The goal of this report is to understand the tradeoffs in speed, optimality, and complexity of progression search as problem size increases.

Search Algorithms

Uninformed:

- breadth_first_search (BFS)
- depth_first_graph_search (DFGS)
- uniform_cost_search (UCS)

With Heuristics:

- greedy_best_first_graph_search (GBFGS) with h_unmet_goals
- greedy_best_first_graph_search with h_pg_levelsum
- greedy_best_first_graph_search with h_pg_maxlevel
- greedy_best_first_graph_search with h_pg_setlevel
- astar_search (A*) with h_unmet_goals
- astar_search with h_pg_levelsum
- astar_search with h_pg_maxlevel
- astar_search with h_pg_setlevel

Air Cargo Problems

Problem	# Airplanes	# Cargo Items	# Airports
1	2	2	2
2	3	3	3
3	2	4	4
4	2	5	4

Analysis

Among the Uninformed algorithms:

Problem	Heuristic	Actions	Expansions	Goal Tests	New Nodes	Plan Length	Seconds
1	breadth_first_search	20	43	56	178	6	0,02
1	depth_first_graph_search	20	21	22	84	20	0,00
1	uniform_cost_search	20	60	62	240	6	0,01
2	breadth_first_search	72	3343	4609	30503	9	1,68
2	depth_first_graph_search	72	624	625	5602	619	2,34
2	uniform_cost_search	72	5154	5156	46618	9	2,65
3	breadth_first_search	88	14663	18098	129625	12	8,44
3	uniform_cost_search	88	18510	18512	161936	12	13,31
4	breadth_first_search	104	99736	114953	944130	14	82,12
4	uniform_cost_search	104	113339	113341	1066413	14	92,26

- In Problem 1:
  - BFS & UCS obtained a plan length value of 6, being the optimal one.
  - DFGS found a path faster (although a much higher Plan Length) with a lower number of node expansions as well.
- DFGS doesn't necessarily achieve the optimal value or guaranteed solution, and that's a good reason to exclude it from Problems 3 and 4
- In Problem 2, 3 & 4:
  - BFS & UCS obtained the optimal plan length again.

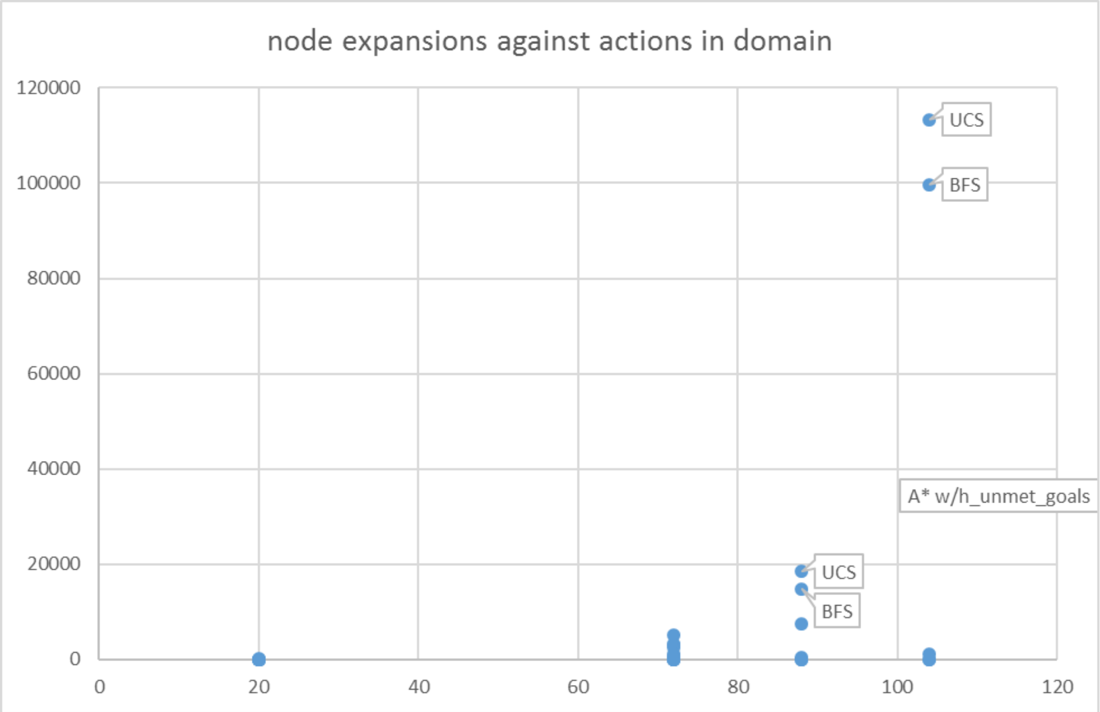
- BFS was the fastest, but time seems to escalate quickly when the solution implies a significant depth

