AIND – Planning Project

# James Mallett

## Part 1 Result Documentation

**Problem 1**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Search Type** | **Node Expansions** | **Goal Tests** | **Time Elapsed**  **(s)** | **Solution Length** |
| breadth-first-search | 43 | 56 | 0.043644952 | 6 |
| breadth-first-tree-search | 1458 | 1459 | 1.0484 | 6 |
| depth-first-graph-search | 21 | 22 | 0.02199 | 20 |
| depth-limited-search | 101 | 271 | 0.1110259 | 50 |
| uniform-cost-search | 55 | 57 | 0.053333 | 6 |
| recursive-best-first-search with h-1 | 4229 | 4230 | 3.380909 | 6 |
| greedy-best-first-graph-search with h-1 | 7 | 9 | 0.008354 | 6 |
| astar-search with h-1 | 55 | 57 | 0.067594 | 6 |
| astar-search with h-ignore-preconditions | 41 | 43 | 0.040059 | 6 |
| Astar-search-with h-pg-levelsum | 11 | 13 | 0.6371 | 6 |

Depth-first-graph-search and depth-limited-search were the only two search types that resulted in a non-optimal solution length. As for the others, greedy-best-first-graph-search-with-h-1 was able to find the optimal solution with the least node expansions and goal tests it did followed by Astar-search-with-h-pg-levelsum which had a longer computing time. it was also the fastest search method. Greedy-best-first-graph-search was also the fastest search type that ended in an optimal solution.

**Problem 2**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Search Type** | **Node Expansions** | **Goal Tests** | **Time Elapsed**  **(s)** | **Solution Length** |
| breadth-first-search | 3343 | 4609 | 11.1005 | 9 |
| breadth-first-tree-search | **Stopped after 120s** | | | |
| depth-first-graph-search | 624 | 625 | 4.52146 | 619 |
| depth-limited-search | **Stopped after 120s** | | | |
| uniform-cost-search | 4852 | 4854 | 15.51258 | 9 |
| recursive-best-first-search with h-1 | **Stopped after 120s** | | | |
| greedy-best-first-graph-search with h-1 | 990 | 992 | 3.07647 | 21 |
| astar-search with h-1 | 4852 | 4854 | 15.05205 | 9 |
| astar-search with h-ignore-preconditions | 1450 | 1452 | 5.316719 | 9 |
| Astar-search-with h-pg-levelsum | 86 | 88 | 58.0178 | 9 |

Breadth-first-tree-search, depth-limited-search and recursive-best-first-search-with-h-1 were all stopped after 2 minutes of run time after seeing that a solution could be reached in a much shorter time. The optimal solution length was 9 moves, and was reached by all except the depth-first-graph-search and greedy-best-first-graph-search-with-h-1 which found solutions with non-optimal lengths. The fastest search methods with optimal solution lengths were astar-search with h-ignore-preconditions, breadth-first-search, astar-search-with-h-1 and uniform-cost-search in that order. Astar-search-with h-pg-levelsum expanded the least nodes and goal tests to find an optimal solution, the next best being h-ignore-preconditions.

**Problem 3**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Search Type** | **Node Expansions** | **Goal Tests** | **Time Elapsed**  **(s)** | **Solution Length** |
| breadth-first-search | 8602 | 11196 | 29.7487 | 12 |
| breadth-first-tree-search | **Stopped after 360s** | | | |
| depth-first-graph-search | 1292 | 1293 | 3.4746 | 875 |
| depth-limited-search | **Stopped after 360s** | | | |
| uniform-cost-search | 11483 | 11485 | 32.6559 | 12 |
| recursive-best-first-search with h-1 | **Stopped after 360s** | | | |
| greedy-best-first-graph-search with h-1 | 907 | 909 | 2.28907 | 19 |
| astar-search with h-1 | 11483 | 11485 | 33.0492 | 12 |
| astar-search with h-ignore-preconditions | 4117 | 4119 | 13.9255 | 13 |
| Astar-search-with h-pg-levelsum | 279 | 281 | 126.4737 | 12 |

Breadth-first-tree-search, depth-limited-search and recursive-best-first-search-with-h-1 were all once again stopped early (after 6 minutes each) as it was taking too long to find a solution. The optimal solution length was found to be 12 moves, and the only functions able to reach this optimal solution length were breadth-first-search, uniform-cost-search, astar-search-with-h-1 and Astar-search-with-h-pg-levelsum. The fastest of which was breadth-first-search, once again Astar-search-with-h-pg-levelsum expanded significantly less nodes than the others, but did however sacrifice computing time to do so. Uniform-cost-search, and astar-search-with-h-1 both expanded 11483 nodes with 11485 goal tests, they too 32.65 and 33.04 seconds respectively.

