Fall 2022 - CSC545/645 Artificial Intelligence - Assignment 3

Due date: Thursday, September 22, 2022, 2pm. Please create a folder called assignment3 in your local working copy of the repository and place all files and folders necessary for the assignment in this folder. Once done with the assignment, add the files and folders to the repo with svn add *files,folders* and then commmit with svn ci -m "SOME USEFUL MESSAGE" *files,folders* .

Exercise 3.1

Read chapter 3.5 – 3.6 (informed search strategies and heuristic functions) of the textbook.

The heuristic path algorithm is a best-first search in which the objective function is $f(n) = (2 - w)g(n) + wh(n)$. For what values of w is this algorithm guaranteed to be optimal? What kind of search does this perform when $w = 0$? When $w = 1$? When $w = 2$? [4 points]

When w = 0 this means we do not consider the heuristic at all and only calculate the path based on the true cost of taking an action (technically 2 * the cost but since everything is scaled equally it's essentially just the cost). This is equivalent to a greedy search and this algorithm will always just pick the shortest option available to it.

When w = 1 this is A* search. The function takes a combination of the true cost and a heuristic cost (which always is an underestimation of the real cost but still reflects in some way actions that would need to take place). A* search is the most optimal search algorithm we have learned.

When w = 2 this is just a purely-heuristic search where only the heuristic is considered. Only the value of the heuristic function is used.

Exercise 3.2
Consider the route finding problem between 2 points in a plane that has convex polygons as obstacles (see figure 1). Find the shortest path. This is an idealistic navigation problem for a robot (e.g. navigating through a museum).
[16 points in total]

1. **Suppose the state space consists of all positions (*x, y*) on the plane. How many states are there? How many paths are there to the goal?**
   [2 points]

   If every position available is on an x, y plane then there are x * y (where x and y are the number of x positions and y positions, in cases such as being able to be located at decimal places on a plane) possible states. Despite a limited number of states there are infinitely many paths to the goal if you allow loops.

2. **Explain briefly why the shortest path from one polygon vertex to any other in the scene must consist of straight-line segments joining some of the vertices of the polygons. Define a good state space now. How large is this state space?**
   [2 points]

   We already know the shortest path between any two points is a straight line (Pythagorean theorem), so that is why we consider straight lines and not curved lines or triangles. Therefore, since we can only move between vertices of polygons, the shortest path is made by moving in this way.

   Since these are the only positions we can choose, the state space would be the total number of vertices across every shape in the plane.

3. **Define the necessary functions to implement the search problem, including a successor function that takes a vertex as input and returns the set of vertices that can be reached in a straight line from the given vertex. (Also think about the neighbors on the same polygon.) Use the straight-line distance for the heuristic function.**
   [2 points]
   For clarity imagine some line (parent) ——-------> (point)

   g(n) = the distance from the parent to point
   [sqrt((parent.x-point.x)^2 + (parent.y-point.y)^2]
   h(n) = the distance from this point to the goal state
   [sqrt((point.x-goal.x)^2 + (point.y-goal.y)^2]
   f(n) = g(n) + h(n)

   `Successor` = Check all possible connections that can be made from point, and remove those which have a shape in the way. Points along the same polygon (according to class example) are allowed to be possible next points. All successors have point as their parent and a calculated f(n) cost.

   `findCheapestState` = For each successor we have in a list, find the one with the current lowest f(n) cost. If this is the goal function, return success! Otherwise, make this state the new point, and repeat the successor function.

4. **Implement and apply one or more of the classical search algorithms that we have discussed in class to solve a range of problems in the domain, and comment on their performance. A* should be among them. CSS645-students: you need to use at least three such as A*, Greedy, and Uniform-Cost search. Make sure that the implementation is independent of the specific domain. This means that the same algorithm should be able to solve other variants of the**

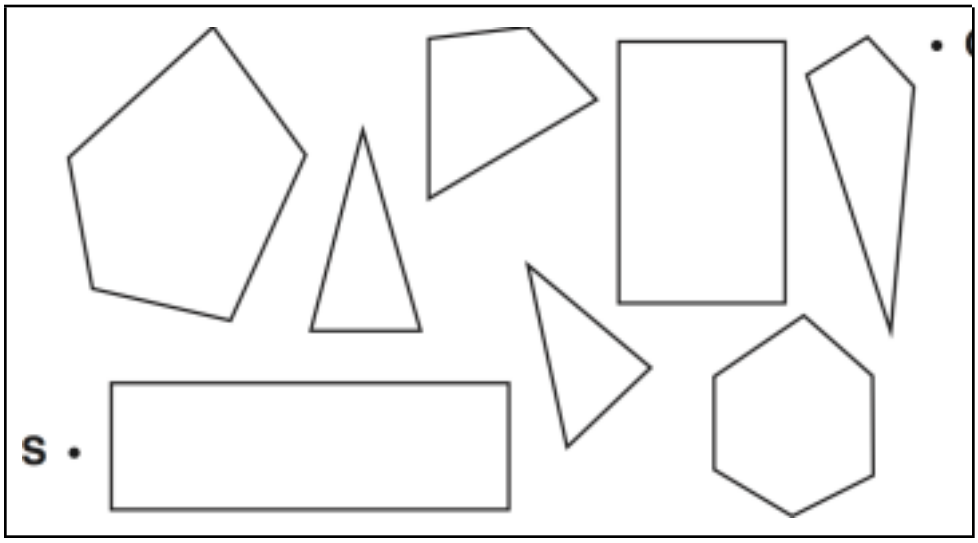

Figure 1: A scene with random polygonal obstacles. S and G are the start and goal states. See page 114 of our textbook (3rd edition).

problem (e.g. the polygons in the plane are different). You may want to use the framework provided by Michael Davis (download framework for Java, C++, Python).
[10 points]