Heuristic Analysis for Air Cargo Problem

Problem

Provide an optimal plan for Problem 1, 2 and 3. The details of problem are in README.md, which is provided by Udacity, and my code is in "my_air_cargo_problems.py" and "my_planning_graph.py".

In [60]:

```
import numpy as np
import pandas as pd
```

Compare non-heuristic search metrics

I choose 3 search methods for comparing: "breadth_first_search", "depth_first_graph_search" and "uniform_cost_search." The results is below:

In [71]:

```
df_1 = pd.DataFrame({'Expansions':[43,21,55,3343,624,4853,14663,408,18223],
                     'Goal_Tests':[56,22,57,4609,625,4855,18098,409,18225],
                    "New_Nodes":[180,84,224,30509,5602,44041,128605,3364,158174],
                    "Plan_length":[6,20,6,9,619,9,12,392,12],
                    "Time/s":[0.027,0.013,0.033,11.237,2.767,9.701,79.519,1.432,42.641]},
                  index=["problem_1:breadth_first","problem_1:depth_first","problem_1:uniform_co
st",
                         "problem_2:breadth_first","problem_2:depth_first","problem_2:uniform_co
st",
                         "problem_3:breadth_first","problem_3:depth_first","problem_3:uniform_co
st"])
df_1
```

Out[71]:

	Expansions	Goal_Tests	New_Nodes	Plan_length	Time/s
problem_1:breadth_first	43	56	180	6	0.027
problem_1:depth_first	21	22	84	20	0.013
problem_1:uniform_cost	55	57	224	6	0.033
problem_2:breadth_first	3343	4609	30509	9	11.237
problem_2:depth_first	624	625	5602	619	2.767
problem_2:uniform_cost	4853	4855	44041	9	9.701
problem_3:breadth_first	14663	18098	128605	12	79.519
problem_3:depth_first	408	409	3364	392	1.432
problem_3:uniform_cost	18223	18225	158174	12	42.641

What we care about is the plan length and time-consuming of the algorithm. Depth first search is unacceptable, although it is fast, the plan length is not optimal. Breadth first search and Uniform cost search show the same plan length, and uniform cost search is faster than breadth first search.

Compare heuristic search metrics

I use A* search with the "ignore preconditions" and "level-sum" heuristics for Problems 1, 2, and 3. The caculation results is below:

In [72]:

```
df_2 = pd.DataFrame({'Expansions':[41,11,1450,86,5040,403],
                    'Goal_Tests':[43,13,1452,88,5042,405],
                    "New_Nodes":[170,50,13303,841,44769,3703],
                    "Plan_length":[6,6,9,9,12,12],
                    "Time/s":[0.034,0.52,3.43,64.82,13.76,550.44]},
                   index=["problem_1:ignore-preconditions","problem_1:level-sum",
                          "problem_2:ignore preconditions","problem_2:level-sum",
                          "problem_3:ignore preconditions","problem_3:level-sum"])
df_2
```

Out[72]:

	Expansions	Goal_Tests	New_Nodes	Plan_length	Time/s
problem_1:ignore-preconditions	41	43	170	6	0.034
problem_1:level-sum	11	13	50	6	0.520
problem_2:ignore preconditions	1450	1452	13303	9	3.430
problem_2:level-sum	86	88	841	9	64.820
problem_3:ignore preconditions	5040	5042	44769	12	13.760
problem_3:level-sum	403	405	3703	12	550.440

The two heuristics A* search give same plan length for each problem, but “level-sum” is slower than "ignore preconditions", especially “level-sum” spends about 550 seconds to solve Problem3. The "ignore preconditions" A* search is also better than “uniform_cost” search in computation speed. The results show that using suitable heuristics can help us to find the solution faster. All in all, I recommend "ignore preconditions" A* search for air cargo problems.

Optimal plan

I use "ignore preconditions" A* search to compute the optimal plan for Problem 1,2 and3.

Optimal plan of 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)

Optimal plan of Problem 2

Load(C3, P3, ATL)
Fly(P3, ATL, SFO)
Unload(C3, P3, SFO)
Load(C1, P1, SFO)
Fly(P1, SFO, JFK)
Unload(C1, P1, JFK)
Load(C2, P2, JFK)
Fly(P2, JFK, SFO)
Unload(C2, P2, SFO)

Optimal plan of Problem 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)