ECE 5984: Advanced Robot Motion Planning (Spring 2017)

Homework 1

Due: Monday February 6th, 9PM

January 25, 2017

Instructions. This homework constitutes 5% of the course grade. You must work on it individually. It is okay to discuss the problems with other students in class. However, your submission must be your original work. This implies:

- If you discuss the problems with other students, you should write down their names in the pdf report.
- If you refer to any other source (textbook, websites, etc.), please cite them in the report at the relevant places.
- The answers in the pdf report must be written entirely by you from scratch. No verbatim copy-paste allowed without citations.
- Any software you submit must be written entirely by you with no copy-pasting of significant portions of code from other sources.

Please follow the submission instructions posted on canvas exactly. You must submit your assignment online on canvas by the due date. Your submission must include one pdf file with the answers to all the problems and one or more files containing your code for Problem 4. It is okay to scan your answers and create the pdf submission.

Problem 1 — 10+10 points

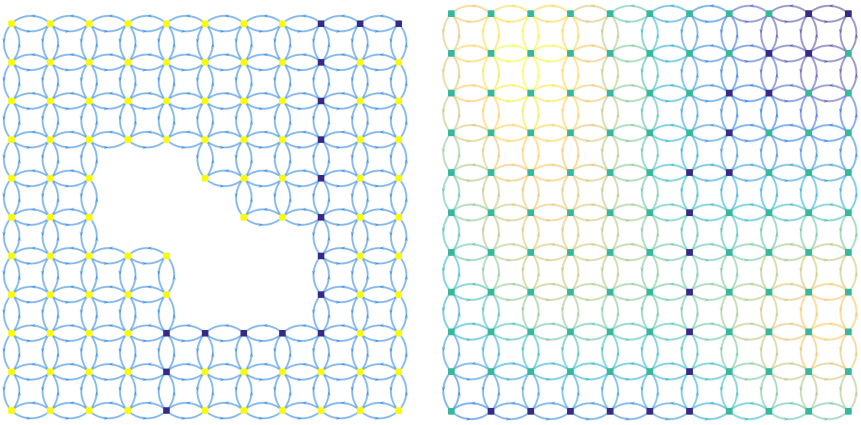
(a) Draw one environment where the Bug 1 strategy travels more distance to reach the destination than the Bug 2 strategy. (b) Draw one environment where the Bug 2 strategy travels more distance to reach the destination than the Bug 1 strategy.

Problem 2 — 20 points

Gas Station Problem. Suppose that you have a robot that is tasked with traveling from start to goal. In this problem, we assume that we already know the path that will be followed by the robot. This path is a straight road from the start, x_0, to the goal x_n. Our robot has limited fuel and will need to stop along this path (possibly multiple times) to refuel. The robot spends one unit of fuel per unit distance traveled. There are n gas stations along the path: $x_0 \to x_1 \to x_2 \to \ldots \to x_n$. The i^{th} gas station, x_i, is at a distance d_i from the start. It costs p_i dollars per unit of fuel to refuel at the i^{th} station. Our objective is to minimize the total cost of refueling while ensuring the robot reaches the goal position.

(a) Can this problem be formulated and solved by dynamic programming?

(b) If your answer is yes, then write down the Bellman recurrence equation (similar to the one we saw in class) for this problem. Define and explain your notation clearly. You only need to specify the recurrence relation and not actually solve the DP. If your answer to part (a) is no, then give a counter example to show that the problem does not satisfy the optimal sub-structure property.



(a) `input1.txt` (b) `input2.txt`

Figure 1: The two sample instances. The input is a directed graph given in the `input1.txt` and `input2.txt` files. The color of the edges indicates the cost (yellower edges have higher costs than bluer edges). The minimum cost path from start to goal is shown in blue.

You may assume that initially the robot has B units of fuel. You can also assume that $B \geq d_{i+1} - d_i$ and that all d_i values are integers.

Hint. In this problem, we know that the robot is going to $x_0, x_1, \ldots, x_n$ in order. The problem asks how many units to refuel at each x_i. Think of a directed graph with B repeated vertices at each x_i, representing each possible fuel level, say x_i^k indicates that the robot reaches x_i with k units of fuel remaining. The graph would have edges going from x_i^k to x_{i+1}^l. What should the costs on the edges be?

Problem 3 — 10 points

Assume that we have a bounded environment where there are a finite number of *convex* obstacles. Under these assumptions, does the Bug 0 strategy give us a *complete* algorithm? If no, then you must give a counter-example. If yes, then justify with a proof sketch.

Problem 4 — 50 points

Implement the dynamic programming algorithm to solve the minimum cost path planning problem on a given graph. You may implement your algorithm using one of the following programming languages: MATLAB, C/C++, or Python.

The input to your algorithm should be a file called `input.txt`. This file has a specific format as given below:

- The first line of the input file gives the total number of vertices, n, in the graph.
- The second line gives the starting vertex index (indices go from 1 to n).
- The third line gives the goal vertex index (indices go from 1 to n).
- Starting from the fourth line, we have the edges in the graph specified in the form of an edge list. Each line specifies one edge: `i j wij` which indicates that there is a (directed) edge from `i` to `j` with a cost of `wij`. Note that this is a directed graph.

Your code must produce a file called `output.txt`. This file must contain the vertices along the shortest path as well as the optimal value function (which is the first column in the table). The output file must have the following format:

- The first line must list the indices of the vertices on the shortest path – from the start to the goal vertex.
- The second line must list all the optimal value functions, i.e., $V(x_0), V(x_1), \ldots, V(x_n)$.

Two sample input and output files are given on canvas (also shown in Figure **??**). The figures were plot using the MATLAB graph function: `https://www.mathworks.com/help/matlab/ref/graph.plot.html`.

In addition to the code, you must write a short paragraph describing your implementation in the pdf report. This description should include instructions on how to compile and run your code. We will test your code on instances other than the two input files. Make sure you follow the input/output conventions exactly.