1. SEARCH COMPLEXITY AS A FUNCTION OF DOMAIN SIZE, SEARCH ALGORITHM, AND HEURISTIC.

In this section, the number of nodes expanded against number of actions in the domain will be analyzed. Charts will be plotted, displaying search values for these two criteria for all four air cargo problems. This will aid the task of tracking growth trends while the problem domain is expanded with each problem.

Table 1.1 illustrates growing complexity in logistical solutions beginning with problem 1, the simplest, and ending with problem 4, most complex. Each consecutive problem brings a number of additional cargos [C1, C2,…], planes[P1, P2,…], airports[ATL, SFO, JFK,…], and goals to the table. Four air cargo problems are formulated in AirCargoProblem class and, in detail, in its air_cargo_p1() - air_cargo_p4() member functions. Air cargo problems 1 and 2 are solved using three uninformed search algorithms, four greedy best-first heuristic searches, and four A-star heuristic search algorithms. Air cargo problems 3 and 4 are attempted by at least one uninformed search, two heuristics with greedy best first search, and two heuristics with A-star.

Table 1.1 Air Cargo Problem 1-4

	Cargos	Planes	Airports	Goals
Problem 1	2	2	2	2
Problem 2	3	3	3	3
Problem 3	4	2	4	4
Problem 4	5	2	4	5

Search algorithm legend that is being used in the charts:

bfs: breadth first search

dfs: depth first graph search

ucs: uniform cost search

gbfs un_g: greedy best first graph search with h_unmet_goals

gbfs lsum: greedy best first graph search with h_pg_levelsum

gbfs maxl: greedy best first graph search with h_pg_maxlevel

gbfs setl: greedy best first graph search with h_pg_setlevel

A* un_g: A-star search with h_unmet_goals

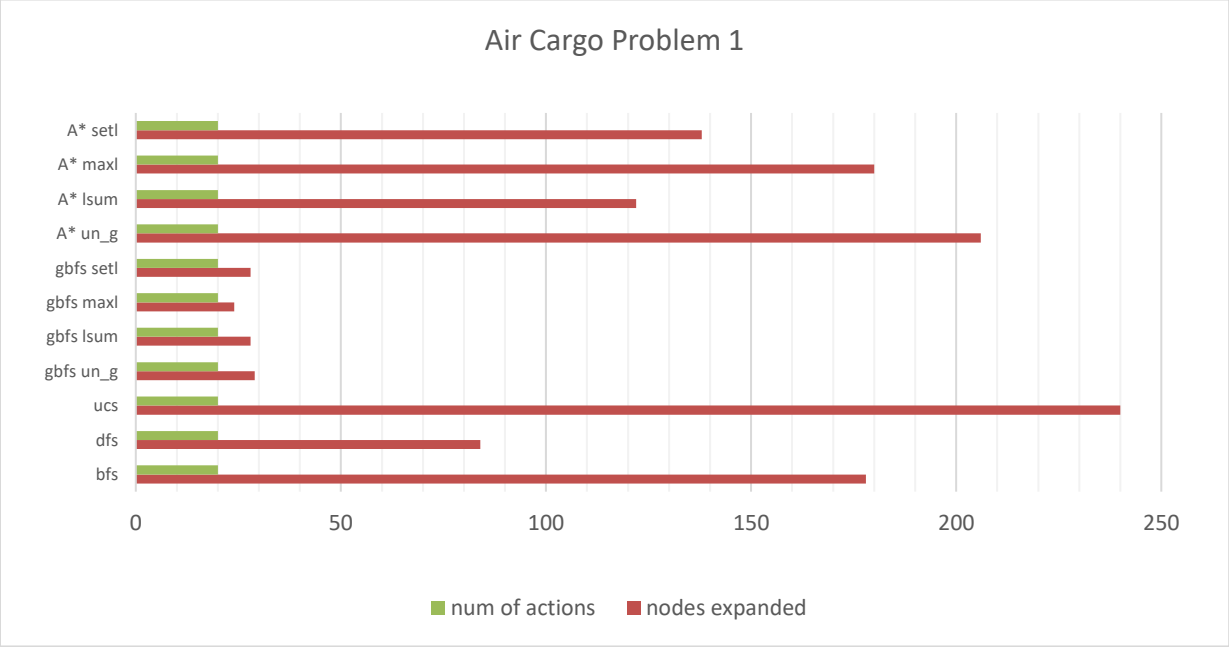
A* lsum: A-star search with h_pg_levelsum

A*maxl: A-star search with h_pg_maxlevel

A* setl: A-star search with h_pg_setlevel

Where h stands for heuristic as pg does for planning graph.

