Data of Search algorithms for air cargo problems

Cargo 1 and 2 problems

		breadth first search	depth first search	uniform cost search	greedy best first search - h_unmet_goals	greedy best first search - h_pg_levelsum	greedy best first search - h_pg_maxlevel	greedy best first search - h_pg_setlevel	a* search - h_unmet_goals	a* search - h_pg_levelsum	a* search - h_pg_maxlevel	a* search - h_pg_setlevel
		uniformed searches			informed searches							
Air cargo problem 1	Actions	20	20	20	20	20	20	20	20	20	20	20
	Expansions	43	21	60	7	6	6	6	50	28	43	33
	Goal Tests	56	22	62	9	8	8	8	52	30	45	35
	New Nodes	178	84	240	29	28	24	28	206	122	180	138
	Plan length	6	6	6	6	6	6	6	6	6	6	6
	time to complete the plan search (seconds)	0.0029038	0.0015582	0.0045175	0.0016421	0.2509892	0.1796152	0.3881986	0.0050629	0.68455	0.6321882	0.8265409
Air cargo problem 2	Actions	72	72	72	72	72	72	72	72	72	72	72
	Expansions	3343	624	5154	17	9	27	9	2467	357	2887	1037
	Goal Tests	4609	625	5156	19	11	29	11	2469	359	2889	1039
	New Nodes	30503	5602	46618	170	86	249	84	22522	3426	26594	9605
	Plan length	9	619	9	9	9	9	9	9	9	9	9
	time to complete the plan search (seconds)	1.1829658	1.5887878	1.7790036	0.013352	5.8497271	11.6888368	9.29709	1.3056961	183.3792411	1115.435711	993.091443

Cargo 3 and 4 problems

=> Algorithms exclusion for Cargo problem 3 and 4

* To balance between run time and requirements (=at least one uninformed search, two heuristics with greedy best first search, and two heuristics with A*)
the following algorithms are excluded :
- depth first search as it does not find an optimal solution (619 plan length vs optimal solution consisting of 9 for Air Cargo Problem 2)
- greedy best first search (h_pg_setlevel)
- a* search (h_pg_maxlevel)
- a* search (h_pg_setlevel)

		breadth first search	depth first search	uniform cost search	greedy best first search - h_unmet_goals	greedy best first search - h_pg_levelsum	greedy best first search - h_pg_maxlevel	greedy best first search - h_pg_setlevel	a* search - h_unmet_goals	a* search - h_pg_levelsum	a* search - h_pg_maxlevel	a* search - h_pg_setlevel
		uniformed searches			informed searches							
Air cargo problem 3	Actions	88	*skipped	88	88	88	88	*skipped	88	88	*skipped	*skipped
	Expansions	14663	*skipped	18510	25	14	21	*skipped	7388	369	*skipped	*skipped
	Goal Tests	18098	*skipped	18512	27	16	23	*skipped	7390	371	*skipped	*skipped
	New Nodes	129625	*skipped	161936	230	126	195	*skipped	65711	3403	*skipped	*skipped
	Plan length	12	*skipped	12	15	14	13	*skipped	12	12	*skipped	*skipped
	time to complete the plan search (seconds)	5.7938158	*skipped	8.7262622	0.0319108	12.0065057	16.5821047	*skipped	4.8949482	279.9384	*skipped	*skipped
Air cargo problem 4	Actions	104	*skipped	104	104	104	104	*skipped	104	104	*skipped	*skipped
	Expansions	99736	*skipped	113339	29	17	56	*skipped	34330	1208	*skipped	*skipped
	Goal Tests	114953	*skipped	113341	31	19	58	*skipped	34332	1210	*skipped	*skipped
	New Nodes	944130	*skipped	1066413	280	165	580	*skipped	328509	12210	*skipped	*skipped
	Plan length	14	*skipped	14	18	17	17	*skipped	14	15	*skipped	*skipped
	time to complete the plan search (seconds)	61.0138178	*skipped	105.2413447	0.0395886	25.2737602	60.300899	*skipped	37.470326	1349.7638	*skipped	*skipped

Analysis of Search algorithms for air cargo problems

Analysis of Search algorithms for air cargo problems

Analysis: Run time

Run time grew exponentially with number of actions for most algorithms.
(Actions: 20 -> 72 ->88 -> 104)

The sharpest increase in run time had A* search:
- A* search (h_pg_level_sum) 0.68s -> 183.37s ->279s -> 1349s