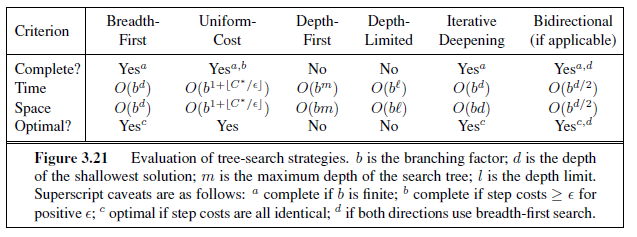
## Top Performing Searches to reach optimal solution:

The main factor considered in this evaluation was the time in which the search types took to find a solution.

|  |  |  |
| --- | --- | --- |
| **Problem 1** | **Problem 2** | **Problem 3** |
| greedy-best-first-graph-search with h-1 | astar-search with h-ignore-preconditions | breadth-first-search |
| astar-search with h-ignore-preconditions | breadth-first-search | uniform-cost-search |
| breadth-first-search | astar-search with h-1 | astar-search with h-1 |
| uniform-cost-search | uniform-cost-search | Astar-search-with h-pg-levelsum |
| astar-search with h-1 | Astar-search-with h-pg-levelsum |
| Astar-search-with h-pg-levelsum |
| breadth-first-tree-search |
| recursive-best-first-search with h-1 |

So the only search types to reach an optimal solution for each problem were breadth-first-search, uniform-cost-search, astar-search-with-h-1 and Astar-search-with-h-pg-levelsum. And the overall best performer of these three was breadth-first-search, followed by uniform-cost-search. This was based on time, for least node expansions, Astar-search-with-h-pg-levelsum expanded significantly less nodes in the more complex problem, this came at a cost however, and its heuristic function took much longer to compute and end up with a solution.

The reasons for the different uninformed search functions performing the way they do are justified with the comparison that is shown in the following Figure from AIMA Section, this shows what their expected/theoretical performance is in relation to some of their properties:



It is evident that the depth-first and depth-limited search types are not expected to complete the search and also not able to find the optimal solution, as was proven in the results that were obtained. All the remaining search functions shown in the image that were used, were able to find optimal solutions, as they were expected to do.

To summarize, this was a thoroughly enjoyable project with many challenges, it did however feel good to overcome them as I learnt a lot about logical statements and the use of them in planning problems. It was also very helpful to be able to compare the different search types against each other to see how they fared in problems of different complexity.

## Optimal Solutions for the Problems:

### Problem 1:

Using greed-best-first-graph-search with h-1:

**Expansions** = 7

**Goal tests** = 9

**Time Elapsed**: 0.0084166s

**Plan Length** = 6

**Plan**: Load(C1, P1, SFO)

Load(C2, P2, JFK)

Fly(P1, SFO, JFK)

Fly(P2, JFK, SFO)

Unload(C1, P1, JFK)

Unload(C2, P2, SFO)

### Problem 2:

Using astar-search with h-ignore-preconditions:

**Expansions** = 1450

**Goal** **tests** = 1452

**Time** **Elapsed**: 4.51015s

**Plan Length** = 9

**Plan**: Load(C3, P3, ATL)

Fly(P3, ATL, SFO)

Unload(C3, P3, SFO)

Load(C2, P2, JFK)

Fly(P2, JFK, SFO)

Unload(C2, P2, SFO)

Load(C1, P1, SFO)

Fly(P1, SFO, JFK)

Unload(C1, P1, JFK)

### Problem 3:

Using breadth-first-search:

**Expansions** = 8602

**Goal** **tests** = 11196

**Time** **Elapsed**: 24.5647s

**Plan Length** = 12

**Plan**: Load(C1, P1, SFO)

Fly(P1, SFO, ATL)

Load(C3, P1, ATL)

Fly(P1, ATL, JFK)

Load(C2, P1, JFK)

Unload(C1, P1, JFK)

Unload(C3, P1, JFK)

Fly(P1, JFK, ORD)

Load(C4, P1, ORD)

Fly(P1, ORD, SFO)

Unload(C2, P1, SFO)

Unload(C4, P1, SFO)