A* Solution to the Rush Hour Game

James Devine

May 5, 2008

1 About A*

The A* search algorithm is a search algorithm that uses a heuristic to estimate the cost of taking a given path in the solution tree to the goal state. The cost is calculated by adding the $g + h$ values. The g value is the cost in steps to get to the current state and the h value is the estimated distance from the solution. For the algorithm to work correctly the heuristic must be at best an underestimate of the actual distance from a solution.

2 How the Code Works

The actual A* search is only a few lines of code. It first takes a board node from the top of the priority queue, if the node is a goal node the algorithm quits, if not it expands all of the board configurations that are reachable from the current board node and adds then to the queue(after checking to make sure that the board node has not been explored). The process is repeated until the queue is emptied or a solution is found. In the event that the queue is emptied it signifies that there is no solution.

The implementation used generates board configurations and then adds then to board nodes. Each board has the ability to calculate its heuristic, generate all moves that can be made, generate a hash, and determine if the board is at a goal state. To determine if a board is at a goal state the method isGoal checks to make sure there are spaces before the car trying to exit and the exit. The heuristic is determined by adding the number of spaces that are blocked in front of the car trying to exit.

The driver program can take an integer command line argument from 1-10 that will use a stored board to solve. The boards range in complexity, with numbers closer to 10 being harder to solve. All of the boards are solvable though.

3 Program Evaluation

The implementation used is rather general. In the test performed by running the included configurations, the number of moves varied greatly. Two of the hardest puzzles were 9 and 10. Each were difficult in the fact that there were a lot of cars on the board with only a few blank spaces. Configuration 9 took 59 moves and configuration 10 took 102 moves. This is compared to around 3-11 moves for the other configurations. The huge increase in moves had very little impact on performance. Each test printed out results nearly instantly.

An inherent problem with the Rush Hour Game is that it takes up quite a bit of memory. This is especially true of this implementation. Each node contains a board and a link to a parent node. Memory was not an issue with the set size if a 6x6 board in this implementation. If this were increased to a larger size the program would still run quickly, but the space growth in relation to the puzzle size would be very large.