Artificial Intelligence Project 1 Report:

Traffic Jam Puzzle

Gary Chen

CSCI 373 Prof. Jon Park

October 7, 2016

Abstract

The traffic jam puzzle, like the games implemented in the retail board game "Rush Hour" and the popular iOS app "Unblock Me", is a game played with 2- or 3-length pieces laid on a grid with the goal of moving a specific piece out of a specified exit on one edge of the grid. At any point, any piece can be moved forwards or backwards in its lengthwise direction until it hits a wall or another piece. To formulate this as an AI search problem in Python, game states were represented by data structures that kept track of the grid size, the exit location on the top edge, and every piece's location, length, and orientation. The initial search approach was a simple breadth-first search with a check to prevent cycling game moves; this worked decently well, finding the optimal solution in an average of 1,229 game states checked for solutions for the three test boards. Later, a simple heuristic was implemented that counted the number of pieces blocking the solution piece from the exit and the search algorithm was changed to A*; this improved the performance slightly, reducing the average number of game states checked to 1,029. If a better heuristic method was employed, the performance of the search could probably improve more drastically; however, the current solution runs in under thirty seconds without any parallel processing.

Introduction

This traffic-jam puzzle is representative of a larger category of search problems that attempt to find a solution path through a search tree without knowing what depth the solution is at and where the branching factor can be relatively high. Another feature that shaped the approach to solving this problem which may be relevant to others is that there can be many cycles between various nodes or game states, meaning the search algorithm must be careful not

to become caught in search loops or end up finding a solution which is sub-optimal when there is another better path. Many puzzles, games, and real-life scenarios can be described with these characteristics, making it important that we understand the best methods for AI search in these situations.

Formulation

The data structures used to represent the game state did not check for valid arrangements of pieces on the board, meaning pieces could have coordinates that were off the grid or which overlapped other pieces. However, the functions which performed moves on game states did only allow valid moves; assuming the initial state given to the search agent is valid, this means the agent will never explore a game state which isn't a valid state for the traffic jam puzzle. The puzzle's exit was represented as a x-coordinate and was assumed to be along the top edge of the grid. The program didn't ever need to keep track of a "solution" piece; the goal test was just to check if any piece was vertically aligned with and adjacent to the exit, and similarly the heuristic function counts pieces in the column of the exit until a vertically-oriented piece is encountered. The valid actions for any given game state are given by the set of all valid moves for each piece on the board, which means moves in the forwards or backwards direction of the piece's orientation until a wall or other piece is hit (excluding zero moves).

Approaches

Initially, a simple breadth-first search was used to find the solution, with an extra data structure to track seen game states (added to the search queue, not necessarily goal-tested yet) to prevent cycles, which are very common for this problem. The BFS was chosen because the

solution depth of a puzzle could be any number zero or greater. However, an A* search was a very simple heuristic was later implemented to see if the agent's performance would improve significantly. The heuristic was a count of the number of pieces between the puzzle exit and the solution piece; in any game state, at least the pieces directly blocking the solution piece from the exit must be moved for the solution car to move through, so the function was definitely admissible. Unexpectedly, implementing this algorithm significantly improved performance but caused the agent to return a sub-optimal solution in one of the test puzzles; this was determined to be because the agent was evaluating certain moves that reached a node on the optimal solution path with a promising heuristic but sub-optimal moves leading up to it. The node on the solution path was then added to the "seen states" list, and when the optimal path (with a worse heuristic) reached that node later, it was ignored because the state had been seen before. To correct for this, seen-states were not stored by the state of the board but by reference to the actual search node so they could be updated in the priority queue with a better so-far path if the new search node's state had been seen before but took less moves to get there. This doesn't seem like a problem unique to the traffic-jam puzzle; it is likely applicable to any search problem where a severely underestimating heuristic is used and search cycles must be prevented. For this problem, any move can only improve the heuristic by one at most, so the more optimal path was reached one search depth later, but one can imagine another problem where heuristic improves drastically with a move that isn't actually the best one.

Experiment and Analysis

The breadth-first search found the optimal solution relatively quickly, running in under 30 seconds and goal-testing an average of 1,229 game states across three test boards. The A*

algorithm with a simple "number of blocking pieces" heuristic improved performance slightly, goal-testing an average of 1,029 states, but also included some higher overhead making the real-world runtime about the same. The meager improvement is likely due to the heuristic being rather inaccurate, especially since in many cases the optimal move was one that increased the heuristic of game state. More specifically for these initial states, moves which did decrease the heuristic were not possible until near the end of the solution, not allowing much room for improvement. For Initial State B, one move actually increased the heuristic, blocking the solution piece more before it could be moved away. Initial State C's pieces were rather "gridlocked", meaning the number of possible moves for each state was low, probably explaining why there wasn't much room for improvement.

