Report : Implement a Planning Search

By Vijai Narayanan

A planning search agent was developed to solve logistics planning problems for an Air Cargo transport system. Using both informed and uninformed progression search algorithms, optimal plans to solve each problem were identified. Table 1 shows each problem description and a solution of optimal length. Table 2-4 compare the performance of various search algorithms for this planning graph.

No	Description	Solution
1	Init(At(C1, SFO) ^ At(C2, JFK) ^ At(P1, SFO) ^ At(P2, JFK) ^ Cargo(C1) ^ Cargo(C2) ^ Plane(P1) ^ Plane(P2) ^ Airport(JFK) ^ Airport(SFO)) Goal(At(C1, JFK) ^ At(C2,SFO))	Load(C1, P1, SFO) Fly(P1, SFO, JFK) Load(C2, P2, JFK) Fly(P2, JFK, SFO) Unload(C2, P2, SFO) Unload(C1, P1, JFK)
2	Init(At(C1, SFO) ^ At(C2, JFK) ^ At(C3, ATL) ^ At(P1, SFO) ^ At(P2, JFK) ^ At(P3, ATL) ^ Cargo(C1) ^ Cargo(C2) ^ Cargo(C3) ^ Plane(P1) ^ Plane(P2) ^ Plane(P3) ^ Airport(JFK) ^ Airport(SFO)) ^ Airport(ATL)) Goal(At(C1, JFK) ^ At(C2,SFO) ^ At(C3,SFO))	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)
3	Init(At(C1, SFO) ^ At(C2, JFK) ^ At(C3, ATL) ^ At(C4, ORD) ^ At(P1, SFO) ^ At(P2, JFK) ^ Cargo(C1) ^ Cargo(C2) ^ Cargo(C3) ^ Cargo(C4) ^ Plane(P1) ^ Plane(P2) ^ Airport(JFK) ^ Airport(SFO)) ^ Airport(ATL)) ^ Airport(ORD)) Goal(At(C1, JFK) ^ At(C3,JFK) ^ At(C2,SFO) ^ At(C4,SFO))	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)

Table 1 : Summary of problem descriptions and solutions

All ten algorithms for problem 1 returned a solution within a reasonable amount of time (less than five seconds.). Eight returned a plan with the minimum (optimal) length. For this problem, it was interesting to observe that *greedy best first graph search* (GBFS), an uninformed search algorithm, returned the result with the fewest node expansions, goal tests, and new nodes. GBFS makes decisions about node expansions without considering the overall path cost[1]. For a problem with such few initial propositions it could maximize performance in this way and still return an accurate solution.

		Air Cargo Problem 1				
No	**Search Type**	**Expansions**	**Goal Tests**	**New Nodes**	**Plan Length**	**Time**
1	breadth_first_search	43	56	180	6	0.027
2	breadth_first_tree_search	1458	1459	5960	6	0.834
3	depth_first_graph_search	21	22	84	20	0.011
4	depth_limited_search	101	271	414	50	0.078
5	uniform_cost_search	55	57	224	6	0.030
6	recursive_best_first_search with h_1	4229	4230	17023	6	2.321
7	greedy_best_first_graph_search with h_1	7	9	28	6	0.004
8	astar_search with h_1	55	57	224	6	0.030
9	h_ignore_preconditions	41	43	170	6	0.051
10	astar_search with h_pg_levelsum	11	13	50	6	0.445

Table 2 : Algorithm performance for Air Cargo Problem 1

Using the available hardware, it was not possible to return a solution for three of the algorithms for problem 2. In this particular problem, *astar_search with h_pg_levelsum* was the most efficient search algorithm. It had considerably fewer expansions, goal tests, and new nodes than any other. GBFS didn't even return an optimal plan length for this problem. The downside, however, is that this algorithm took quite some time to run (appx 40 seconds). The algorithm *astar_search with h_1* was less efficient but took much less time to execute.

		Air Cargo Problem 2				
No	**Search Type**	**Expansions**	**Goal Tests**	**New Nodes**	**Plan Length**	**Time**
1	breadth_first_search	3343	4609	30509	9	7.088
2	breadth_first_tree_search	Timeout				
3	depth_first_graph_search	624	625	5602	619	2.907
4	depth_limited_search	Timeout				
5	uniform_cost_search	4852	4854	44030	9	14.570
6	recursive_best_first_search with h_1	Timeout				
7	greedy_best_first_graph_search with h_1	990	992	8910	15	2.113
8	astar_search with h_1	4852	4854	44030	9	10.714
9	h_ignore_preconditions	1450	1452	13303	9	4.475
10	astar_search with h_pg_levelsum	86	88	841	9	40.887

Table 3 : Algorithm performance for Air Cargo Problem 2

Similar to problem 2, it was not possible to find a solution for a few algorithms for problem 3. At first glance it appears that *depth_first_search*, an uninformed planning search, was the most efficient. However, the plan yielded by this algorithm is considerably longer (288 steps) than the rest. If real costs were taken into account (fuel costs, labor, etc), this would be unacceptable. Paying the time cost once again, *astar_search with h_pg_levelsum* yields the fewest expansions, goal tests, etc while returning the optimal plan length. Experimentation with the set level heuristic might lead to an even more efficient solution[2].

		Air Cargo Problem 3				
No	**Search Type**	**Expansions**	**Goal Tests**	**New Nodes**	**Plan Length**	**Time**
1	breadth_first_search	14120	17673	124926	12	34.711
2	breadth_first_tree_search	Timeout				
3	depth_first_graph_search	292	293	2388	288	0.977
4	depth_limited_search	Timeout				
5	uniform_cost_search	18223	18225	159618	12	43.435
6	recursive_best_first_search with h_1	Timeout				
7	greedy_best_first_graph_search with h_1	5578	5580	49150	22	13.554
8	astar_search with h_1	18223	18225	159618	12	43.840
9	h_ignore_preconditions	5040	5042	44944	12	15.573
10	astar_search with h_pg_levelsum	316	318	2912	12	203.561

Table 4 : Algorithm performance for Air Cargo Problem 3

Citations

[1] “Best-First search.” Wikipedia, Wikimedia Foundation, 20 Dec. 2017, en.wikipedia.org/wiki/Best-first_search.

[2] Russell, S., and P. Norvig. "Artificial Intelligence: A Modern Approach, 3rd edt. chapter 10." *Essex, UK: Pearson* (2016).