# Planning Search Project – Heuristic Analysis

By: Michael Shum

**Problem 1:**

Results for non-heuristic search:

| **Air_cargo_p1** | **Expansions** | **Goal Tests** | **New Nodes** | **Plan Length** | **Time Elapsed (s)** |
|---|---|---|---|---|---|
| Breadth First Search | 43 | 56 | 180 | 6 | 0.0246 |
| Breadth First Tree Search | 1458 | 1459 | 5960 | 6 | 0.7821 |
| Depth First Graph Search | 21 | 22 | 84 | 20 | 0.0124 |
| Depth Limited Search | 101 | 271 | 414 | 50 | 0.0716 |
| Uniform Cost Search | 55 | 57 | 224 | 6 | 0.0321 |

Breadth first search (BFS) was the best non-heuristic search for Problem 1, using the least number of expansions, goal tests, and time, all while reaching an optimal plan of length 6 (listed below). Depth limited search created the least efficient plan. The reason that BFS was so quick was due to the smaller initial state and small number of planes, airports, and cargo, allowing the algorithm to traverse through each layer of the graph quickly.

Optimal Plan:

Load(C1, P1, SFO)

Load(C2, P2, JFK)

Fly(P2, JFK, SFO)

Unload(C2, P2, SFO)

Fly(P1, SFO, JFK)

Unload(C1, P1, JFK)

Results for heuristic search:

| **Air_cargo_p1** | **Expansions** | **Goal Tests** | **New Nodes** | **Plan Length** | **Time Elapsed (s)** |
|---|---|---|---|---|---|
| Recursive Best First Search | 4229 | 4230 | 17023 | 6 | 2.1787 |
| Greedy Best First Graph Search | 7 | 9 | 28 | 6 | 0.0038 |
| A* Search h_l | 55 | 57 | 224 | 6 | 0.0298 |
| A* Search h_ignore preconditions | 41 | 43 | 170 | 6 | 0.0236 |
| A* Search h_pg_levelsum | 11 | 13 | 50 | 6 | 0.7038 |

All of the heuristic searches were able to come up with an optimal plan requiring a length of 6. Greedy best first graph search was the most efficient by using the least number of expansions, goal tests, and new nodes, and thus, the least time. Recursive best first search is the standout here in terms of taking the longest to run. Greedy best first graph search was the most efficient because it used a heuristic estimate to the goal, and since the plan states are few, the path it took happened to be one with the optimal plan length as well.

Optimal Plan:

Load(C1, P1, SFO)

Fly(P1, SFO, JFK)

Load(C2, P2, JFK)

Fly(P2, JFK, SFO)

Unload(C1, P1, JFK)

Unload(C2, P2, SFO)

Summary:

For Problem 1, the non-heuristic search algorithms were quick, but only BFS and UCS found the optimal plan. In contrast, every heuristic search algorithm found an optimal plan, and in the case of Greedy Best First Graph Search and A* Search, was able to do it in roughly the same time or with less new nodes. Given the reduced number of parameters, every search managed to finish relatively quickly. The fact that A* Search h_ignore_preconditions was able to find the optimal plan and beat BFS in terms of expansions, goal tests, new nodes, and time elapsed shows the power of even a simple heuristic such as calculating the number of remaining goals.

**Problem 2:**

Results for non-heuristic search:

| Air_cargo_p2 | Expansions | Goal Tests | New Nodes | Plan Length | Time Elapsed (s) |
|---|---|---|---|---|---|
| Breadth First Search | 3343 | 4609 | 30509 | 9 | 13.2842 |
| Breadth First Tree Search | Timed out | | | | |
| Depth First Graph Search | 624 | 625 | 5602 | 619 | 3.3026 |
| Depth Limited Search | Timed out | | | | |
| Uniform Cost Search | 4853 | 4855 | 44041 | 9 | 10.587 |

Both BFS and UCS were able to find an optimal plan of length 9, but they both required thousands of expansions, goal tests, and new nodes to do it and over 10 seconds of processing time. DFS was extremely quick, but the plan it produced was incredibly inefficient, requiring 619 steps. BFS was the most efficient in terms of expansions, goal tests, and new nodes since it always considers the shortest path first. UCS would take a bit more time since it keeps going until the goal node is popped off the frontier.

Optimal Plan:

Load(C1, P1, SFO)

Load(C2, P2, JFK)

Load(C3, P3, ATL)

Fly(P2, JFK, SFO)

Unload(C2, P2, SFO)

Fly(P1, SFO, JFK)

Unload(C1, P1, JFK)

Fly(P3, ATL, SFO)

Unload(C3, P3, SFO)

Load(C1, P1, SFO)

Load(C2, P2, JFK)

Fly(P2, JFK, SFO)

Unload(C2, P2, SFO)

Fly(P1, SFO, JFK)

Unload(C1, P1, JFK)

