Planning Heuristics Analysis

I aim to compare the performance of different path finding algorithms. Looking at planning graph and automatic domain independent heuristic with A* search and compare against the performance of uniformed heuristic search.

Planning Problem

Results for Problem 1

Algorithm	Node Expansions	Goal Tests	New Nodes	Time (s)	Optimal
Breadth First Search	43	56	180	0.0518	TRUE
Depth First Graph Search	12	13	48	0.0102	FALSE
Uniform Cost Search	55	57	224	0.555	TRUE
A* Search h_1	55	57	224	0.0489	TRUE
A* Search h_ignrore_preconditions	41	43	170	0.0478	TRUE
A* Search h_pg_levelsum	11	13	50	0.5966	TRUE

Results for problem 2

Algorithm	Node Expansions	Goal Test	New Nodes	Time (s)	Optimal
Breadth First Search	3399	4667	31207	15.67	TRUE
Depth First Graph Search	1522	1523	13561	15.45	FALSE
Uniform Cost Search	4853	4855	44041	21.91	TRUE
A* Search h_1	4853	4855	44041	22.17	TRUE
A* Search h_ignrore_preconditions	1450	1452	13301	7.25	TRUE
A* Search h_pg_levelsum	86	88	841	53.93	TRUE

Results for problem 3

Alorithm	Node Expansions	Goal Test	New Nodes	Time (s)	Optimal
breadth first search	14664	18098	129641	106.10	TRUE
depth first search	780	781	6439	6.79	FALSE
Uniform Cost Search	18223	1825	159618	108.75	TRUE
A* Search h_1	18223	18225	159618	120.37	TRUE
A* Search h_ignrore_preconditions	5040	5042	44944	38.01	TRUE
A* Search h_pg_levelsum	316	318	2912	273.38	TRUE

- Compare and contrast heuristic search result metrics using A* with the "ignore preconditions" and "level-sum" heuristics for Problems 1, 2, and 3.
- What was the best heuristic used in these problems? Was it better than non-heuristic search planning methods for all problems? Why or why not?

Comparison

Looking at the results for breadth first search it always find the optimal path

Depth First search can be seen to solve this problem faster and out performs all other algorithms this is due to traversal pattern of DFS. However it is not guaranteed to find an optimal solution.

Seeing that non heuristic search algorithms problem performed better when the problem is simpler (smaller state space). Heuristic search algorithms perform better when

Heuristic search algorithms seem to out perform non heuristic search when the problem becomes more complex (When the number of state variables increase). It can be seen from the results table for all problems, that heuristic search algorithms have a lower amount of node expansions, goal tests and node creation. When looking at algorithms that are guaranteed to find optimal algorithms. This in-turn means that these algorithms were less computational intensive.

1. breadth_first_search
2. breadth_first_tree_search
3. depth_first_graph_search
4. depth_limited_search
5. uniform_cost_search
6. recursive_best_first_search h_1
7. greedy_best_first_graph_search h_1
8. astar_search h_1
9. astar_search h_ignore_preconditions
10. astar_search h_pg_levelsum

Problem 1 Optimal path:
Load(C2, P2, JFK)
Load(C1, P1, SFO)
Fly(P2, JFK, SFO)
Unload(C2, P2, SFO)
Fly(P1, SFO, JFK)
Unload(C1, P1, JFK)

Problem 2 Optimal Path:
Load(C2, P2, JFK)
Load(C3, P3, ATL)
Load(C1, P1, SFO)
Fly(P2, JFK, SFO)
Unload(C2, P2, SFO)
Fly(P3, ATL, SFO)
Unload(C3, P3, SFO)
Fly(P1, SFO, JFK)
Unload(C1, P1, JFK)

Problem 3 Optimal Path:
Load(C2, P2, JFK)
Load(C1, P1, SFO)
Fly(P2, JFK, ORD)
Load(C4, P2, ORD)
Fly(P1, SFO, ATL)
Load(C3, P1, ATL)

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