Considering Now the Algorithms with Heuristics:

Problem	Heuristic	Actions	Expansions	Goal Tests	New Nodes	Plan Length	Seconds
1	greedy_best_first_graph_search with h_unmet_goals	20	7	9	29	6	0,00
1	greedy_best_first_graph_search with h_pg_levelsum	20	6	8	28	6	0,29
1	greedy_best_first_graph_search with h_pg_maxlevel	20	6	8	24	6	0,22
1	greedy_best_first_graph_search with h_pg_setlevel	20	6	8	28	6	1,12
1	astar_search with h_unmet_goals	20	50	52	206	6	0,01
1	astar_search with h_pg_levelsum	20	28	30	122	6	0,73
1	astar_search with h_pg_maxlevel	20	43	45	180	6	0,77
1	astar_search with h_pg_setlevel	20	33	35	138	6	2,86
2	greedy_best_first_graph_search with h_unmet_goals	72	17	19	170	9	0,02
2	greedy_best_first_graph_search with h_pg_levelsum	72	9	11	86	9	6,40
2	greedy_best_first_graph_search with h_pg_maxlevel	72	27	29	249	9	13,09
2	greedy_best_first_graph_search with h_pg_setlevel	72	9	11	84	9	31,14
2	astar_search with h_unmet_goals	72	2467	2469	22522	9	1,82
2	astar_search with h_pg_levelsum	72	357	359	3426	9	201,11
2	astar_search with h_pg_maxlevel	72	2887	2889	26594	9	957,38
2	astar_search with h_pg_setlevel	72	1037	1039	9605	9	2.392,57
3	greedy_best_first_graph_search with h_unmet_goals	88	25	27	230	15	0,03
3	greedy_best_first_graph_search with h_pg_levelsum	88	14	16	126	14	14,21
3	astar_search with h_unmet_goals	88	7388	7390	65711	12	6,70
3	astar_search with h_pg_levelsum	88	369	371	3403	12	263,73
4	greedy_best_first_graph_search with h_unmet_goals	104	29	31	280	18	0,07
4	greedy_best_first_graph_search with h_pg_levelsum	104	17	19	165	17	25,79
4	astar_search with h_unmet_goals	104	34330	34332	328509	14	45,96
4	astar_search with h_pg_levelsum	104	1208	1210	12210	15	3.128,92