Chart 1.1 Nodes expanded vs. number of actions in problem 1.



Next three charts have been plotted in logarithmic scale to make the two criteria comparable as the number of added nodes supersedes the number of actions by as much as 10^3 or 10^4 in uninformed and A* search results for air cargo problems 3 and 4.

Chart 1.2 Nodes expanded vs. number of actions in problem 2.

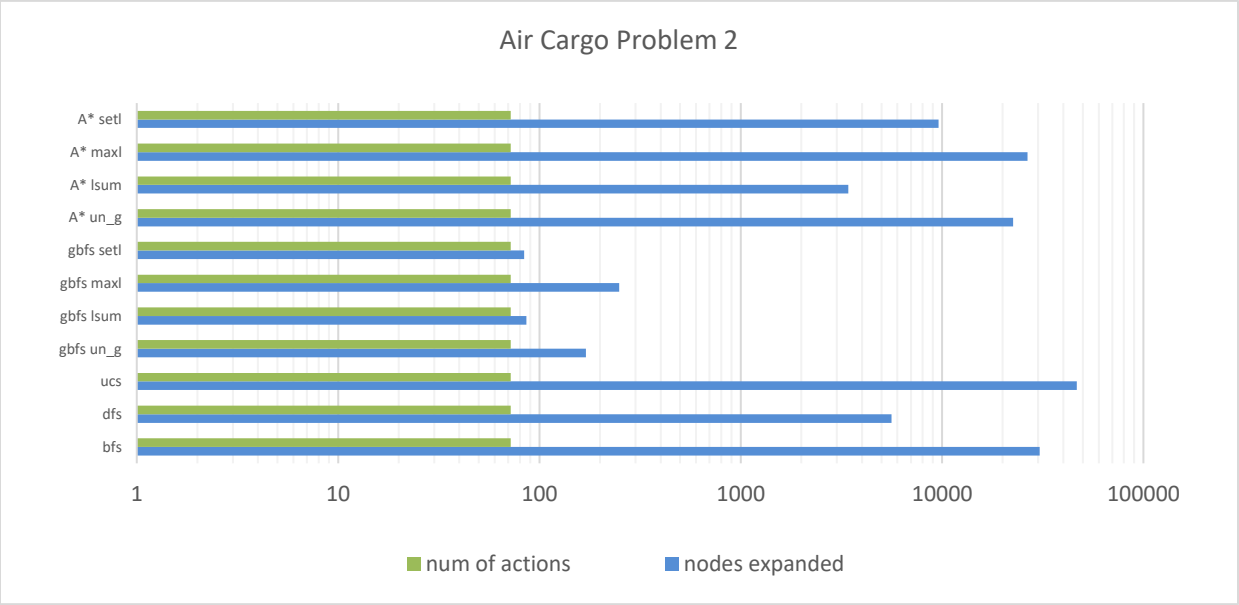


Chart 1.3 Nodes expanded vs. number of actions in problem 3.

