Heuristic Analysis of Project 3
By Brennon York
On 2017/03/19

1. Provide an optimal plan for Problems 1, 2, and 3.

Problem 1

Plan length: 6
Load(C1, P1, SFO)
Load(C2, P2, JFK)
Fly(P2, JFK, SFO)
Unload(C2, P2, SFO)
Fly(P1, SFO, JFK)
Unload(C1, P1, JFK)

Problem 2

Plan length: 9
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

Plan length: 12
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)

2. Compare and contrast non-heuristic search result metrics (optimality, time elapsed, number of node expansions) for Problems 1,2, and 3. Include breadth-first, depth-first, and at least one other uninformed non-heuristic search in your comparison; Your third choice of non-heuristic

search may be skipped for Problem 3 if it takes longer than 10 minutes to run, but a note in this case should be included.

When comparing breadth-first, depth-first-graph, and uniform-cost search there are some major differences that arise. First, lets show the raw data:

		Breadth First	Depth First Graph	Uniform Cost	A* Ignore Precond.	A* Level Sum
Problem 1	Expansions	43	21	55	41	11
	Goal Tests	56	22	57	43	13
	New Nodes	180	84	224	170	50
	Plan Length	6	20	6	6	6
	Elapsed Time	0.03	0.01	0.03	0.02	1.64
Problem 2	Expansions	3343	624	4853	1506	86
	Goal Tests	4609	625	4855	1508	88
	New Nodes	30509	5602	44041	13820	841
	Plan Length	9	619	9	9	9
	Elapsed Time	14.14	3.86	48.23	26.30	196.37
Problem 3	Expansions	14663	408	18223	5118	DNF
	Goal Tests	18098	409	18225	5120	DNF
	New Nodes	129631	3364	159618	45650	DNF
	Plan Length	12	392	12	12	DNF
	Elapsed Time	123.67	1.65	379.76	77.83	DNF

There are a number of observations from the data that we can make. First, it is shown that both breadth-first and uniform-cost both achieve an optimal plan length for every problem statement while depth-first-graph does not. So far so that for problem two depth-first-graph actually performs worse than in problem three. On the note of the uniqueness of depth-first-graph it presents a near linear correlation between the plan length and the elapsed time to complete a given problem. What is more interesting is that depth-first-graph, while presenting a much larger plan than optimal, always finishes very quickly relative to uniform-cost and breadth-first searches. This can be attributed to the fact that depth-first-graph expands far fewer nodes, up to a whole factor less in problem three, which, while not presenting an optimal solution, returns with **a** solution much quicker. Looking one step deeper one can see that for all problem statements solved by depth-first-graph the number of expansions and total plan length are nearly identical. One last observation is that, when looking for an optimal solution, it seems to be preferred that a breadth-first traversal

of the problem space will be most efficient. When comparing breadth-first against only uniform-cost search breadth-first finishes around three times faster and with fewer node expansions.

3. Compare and contrast heuristic search result metrics using A* with the "ignore preconditions" and "level-sum" heuristics for Problems 1, 2, and 3.

With the raw data for both the “ignore preconditions” and “level sum” heuristics of the A* algorithm presented in the previous graph let us begin our analysis. First we can see that, for problem statement three, the “level sum” heuristic did not finish in the allotted ten minute timeframe. Looking at the comparison between the two algorithms at problem statement two we can see that the “ignore preconditions” heuristic finished greater than one order of magnitude faster than its competition. This is also apparent with the first problem statement. That said, when looking at the number of node expansions the “ignore preconditions” heuristic searches a much wider state space than the “level sum” heuristic. This seems to conclude that, while the “ignore preconditions” heuristic is constantly searching a large number of states, the “level sum” heuristic is working out to determine the next best node to expand. In the end though each heuristic lands on the optimal traversal.

4. What was the best heuristic used in these problems? Was it better than non-heuristic search planning methods for all problems? Why or why not?

The best heuristic was the “ignore preconditions” method. It outperformed all other search implementations, heuristic and non-heuristic, **when searching for the optimal path**. If the optimal path does not matter then the depth-first-graph search outperforms all methods purely from an elapsed time measure. The reason the heuristic searches outperform all other search methods when discussing optimal traversals is through the usage of a constraint. When applying the heuristic (constraint) while traversing a large state space it efficiently allows the machine to apply some level of logic to each choice thus reducing the overall state space. Depending on the heuristic it can have anywhere from minor to drastic effects on the Big O runtime of the original algorithm.