AI Intro (IT-3105) Homework Module #1:
Using the A* Algorithm to Solve Rush Hour Puzzles

August 4, 2017

Purpose: Gain hands-on experience with best-first search using the A* Algorithm.

In this assignment, you must first become familiar with the A* code by implementing it from scratch in the language of your choice. Once you have a working version of A*, you will apply it to several examples of the Rush Hour puzzle. Your system should be highly modular such that the core A* code can be used to solve many other search problems, including that of homework module 2.

1 Rush Hour

Rush Hour is a popular puzzle (and also a free App from the Apple Store). The standard board is 6 x 6, with the pieces having dimensions 2 x 1 (cars) and 3 x 1 (trucks). One special car (car-0) needs to get to a particular spot on the edge of the board, i.e., the exit. Most (if not all) other cars and trucks must be moved to enable car-0 to reach this goal . Each vehicle is positioned either horizontally or vertically, and it can only move along that row or column, respectively, throughout the solution sequence. So a car or truck positioned horizontally along row 3 can only move back and forth within row 3.

Figure 1 shows a typical scenario, with 5 cars and one (yellow) truck. The red vehicle (car 0) needs to get to the exit in as a few moves (of car 0 and all other vehicles) as possible. Optimal solutions are those involving a minimum amount of total vehicle movement. A *move* is thus any one-cell translation of a vehicle. So if car-0 in Figure 1 drives from (2,2) to (0,2), that counts as 2 moves. Vehicles 0 - 3 can only move horizontally, while vehicles 4 and 5 move only vertically. The grid is 6 x 6, with dots marking the center of each grid cell. No part of any vehicle can move beyond the square boundary.

For our purposes, the origin of this 6 x 6 discrete cartesian plane is in the upper left. So the upper left cell has coordinates (0,0), while the lower right cell has coordinates (5,5). As shown in Figure 1, a scenario is described by a set of quads, one for each vehicle. The 4 elements of a quad, (O, X, Y, S), denote the orientation, horizontal (0) or vertical (1)), the coordinates (X and Y) of the upper left portion of the vehicle, and its size (S). A specification with K vehicles consists of K quads, listed in order of the vehicle number; the quad for car-0 is always first. In the figure, the quad for car 3 is (0,4,1,2), indicating that it is oriented horizontally and that the leftmost cell that it currently occupies is (4,1) - 4 moves over and 1 move down from the origin. Finally, the 2 represents the size of car 3: it spans 2 grid cells.

For all examples given in this homework exercise, the exit is at the same location (5,2) as shown in Figure 1, and all vehicles have size 2 or 3. Solving the puzzle entails finding a combination of moves that enables

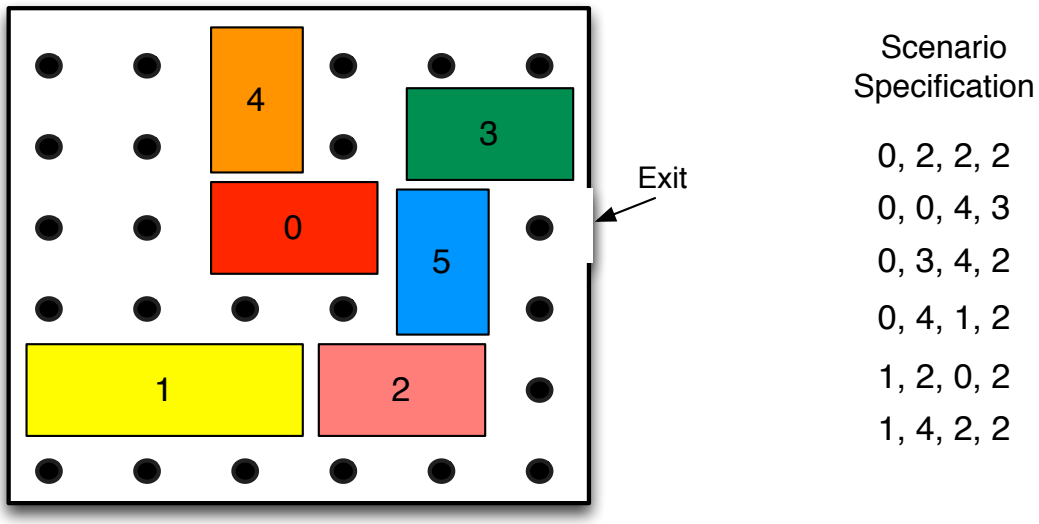

Figure 1: A simple Rush Hour puzzle (left) and its corresponding text specification (right). Small circles denote the centers of each grid cell; the upper left circle, the origin, has coordinates (0,0).

the right half of car-0 to cover location (5,2); from there it can obviously move *out of the traffic jam*.