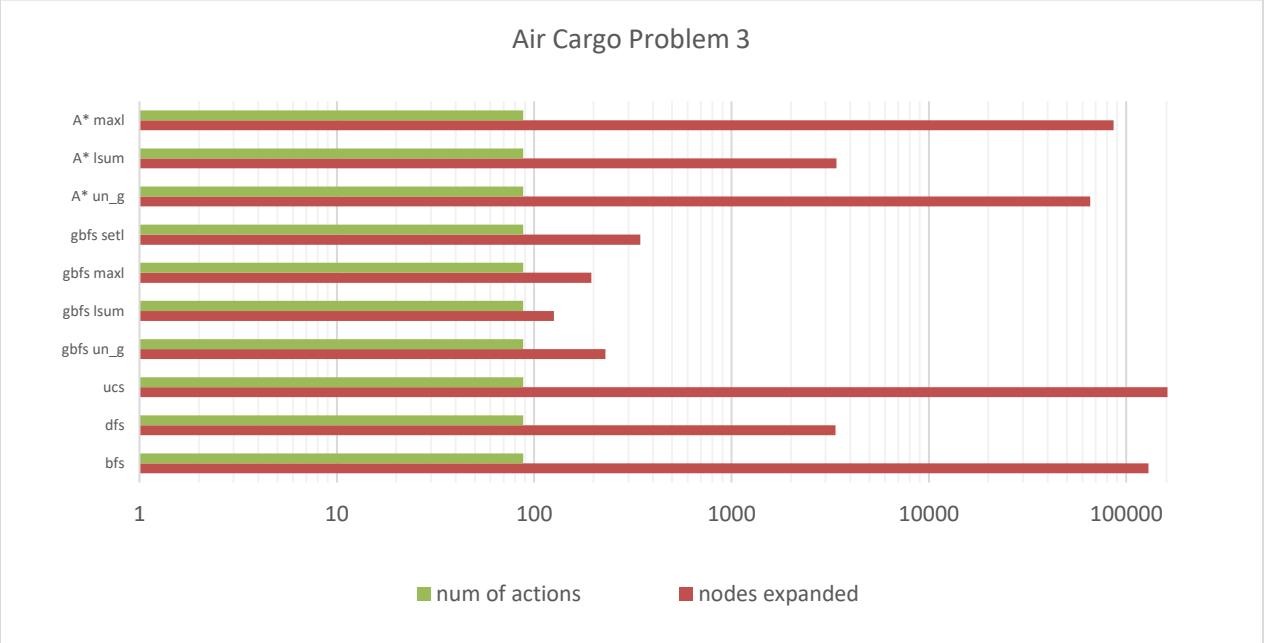


Chart 1.4 Nodes expanded vs. number of actions in problem 4.

