Harvard CS 182: Artificial Intelligence September 14, 2022

Section 2: Informed Search and Motion Planning

Lecturer: Stephanie Gil *Authors: Lauren Cooke and Chris Lee*

Informed Search

Unlike the uninformed search algorithms we discussed last week, *informed search* algorithms use information beyond that given in the problem statement via a heuristic function.

- We can think of a *heuristic* as a shortcut strategy to find our solution within a quantifiable window of accuracy. These shortcuts can be useful in situations where looking for the exact answer with a more brute force method will take too long and where your heuristic will provide a solution that is close enough to the true answer you are looking for.

Example 1 *One way we can think about the benefits of using a heuristic would be in calculating tip after eating out at a restaurant. Say your bill comes out to* \$25.13*, and tax ends up being* \$2.60*. If your goal is to leave a* 20% *tip, knowing that sales tax percentage is roughly between* 8% *and* 11%*, you could apply the heuristic of calculating your tip by doubling the tax amount. In this case, you would be leaving* \$5.20*, or a* 20.7% *tip; which is close enough to your target and saves you from the hassle of doing long division.*

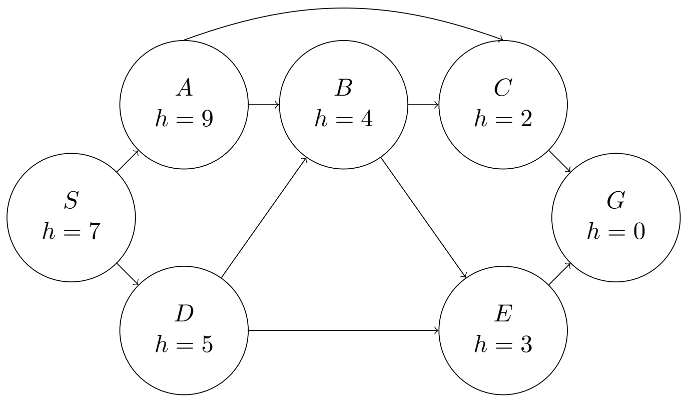
Problem 1 *You find yourself on the main road in a city that you have never been to before, and you are really hungry. This road has 100 restaurants. Your goal is to find the best restaurant on this road, as you are looking for somewhere delicious to eat. You could hypothetically go eat at every single restaurant on the street, but this would take a long time and be quite expensive. What heuristic could you employ to solve this problem?*

Informed Search Algorithms

Different informed search algorithms will take advantage of unique heuristics in trying to solve problems:

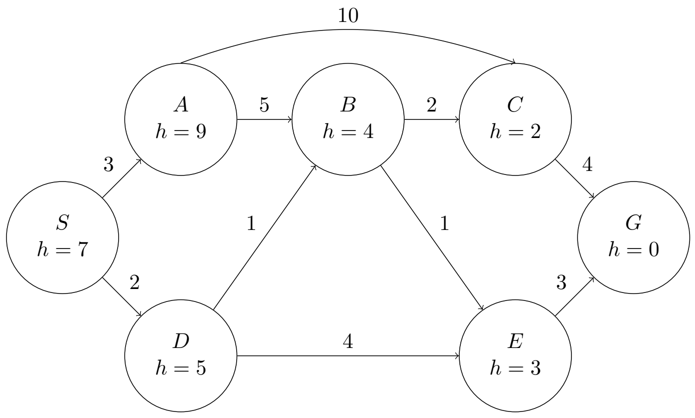
- For *Greedy Search*, we expand the node of our search problem closest to the goal node, measuring this closeness using a heuristic estimate $h(x)$. The lower the output of $h(x)$, the closer state x is to the goal state.

  Problem 2 *Find the path from the start state, s, to the goal state, g using the greedy search algorithm and the heuristic h values located inside each node. (From Source 2)*



- For *A* Tree Search*, we combine the closeness heuristic $h(x)$ from greedy search with the cost summation, $g(x)$ from uniform cost search from last week to create a new heuristic, $f(x) = h(x) + g(x)$. We choose to sum these two values to get a rough understanding of the total cost of a path through node x:
  - $h(x)$ estimates the distance from the current node x to the goal state
  - $g(x)$ estimates the cost from the start node to the current node x
- *A* Graph Search* is largely similar to A* Tree search, as it uses the same heuristic in making its expansion decisions. However, this strategy avoids the problem of expanding the same node more than once in different branches by using graph search to keep track of nodes that have already been expanded and therefore never expanding the same node more than once.

