Timothy Ryan – Planning Heuristic Analysis

Optimal Plans:

Problem 1:
Load(C1, P1, SFO)
Load(C2, P2, JFK)
Fly(P2, JFK, SFO)
Unload(C2, P2, SFO)
Fly(P1, SFO, JFK)
Unload(C1, P1, JFK)

Problem 2:
Load(C3, P3, ATL)
Fly(P3, ATL, SFO)
Unload(C3, P3, 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)

Problem 3:
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(C1, P1, JFK)
Unload(C2, P2, SFO)

Breadth-first search was remarkable for always find the best solution. It was also remarkable for the expansive number of expansions it had to complete to get to a solution, in problem 3 numbering over 10,000. It ran much more quickly than some of the other methods, but overall for the sample of methods had about middling operation time. It's clear that if the information size became much larger than that in problem 3, it would not have been able to complete the search. Memory-usage wise, because of the expansive number of expansions, the algorithm would not be feasible for large problems.

Depth-first search always yielded a solution. But it did not guarantee the best solution. It was remarkable for its speed and the relatively sparse number of expansions it had to make to get to a solution and its quickness. It is feasible for large data sets where an solution is necessary but the optimal solution isn't, and where there were memory and time constraints.

Uniform-cost search had very similar results to breath first search (for obvious reasons given the structure of the algorithm) but was always less efficient than breath first search in memory-usage and time. It always came to an optimal solution. Like breadth-first search, it would not be appropriate for problems much more complicated than problem 3.

The level-sum heuristic was remarkable for always coming to the optimal plan and for doing so in very, very few expansions (solving problem 2 in only 88 expansions when others took thousands to come to the same solution!) It was also remarkable for how long it took to do so. The lead the algorithm to be unable to finish Problem 3 within 10 minutes. It was orders of magnitude slower than other methods that came to optimal solutions. Therefore, it can be used in situations where memory is at a limited, but time is not.

The ignore preconditions heuristic was consistently the most time-efficient solution that came up with optimal solutions. It was also fairly memory efficient doing so, having about 1/3 the number of expansions of breadth-first search in complicated problems (though not in exceedingly simple problems like problem 1.)

The best heuristic was clearly the ignore preconditions heuristic, and though it was not as efficient as a simple breadth-first search in very simple problems, its performance was comparable so in most situations it would be best. In situations where algorithmic simplicity was the goal and problems were exceedingly simple, breadth-first search might be best. In situations where algorithmic simplicity was valued, and optimal solutions were not necessary, depth-first search would be practical. In situations where optimal solutions were not necessary, but speed was the absolute goal, greedy best-first search was an absolute speed-demon and could be used.

Planning results organized by problem and search technique								
Problem	S-number	Search technique	Heuristic	Expansions	Goal Tests	New Nodes	Plan Length	Time Elapsed in seconds
problem 1	1	"breadth_first_search"		43	56	180	6	0.072496054
problem 1	2	breadth_first_tree_search'		1458	1459	5960	6	1.154941917
problem 1	3	depth_first_graph_search'		21	22	84	20	0.040319549
problem 1	4	depth_limited_search'		101	271	414	50	0.193151058
problem 1	5	uniform_cost_search'		55	57	224	6	0.076116514
problem 1	6	recursive_best_first	h_1'	4229	4230	17023	6	3.198005085
problem 1	7	greedy_best_first_g	h_1'	7	9	28	6	0.019482523
problem 1	8	astar_search'	h_1'	55	57	224	6	0.086440863
problem 1	9	astar_search'	h_ignore_precon	55	57	224	6	0.081583588
problem 1	10	astar_search'	h_pg_levelsum'	11	13	50	6	1.605484455
problem 2	1	"breadth_first_search"		3343	4609	30509	9	6.196864863
problem 2	2	breadth_first_tree_search'		x	x	x	x	x
problem 2	3	depth_first_graph_search'		624	625	5602	619	7.624194147
problem 2	4	depth_limited_search'		x	x	x	x	x
problem 2	5	uniform_cost_search'		4852	4854	44030	9	22.47644904
problem 2	6	recursive_best_first	h_1'	x	x	x	x	x
problem 2	7	greedy_best_first_g	h_1'	990	992	8910	21	4.870342615
problem 2	8	astar_search'	h_1'	4852	4854	44030	9	24.04302116
problem 2	9	astar_search'	h_ignore_precon	1450	1452	13303	9	8.69741261
problem 2	10	astar_search'	h_pg_levelsum'	86	88	841	9	340.8142909
problem 3	1	"breadth_first_search"		14663	18098	129631	12	84.24685141
problem 3	2	breadth_first_tree_search'		x	x	x	x	x
problem 3	3	depth_first_graph_search'		408	409	3364	392	3.544802587
problem 3	4	depth_limited_search'		x	x	x	x	x
problem 3	5	uniform_cost_search'		18223	18225	159618	12	108.5463537
problem 3	6	recursive_best_first	h_1'	x	x	x	x	x
problem 3	7	greedy_best_first_g	h_1'	5578	5580	49150	22	30.65972572
problem 3	8	astar_search'	h_1'	18223	18225	159618	12	101.4482308
problem 3	9	astar_search'	h_ignore_precon	5040	5042	44944	12	33.40924325
problem 3	10	astar_search'	h_pg_levelsum'	x	x	x	x	x
* x denotes tests that were not able to run in 10 minutes								