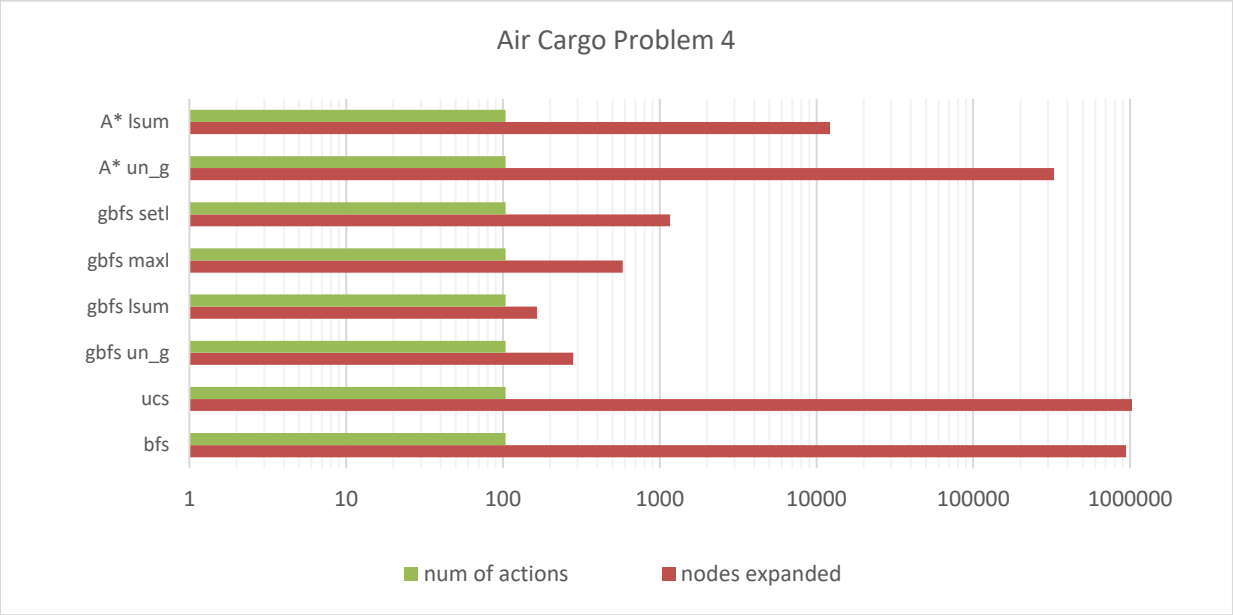
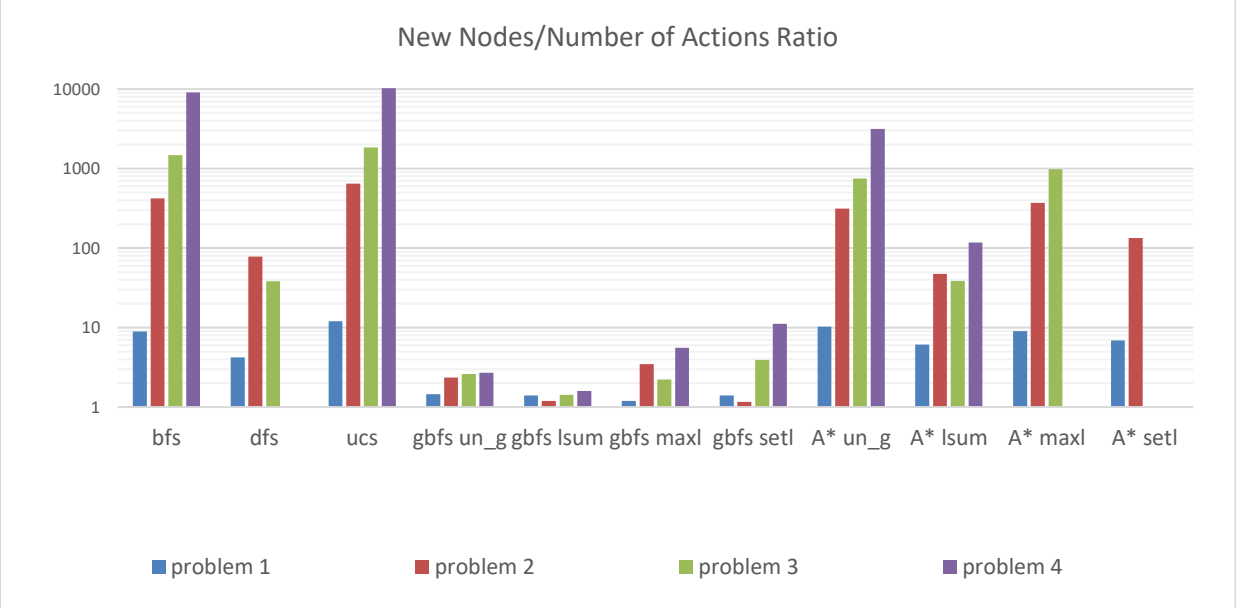


