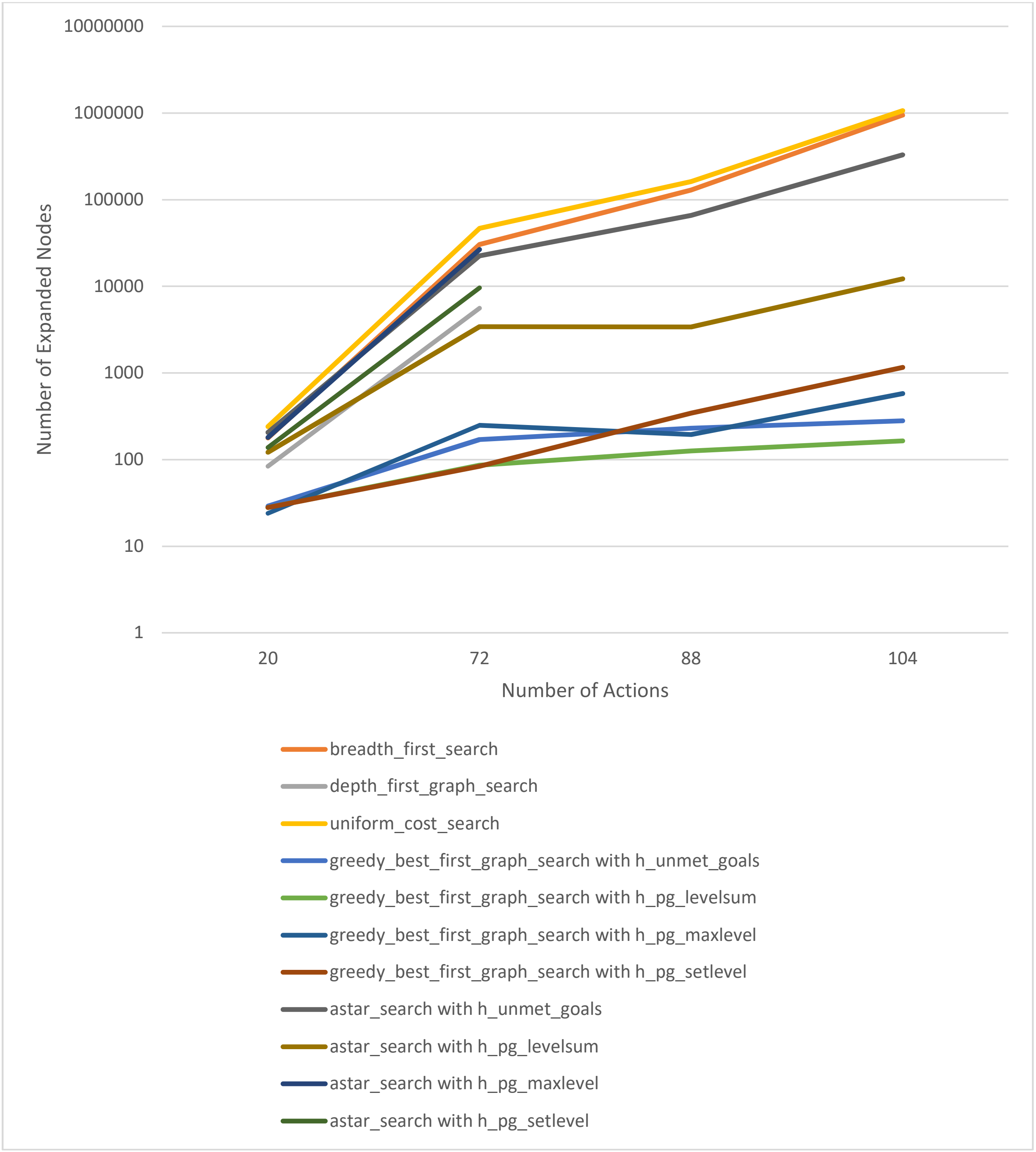
# Classical Planning Project Report

## Introduction

The objective of this report is to analyze various search algorithms used in the planning graph problem in terms of search time, number of expanded nodes and the optimally of the solution as a function of domain size. The algorithms is ran on 4 different variants of the air cargo problem.

