AI Nanodegree Project 3: Planning

Written Analysis

Optimal plan

Problem 1

Load(C1, P1, SFO)
Load(C2, P2, JFK)
Fly(P1, SFO, JFK)
Fly(P2, JFK, SFO)
Unload(C1, P1, JFK)
Unload(C2, P2, SFO)

Problem 2

Load(C1, P1, SFO)
Load(C2, P2, JFK)
Fly(P2, JFK, ATL)
Load(C3, P2, ATL)
Fly(P1, SFO, JFK)
Fly(P2, ATL, SFO)
Unload(C1, P1, JFK)
Unload(C2, P2, SFO)
Unload(C3, P2, SFO)

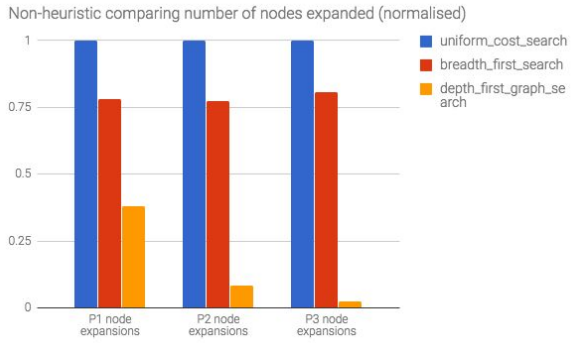
Problem 3

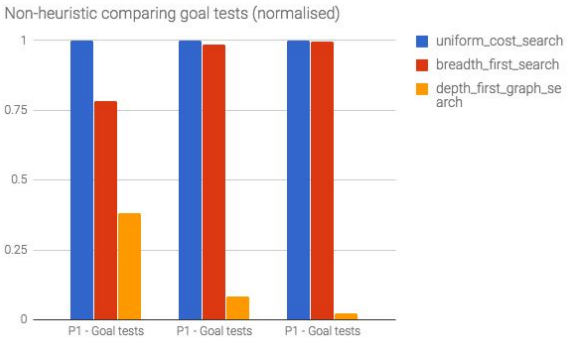
Load(C1, P1, SFO)
Load(C2, P2, JFK)
Fly(P1, SFO, ATL)
Load(C3, P1, ATL)
Fly(P2, JFK, ORD)
Load(C4, P2, ORD)
Fly(P2, ORD, SFO)

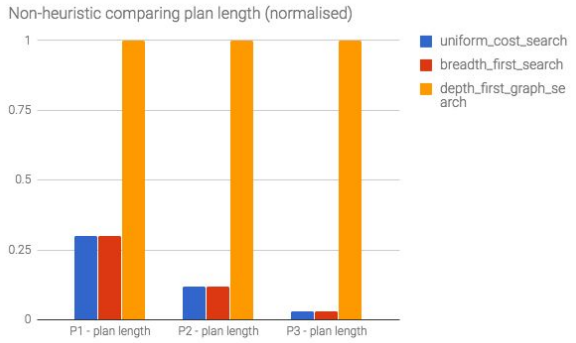
Fly(P1, ATL, JFK)
Unload(C1, P1, JFK)
Unload(C2, P2, SFO)
Unload(C3, P1, JFK)
Unload(C4, P2, SFO)

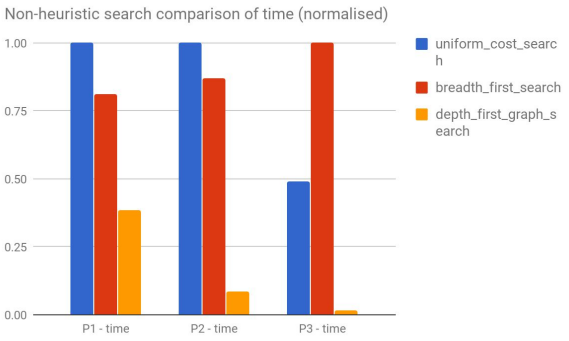
A comparison between non-heuristic search results

The following plots show a comparison between **uniform cost search**, **breadth first search**, and **depth first graph search** comparing **node expansion**, **goal tests**, **plan length** and **time** for each problem.