Chart1.5 Number of expanded nodes-number of actions ratios for problem1-4.

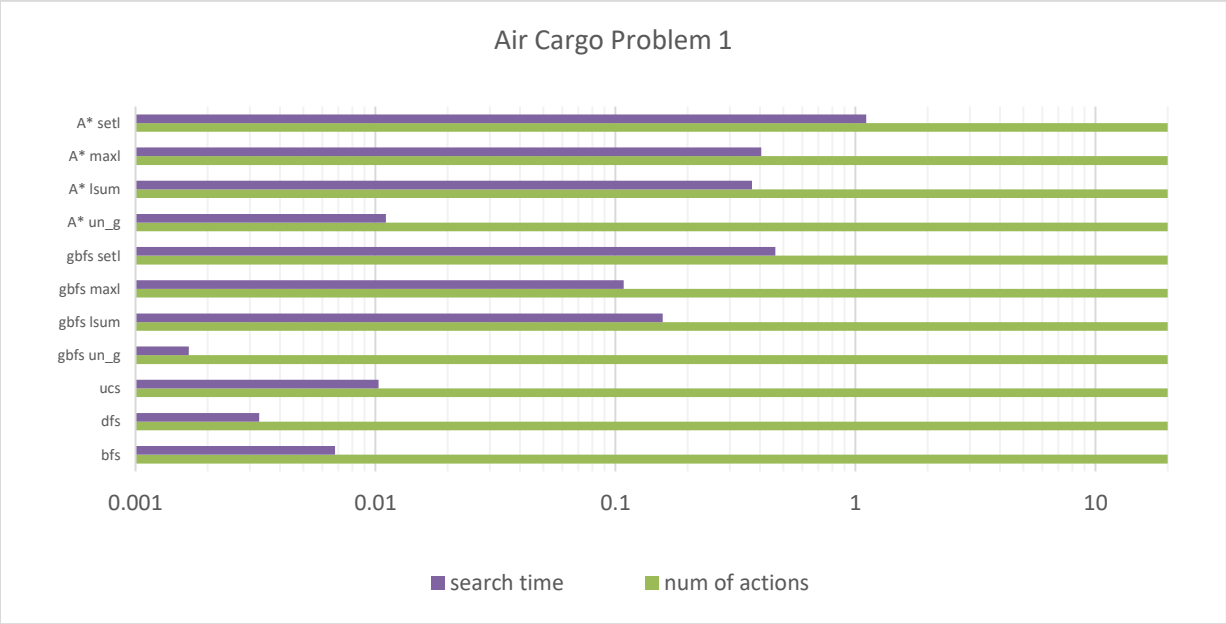


For both uninformed and A* search algorithms, according to the charts, it becomes evident that not only the number of new nodes, but also the ratio of the number of nodes expanded over number of actions,

is increased. The latter goes up from approximately 10 in problem 1 to 10^4 in problem 4. The exception is greedy best first graph searches whose most conservative (number of new nodes over number of actions) ratios are in the 1.2-1.45 range for problem 1, and in 1.58-11.2 range for problem 4. Even in this case, the overall tendency of the number of added nodes is to go up as the domain size is increased from 20 actions in problem 1 to 104 in problem 4.

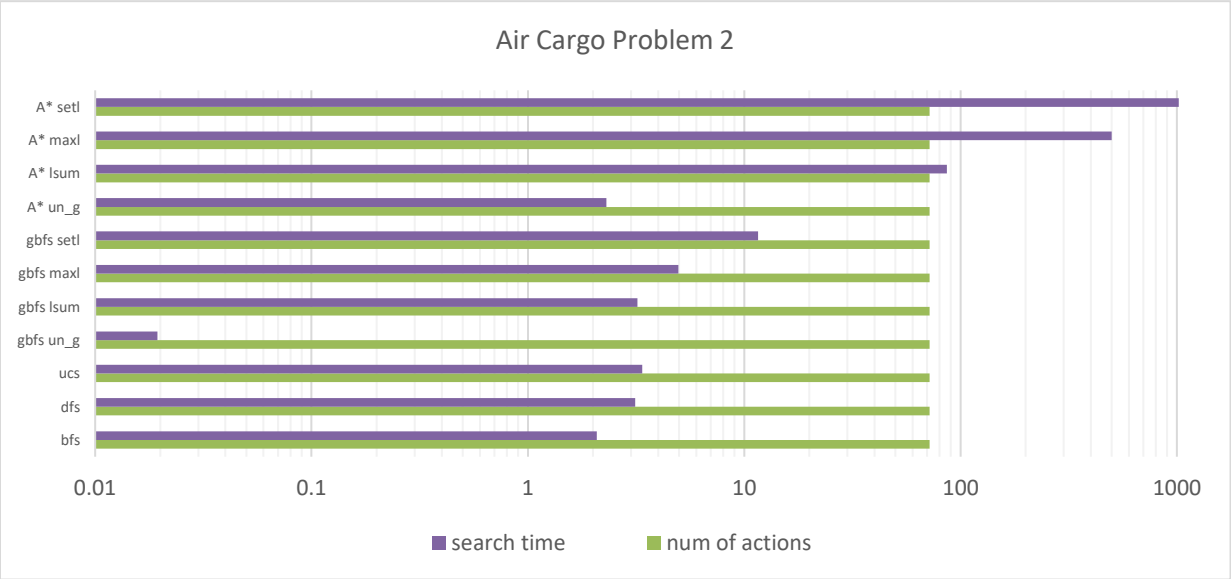
2. SEARCH TIME AS A FUNCTION OF DOMAIN SIZE, SEARCH ALGORITHM, AND HEURISTIC.

Chart 2.1 Search time vs. number of actions in problem 1.



For a small size planning problem, such as air cargo problem 1, whose number of actions does not exceed 20, the planning is calculated very expeditiously. Search time barely exceeds 1 second even for time-consuming algorithms such as astar_search with h_pg_maxlevel or astar_search with h_pg_setlevel.

Chart 2.2 Search time vs. number of actions in problem 2.



