Artificial Intelligence Nanodegree

Building a Forward Planning Agent

In this project we focused on the problem of solving a deterministic logistics planning problem for Air Cargo Transport System. Our search and planning agent efficiently plans a strategy for this process.

Problem Definition

We are given three different types of planning problems that need to be solved using the same action schema.

Action(Fly(p, from,to),
PRECOND: At(p, from) ∧ Plane(p) ∧ Airport(from) ∧ Airport(to) EFFECT: ¬At(p, from) ∧ At(p,to))
Action(Load(c, p, a),
PRECOND: At(c, a) ∧ At(p, a) ∧ Cargo(c) ∧ Plane(p) ∧ Airport(a) EFFECT: ¬ At(c, a) ∧ In(c, p))
Action(Unload(c, p, a),
PRECOND: In(c, p) ∧ At(p, a) ∧ Cargo(c) ∧ Plane(p) ∧ Airport(a) EFFECT: At(c, a) ∧ ¬ In(c, p))

Tables and Charts

Table 1 - Results for all search algorithms applied to Problem 1

problem	algorithm	#Actions	Expansions	Goal Tests	*New Nodes*	*Plan Length*
1. Air Cargo Problem 1	1. breadth_first_search	20	43	56	178	6
1. Air Cargo Problem 1	2. depth_first_graph_search	20	21	22	84	20
1. Air Cargo Problem 1	3. uniform_cost_search	20	60	62	240	6
1. Air Cargo Problem 1	4. greedy_best_first_graph_search h_unmet_goals	20	7	9	29	6
1. Air Cargo Problem 1	5. greedy_best_first_graph_search h_pg_levelsum	20	6	8	28	6
1. Air Cargo Problem 1	6. greedy_best_first_graph_search h_pg_maxlevel	20	6	8	24	6
1. Air Cargo Problem 1	7. greedy_best_first_graph_search h_pg_setlevel	20	6	8	28	6
1. Air Cargo Problem 1	8. astar_search h_unmet_goals	20	50	52	206	6
1. Air Cargo Problem 1	9. astar_search h_pg_levelsum	20	28	30	122	6
1. Air Cargo Problem 1	10. astar_search h_pg_maxlevel	20	43	45	180	6
1. Air Cargo Problem 1	11. astar_search h_pg_setlevel	20	33	35	138	6

Table 2 - Results for all search algorithms applied to Problem 2

problem	algorithm	#Action s	Expansions	Goal Tests	*New Nodes*	*Plan Length*
2. Air Cargo Problem 2	1. breadth_first_search	72	3343	4609	30503	9
2. Air Cargo Problem 2	2. depth_first_graph_search	72	624	625	5602	619
2. Air Cargo Problem 2	3. uniform_cost_search	72	5154	5156	46618	9
2. Air Cargo Problem 2	4. greedy_best_first_graph_search h_unmet_goals	72	17	19	170	9
2. Air Cargo Problem 2	5. greedy_best_first_graph_search h_pg_levelsum	72	9	11	86	9
2. Air Cargo Problem 2	6. greedy_best_first_graph_search h_pg_maxlevel	72	27	29	249	9
2. Air Cargo Problem 2	7. greedy_best_first_graph_search h_pg_setlevel	72	9	11	84	9
2. Air Cargo Problem 2	8. astar_search h_unmet_goals	72	2467	2469	22522	9
2. Air Cargo Problem 2	9. astar_search h_pg_levelsum	72	357	359	3426	9
2. Air Cargo Problem 2	10. astar_search h_pg_maxlevel	72	2887	2889	26594	9
2. Air Cargo Problem 2	11. astar_search h_pg_setlevel	72	1037	1039	9605	9

Table 3 - Results for selected search algorithms applied to Problem 3

problem	algorithm	#Action s	Expansion s	Goal Tests	*New Nodes*	*Plan Length*
3. Air Cargo Problem 3	1. breadth_first_search	88	14663	18098	129625	12
3. Air Cargo Problem 3	4. greedy_best_first_graph_search h_unmet_goals	88	25	27	230	15
3. Air Cargo Problem 3	5. greedy_best_first_graph_search h_pg_levelsum	88	14	16	126	14
3. Air Cargo Problem 3	8. astar_search h_unmet_goals	88	7388	7390	65711	12
3. Air Cargo Problem 3	9. astar_search h_pg_levelsum	88	369	371	3403	12

Table 4 - Results for selected search algorithms applied to Problem 4

problem	algorithm	#Action s	Expansion s	Goal Tests	*New Nodes*	*Plan Length*
4. Air Cargo Problem 4	1. breadth_first_search	104	99736	114953	944130	14
4. Air Cargo Problem 4	4. greedy_best_first_graph_search h_unmet_goals	104	29	31	280	18
4. Air Cargo Problem 4	5. greedy_best_first_graph_search h_pg_levelsum	104	17	19	165	17