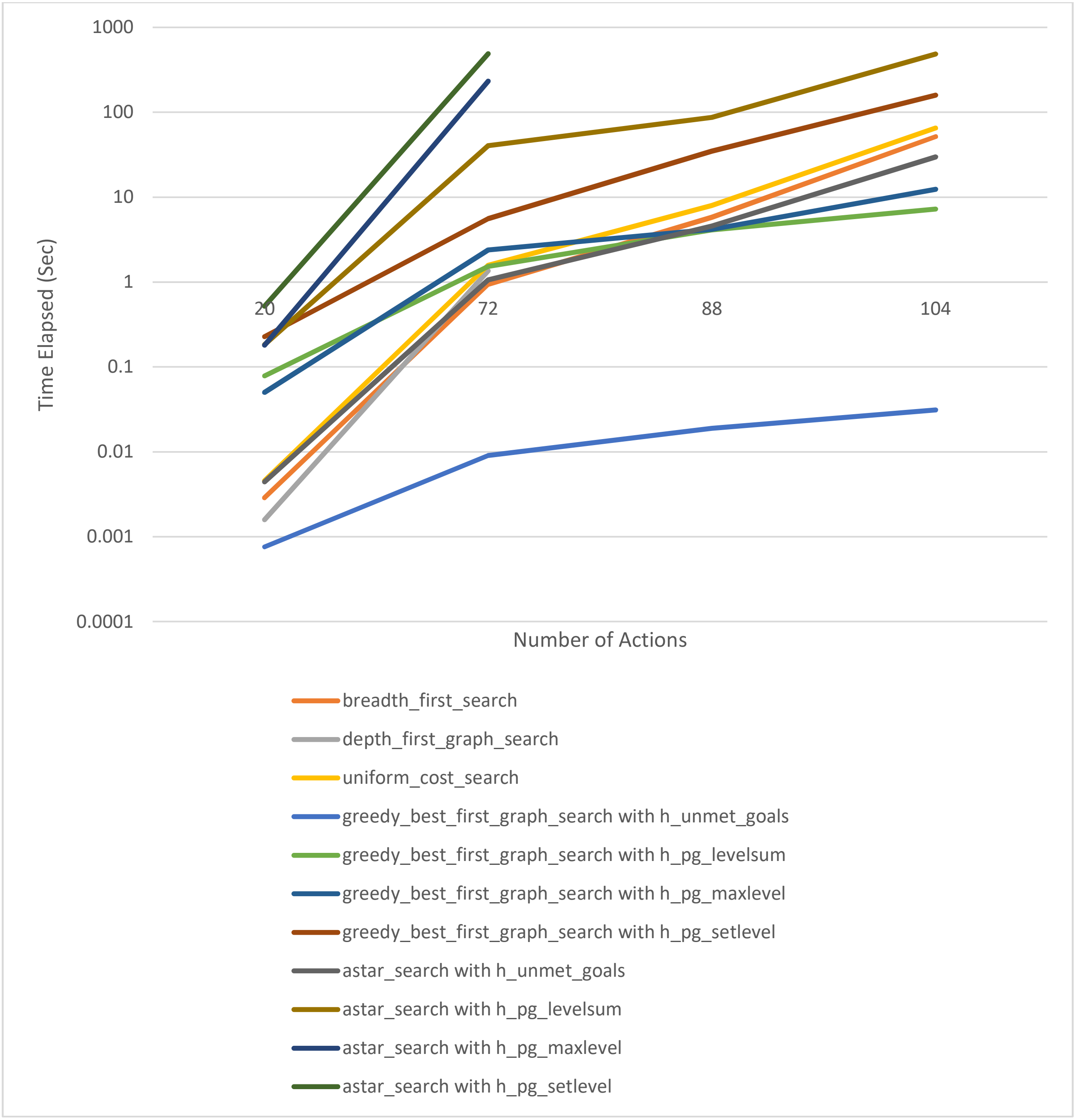
*depth_first_graph_search, astar_search with h_pg_maxlevel and astar_search with h_pg_setlevel were excluded from running in problems 3 and 4 as they were expected to take too much time to finish.*

