Project Analysis

Zeyu Tao

Mar 19, 2017

Data (first part):

	breadth_first_search	**depth_first_graph_search**
air_cargo_p1	40 (Node Expansion) 56 (Goal Test) 0.12264959400636144s Output: Load(C1, P1, SFO) Load(C2, P2, JFK) Fly(P2, JFK, SFO) Unload(C2, P2, SFO) Fly(P1, SFO, JFK) Unload(C1, P1, JFK) (which is optimal)	21 (Node Expansion) 22 (Goal Test) 0.05040333599026780s Output is of length 20 (which is not optimal)
air_cargo_p2	3343 (Node Expansion) 4609 (Goal Test) 46.63726615400083s Output: Load(C1, P1, SFO) Load(C2, P2, JFK) 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) (which is optimal)	624 (Node Expansion) 625 (Goal Test) 9.691954628004169s Output is of length 619 (which is not optimal)
air_cargo_p3	7320 (Node Expansion) 11132 (Goal Test) 136.91961661000096s Output: Load(C1, P1, SFO) Fly(P2, JFK, ATL) Load(C3, P2, ATL) Fly(P1, SFO, JFK) Unload(C1, P1, JFK) Fly(P2, ATL, ORD) Load(C4, P2, ORD) Fly(P2, ORD, SFO) Unload(C3, P2, SFO) Unload(C4, P2, SFO) (which is optimal)	273 (Node Expansion) 274 (Goal Test) 4.027164797000296s Output is of length 261 (which is not optimal)

Data (second part):

	astar_search h_1	**astar_search h_ignore_preconditions**	**astar_search h_pg_levelsum**
air_cargo_p1	55 (Node Expansion) 57 (Goal Test) 0.1421329010045156s Output is of length 6 (which is optimal)	41 (Node Expansion) 43 (Goal Test) 0.11284507800883148s Output is of length 6 (which is optimal)	11 (Node Expansion) 13 (Goal Test) 1.8732714319921797s Output is of length 6 (which is optimal)
air_cargo_p2	4852 (Node Expansion) 4854 (Goal Test) 98.73611211500247s Output is of length 9 (which is optimal)	1506 (Node Expansion) 1508 (Goal Test) 29.049476678002975s Output is of length 9 (which is optimal)	86 (Node Expansion) 88 (Goal Test) 183.26364141399972s Output is of length 9 (which is optimal)
air_cargo_p3	4852 (Node Expansion) 4854 (Goal Test) 103.87692542899458s Output is of length 10 (which is optimal)	2036 (Node Expansion) 2038 (Goal Test) 45.70321316999616s Output is of length 10 (which is optimal)	189 (Node Expansion) 191 (Goal Test) 568.3686694179924s Output is of length 10 (which is optimal)

Analysis: From the two tables above, we can have these observations:

- Breath-first-search is optimal.
- Depth-first-graph-search is much faster than breath-first-search: it visited less nodes and conducted less goal tests. However, depth-first-graph-search is not optimal.
- A^* algorithm is better than the previous two kind of search: it is optimal and fast (less node expansion goal tests) at the same time.
- Among the A^* heuristics, **h_1** had largest node expansion and goal test number. **h_ignore_preconditions** is better than **h_1** but **h_pg_levelsum** had the minimum number of node expansion and goal tests.
- Among the A^* heuristics, **h_pg_levelsum** had the lowest speed. **h_pg_levelsum** is better than **h_1** but **h_ignore** had the fastest performance.

From these observations, it is clear that using **h_ignore_preconditions** will give better overall performance. **h_1** is not very "focused" on our direction and had a lot unnecessary node expansions. **h_pg_levelsum** managed to decrease node expansion and goal test further compared to **h_ignore_preconditions**. But since it took too much time to solve same question, I think **h_ignore_preconditions** is the best heuristics. One possible reason that why **h_pg_levelsum** is very slow is that it took long time to generate the planning graph, especially in my implementation. So I think after improvements, **h_pg_levelsum** may over-perform **h_ignore_preconditions**. Besides, here we don't have an expensive goal test function, if we do, then **h_pg_levelsum** would be our best choice because it took far less goal tests than the previous two heuristics.

As for the non-heuristic search planning methods, I think they will generally perform poorly compared to A^* with proper heuristics. The reason is that without a heuristic, the program would waste a lot of time search through irrelevant states. However, since it can be totally random for non-heuristic search planning methods to choose its node expansion, it is possible that in certain case (with very small possibility), they will over-perform A^* algorithm.