As the problem domain increases from 20 actions to 72, search time for all search algorithms rises significantly. Uninformed searches take over 3 seconds long as opposed to 0.01 seconds. Informed searching time rises from 0.011 to 2.3 seconds for A* with unmet goals and from 1.1 to 1020 seconds (17 min) for A* with set-level heuristic.

One searching algorithm that remains virtually unaffected is greedy-best-first-graph-search with unmet goals heuristic. Throughout all four domains, its search time does not exceed 0.06 seconds.

Chart 2.3 Search time vs. number of actions in problem 3.

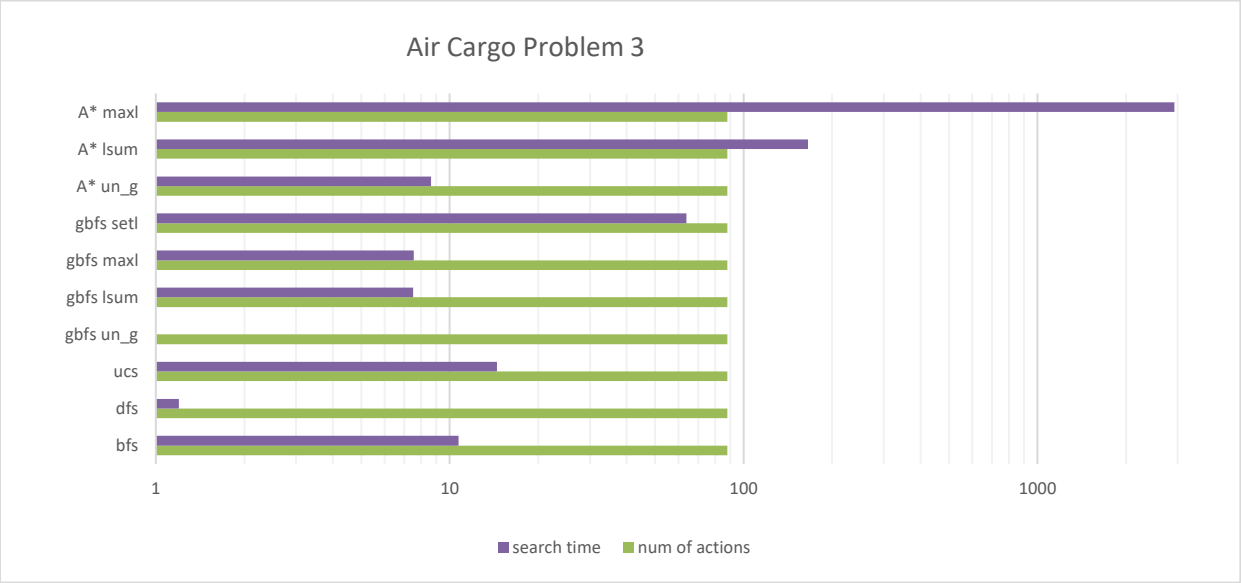
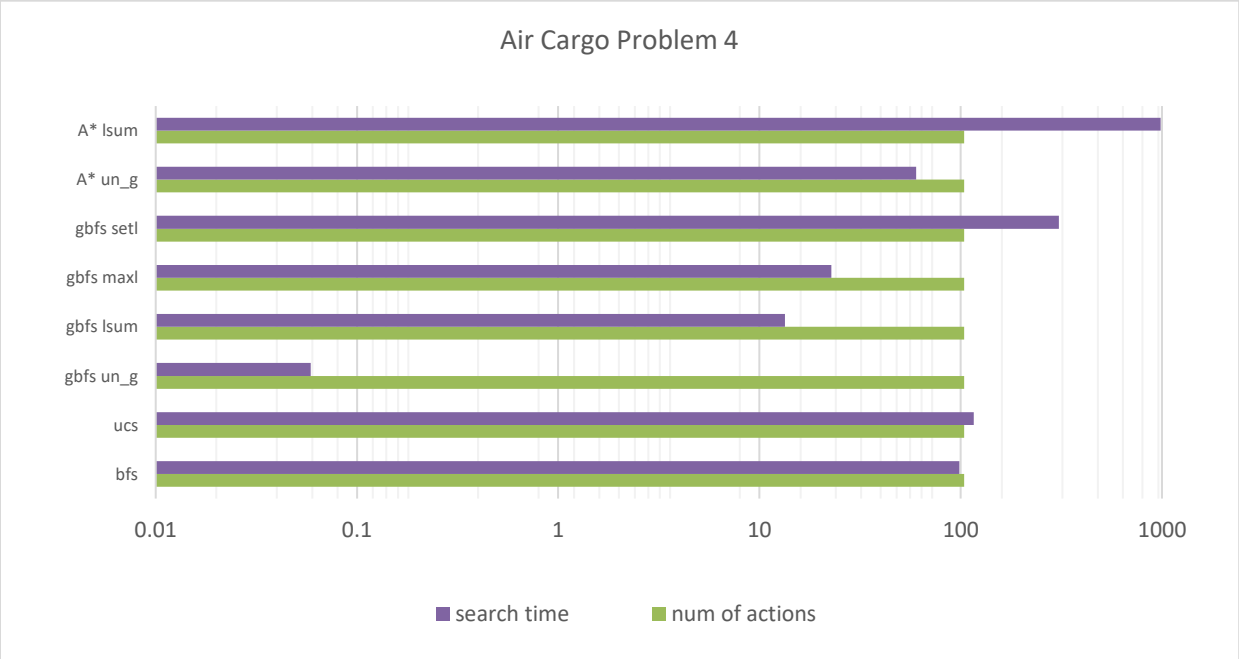


