Project 3: Heuristic Analysis

Ronald Eddings

The analysis in this paper is the comparison of non-heuristic search result metrics (optimality, time elapsed, number of node expansions) for Problems 1,2, and 3. Also, the comparison of heuristic search result metrics using A* with the "ignore preconditions" and "level-sum" heuristics for Problems 1, 2, and 3 are included.

Problem 1

	Expansions	Goal Tests	New Nodes	Plan Length	Time Elapsed
breadth_first_search	43	56	180	6	.02
breadth_first_tree_search	1458	1459	5960	6	.74
depth_first_graph_search	21	22	84	20	.011
depth_limited_search	101	271	414	50	.086
uniform_cost_search	55	57	224	6	.03
recursive_best_first_search h_1	4229	4230	17023	6	2.71
greedy_best_first_graph_search h_1	7	9	28	6	.006
astar_search h_1	55	57	224	6	.03
astar_search h_ignore_preconditions	41	43	170	6	.03
astar_search h_pg_levelsum	11	13	50	6	.799

Problem 1 is a fairly easy problem to solve. Additionally, each search algorithm was able to find an goal state in a reasonable amount of time. In the case of non-heuristic search, greedy best graph search outperforms the other search algorithms. Furthermore, out of the non-heuristic searches it had the least amount of expansions, goal tests, new nodes, plan length and time elapsed. In the case of heuristic search, A* search with ‘h_ignore_preconditions’ performed the best with the least amount of time elapsed and less expansions than other A* with ‘h_1’

Optimal Plan

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

	Expansions	Goal Tests	New Nodes	Plan Length	Time Elapsed
breadth_first_search	3343	4609	30509	9	13.76
breadth_first_tree_search	N/A	N/A	N/A	N/A	N/A
depth_first_graph_search	624	625	5602	619	3.57
depth_limited_search	N/A	N/A	N/A	N/A	N/A
uniform_cost_search	4849	4851	44001	9	13.17
recursive_best_first_search h_1	N/A	N/A	N/A	N/A	N/A
greedy_best_first_graph_search h_1	996	968	8694	16	2.77
astar_search h_1	4849	4851	44001	9	13.48
astar_search h_ignore_preconditions	1443	1445	13234	9	5.06
astar_search h_pg_levelsum	85	87	831	9	86.68

Problem 2 introduces more complexity. Breadth first tree search, depth limited search, and recursive best first search were unable to find a plan in a reasonable matter of time. Out of the non-heuristic search, greedy best first graph search outperforms the other search techniques in speed. One thing to note is that greedy best first graph search did not outperform other search techniques in other areas such as expansions, and new nodes. From heuristic search, A* search with 'h_ignore_preconditions' outperformed the other search techniques in speed. However, A* with 'levelsum' had the least amount of expansions, new nodes, and goal tests

Optimal Plan

Load(C1, P1, SFO)
Load(C2, P2, JFK)
Load(C3, P3, ATL)
Fly(P2, JFK, SFO)
Unload(C2, P2, SFO)
Fly(P1, SFO, JFK)
Unload(C1, P1, JFK)
Fly(P3, ATL, SFO)
Unload(C3, P3, SFO)

Problem 3

	Expansions	Goal Tests	New Nodes	Plan Length	Time Elapsed
breadth_first_search	14663	18098	129631	12	114.131
breadth_first_tree_search	N/A	N/A	N/A	N/A	N/A
depth_first_graph_search	408	409	3364	392	1.84
depth_limited_search	N/A	N/A	N/A	N/A	N/A
uniform_cost_search	18235	18237	159716	12	79.97
recursive_best_first_search h_1	N/A	N/A	N/A	N/A	N/A
greedy_best_first_graph_search h_1	5462	5464	48176	21	23.44
astar_search h_1	18235	18237	159716	12	78.37
astar_search h_ignore_preconditions	4945	4947	43991	12	23.70
astar_search h_pg_levelsum	290	292	2670	12	458.56

Again, breadth first tree search, depth limited search, and recursive best first search were unable to find a plan in a reasonable matter of time for this problem. From a time perspective, it would appear that depth first graph search is the winner of non-heuristic search but this search has a plan length of 392 which would probably be an expensive plan. For this reason, greedy best first graph search is again the winner since it has the shortest plan and can find a goal state fastest out of the non-heuristic search techniques. On problem 3, A* search was the winner in speed and plan length again. It appears that A* search with 'levelsum' searches very efficiently but unfortunately, it's slower for the problems in this project.

Optimal Plan

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

Unload(C2, P2, SFO)

Summary

Forward and Backward search are not as efficient without a good heuristic function. More specifically, it's good to have a heuristic that does not overestimate the goal. In the case of heuristic search techniques, A* search with ignoring preconditions outperformed most search techniques. Chapter 10 in AIMA states, "An Admissible heuristic can be derived by defining a relaxed problem that is easier to solve. The exact cost of a solution to this easier problem then becomes the heuristic for the original problem." Using A* search with ignoring preconditions performed well because it relaxes the problem while not overestimating the goal.

References

Russell, Stuart J., et al. Artificial Intelligence: a Modern Approach. Prentice Hall, 2016.