Sean Batir
AIND - Planning

Heuristic Analysis

0. Optimal Plan for Problem 1, 2, and 3

We define the optimal plan as the algorithm that arrives at the goal with the shortest path length.

	Problem 1	Problem 2	Problem 3
Path Length	6	9	12
Time Elapsed	0.0464265490	16.643273759	117.58682158
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)	Load(C1, P1, SFO) Load(C2, P2, JFK) Load(C3, P3, ATL) Fly(P1, SFO, JFK) Fly(P2, JFK, SFO) Fly(P3, ATL, SFO) Unload(C3, P3, SFO) Unload(C2, P2, SFO) Unload(C1, P1, JFK)	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)

1. Non Heuristic Search Metrics

Breadth First Search Algorithm Analysis

	Problem 1	Problem 2	Problem 3
Path Length	6	9	12
Time Elapsed	0.046426549000898376	16.643273759000294	117.586821582001
Expansions	43	3346	14120
Goal Tests	56	4612	17673
New Nodes	180	30534	124926

Sean Batir
AIND - Planning

Depth First Search Algorithm Analysis

	Problem 1	Problem 2	Problem 3
Path Length	12	1085	660
Time Elapsed	0.010793465000460856	9.589888878996135	4.247559849995014
Expansions	12	1124	677
Goal Tests	13	1125	678
New Nodes	48	10017	5608

Uniform Cost (Non-Heuristic) Search Algorithm Analysis

	Problem 1	Problem 2	Problem 3
Path Length	6	9	12
Time Elapsed	0.04852645599748939	15.063412668008823	62.33303530601552
Expansions	55	4853	18223
Goal Tests	57	4855	18225
New Nodes	24	44041	159618

The algorithm trades off optimality for speed. The trade-off between the suggested path length and the time elapsed is consistent with our understanding the breadth first search and the depth first search algorithm. While the depth first search algorithm searches the tree of possibilities rapidly, it does not arrive at an optimal solution, suggesting path lengths of 12, 1085, and 660 for problems 1,2, and 3 respectively. This is a consequence of the algorithm traversing to the deepest layer of the tree, and only exploring a singular subtree in the process. In contrast, although the breadth first search algorithm takes longer, it reports an optimal path length of 6, 9 and 12 for problems 1,2, and 3 respectively. This is also aligned with the nature of the breadth first search algorithm which essentially explores the entire frontier of all subtrees starting from the root . While run time may suffer from the drastic increase in the number of expansions and and new nodes explored, the algorithm returns an optimal path length. A small improvement to the breadth first search algorithm, the uniform cost search algorithm, produces the best non-heuristic search performance by yielding the optimal path length with a short time elapsed. The simple fix is that instead of expanding the shallowest node, the node with the lowest *path cost g(n)* is expanded. Therefore, this ensures that uniform cost search

expands nodes in their optimal path cost, explaining the resulting optimal path length and reduction in time elapsed.

2. Heuristic Search Metrics

A* Search Algorithm Analysis

	Problem 1	Problem 2	Problem 3
Path Length	6	9	12
Time Elapsed	0.05020688200602308	14.830767720006406	64.43145233998075
Expansions	55	4853	18223
Goal Tests	57	4855	18225
New Nodes	224	44041	159618

A* Ignore Preconditions Search Algorithm Analysis

	Problem 1	Problem 2	Problem 3
Path Length	6	9	12
Time Elapsed	0.04958194699429441	5.136973121989286	21.63100447699253
Expansions	41	1450	5040
Goal Tests	43	1452	5042
New Nodes	170	13303	44944

A* Level Sum Search Algorithm Analysis

	Problem 1	Problem 2	Problem 3
Path Length	6	9	12
Time Elapsed	1.376313813001616	221.07149807699898	1429.6793429190002
Expansions	11	86	316
Goal Tests	13	88	318
New Nodes	50	841	2912

Sean Batir
AIND - Planning

3. Compare and Contrast

We observe that the **ignore preconditions** heuristic resulted in the fastest run-time among all the A* heuristics, with a range of 0.0496 21.6310 seconds. However, the **"level sum"** heuristic resulted in the least number of expansions, goal tests , and new nodes, in contrast to both the ignore preconditions heuristic and the baseline A* search algorithm. All three A* heuristic searches resulted in a path length of 6 for Problem 1, 9 for Problem 2, and 12 for Problem 3.

The "best heuristic" used in these problems can be best described by what the user's definition of "best" entails. If "best" is defined as a fast execution time, then the clearest candidate is **A* ignore preconditions.** However, if the "best heuristic" is defined as minimizing the number of expansions, goal tests, and total new nodes, then the clear winner is the A* level sum search algorithm.

If we consider individual nodes as states, and all possible edges as actions between those states, then the ignore preconditions heuristic would function to increase the number of possible edges and therefore connectivity across all possible states. As a result, the elimination of preconditions and enhanced connectivity between nodes could explain the drastic increase in run time performance to get form the start state to the goal state.

The performance of the **level sum heuristic** is also expected, due to the heuristic's traversal of the planning graph which, although temporally expensive, is ultimately accurate in returning the optimal path. As a brief overview, according to Norvig et al., in Artificial Intelligence, 3rd edition, the level sum heuristic exists as an extension of the subgoal independence assumption. By assuming that individual subgoals are decomposable, and the cost to achieve the optimal goal may be approximated by the sum of all subgoals required to achieve the end goal state. In the context of air cargo loading, the problem space is sufficiently decomposable for the level sum heuristic to return an optimal path.