1.1 Puzzle Variants

You will need to run A* on the 4 scenarios of Table 1. These constitute easy, medium, hard, and expert-level puzzles found in the free Rush-Hour App.

2 Visualization

Your system must be able to display states of the Rush Hour game in an easily-understandable, graphic format. When run in *display* mode, it must show both:

1. The board state corresponding to the node that was most recently popped from the agenda and expanded.
2. The complete sequence of states along the final solution path.

In other words, display mode allows you to see the nodes that A* is currently considering, and then, when a solution has been found, display mode allows you to show all states along the path from start to goal state.

Rush Hour Scenarios				
Vehicle	Easy-3	Medium-1	Hard-3	Expert-2
0	(0,2,2,2)	(0,1,2,2)	(0,2,2,2)	(0,0,2,2)
1	(0,0,4,3)	(0,0,5,3)	(0,0,4,2)	(0,0,1,3)
2	(0,3,4,2)	(0,1,3,2)	(0,0,5,2)	(0,0,5,2)
3	(0,4,1,2)	(0,3,0,2)	(0,2,5,2)	(0,1,0,2)
4	(1,2,0,2)	(1,0,2,3)	(0,4,0,2)	(0,2,3,2)
5	(1,4,2,2)	(1,2,0,2)	(1,0,0,3)	(0,3,4,2)
6		(1,3,1,2)	(1,1,1,3)	(1,0,3,2)
7		(1,3,3,3)	(1,2,0,2)	(1,2,4,2)
8		(1,4,2,2)	(1,3,0,2)	(1,3,0,3)
9		(1,5,0,2)	(1,4,2,2)	(1,4,0,2)
10		(1,5,2,2)	(1,4,4,2)	(1,4,2,2)
11			(1,5,3,3)	(1,5,2,2)
12				(1,5,4,2)

Table 1: Specification quads for four different Rush Hour scenarios. The name of each scenario indicates its source in the free Rush-Hour App. For example, Hard-3 denotes the 3rd scenario in the set of puzzles that the App classifies as *Hard*. For all scenarios, car-0 is the vehicle that must drive to the exit, location (5,2), in order to complete the puzzle.

3 Demo Examples

At the demo session, you will be required to:

- Run your system on 2 of the RUSH HOUR examples presented above, as chosen by the course instructor or assistant.
- Run your system on 2 other examples that will be given to you at the demo session. These will also involve a 6-by-6 board and will use the same file format as the other examples.

At the demo session, all of the scenarios provided to you ahead of time (on the course web page) should be easily available to your system during the demo. For example, you should be able to enter an index or file name and have any of these cases loaded automatically. Failure to streamline the scenario-entry process can lead to a loss of points.

4 Report Contents

Your report (of no more than 3 pages) must be written in Norwegian or English and include the following:

For each of the 4 puzzles above, you must clearly document the following **key aspects** in your report:

1. A clear, concise description (using **mathematical expressions** and text) of the heuristic function (h) used to solve the puzzle.
2. A thorough description of the procedure used to generate successor states when expanding a node.

3. A comparison of A*'s performance to that of both depth-first and breadth-first search when applied to each of the 4 puzzles above.

The comparison of A* to depth- and breadth-first search should consist of two factors:

- the total number of nodes generated by the search tree,
- the number of moves in the first solution that the method finds. All of this should be expressed in the **exact same format** as shown in Table 2

Your comparison will largely consist of a table, an example of which appears in Table 2, along with a short paragraph or two explaining the main results.

	Puzzle Variants			
Search Method	Easy-3	Medium-1	Hard-3	Expert-2
Breadth-1st	(103, 16)	(12987, 39)	(2053, 33)	(10938, 73)
Depth-1st	(82,31)	(4816, 1180)	(2258, 568)	(5283, 1124)
Best-1st (A*)	(77,16)	(9429, 39)	(923, 33)	(5685, 73)

Table 2: Comparison of A*, depth-first and breadth-first search on 4 rush-hour puzzles. Pairs in parentheses are the node count and number of moves (from start to goal), respectively.

5 Deliverables

1. Well-commented source code for the general A* procedure.
2. Well-commented source code for the specialization of A* needed to handle Rush Hour problems.
3. Your report of 3 or less pages. Reports that fail to properly convey the requested information or that exceed 3 pages may not receive full credit. **(3 points)**
4. Demonstration of your system on 4 different scenarios **(12 points)**

The 15 total points for this module are 15 of the 100 points that are available for the entire semester. The due date for this module is the first demo session.

A zip file containing your report along with the commented code must be uploaded to BLACKBOARD prior to the demo session in which this project module is evaluated. You will not get explicit credit for the code, but it is crucial that we have the code online in the event that you decide to register a formal complaint about your grade (for the entire course).