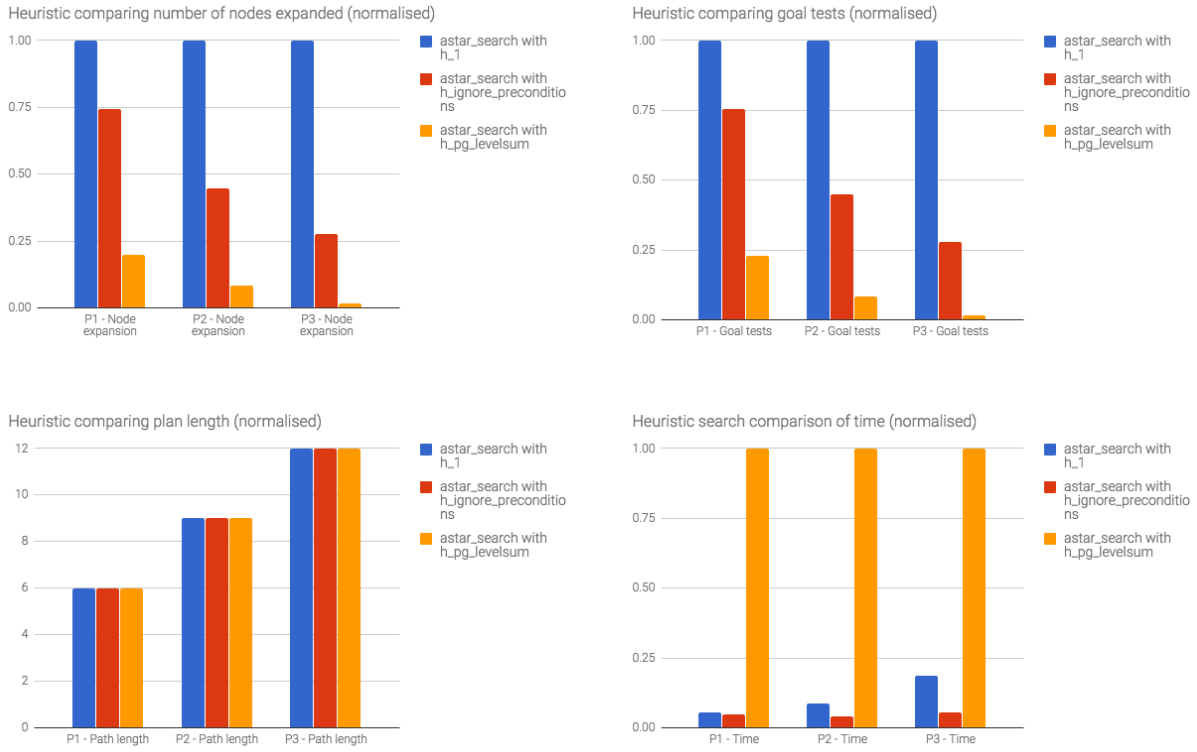


As highlighted in the plots above; **depth first** outperforms the other search approaches in terms of time but performs poorly finding the optimum path as highlighted in the **path length** plot, which makes sense as depth first will terminate its search as soon as the goal is reached while the other searches evaluate nodes per level/depth, therefore being able to discover the optimum path before terminating prematurely. The others, excluding depth first, find the optimum plan therefore the comparison is made on the time taken between **uniform cost** and **breadth first**

search. Interestingly, **breadth first search** outperforms **uniform cost search** for problems 1 and 2 but this is switched for problem 3 which suggests **uniform cost** favors complexity more than a **breadth first search**.