Fig. 1: Number of Goal Tests for Different Search Algorithms

	Initial State A	*Initial State B*	*Initial State C*	*Average*
BFS	917	2384	385	1228.67
*A**	533	2189	366	1029.33

Conclusions

This implementation of a solution to the traffic jam puzzle using an A* search AI was largely successful; even the basic breadth-first-search ran faster than expected, and adding a heuristic to make it an A* algorithm only decreased the number of nodes goal-tested. We could definitely improve the performance of the agent more dramatically by implementing a better heuristic, for example one which recursively counts how many pieces block the pieces that block the exit, and how many pieces block those pieces, etc. Another interesting option could be to

explore are more probabilistic search algorithms that rank moves by how good they're estimated to be, perhaps by counting how many possible moves for the next step they open up. This implementation's weak heuristic prevented it from improving much over a more naïve breadth-first search, though perhaps for larger puzzles we might see a better improvement.

Appendix

Appendix A: *A* Search Solution for Initial State A*

```
          *
_ _ _ A B B
_ _ _ A C C
D E E E F F
D _ _ _ _ #
_ _ G G G #
_ _ _ _ _ _

Move car at (0, 2) by 2
          *
_ _ _ A B B
_ _ _ A C C
_ E E E F F
_ _ _ _ _ #
D _ G G G #
D _ _ _ _ _

Move car at (1, 2) by -1
          *
_ _ _ A B B
_ _ _ A C C
E E E _ F F
_ _ _ _ _ #
D _ G G G #
D _ _ _ _ _

Move car at (3, 0) by 2
          *
_ _ _ _ B B
_ _ _ _ C C
E E E A F F
_ _ _ A _ #
D _ G G G #
D _ _ _ _ _

Move car at (4, 0) by -4
          *
B B _ _ _ _
_ _ _ _ C C
E E E A F F
_ _ _ A _ #
D _ G G G #
D _ _ _ _ _

Move car at (4, 1) by -4
          *
B B _ _ _ _
C C _ _ _ _
E E E A F F
_ _ _ A _ #
D _ G G G #
D _ _ _ _ _

Move car at (3, 2) by -2
          *
B B _ A _ _
C C _ A _ _
E E E _ F F
_ _ _ _ _ #
D _ G G G #
D _ _ _ _ _

Move car at (4, 2) by -1
          *
B B _ A _ _
C C _ A _ _
E E E F F _
_ _ _ _ _ #
D _ G G G #
D _ _ _ _ _

Move car at (5, 3) by -3
          *
B B _ A _ #
C C _ A _ #
E E E F F _
_ _ _ _ _ _
D _ G G G _
D _ _ _ _ _

Puzzle completed in 8 moves.
Number of nodes visited in search:  533
```