Problem 3 *Find the path from the start state, s, to the goal state, g, using the A* graph search algorithm, heuristic h values per node, and edge weights specified in the graph below (from source 2):*



Heuristic Quality and Optimality

We can categorize the quality of a heuristic both in terms of the problem space and in terms of other heuristics:

- A heuristic h *dominates* the heuristic h' if and only if $\forall x$, $h(x) \geq h'(x)$. A heuristic h' is therefore *dominated* by heuristic h because h will produce a larger or equal value than h' for all inputs.
- A heuristic is *admissible* if, $\forall$ nodes x, $h(x) \leq h^*(x)$, where $h^*(x)$ represents the cost of the optimal path to a goal. In other words, a heuristic is admissible when it either underestimates or is exactly correct with its estimate of the cost of the optimal path.
- A heuristic is *consistent* if, for every 2 nodes x, y, i.e. $h(x) \leq c(x, y) + h(y)$ where $c(x, y)$ is the cost of the cheapest path between x and y. In other words, a heuristic is consistent if taking a detour is more costly then remaining on the current path.
- Note that consistency implies admissibility when $h(t) = 0$ for all goals t.

Using these definitions for heuristic quality, we can assert the following about optimality for our different A* Search strategies:

Theorem 1 *A* tree search with an admissible heuristic returns and optimal solution.*

Proof: Let the suboptimal goal t be expanded before the optimal goal $t*$. Then $\exists$ a node x on the optimal path to t^* that has been discovered but not expanded because t is suboptimal. So:

$$\begin{aligned} f(x) &= g(x) + h(x) && \text{by definition of } f \\ &\leq g(x) + h^*(x) && \text{by admissibility} \\ &= g(t^*) < g(t) && \text{because not yet on the optimal path} \\ &= f(t) && \text{because } g(t) = f(t) \text{ because } h(t) = 0 \end{aligned}$$

Therefore, $f(x)$ is strictly smaller than $f(t)$, and thus x should have been expanded before t, meaning A* tree search with an admissible heuristic returns an optimal solution. ■

Theorem 2 *A* graph search with a consistent heuristic returns an optimal solution.*

Motion Planning

We will now see how it is possible to apply search algorithms to the task of motion planning, e.g. programming a robot to move to a goal while avoiding some obstacles. This brings with it a two main challenges:

- Practical challenges: Physical limitations robots, and often times there are many degrees of freedom, creating a high-dimensional state space

- The search space is often continuous, meaning that there would naively be infinitely many nodes that are infinitesimally close to one another

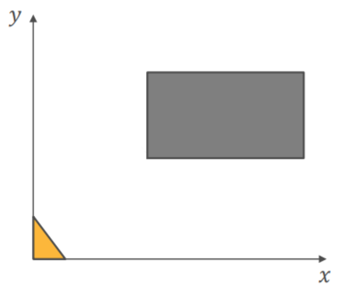
To rectify both problems, the main idea is to determine a better way of modeling the state space. The two main methods discussed in lecture were to represent the search space as a configuration space and to discretize the state space.

Problem 4 *What is the problem with treating every possible location as its own node? Notably, search algorithms can be directly used on other infinite state spaces – what is it about this particular problem formulation that gives these algorithms difficulty?*

Configuration Space

Often times, the variables that the robot can directly control are nontrivial to convert into 3D space. For example, consider a robot with n joints; the variables it is able to directly control are the angles of each of these joints, which is difficult to visualize in space. Thus, it is often easier to solve the search problem within this n-dimensional space rather than by using x, y, and z-coordinates.

Problem 5 *Consider the search space below, where the our robot is the yellow triangle in the bottom left and the gray rectangle is an obstacle. Suppose our robot cannot rotate or flip, and let the configuration space be all the (x, y)-coordinates that the bottom left corner of our robot can occupy. What will the configuration space look like?*



Visibility Graphs

A *visibility graph* is a convenient way of representing search spaces in which all obstacles are polygonal. In a visibility graph, the nodes are the vertices of the obstacles as well as start and goal, and the edges are the paths between nodes that do not enter the interior of any obstacle (including those between vertices on the same object, i.e. the edges' obstacles).

The benefit of this representation is that the shortest path on this graph will always be optimal. This can be seen (as in lecture) through a proof by contradiction, by considering an 'optimal' path that has a vertex in free space or on an edge of an obstacle. Thus, we can use any optimal graph search algorithm to find the shortest path.

This method unfortunately does not generalize to 3D space – finding the optimal path in 3D space is NP-hard.

Problem 6 *Consider the following two hexagonal obstacles below and the red equilateral triangles that can be formed from some of their vertices. Consider two different state spaces, each with one of these obstacles and start and goal nodes in random locations. Suppose we draw the visibility graphs as if the triangles (and not the hexagons) are the obstacles. Will either of these visibility graphs lead to the optimal solution in their respective state space?*