4. Air Cargo Problem 4	8. astar_search h_unmet_goals	104	34330	34332	328509	14
4. Air Cargo Problem 4	9. astar_search h_pg_levelsum	104	1208	1210	12210	15

Number of nodes expanded vs Number of Actions

Algorithm	**Action- problem 1**	**Nodes-problem 1**	**Action-problem 2**	**Action-problem 2**
BFS	20	43	72	3343
DFS	20	21	72	624
UCS	20	60	72	5154
GBFS 1	20	7	72	17
GBFS 2	20	6	72	9
GBFS 3	20	6	72	27
GBFS 4	20	6	72	9
A* Search 1	20	50	72	2467
A* Search 2	20	28	72	357
A* Search 3	20	43	72	2887
A* Search 4	20	33	72	1037

Algorithm	**Action-problem 3**	**Nodes-problem 3**	**Action-problem 4**	**Nodes-problem 4**
BFS	88	14663	104	99736
DFS	88	408	104	-
UCS	88	18512	104	113339
GBFS 1	88	25	104	29
GBFS 2	88	14	104	17
A* Search 1	88	7388	104	34330
A* Search 2	88	369	104	1208

Analysis - The amount of nodes expanded increases as the number of actions for the problem increases. The number of nodes expand for each next problem. However the increase is observed to be different depending on search algorithms and heuristics. In case of uninformed search like BFS, UCS, DFS and A* search, the amount of nodes significantly increases as the number of actions increases. However the amount of increase in case of Greedy Best First Search algorithms is slightly slow but it also increases with actions.

Search Time vs Number of Actions

Algorithm	Search Time 1	Actions Problem 1	Search Time 2	Actions Problem 2
BFS	0.00594411	20	1.81303950	72
DFS	0.00303032	20	2.78556195	72
UCS	0.00894590	20	3.08115778	72
GBFS 1	0.00164158	20	0.01767745	72
GBFS 2	0.41193465	20	9.00224264	72
GBFS 3	0.29929283	20	18.50740043	72
GBFS 4	0.50776247	20	12.47610014	72
A* Search 1	0.00890677	20	2.095102238	72
A* Search 2	1.02987356	20	227.82771191	72
A* Search 3	1.05396838	20	1228.34163389	72
A* Search 4	1.21358165	20	1042.03801595	72

Algorithm	Search Time 3	Actions Problem 3	Search Time 4	Actions Problem 4
BFS	8.53046281	88	76.85300303	104
DFS	0.98303380	88	-	104

UCS	11.76837428	88	93..64420320	104
GBFS 1	0.03065162	88	0.04785316	104
GBFS 2	18.51682551	88	33.73382394	104
A* Search 1	6.90584898	88	44.22115365	104
A* Search 2	339.75939135	88	1877.04277142	104

Analysis - Search time also increase as the number of actions increases corresponding to the different problems. The number of actions for each problem increases as the complexity of the problem increases. Run time increases greatly for A* search which is way more than other algorithms like BFS, DFS, UCS, GBFS etc.

Complexity Analysis

Greedy Best First Search 1, 2 uses less memory than Depth first Search. BFS uses less memory than Uniform Cost Search. In case of larger domains A* search uses less memory than BFS and DFS..

Search Time Analysis

Of all the strategies, Breadth first search always finds the optimal path with quite less search time. However as the size of the problem domain increases, Greedy Best first search 1 algorithm outperforms BFS with lesser search time. Depth First Search also needs less time to search however the path planning is not optimal just like Greedy best first search 1.

Questions

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 very restricted domain, such as the one seen in problem 1 (Air Cargo Problem 1), algorithm 4 (greedy_best_first_graph_search h_unmet_goals) was shown to find near-optimal plans within a millisecond. As the number of actions grows, plans grow up to 25% larger than optimal, but elapsed time remained within a few milliseconds.

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)

The answer to this question depends on three factors: (1) how long one would be willing to wait for a plan to be generated, (2) how important it is for plans to be optimized for length and (3) how hard it is to achieve a solution within those constraints, given the size of the domain. The largest domain we have experimented with was problem 4 **(Air Cargo Problem 4)**, where three possible solutions to this question emerged:

Algorithm 4 **(greedy_best_first_graph_search h_unmet_goals)** reached a solution quickest, but generated a plan that was 25% larger than optimal
Algorithm 5 **(greedy_best_first_graph_search h_pg_levelsum)** was second quickest, even though it was 100 times slower, and generated a plan that was 17% larger than optimal ● Algorithm 8 **(astar_search h_unmet_goals)** was third quickest, even though it was 725 time slower, but generated the optimal plan

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

Algorithms 8-11 (astar_search with multiple heuristics) would be the most appropriate in this case, since A* is guaranteed to find only optimal place if we have an admissible heuristic.