If the path start from E (or any points close to the E's area), there is obviously no edges connecting to A and B because E cannot see A, and B. If we start from point F (or any points inside the F's area), we will have the direct edges connecting to vertices C and D, and we do not connecting back to A and B. Therefore, there is no path will going through A and B, and we can remove any edges connecting to A and B, and it will not affect to the paths generated by the algorithm. In general, we should remove all the edges if its vertices lie in non-convex shape area.

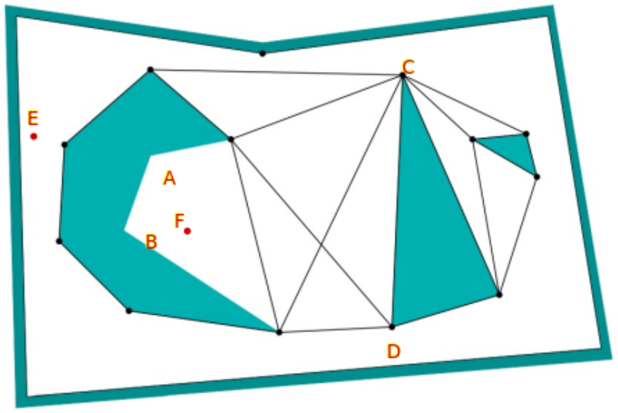

Fig. 1: Shortest Path Roadmap

II. State Space Search

A. Random-first Search

If there is loops in graphs, DFS may never terminate. Instead of DFS and BFS, we pick the next path randomly to expand.

- The likelihood the agent will reach the goal is $1/n$ - n is number of state in the path's list. As n increases, the probability of taking the right path is lower.
- In the best scenario, the random approach can get the shortest path by picking up the right states 100%. This ratio will rarely happen.
- In the worst case, the algorithm will stuck in a loop and never terminate. Because it can pick the visited states again.

B. Manhattan Distance Heuristic I

In grid-based search in which the robot can traverse diagonals, and so has 8 moves from each grid cell. If we use Manhattan Distance for the heuristic function, it is **not admissible** anymore. The requirement for admissible heuristic is it must be underestimate the distance to the goal. Assuming we need to move from node $A(1,1)$ to node $B(2,3)$. If using Manhattan distance, the heuristic distance is 3 unit $>$ distance to travel = $1+\sqrt{2}$, it violates the admissible requirements.

C. Manhattan Distance Heuristic II

In grid-based search in which the robot can not traverse diagonals, and so has 4 moves from each grid cell. Manhattan Distance is a heuristic function, because it meets the **admissible** requirement. For example, if the node A and B are in the diagonal position. With 4 moves, the distance to travel from A to B is 2. If using Manhattan distance, the cost to travel is also 2. Thus, the heuristic distance = the actual distance = 2, and it meets the requirement of admissible heuristic: $h \leq \textit{actual distance}$

D. Admissible and Inadmissible Heuristics

Consider our grid-based search example from class (in which we are traveling on a grid including diagonals).

- If we used an inadmissible heuristic in A* search, it did not change the completeness of the search. Because the algorithm will keep finding the path by trying all the nodes in the list until it will find one.
- If we used an inadmissible heuristic in A* search, it could change the optimality of the search. Because if the heuristic is inadmissible, the search will overestimate the

distance to the goal, and perform as a greedy approach. As having a solution, because all other nodes are overestimated, the search could find the current solution optimal, and stop discovering alternative solutions.

- An inadmissible heuristic has advantage over an admissible one: With an inadmissible heuristic, the algorithm works very fast. For example for sliding puzzle with size 4x2 in project 1, with heuristic Manhattan distance, the number of iteration is 255, meanwhile with heuristic Manhattan distance square, the number of iteration is only 120 - a half of Manhattan distance one. However, it usually provides the sub-optimal one (path lengths is 23 vs 21 in). As the admissible heuristic is 0, the algorithm will work too long and take computation time as Depth Breath Search (4594 iteration number).