Analysis Report

COMP-472 - F - Fall 2022

Mini Project 2

Instructor:

Dr. Leila Kosseim

Team Members:

Kevin Nguyen	40158397
Vatanak So	40080207

Analysis

A link to our analysis spreadsheet

1. Lowest Solution Path

For each puzzle, the lowest solution path out of the three algorithms was calculated by the A* algorithm and the UCS algorithm. This falls in line with A* as an algorithm since A* is *admissible*. The UCS solution path length and the A* solution path lengths of h1, h2, and h4 are identical. However, h3 of A* would occasionally give solution path lengths +1 more than optimal solutions. I think this can be attributed to the multiplier used for h3, being 4. This might have made A* inadmissible.

2. Admissibility of Each Heuristic

H1 is admissible. If there are no cars blocking car 'A' from its goal, the heuristic will return 0, otherwise h1 will always return the number of cars blocking car 'A' from its goal. From the data, h1 consistently returns the optimal path for each puzzle in A* as seen also in UCS which is considered admissible.

H2 is admissible. If there are no positions between car 'A' and its goal that are blocked, the heuristic will return 0, otherwise h2 will always return the number of positions that are not empty space between car 'A' and its goal. From the data, h2 consistently returns the optimal path for each puzzle in A* as seen also in UCS which is considered admissible.

H3 is inadmissible. Logically, h3 should be admissible as it follows the logic of h1 except with a coefficient attached to the result. However, from the data, there are instances where a* with h3 returns an nonoptimal solution path. That being said, the solution path h3 provides is only one move up from the optimal solution path.

H4 is admissible. It is the absolute sum of both h1 and h2. From the logic of both h1 and h2, h4 will return 0 only when the path has no blocked positions or cars blocking the goal. It returns the optimal path as its result for each puzzle is identical to the UCS solution.

3. The execution time across algorithms and heuristics

Algorithms

Given a well-designed and easy to compute heuristic, an informed search is always faster with the extra knowledge of the search tree. However, GBFS is the slowest of all in the analysis table of the 50 generated puzzles. Since GBFS mostly gives the shortest search path, one of the reasons behind the bad performance can be poor implementations of the heuristics. GBFS keeps all the children nodes in the priority queue which is sorted each time new children are added, so slightly worse implementation of heuristics will result in a dramatic performance drop. Another reason is that the heuristics are not specific enough (only consider the positions in front of car A), therefore, many nodes can have the same cost to goal, which leads to the algorithm being stuck in unnecessary loops. One example is the puzzle number 34, on which GBFS performed more than 100 times slower than the rest.

The Algorithm A* is the best performer in terms of execution time, because it also considers the cost from root to node *n* which reduces the indistinguishability of the heuristics.

Heuristics

h1 and h3 are almost identical for GBFS, because all the costs will increase by the same factor and will have no effect on the algorithm. For the algorithm A*, on the other hand, h3 allows for better distinguishing the cost of each node since it adds *g(n)*, also we can have an infinite range of coefficients to pick from. So, h3 is the fastest performer heuristic alongside A*, with the results shown on the table.

4. Other interesting facts

1. h1, h2, h3 are technically the same heuristics on GBFS, they only differ slightly in execution time. Because once we have the number of blockings in front of vehicle A, any moves outside the blocking position are treated the same.

2. If the vehicles in front of vehicle A are all vertical, then h1 and h2 are exactly the same, since any one blocking vehicle can only block one position, so #blocking_positions = #blocking_vehicles.

3. The algorithm A* combined with h4 is the same as UCS when all the blocking vehicles are vertical. This follows from 2, so that #blocking_positions - #blocking_vehicles = 0, the only cost is the cost from the root to node *n*.

Tasks

Kevin Nguyen	- UCS - A2* Search - h1, h2, h3, h4 - Rush Hour Puzzle Reader and Implementation - Analysis 1 & 2
Vatanak So	- GBFS - Heuristics h1, h2, h3 - Search paths GBFS - Solution paths GBFS - Analysis 3 & 4

References

- 50 puzzles: https://www.michaelfogleman.com/rush/#DatabaseFormat