Juliana Yurie de Heer
Project: Planning Search
September 24, 2017

Heuristic Analysis

Part 1 - Planning problems - uninformed planning searches

The results of running the 7 different searches on `air_cargo_p1`, `air_cargo_p2`, and `air_cargo_p3` were compiled in the following spreadsheet: https://docs.google.com/spreadsheets/d/1mi5BjTHmZmh-NkRIr-kW1xVMsQ8Di7z6KeZie0KII14/edit?usp=sharing.

For problem 1, except for depth_first_graph_search and depth_limited_search, they all had optimal results (plan length was 6), and greedy was the most efficient, with 7 expansions, 9 goal tests, 28 new nodes created, and ~0.00566 seconds, versus 4229 expansions, 4230 goal tests, 17029 new nodes created, and ~3.58 seconds from recursive_best_first_search with h_1, the worst performing, and an average of 843.57 expansions, 870.71 goal tests, 3411.86 new nodes created, and ~0.728 seconds. Although depth_first_graph_search had a low running time of ~0.018 seconds, its plan length was, 12, twice as large as the others. As for depth_limited_search, the plan length was 50, the worst of all, and its running time was ~0.128 seconds. From these results, I conclude that for a simple problem such as `air_cargo_p1`, BFS, UCS, or Greedy would all be reasonable choices for search algorithms.

For problem 2, I had to force quit breadth_first_tree_search and recursive_best_first_search with h_1 because they had not finished running after 10 minutes (in fact, I left the former running over night and it still didn't finish, leading me to think that it was on an infinite loop, but I did not investigate further). Both breadth_first_search and uniform_cost_search found optimal solutions (length 9), but BFS performed slightly better, with 3343 expansions, 4609 goal tests, 30509 new nodes created, and ~20.29 seconds, versus UCS's 4853 expansions, 4855 goal tests, 44041 new nodes created, and ~24.83 seconds. The average was 46499 expansions, 412957.6 goal tests, 428572.4 new nodes created, and ~291.22 seconds. The worst performance was from depth_limited_search, which took ~1399.66 seconds to produce a solution of length 50, and the least optimal solution came from depth_first_graph_search, which ran in ~5.72 seconds, but produced a plan of length 575. From these results, I conclude that BFS and UCS remain the best choices.

For problem 3, I had to force quit breadth_first_tree_search, depth_limited_search, and recursive_best_first_search with h_1 after they ran for longer than 10 minutes. Again, both breadth_first_search and uniform_cost_search found optimal solutions (length 12). BFS had ran slower, at ~169.46 seconds, but produced 14663 expansions, 18098 goal tests, and 129631 new nodes, which is slightly lower than UCS's, which had 18222 expansions, 18224 goal tests, 159608 new nodes created, and ran in ~131.93 seconds. The average was 9770.25 expansions, 10630.25 goal tests, 85874.75 new nodes created, and ~85.84 seconds. Again, the least optimal solution came from depth_first_graph_search, which produced a plan of length 596, but was much more efficient in terms of space and time complexity, with ~4.99 seconds

time elapsed, 627 expansions, 628 goal tests, and 5176 new nodes. From these results, I conclude that BFS and UCS remain the best choices if the solution optimality is a priority, but I'd consider DFS is actions are cheap and time/space complexity are important.

Part 2 - Domain-independent heuristics

Air Cargo Problem 1 using astar_search with...

Heuristics:	h_1	h_ignore_preconditions	h_pg_levelsum
Expansions	55	41	11
Goal Tests	57	43	13
New Nodes	224	170	50
Plan length	6	6	6
Time elapsed in seconds	0.052871026971843094	0.041102360002696514	0.5194219419499859
Nodes expanded	Load(C1, P1, SFO) Load(C2, P2, JFK) Fly(P1, SFO, JFK) Fly(P2, JFK, SFO) Unload(C1, P1, JFK) Unload(C2, P2, 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)	Load(C1, P1, SFO) Fly(P1, SFO, JFK) Load(C2, P2, JFK) Fly(P2, JFK, SFO) Unload(C1, P1, JFK) Unload(C2, P2, SFO)

Air Cargo Problem 2 using astar_search with...

Heuristics:	h_1	h_ignore_preconditions	h_pg_levelsum
Expansions	4853	1450	86
Goal Tests	4855	1452	88
New Nodes	44041	13303	841
Plan length	9	9	9
Time elapsed in seconds	23.503614104993176	7.167053882963955	45.14674474200001
Nodes expanded	Load(C1, P1, SFO) Load(C2, P2, JFK) Load(C3, P3, ATL) Fly(P1, SFO, JFK) Fly(P2, JFK, SFO) Fly(P3, ATL, SFO) Unload(C3, P3, SFO) Unload(C2, P2, SFO) Unload(C1, P1, JFK)	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)	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)

Air Cargo Problem 3 using astar_search with...

Heuristics:	h_1	h_ignore_preconditions	h_pg_levelsum
Expansions	18222	5040	316
Goal Tests	18224	5042	318
New Nodes	159608	44944	2912
Plan length	12	12	12
Time elapsed in seconds	114.94243180501508	32.901511483010836	221.7921866599936
Nodes expanded	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)	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)	Load(C2, P2, JFK) Fly(P2, JFK, ORD) Load(C4, P2, ORD) Fly(P2, ORD, SFO) Load(C1, P1, SFO) Fly(P1, SFO, ATL) Load(C3, P1, ATL) Fly(P1, ATL, JFK) Unload(C4, P2, SFO) Unload(C3, P1, JFK) Unload(C2, P2, SFO) Unload(C1, P1, JFK)

As we can see from the above results, the inexistence of a heuristic (ie, h_1) doesn't affect much the results for A* when working with very simple problems, but it quickly becomes a problem when the problem becomes slightly more complex. As for ignore_preconditions and levelsum, it seems that the first tends to have better time complexity, while the former tends to run slower but have better space complexity (lower number of expansions and nodes created).

Part 3: Written Analysis

One possible optimal plan for each of the 3 problems would be:

Optimal plan for Problems 1:	Optimal plan for Problems 2:	Optimal plan for Problems 3:
Load(C1, P1, SFO) Load(C2, P2, JFK) Fly(P1, SFO, JFK) Fly(P2, JFK, SFO) Unload(C1, P1, JFK) Unload(C2, P2, SFO)	Load(C1, P1, SFO) Load(C2, P2, JFK) Load(C3, P3, ATL) Fly(P1, SFO, JFK) Fly(P2, JFK, SFO) Fly(P3, ATL, SFO) Unload(C3, P3, SFO) Unload(C2, P2, SFO) Unload(C1, P1, JFK)	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)

For these problems A* search was comparable to the best performing uninformed search algorithms. Generally speaking, DFS had better time/space complexity than its competitors, but it did not produce optimal plans, whereas levelsum produced optimal results and also had great space complexity, but it was amongst the slowest in time elapsed.