## Number of Expanded Nodes vs Number of Actions



The graph shown above shows number of actions against number of expanded nodes of different algorithms. Problems 1, 2, 3 and 4 have 20, 72, 88 and 104 number of actions, respectively. The trend is very clear here. As the number of possible actions in the problem increases, the number of expanded nodes increases. Uniform cost search is having the maximum number of expanded nodes in all the problems. Breadth first search almost have the same number of expanded nodes as uniform cost search. This should be taken into consideration if there are memory constraints. Greedy best search is having a low number of expanded nodes in all of the problems. Level sum heuristics seems that it decreases the number of expanded nodes quite well as it has the minimum value across all the other heuristics in both A* and greedy best first search. Although all actions have the same cost in this problem, uniform cost search expanded more nodes than breadth first search as they processed the nodes in different order. The reason for this is that priority queue does not guarantee “popping” the items in the some order as it is “pushed”.

## Search Time vs Number of Actions



The graph shown above shows number of actions against the search time of different algorithms. Here, as for the number of expanded nodes, the search time increases as the number of possible actions in the problem increases. *greedy_best_first_graph_search with h_unmet_goals* had minimum search time in all the problems. In general, Unmet goals heuristic seems to be very good in decreasing the elapsed time. A* took much time more than greedy best first as it expands more nodes. Uniform cost search took slightly more time than breadth first search because of its priority queue overhead and it processes the nodes in a different order.

## Length of Plans

| **Search Method** | **Problem 1** | **Problem 2** | **Problem 3** | **Problem 4** |
|---|---|---|---|---|
| breadth_first_search | 6 | 9 | 12 | 14 |
| depth_first_graph_search | 20 | 619 | N/A | N/A |
| uniform_cost_search | 6 | 9 | 12 | 14 |
| greedy_best_first_graph_search with h_unmet_goals | 6 | 9 | 15 | 18 |
| greedy_best_first_graph_search with h_pg_levelsum | 6 | 9 | 14 | 17 |
| greedy_best_first_graph_search with h_pg_maxlevel | 6 | 9 | 13 | 17 |
| greedy_best_first_graph_search with h_pg_setlevel | 6 | 9 | 17 | 23 |
| astar_search with h_unmet_goals | 6 | 9 | 12 | 14 |
| astar_search with h_pg_levelsum | 6 | 9 | 12 | 15 |
| astar_search with h_pg_maxlevel | 6 | 9 | N/A | N/A |
| astar_search with h_pg_setlevel | 6 | 9 | N/A | N/A |

The table shown above shows length of the plan extracted for different algorithms. *breadth_first_search*, *uniform_cost_search* and *astar_search with h_unmet_goals* always got the minimum path length in all the problems which is expected as they all optimal and has admissible heuristics (for informed ones). Although, A* search being optimal, astar_search with h_pg_levelsum did not get the minimum plan length in Problem 4 as its heuristic is not admissible. However, the difference is only one step. Greedy best first search is not optimal by definition that is why it did not get the best plan in problems 1 and 2. *greedy_best_first_graph_search with h_pg_maxlevel* was closest to the shortest plan length.

## Answers to 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?**

greedy_best_first_graph_search with h_unmet_goals,  breadth_first_search and depth_first_graph_search should be appropriate as they took short execution time in problems 1 and 2 which have low number of actions.

depth_first_graph_search, although not being optimal, it should get a solution in a short time.

**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)**

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_unmet_goals as all of them took considerably short execution time in problems 3 and 4.

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

breadth_first_search, uniform_cost_search (if actions have different costs), astar_search with h_unmet_goals, astar_search with h_pg_maxlevel and astar_search with h_pg_setlevel.

Greedy best first and depth first search are not appropriate as they are both do not optimal plans by definition.

astar_search with h_pg_levelsum is not appropriate as level sum heuristic is not admissible.