Optimal Plans

By looking at plan length in run_search.py execution results the optimal plans for the project’s problems is as follows:

Problem 1:

Load(C1, P1, SFO)

Fly(P2, JFK, SFO)

Unload(C2, P2, SFO)

Fly(P1, SFO, JFK)

Unload(C1, P1, JFK)

Problem 2:

Load(C1, P1, SFO)

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)

Problem 3:

Load(C1, P1, SFO)

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)

Problem	Search Algorithm	Heuristic	Expansions	Goal Tests	New Nodes	Plan Length	Elapsed Time
Air Cargo Problem 1	breadth_first_search		43	56	180	6	0.037904897006228566
Air Cargo Problem 1	breadth_first_tree_search		1458	1459	5960	6	1.2023799820017302
Air Cargo Problem 1	depth_first_graph_search		12	13	48	12	0.010533813998335972
Air Cargo Problem 1	depth_limited_search		101	271	414	50	0.11511100200004876
Air Cargo Problem 1	uniform_cost_search		55	57	224	6	0.04769312900316436
Air Cargo Problem 1	recursive_best_first_search	with h_1	4229	4230	17029	6	3.4296419210004387
Air Cargo Problem 1	greedy_best_first_graph_search	with h_1	7	9	28	6	0.006546022996190004
Air Cargo Problem 1	astar_search	with h_1	55	57	224	6	0.04686764800862875
Air Cargo Problem 1	astar_search	with h_ignore_preconditions	41	43	170	6	0.047879542005830444
Air Cargo Problem 1	astar_search	with h_pg_levelsum	39	41	158	6	1.0192712040006882
Air Cargo Problem 2	breadth_first_search		3343	4609	30509	9	8.829687690013088
Air Cargo Problem 2	breadth_first_tree_search						timeout
Air Cargo Problem 2	depth_first_graph_search		582	583	5211	575	3.5501317769958405
Air Cargo Problem 2	depth_limited_search						timeout
Air Cargo Problem 2	uniform_cost_search		4852	4854	44030	9	14.094473118006135
Air Cargo Problem 2	recursive_best_first_search	with h_1					timeout
Air Cargo Problem 2	greedy_best_first_graph_search	with h_1	990	992	8910	15	2.84201829500671
Air Cargo Problem 2	astar_search	with h_1	4852	4854	44030	9	14.189728138997452
Air Cargo Problem 2	astar_search	with h_ignore_preconditions	1450	1452	13303	9	5.286520724999718
Air Cargo Problem 2	astar_search	with h_pg_levelsum	1129	1131	10232	9	311.7123961659963
Air Cargo Problem 3	breadth_first_search		14663	18098	129631	12	39.156475898009376
Air Cargo Problem 3	breadth_first_tree_search						timeout
Air Cargo Problem 3	depth_first_graph_search		627	628	5176	596	2.949835395993432
Air Cargo Problem 3	depth_limited_search						timeout
Air Cargo Problem 3	uniform_cost_search		18235	18237	159716	12	47.79782148900267
Air Cargo Problem 3	recursive_best_first_search	with h_1					timeout
Air Cargo Problem 3	greedy_best_first_graph_search	with h_1	5614	5616	49429	22	15.539999439002713
Air Cargo Problem 3	astar_search	with h_1	18235	18237	159716	12	49.069883903997834
Air Cargo Problem 3	astar_search	with h_ignore_preconditions	5040	5042	44944	12	15.476714656004333
Air Cargo Problem 3	astar_search	with h_pg_levelsum					timeout

Table 1: run_search.py execution results

The tests results show that the best heuristic used was “h_ignore_preconditions”