Planning results organized by problem and plan length and search time								
Problem	S-number	Search technique	Heuristic	Expansions	Goal Tests	New Nodes	Plan Length	Time Elapsed in seconds
problem 1	**7**	**greedy_best_first_**	**h_1'**	**7**	**9**	**28**	**6**	**0.019482523**
problem 1	1	"breadth_first_search"		43	56	180	6	0.072496054
problem 1	5	uniform_cost_search'		55	57	224	6	0.076116514
problem 1	9	astar_search'	h_ignore_precon	55	57	224	6	0.081583588
problem 1	8	astar_search'	h_1'	55	57	224	6	0.086440863
problem 1	2	breadth_first_tree_search'		1458	1459	5960	6	1.154941917
problem 1	10	astar_search'	h_pg_levelsum'	11	13	50	6	1.605484455
problem 1	6	recursive_best_first	h_1'	4229	4230	17023	6	3.198005085
problem 1	3	depth_first_graph_search'		21	22	84	20	0.040319549
problem 1	4	depth_limited_search'		101	271	414	50	0.193151058
problem 2	**1**	**"breadth_first_search"**		**3343**	**4609**	**30509**	**9**	**6.196864863**
problem 2	9	astar_search'	h_ignore_precon	1450	1452	13303	9	8.69741261
problem 2	5	uniform_cost_search'		4852	4854	44030	9	22.47644904
problem 2	8	astar_search'	h_1'	4852	4854	44030	9	24.04302116
problem 2	10	astar_search'	h_pg_levelsum'	86	88	841	9	340.8142909
problem 2	7	greedy_best_first_g	h_1'	990	992	8910	21	4.870342615
problem 2	3	depth_first_graph_search'		624	625	5602	619	7.624194147
problem 2	2	breadth_first_tree_search'		x	x	x	x	x
problem 2	4	depth_limited_search'		x	x	x	x	x
problem 2	6	recursive_best_first	h_1'	x	x	x	x	x
problem 3	**9**	**astar_search'**	**h_ignore_precor**	**5040**	**5042**	**44944**	**12**	**33.40924325**
problem 3	1	"breadth_first_search"		14663	18098	129631	12	84.24685141
problem 3	8	astar_search'	h_1'	18223	18225	159618	12	101.4482308
problem 3	5	uniform_cost_search'		18223	18225	159618	12	108.5463537
problem 3	7	greedy_best_first_g	h_1'	5578	5580	49150	22	30.65972572
problem 3	3	depth_first_graph_search'		408	409	3364	392	3.544802587
problem 3	2	breadth_first_tree_search'		x	x	x	x	x
problem 3	4	depth_limited_search'		x	x	x	x	x
problem 3	6	recursive_best_first	h_1'	x	x	x	x	x
problem 3	10	astar_search'	h_pg_levelsum'	x	x	x	x	x
* x denotes tests that were not able to run in 10 minutes								