AIND Classical Planning Project

Franz Williams

Experiment Results

Problem 1:

- 2 cargos (*C1*, *C2*), 2 planes (*P1*, *P2*), and 2 airports (*JFK*, *SFO*)
- Initial state: *At(C1, SFO); At(C2, JFK); At(P1, SFO); At(P2, JFK)*
- Goal: *At(C1, JFK); At(C2, SFO)*
- Action count: 20

Search	Heuristic	Plan Length	Expansions	Goal Tests	New Nodes	Time (s)
`breadth_first_search`	-	6	43	56	178	0.0073
`depth_first_graph_search`	-	20	21	22	84	0.0036
`uniform_cost_search`	-	6	60	62	240	0.0121
`greedy_best_first_graph_search`	`h_unmet_goals`	6	7	9	29	0.0018
`greedy_best_first_graph_search`	`h_pg_levelsum`	6	6	8	28	0.2250
`greedy_best_first_graph_search`	`h_pg_maxlevel`	6	6	8	24	0.1546
`greedy_best_first_graph_search`	`h_pg_setlevel`	6	6	8	28	0.6094
`astar_search`	`h_unmet_goals`	6	50	52	206	0.0099
`astar_search`	`h_pg_levelsum`	6	28	30	122	0.5239
`astar_search`	`h_pg_maxlevel`	6	43	45	180	0.5519
`astar_search`	`h_pg_setlevel`	6	33	35	138	1.4153

Problem 2:

- 3 cargos (*C1-C3*), 3 planes (*P1-P3*), and 3 airports (*JFK*, *SFO*, and *ATL*)
- Initial state: *At(C1, SFO); At(C2, JFK); At(C3, ATL); At(P1, SFO); At(P2, JFK); At(P3, ATL)*
- Goal: *At(C1, JFK); At(C2, SFO); At(C3, SFO)*
- Action count: 72

Search	Heuristic	Plan Length	Expansions	Goal Tests	New Nodes	Time (s)
`breadth_first_search`	-	9	3343	4609	30503	2.1669
`depth_first_graph_search`	-	619	624	625	5602	3.3154
`uniform_cost_search`	-	9	5154	5156	46618	3.4485
`greedy_best_first_graph_search`	`h_unmet_goals`	9	17	19	170	0.0197
`greedy_best_first_graph_search`	`h_pg_levelsum`	9	9	11	86	4.5951
`greedy_best_first_graph_search`	`h_pg_maxlevel`	9	27	29	249	7.1148
`greedy_best_first_graph_search`	`h_pg_setlevel`	9	9	11	84	15.3752
`astar_search`	`h_unmet_goals`	9	2467	2469	22522	2.3143
`astar_search`	`h_pg_levelsum`	9	357	359	3426	124.6904
`astar_search`	`h_pg_maxlevel`	9	2887	2889	26594	727.5697
`astar_search`	`h_pg_setlevel`	9	1037	1039	9605	1397.4360

Problem 3:

- 4 cargos, 2 planes, and 4 airports (*JFK, SFO, ATL, ORD*)
- Initial state: *At(C1, SFO); At(C2, JFK); At(C3, ATL); At(C4, ORD); At(P1, SFO); At(P2, JFK)*
- Goal: *At(C1, JFK); At(C2, SFO); At(C3, JFK); At(C4, SFO)*
- Action count: 88

Search	Heuristic	Plan Length	Expansions	Goal Tests	New Nodes	Time (s)
`breadth_first_search`	-	12	14663	18098	129625	11.0925
`greedy_best_first_graph_search`	`h_unmet_goals`	15	25	27	230	0.0386

Search	Heuristic	Plan Length	Expansions	Goal Tests	New Nodes	Time (s)
`greedy_best_first_graph_search`	`h_pg_levelsum`	14	14	16	126	11.0574
`astar_search`	`h_unmet_goals`	12	7388	7390	65711	9.0639
`astar_search`	`h_pg_levelsum`	12	369	371	3403	293.1146

Problem 4:

- 5 cargos, 2 planes, and 4 airports (*JFK, SFO, ATL, ORD*)
- Initial state: *At(C1, SFO); At(C2, JFK); At(C3, ATL); At(C4, ORD); At(C5, ORD); At(P1, SFO); At(P2, JFK)*
- Goal: *At(C1, JFK); At(C2, SFO); At(C3, JFK); At(C4, SFO); At(C5, JFK)*
- Action count: 104

Search	Heuristic	Plan Length	Expansions	Goal Tests	New Nodes	Time (s)
`breadth_first_search`	-	14	99736	114953	944130	151.2396
`greedy_best_first_graph_search`	`h_unmet_goals`	18	29	31	280	0.0623
`greedy_best_first_graph_search`	`h_pg_levelsum`	17	17	19	165	19.3828
`astar_search`	`h_unmet_goals`	14	34330	34332	328509	57.4568
`astar_search`	`h_pg_levelsum`	15	1208	1210	12210	1342.2847

Analysis

When running the 5 algorithms (breadth-first, greedy best-first with `h_unmet_goals` and `h_pg_levelsum`, and A* with `h_unmet_goals` and `h_pg_levelsum`) across each of the 4 problems, the following average number of nodes were expanded: Problem 1 (20 actions): 26.8, Problem 2 (72 actions): 1238.6, Problem 3 (88 actions): 4481.8, and Problem 4 (104 actions): 27064. As the number of actions increases with each problem, the number of node expansions seems to increase either polynomially or exponentially. Similarly, across the 4 problems, the following average run times were observed: Problem 1: 0.15358s, Problem 2: 26.7573s, Problem 3: 64.8734s, Problem 4: 314.085s. Run time seems to increase polynomially with the number of actions in the problem domain. `breadth_first_search` will always be optimal if the path cost is non-decreasing. `astar_search` with `h_unmet_goals` was also optimal in each of the 4 problems. `depth_first_graph_search` was non-optimal in the first two problems, and would most likely be non-optimal in the other two problems if attempted.

Which algorithm or algorithms would be most appropriate for planning in a very restricted domain (i.e., one that has only a few actions) and needs to operate in real time?

In a restricted domain, `greedy_best_first_graph_search` with `h_unmet_goals` would be the most optimal algorithm based on Problem 1's results. `breadth_first_search` is also a potential choice if optimality is a requirement.

Which algorithm or algorithms would be most appropriate for planning in very large domains (e.g., planning delivery routes for all UPS drivers in the U.S. on a given day)

In larger domains, `greedy_best_first_graph_search` with `h_unmet_goals` would likely be a good choice based on run time. If the problem is largely decomposable, `greedy_best_first_graph_search` with `h_pg_levelsum` may also work well.

Which algorithm or algorithms would be most appropriate for planning problems where it is important to find only optimal plans?

If an optimal plan is required, `astar_search` with `h_unmet_goals` should be used. A* will be optimal as long as the heuristic used is admissible. With respect to time, A* had similar performance to, or outperformed the always-optimal `breadth_first_search` when using the `h_unmet_goals` heuristic.