A comparison between heuristic search results

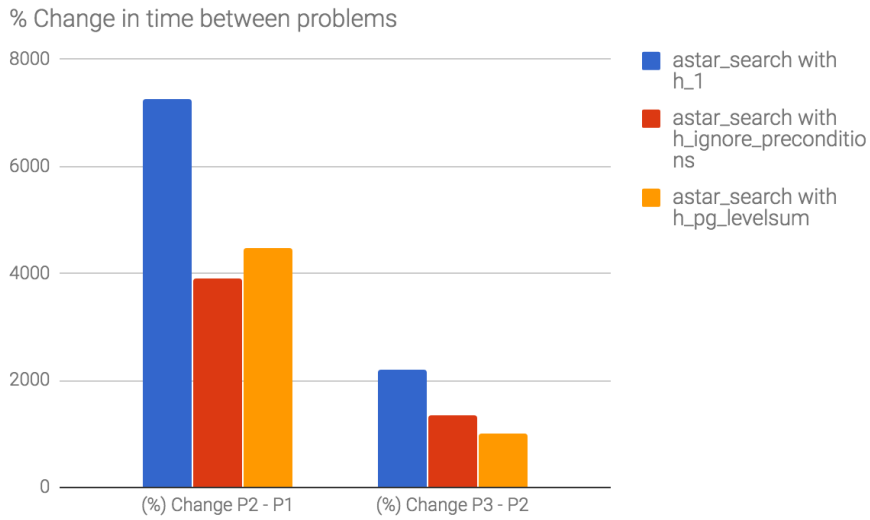
The following plots show a comparison between **astar_search with h_1**, **astar_search with h_ignore_preconditions**, and **astar_search with h_pg_levelsum** comparing **node expansion**, **goal tests**, **plan length** and **time** for each problem.



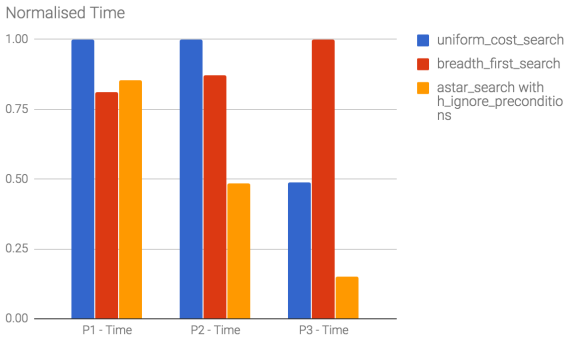
All heuristics achieved equal plan lengths; the differences of the three approaches were then based on the number of **nodes expanded**, **goal tests** performed and the **time** they took. The noticeable difference here was between **astar_search with h_pg_levelsum** and the other two**, h_1, astar_search ignore_preconditions.** While **h_pg_levelsum** had significantly expanded less nodes and performed less goal tests, it took a great more time to perform the search.

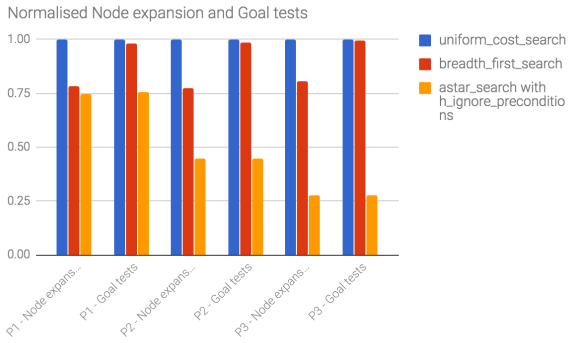
Another important comparison is how well the searches performed with increasing complexity of the search, the following plot shows the percentage change for each of the searches between Problem 2 and Problem 1, and Problem 3 and Problem 2. As illustrated below, an increase in complexity had the

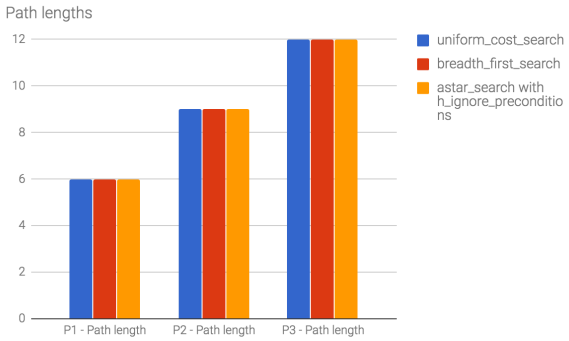
greatest impact on h_1 compared to the others, suggesting that h_1 doesn't scale well with increases in the size of the search space (complexity).



Therefore the best heuristic for these problems would be **h_ignore_preconditions**. So how does it compare with the non-heuristic approaches presented above; below we present the same plots comparing **uniform_cost_search** and **breadth_first_search** (the best 2) with **astar_h_ignore_preconditions**.







As seen above; all searches achieved the same path length but the ignore preconditions heuristic outperformed in time, node expansion, and goal tests. Therefore, for this problem, you would opt for the heuristic approach compared to the others for reasons of better performance and scalability.