Chart 2.4 Search time vs. number of actions in problem 4.

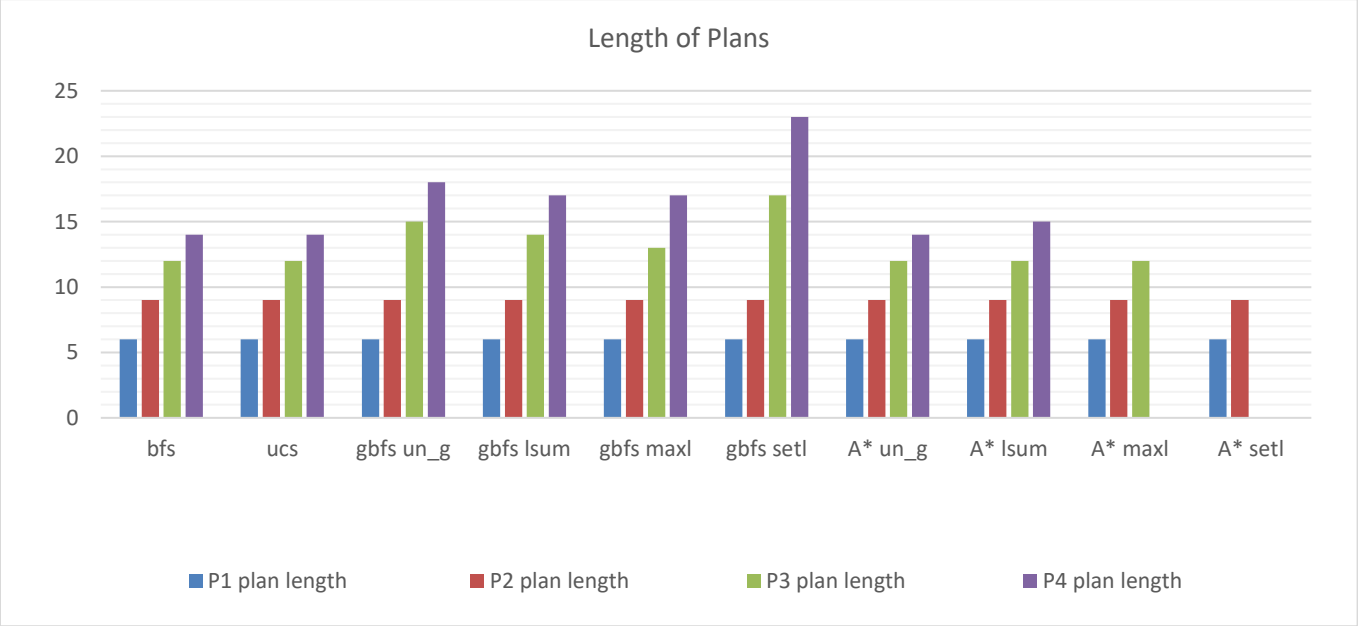


Certain A* searching algorithms were excluded from experiment, due to very long search times required to solve problems 3 and 4. Chart 2.4 lacks data for depth-first-graph search, since it generates a very long plan which is far from optimal. It, however, is calculated much faster in comparison to the rest of search methods. It only took a little over 1 second to come up with a plan in problem 3.

