Heuristic analisys

Following paper contains performance comparisons of various search methods used to solve Cargo Problems 1, 2 and 3, defined in `my_air_cargo_problems.py`. Tests were run on a MacBook Pro Mid 2014 with 4 Intel Core i7 @ 2.2 GHz and 16 GB RAM. Each problem is attempted to be solved by a number of non-heuristic and heuristic search methods. If a search method takes more than 10 minutes to run, it is terminated and no result is shown (marked with -). Statistic tables show number of expansions, number of goal tests, new nodes, length of the plan, and the time elapsed. Time elapsed is shown in seconds.

Problem 1:

Search function	Expansions	Goal Tests	New Nodes	Plan length	Time elapsed
breadth_first_search	**43**	**56**	**180**	**6**	**0.038**
breadth_first_tree_search	**1458**	**1459**	**5960**	**6**	**0.95**
depth_first_graph_search	**21**	**22**	**84**	**20**	**0.015**
depth_limited_search	**101**	**271**	**414**	**50**	**0.091**
uniform_cost_search	**55**	**57**	**224**	**6**	**0.043**
recursive_best_first_search with h_1	**4229**	**4230**	**17023**	**6**	**2.81**
greedy_best_first_graph_search with h_1	**7**	**9**	**28**	**6**	**0.006**
astar_search with h_1	**55**	**57**	**224**	**6**	**0.042**
astar_search with h_ignore_preconditions	**41**	**43**	**170**	**6**	**0.039**
astar_search with h_pg_levelsum	**18**	**20**	**77**	**6**	**0.756**

Optimal solution:

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

Cargo Problem 1 is fairly easy, and almost all search methods found the optimal solution. `greedy_best_first_graph_search with h_1` is by far the most performant, based on all statistics. `astar_search with h_ignore_preconditions` is very close in run time to non-heuristic `breadth_first_search`, beating it on all other statistics.

Problem 2:

Search function	Expansions	Goal Tests	New Nodes	Plan length	Time elapsed
breadth_first_search	3346	4612	30534	9	8.701
breadth_first_tree_search	-	-	-	-	-
depth_first_graph_search	107	108	959	105	0.52
depth_limited_search	-	-	-	-	-
uniform_cost_search	4853	4855	44041	9	12.254
recursive_best_first_search with h_1	-	-	-	-	-
greedy_best_first_graph_search with h_1	998	1000	8982	21	2.446
astar_search with h_1	4853	4855	44041	9	12.246
astar_search with h_ignore_preconditions	1450	1452	13303	9	4.3
astar_search with h_pg_levelsum	1524	1526	14305	9	424.983

Optimal solution:

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)

`breadth_first_tree_search`, `depth_limited_search` and `recursive_best_first_search with h_1` failed to produce a result in 10 minutes run time. `depth_first_graph_search` finished first, by far, but produced a highly inefficient plan, more than 10 times longer then the optimal solution. `greedy_best_first_graph_search with h_1` is second by time elapsed, but also produces a suboptimal plan, albeit a lot better then `depth_first_graph_search`. `astar_search with h_ignore_preconditions` has the best results for Problem 2, producing an optimal plan in the least time, having other statistics pretty low as well.

Problem 3:

Search function	Expansions	Goal Tests	New Nodes	Plan length	Time elapsed
breadth_first_search	14120	17673	124926	12	41.359
breadth_first_tree_search	-	-	-	-	-
depth_first_graph_search	292	293	2388	288	1.142
depth_limited_search	-	-	-	-	-
uniform_cost_search	18223	18225	159618	12	52.76
recursive_best_first_search with h_1	-	-	-	-	-
greedy_best_first_graph_search with h_1	5578	5580	49150	22	16.106
astar_search with h_1	18223	18225	159618	12	53.841
astar_search with h_ignore_preconditions	5040	5042	44944	12	16.679
astar_search with h_pg_levelsum	-	-	-	-	-

Optimal solution:

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)

`breadth_first_tree_search`, `depth_limited_search`, `recursive_best_first_search with h_1` and `astar_search with h_pg_levelsum` failed to produce a result in 10 minutes run time. Once again, `depth_first_graph_search` had the shortest run time, but one again produced a highly inefficient plan. Same as with the Problem 2, `greedy_best_first_graph_search with h_1` had a pretty short run time, but produced a suboptimal solution. And once again, `astar_search with h_ignore_preconditions` produced the optimal plan in the least time.

Conclusion:

For simpler problems, both heuristic and non-heuristic methods mostly find an optimal solution, and have similar run times. As the problems become harder, heuristic methods become much more performant. `depth_first_graph_search` of non-heuristic methods has a very low run time, but generates plans that are very long and very inefficient. `greedy_best_first_graph_search with h_1` of heuristic methods runs fastest of heuristic methods, and produces somewhat suboptimal plans. `astar_search with h_ignore_preconditions` has been shown to produce optimal solutions at lowest run time for harder problems. As the data suggests, solutions should be sought using several different methods, emphasising non-heuristic approaches for simpler problems and heuristic methods for harder ones.