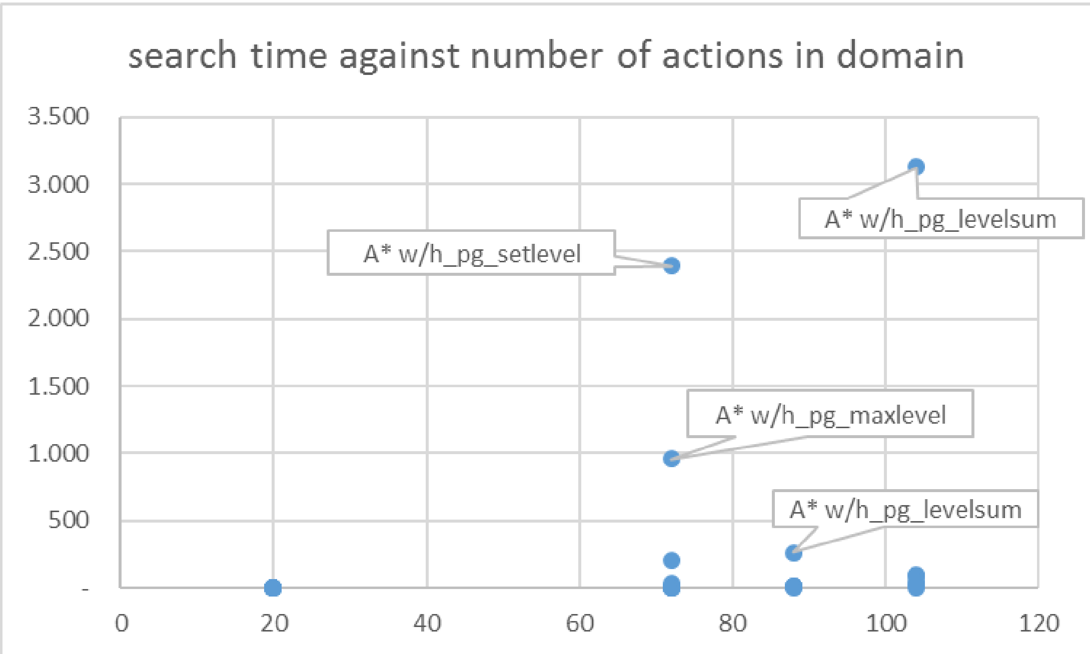
- GBFGS algorithms have shown the lowest node expansion of all algorithms evaluated. From Problems 3 & 4, some cost in optimality is observable. Given this, we could say that informed search gives overall better performance with more complicated graphs.
- If we prioritize preserving optimality, “A* with unmet goals” did it very well, with a lower node expansion and timing than uninformed search algorithms. This is the case when the amount of memory used isn’t an issue.

- Use a table or chart to analyze the number of nodes expanded against number of actions in the domain



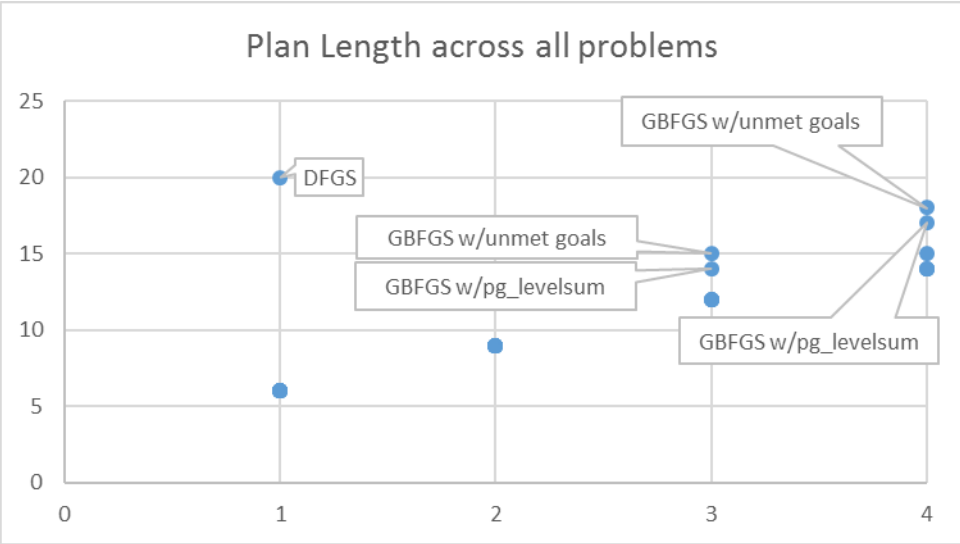
From the chart above, I could infer that uninformed search algorithms are the ones that suffer the more node expansions when the # of actions increases

- Use a table or chart to analyze the search time against the number of actions in the domain



From the chart above, I could infer that selected A* algorithms are the ones with the highest impact in search time when the # of actions increases

- Use a table or chart to analyze the length of the plans returned by each algorithm on all search problems



From the chart above, I could infer that, DFGS excluded for the mentioned reasons, GBFGS algorithms are the ones with highest impact in Plan Length when the complexity of the problem increases.

- **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?**
  Given the results obtained, the best choices would be BFS & UCS.

- **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)**
  Given the results obtained, the best choice would be GBFGS with unmet goals.

- **Which algorithm or algorithms would be most appropriate for planning problems where it is important to find only optimal plans?**
  Given the results obtained, the best choices would be BFS & UCS for any kind of domain (restricted or not restricted)