Air cargo problem 4, being the most complex out of all four, yields longest search times for all informed and uninformed algorithms except greedy-best-first-graph search with unmet goals.

3. OPTIMALITY OF SOLUTION AS A FUNCTION OF DOMAIN SIZE, SEARCH ALGORITHM, AND HEURISTIC.

Chart 3.1 Length of Plans from Searches in Problems 1-4 (P1-P4)



Experimental data gathered from across four air cargo problems, as far as length of plan is concerned, indicates that out of all search algorithms most compact air cargo plans were generated by breadth first, uniform cost, and A* with unmet goals searches. However, a short plan may not turn out to be the most optimal. For example, referring to table 3.1, breadth-first-search produced an optimal plan, whereas A*-search with unmet goals did not, because only plane 2 [P2] is employed to make all cargo deliveries.

All search algorithms generated equal-length plans in problems 1 and 2. And only 5 searching methods out of 11 produced optimal plans. They are listed along with corresponding search times in table 3.2.

Table 3.1. Plans formulated in air cargo problem 3.

initial state	goals	breadth_first_search	A*_search with h_unmet_goals
At(C1, SFO)	At(C1, JFK)	Load(C1, P1, SFO)	Load(C2, P2, JFK)
At(C2, JFK)	At(C3, JFK)	Fly(P1, SFO, ATL)	Fly(P2, JFK, ATL)
At(C3, ATL)	At(C2, SFO)	Load(C3, P1, ATL)	Load(C3, P2, ATL)
At(C4, ORD)	At(C4, SFO)	Fly(P1, ATL, JFK)	Fly(P2, ATL, ORD)
At(P1, SFO)		Unload(C1, P1, JFK)	Load(C4, P2, ORD)
At(P2, JFK)		Unload(C3, P1, JFK)	Fly(P2, ORD, SFO)
		Load(C2, P2, JFK)	Unload(C4, P2, SFO)
		Fly(P2, JFK, ORD)	Unload(C2, P2, SFO)
		Load(C4, P2, ORD)	Load(C1, P2, SFO)
		Fly(P2, ORD, SFO)	Fly(P2, SFO, JFK)
		Unload(C2, P2, SFO)	Unload(C3, P2, JFK)
		Unload(C4, P2, SFO)	Unload(C1, P2, JFK)

Table 3.2 Search times (seconds) for optimal plans in air cargo problems 1 and 2.

	bfs	gbfgs unmet_g	gbfgs levelsum	gbfgs maxlevel	gbfgs setlevel
Problem 1	0.00678	0.00166	0.15723	0.10807	0.46406
Problem 2	2.08065	0.0194	3.215098	4.960219	11.6018

4. Q & A

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?

It is very clear from table 3.2 that *greedy-best-first-graph search with unmet_goals* heuristic outperforms the rest of the field in terms of being optimal and fast, therefore it is the most appropriate algorithm in this case.

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)?

Based on plan search results from problems 3 and 4, plans that come close to optimal ones were solved by greedy-best-first-graph search with maxlevel heuristic and A*search with levelsum heuristic.

These plans may be longer than others by a few steps, but they are better planning solutions, because they involve all available planes in the planning. They are not, however, most optimal.

Table 4.1. Search times (seconds) for algorithms with optimal plans from problems 3 and 4.

	gbfg_s levelsum	gbfg_s maxlevel	A* h_levelsum
Problem 3	7.5128	7.5404	165.5894
Problem 4	13.40498	22.7925	983.5021

And to answer the question:

Since search time will be valued over optimality, *greedy-best-first-graph search with pg_levelsum* heuristic would be most appropriate algorithm for planning in very large domains.

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

Table 3.2 lists all searching algorithms with optimal plan solutions for small domains. And these are:

- *breadth_first_search*
- *greedy_best_first_graph_search 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*

To find best algorithm applicable for planning in large domains, table 4.1 lists three algorithms among which *A* search with levelsum* heuristic produces the plan closest to most optimal. And it is most appropriate for situations requiring only optimal planning.