Uninformed searches (BFS, Uniform Cost Search) ran under 2s for Problem 2, Under 106s for Problem 4 showing exponential grow (although less sharp than for A* search (h_pg_level_sum))
- breadth first search 0.002s ->1.18s -> 5.79 -> 61.01
- uniform cost search 0.004s ->1.17s -> 8.72 -> 105.24

The least runtime increase had Greedy Best-First search (h_unmet_goals). The runtime increase seems to be O (n * log n).
- 0.0016 -> 0.013 -> 0.031-> 0.039

See comparison with A* search using the same heuristics
- A* search (h_unmet_goals) 0.005s -> 1.3s ->4.89s -> 37.47s
- The most likely cause for increased run time is that A* search had to expand many more nodes to get to the goal and due to overhead with maintaining priority queue.

To sum it up, for speed-critical operation Greedy Best-First search (h_unmet_goals) could be here an ideal candidate.

Analysis: Length of the plan

You can see in the table for Deep First Search, that it produces a plan with 619 steps vs optimal plan with 9 steps for Cargo Problem 2. Except that, all algorithms correctly produce an optimal solution.

For Cargo 3, produced plans have between 12-15 steps. For Cargo 4 between 14-18 steps.

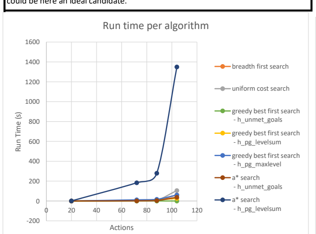
Analysis: Space Complexity (Number of nodes expanded)

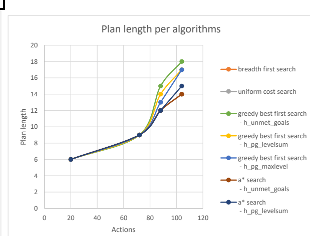
Nodes expanded grew exponentially with number of actions for most algorithms.
(Actions: 20 -> 72 ->88 -> 104)

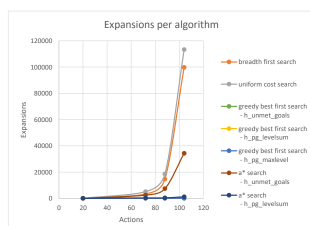
The sharpest exponential increase in a number of nodes expanded had uninformed search algorithms:
- uniform cost search 60 -> 5154 -> 18510 -> 113339
- breath first search 43 -> 3343 -> 14663 -> 99736
- That fits the expectations, as they expand nodes one by one by without any heuristics

A* search shown less step exponential increase than uninformed search algorithms:
- A* search (h_pg_level_sum) new nodes: 28-> 357 -> 369 -> 1208
- A* search (h_unmet_goals) new nodes: 50 -> 2467 -> 7388 -> 34330
- That fits the expectations, as it uses heuristic guiding it to find the goal quicker

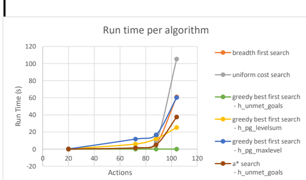
The least increase in number of nodes expanded had Greedy Best-First search (h_unmet_goals). Seems to be lineary increasing.
- 7 -> 17 -> 25 -> 29



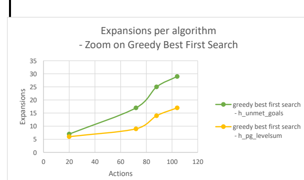




The chart after removing outliers:
- A* search (h_pg_level_sum)



The chart after zooming on Greedy Best First Search



Answer to questions

Question 1: 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?

Answer: In a very restricted domain where a real-time operation is needed and execution speed is important - the ideal choice is Greedy Best First (h_unmet_goals) that was together with Depth-First between the two fastest algorithms for 20 actions. Greedy Best-First search (h_unmet_goals) kept the speed even with increasing numbers of actions and was the fastest algorithm for 72 actions.

Question 2: 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)

Answer: For planning in very large domains (e.g., planning delivery routes for all UPS drivers in the U.S. on a given day) is ideal candidate Greedy Best First (h_unmet_goals), that was one of the fastest and kept the speed with the increasing number of actions.

Question 3: Which algorithm or algorithms would be most appropriate for planning problems where it is important to find only optimal plans?

A* search (h_unmet_goals) will deliver an optimal plan, while keeping reasonable run time (for Cargo Problem 2 with 72 actions it found the solution under 2s)