Heuristic Analysis - Planning

Optimal Plans

Problem 1

> Load(C1, P1, SFO)
> Fly(P1, SFO, JFK)
> Unload(C1, P1, JFK)
> Load(C2, P2, JFK)
> Fly(P2, JFK, SFO)
> Unload(C2, P2, SFO)

Problem 2

> 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)

Peoblem 3

> Load(C2, P2, JFK)
> Fly(P2, JFK, ORD)
> Load(C4, P2, ORD)
> Fly(P2, ORD, SFO)
> Unload(C4, P2, SFO)
> Load(C1, P1, SFO)
> Fly(P1, SFO, ATL)
> Load(C3, P1, ATL)
> Fly(P1, ATL, JFK)
> Unload(C3, P1, JFK)
> Unload(C2, P2, SFO)
> Unload(C1, P1, JFK)

Analysis

Non-heuristic Search

1. breadth_first_search

Problem	Plan Length	Time(s)	Expansions	Goal Tests	New Nodes
Problem1	6	0.024	43	56	180
Problem2	9	6.780	3343	4609	30509
Problem3	12	36.035	14663	18098	129631

2. breadth_first_tree_search

Problem	Plan Length	Time(s)	Expansions	Goal Tests	New Nodes
Problem1	6	0.729	1458	1459	5960
Problem2	-	Timeout	-	-	-
Problem3	-	Timeout	-	-	-

   Foever.
3. depth_first_graph_search

Problem	Plan Length	Time(s)	Expansions	Goal Tests	New Nodes
Problem1	20	0.010	21	22	84
Problem2	619	3.324	624	625	5602
Problem3	392	1.666	408	409	3364

4. uniform_cost_search

Problem	Plan Length	Time(s)	Expansions	Goal Tests	New Nodes
Problem1	6	0.029	55	57	224
Problem2	9	10.886	4853	4855	44041
Problem3	12	48.888	18223	18225	159618

Above are four non-heuristic search result metrics for Problems 1,2, and 3. The `depth_first_graph_search` algorithm took the least time to achieve the goal, but it is not admissible becuase the plen lenth of it is too long. Th other algorithms are all admissible, and `breadth\_first\_search` algorithm may be the best choice becuase it have the fewest nodes. I also noticed that the result of `uniform_cost_search` algorithm was the same as `A* with astar_search h_1` algorithm, but I don't know why.
In addition, the time of sovling Problem 3 is longest, and for Problem 1 is shortest. That's because Problem 3 is more complicated than both Problem 1 and 2, and have more actions and literals, which result in more time cost.

A* Heuristic Search

1. astar_search h_1

Problem	Plan Length	Time(s)	Expansions	Goal Tests	New Nodes
Problem1	6	0.030	55	57	224
Problem2	9	10.105	4853	4855	44041

Problem3	12	46.403	18223	18225	159618

2. astar_search h_ignore_preconditions

Problem	Plan Length	Time(s)	Expansions	Goal Tests	New Nodes
Problem1	6	0.029	41	43	170
Problem2	9	3.694	1450	1452	13303
Problem3	12	14.321	5040	5042	44944

3. astar_search h_pg_levelsum

Problem	Plan Length	Time(s)	Expansions	Goal Tests	New Nodes
Problem1	6	0.532	11	13	50
Problem2	9	47.566	86	88	841
Problem3	12	251.519	324	326	2993

Abobe are three A* heuristic search metrics for Problems 1,2, and 3. `astar_search h_ignore_preconditions` is the most powerful and efficient algorithm, which take the least time to achieve the goal. `astar\_search h\_pg\_levelsum` is also a good choice because it significantly reducing the number of search nodes. Comeparing with non-heuristic search algorithms, heuristic search really improve the procedure of searching.

I think the best heuristic used in these problems is `astar_search h_ignore_preconditions`, because it is really very fast and powerful. Although `astar\_search h\_pg\_levelsum` can reduce the number of search nodes, it take too much time on problem 2 and 3. It is better than non-heuristic search planning methods for all problems, because heuristic function make it easier to evaluate the 'cost', so the search procedure can be more efficient.