COMP-424 - Assignment 1

Posted Monday, January 14, 2013
Due Monday, January 28, 2013

Unless otherwise specified, please submit your assignment by e-mail to cs424@cs.mcgill.ca, by 11:59pm

1. [5 points] **Reading**

   As discussed in class, in 1950, Turing published a seminal paper called “Computing Machinery and Intelligence”. Read the paper (the pdf is linked from the course schedule page). Summarize in one paragraph the main AI issues that Turing anticipated in this paper.

2. [15 points] **Problem formulation**

   (a) Consider the graph given as an example in Lecture 2 (starting in slide 33). Suppose that we want to find a path from the start vertex to the goal vertex such that we always pass through node h. Describe a state-space formulation for this problem. What is the size of the state space in this case?

   (b) Suppose that we have a general graph and we want to find a path from a designated start vertex to a designated goal vertex that passes through k other, given vertices (you want to pick up your friends from their homes before you head off to the party). What is the state space in this case, and what is the size of the state space? Justify your answer.

   (c) Suppose that we want to visit *every single vertex* in the graph at least once and end up at the designated goal vertex. What is the state space in this case, and what is the size of the state space?

3. [10 points] **Uniform-cost search**

   Suppose that we take an initial search problem and we add $c > 0$ to the costs on all edges. Will uniform-cost search return the same answer as in the initial search problem? Justify your answer *precisely* (i.e. either by a proof or by concrete examples).

4. [10 points]A^* **search**

   As discussed in class, A^* works with heuristics that are admissible (i.e., they under-estimate the true cost of getting to the goal). Under these circumstances, A^* finds the optimal path. But what if the heuristic is not exactly admissible? More precisely, suppose that for any node n, its heuristic value $h(n)$ may overestimate the optimal cost-to-go, but only by a small amount ϵ:

   $$h(n) < h^*(n) + \epsilon, \forall n$$

   Let n^* be the optimal goal node, and $g(n^*)$ be the cost of the optimal path. Let n_g be the node returned by running A^* with h as a heuristic. Prove that:

   $$g(n_g) \leq g(n^*) + \epsilon$$

Note: You may assume that the function $f = g+h$ is monotonic (so A^* expands nodes in increasing value of f).

5. [30 points] **Problem formulation and heuristics**

   This is problem 3.27 from Russell & Norvig, Third Edition (with a bit different notation). Suppose you have an $n \times n$ grid and n vehicles are on its top row (numbered 1). The vehicles have to get to the bottom row, such that the vehicle that started at index $(1, i)$ must end up at index $(n, n-i+1)$ (in other words, the order of the vehicle must be reversed). On each time step, each of the vehicles can move one square up, down, left or right, or it can stay put. If the vehicle stays put, one adjacent vehicle (but not more than one) can hop over it. Two vehicles cannot occupy the same square.

   (a) [5 points] Calculate the size of the state space as a function of n

   (b) [5 points] Calculate the branching factor as a function of n

   (c) [5 points] Suppose that vehicle i is at location (x_i, y_i). Write a non-trivial admissible heuristic h_i for the number f moves it will required to get to its goal location, assuming no other vehicles are on the grid.

   (d) [15 points] Which of the following heuristics are admissible for the problem of moving *all* the vehicles to their destination?

      i. $\sum_{i=1}^{n} h_i$

      ii. $\max_{i=1\ldots n} h_i$

      iii. $\min_{i=1\ldots n} h_i$

6. [30 points] **Programming**

   For this part of the assignment, you are provided with a random Maze generator, where each state has a cost associated with it (rather than each operator being associated with a cost). Your job is to program A^* search for the maze, in Java. A^* should use the following heuristics: (a) h_1: the straight-line distance to the goal; (b) h_2: the sum of the displacement along the x and y axes; (c) $0.5h_1 + 0.5h_2$. You can use predefined classes for the priority queue implementation.

   You have to turn in your source code, as well as a small report showing the number of states expanded by each of the heuristics, for a 20 x 20 maze. Explain the results.

   The Maze generator will be posted on Wednesday, January 16.