Results for heuristic search:

| **Air_cargo_p2** | **Expansions** | **Goal Tests** | **New Nodes** | **Plan Length** | **Time Elapsed (s)** |
|---|---|---|---|---|---|
| Recursive Best First Search | Timed out | | | | |
| Greedy Best First Graph Search | 998 | 1000 | 8982 | 21 | 2.1619 |
| A* Search h_l | 4853 | 4855 | 44041 | 9 | 10.5428 |
| A* Search h_ignore preconditions | 1450 | 1452 | 13303 | 9 | 3.2503 |
| A* Search h_pg_levelsum | 86 | 88 | 841 | 9 | 146.1222 |

All of the A* heuristic search algorithms were able to find an optimal plan of length 9, while greedy best first graph search was unable to find an optimal plan. Once again, A* Search h_ignore_preconditions was the most efficient in terms of time elapsed out of the A* searches since its heuristic of calculating remaining goal states was an accurate enough guide for the algorithm. On the other hand, A* Search h_pg_levelsum managed to be the most efficient in terms of expansions, goal tests, and new nodes, but took by far the longest time to complete. This is because it was much more careful at each step, searching through multiple levels for goal states.

Optimal Plan:

Load(C1, P1, SFO)

Fly(P1, SFO, JFK)

Load(C2, P2, JFK)

Fly(P2, JFK, SFO)

Load(C3, P3, ATL)

Fly(P3, ATL, SFO)

Unload(C3, P3, SFO)

Unload(C2, P2, SFO)

Unload(C1, P1, JFK)

Summary:

For Problem 2, the heuristic-search algorithms shined with the performance of A* Search h_ignore_preconditions outperforming all its peers in terms of time elapsed, and A* Search h_pg_levelsum outperforming in terms of expansions, goal tests, and new nodes. This is because the planning graph grew by 50% in terms of planes, cargo, and airports, making simple non-heuristic searches less efficient since they have to explore every node.

**Problem 3:**

Results for non-heuristic search:

| **Air_cargo_p3** | **Expansions** | **Goal Tests** | **New Nodes** | **Plan Length** | **Time Elapsed (s)** |
|---|---|---|---|---|---|
| Breadth First Search | 14663 | 18098 | 129631 | 12 | 91.1065 |
| Breadth First Tree Search | Timed out | | | | |
| Depth First Graph Search | 408 | 409 | 3364 | 392 | 1.7518 |
| Depth Limited Search | Timed out | | | | |
| Uniform Cost Search | 18223 | 18225 | 159618 | 12 | 47.2631 |

Both BFS and UCS were able to find an optimal plan of length 12, but they both required thousands of expansions, goal tests, and new nodes to do it and exponentially more time elapsed compared to Problem 2. DFS was extremely quick, but the plan it produced was incredibly inefficient, requiring 392 steps. BFS was the most efficient in terms of expansions, goal tests, and new nodes since it always considers the shortest path first. UCS would take a bit more time since it keeps going until the goal node is popped off the frontier. With the addition of one extra airport and cargo, but one less plane than in Problem 2, the planning graph was larger again so non-heuristic searches took much longer, causing two of them to time out again.

Optimal Plan:

Load(C1, P1, SFO)

Load(C2, P2, JFK)

Fly(P2, JFK, ORD)

Load(C4, P2, ORD)

Fly(P1, SFO, ATL)

Load(C3, P1, ATL)

Fly(P1, ATL, JFK)

Unload(C1, P1, JFK)

Unload(C3, P1, JFK)

Fly(P2, ORD, SFO)

Unload(C2, P2, SFO)

Unload(C4, P2, SFO)

Results for heuristic search:

| Air_cargo_p3 | Expansions | Goal Tests | New Nodes | Plan Length | Time Elapsed (s) |
|---|---|---|---|---|---|
| Recursive Best First Search | Timed out | | | | |
| Greedy Best First Graph Search | 5578 | 5580 | 49150 | 22 | 15.6394 |
| A* Search h_l | 18223 | 18225 | 159618 | 12 | 46.5494 |
| A* Search h_ignore preconditions | 5040 | 5042 | 44944 | 12 | 14.5949 |
| A* Search h_pg_levelsum | Timed out | | | | |

Once again, A* Search h_ignore_preconditions was the most efficient, but this time it was the best in all categories. This is because A* Search h_pg_levelsum timed out due to the expanded planning graph, taking too much time to iterate through each level searching for goal states since there were now 4 goal states to search for. Greedy best first graph search was unable to find an optimal plan.

Optimal Plan:

Load(C1, P1, SFO)

Load(C2, P2, JFK)

Fly(P1, SFO, ATL)

Load(C3, P1, ATL)

Fly(P2, JFK, ORD)

Load(C4, P2, ORD)

Fly(P2, ORD, SFO)

Fly(P1, ATL, JFK)

Unload(C4, P2, SFO)

Unload(C3, P1, JFK)

Unload(C2, P2, SFO)

Unload(C1, P1, JFK)

Summary:

For Problem 3, A* Search h_ignore_preconditions outperformed all its peers since its main competitors all timed out due to the increased number of states in both the initial and goal state. It shows the exponential run-time of planning problems that adding just one more initial and goal state would render many of the algorithms too slow. It also shows that simpler algorithms like BFS and UCS still have a place given that they scale much better.