Appendix B: *A* Search Solution for Initial State B*

```
        *
A B C D D D
A B C E E E
A B F F G G
H H _ _ # _
_ _ _ _ # _
_ _ I I _ _

Move car at (4, 3) by 1
        *
A B C D D D
A B C E E E
A B F F G G
H H _ _ _ _
_ _ _ _ # _
_ _ I I # _

Move car at (0, 3) by 4
        *
A B C D D D
A B C E E E
A B F F G G
_ _ _ _ H H
_ _ _ _ # _
_ _ I I # _

Move car at (0, 0) by 3
        *
_ B C D D D
_ B C E E E
_ B F F G G
A _ _ _ H H
A _ _ _ # _
A _ I I # _

Move car at (1, 0) by 3
        *
_ _ C D D D
_ _ C E E E
_ _ F F G G
A B _ _ H H
A B _ _ # _
A B I I # _

Move car at (2, 2) by -2
        *
_ _ C D D D
_ _ C E E E
F F _ _ G G
A B _ _ H H
A B _ _ # _
A B I I # _

Move car at (2, 0) by 3
        *
_ _ _ D D D
_ _ _ E E E
F F _ _ G G
A B C _ H H
A B C _ # _
A B I I # _

Move car at (0, 2) by 2
        *
_ _ _ D D D
_ _ _ E E E
_ _ F F G G
A B C _ H H
A B C _ # _
A B I I # _

Move car at (0, 3) by -3
        *
A _ _ D D D
A _ _ E E E
A _ F F G G
_ B C _ H H
_ B C _ # _
_ B I I # _

Move car at (1, 3) by -3
        *
A B _ D D D
A B _ E E E
A B F F G G
_ _ C _ H H
_ _ C _ # _
_ _ I I # _

Move car at (2, 5) by -2
        *
A B _ D D D
A B _ E E E
A B F F G G
_ _ C _ H H
_ _ C _ # _
I I _ _ # _

Move car at (2, 3) by 1
        *
A B _ D D D
A B _ E E E
A B F F G G
_ _ _ _ H H
_ _ C _ # _
I I C _ # _

Move car at (4, 3) by -4
        *
A B _ D D D
A B _ E E E
A B F F G G
H H _ _ _ _
_ _ C _ # _
I I C _ # _

Move car at (2, 4) by -1
        *
A B _ D D D
A B _ E E E
A B F F G G
H H C _ _ _
_ _ C _ # _
I I _ _ # _

Move car at (4, 4) by -1
        *
A B _ D D D
A B _ E E E
A B F F G G
H H C _ # _
_ _ C _ # _
I I _ _ _ _

Move car at (0, 5) by 3
        *
A B _ D D D
A B _ E E E
A B F F G G
H H C _ # _
_ _ C _ # _
_ _ _ I I _

Move car at (2, 3) by 1
```

```
        *
A B _ D D D
A B _ E E E
A B F F G G
H H _ _ # _
_ _ C _ # _
_ _ C I I _

Move car at (0, 3) by 2
        *
A B _ D D D
A B _ E E E
A B F F G G
_ _ H H # _
_ _ C _ # _
_ _ C I I _

Move car at (1, 0) by 3
        *
A _ _ D D D
A _ _ E E E
A _ F F G G
_ B H H # _
_ B C _ # _
_ B C I I _

Move car at (3, 0) by -2
        *
A D D D _ _
A _ _ E E E
A _ F F G G
_ B H H # _
_ B C _ # _
_ B C I I _

Move car at (0, 0) by 3
        *
_ D D D _ _
_ _ _ E E E
_ _ F F G G
A B H H # _
A B C _ # _
A B C I I _

Move car at (2, 2) by -2
        *
_ D D D _ _
_ _ _ E E E
F F _ _ G G
A B H H # _
A B C _ # _
A B C I I _

Move car at (4, 2) by -2
        *
_ D D D _ _
_ _ _ E E E
F F G G _ _
A B H H # _
A B C _ # _
A B C I I _

Move car at (3, 1) by -3
        *
_ D D D _ _
E E E _ _ _
F F G G _ _
A B H H # _
A B C _ # _
A B C I I _

Move car at (4, 3) by -3
        *
_ D D D # _
E E E _ # _
F F G G _ _
A B H H _ _
A B C _ _ _
A B C I I _

Puzzle completed in 24
moves.
Number of nodes visited in
search:  2189
```

Appendix C: *A* Search Solution for Initial State C*

```
          *
_ _ C E E E
_ _ C F H H
_ B C F _ #
A B D D D #
A B _ G G G
A _ _ _ _ _

Move car at (0, 3) by -3
          *
A _ C E E E
A _ C F H H
A B C F _ #
_ B D D D #
_ B _ G G G
_ _ _ _ _ _

Move car at (1, 2) by -2
          *
A B C E E E
A B C F H H
A B C F _ #
_ _ D D D #
_ _ _ G G G
_ _ _ _ _ _

Move car at (2, 3) by -2
          *
A B C E E E
A B C F H H
A B C F _ #
D D D _ _ #
_ _ _ G G G
_ _ _ _ _ _

Move car at (3, 4) by -3
          *
A B C E E E
A B C F H H
A B C F _ #
D D D _ _ #
G G G _ _ _
_ _ _ _ _ _

Move car at (3, 1) by 3
          *
A B C E E E
A B C _ H H
A B C _ _ #
D D D _ _ #
G G G F _ _
_ _ _ F _ _

Move car at (4, 1) by -1
          *
A B C E E E
A B C H H _
A B C _ _ #
D D D _ _ #
G G G F _ _
_ _ _ F _ _

Move car at (5, 2) by -1
          *
A B C E E E
A B C H H #
A B C _ _ #
D D D _ _ _
G G G F _ _
_ _ _ F _ _

Move car at (0, 3) by 3
          *
A B C E E E
A B C H H #
A B C _ _ #
_ _ _ D D D
G G G F _ _
_ _ _ F _ _

Move car at (2, 0) by 1
          *
A B _ E E E
A B C H H #
A B C _ _ #
_ _ C D D D
G G G F _ _
_ _ _ F _ _

Move car at (3, 0) by -1
          *
A B E E E _
A B C H H #
A B C _ _ #
_ _ C D D D
G G G F _ _
_ _ _ F _ _

Move car at (5, 1) by -1
          *
A B E E E #
A B C H H #
A B C _ _ _
_ _ C D D D
G G G F _ _
_ _ _ F _ _

Puzzle completed in 11
moves.
Number of nodes visited in
search:  366
```