Implement a Planning Search Heuristic Analysis

Ju Yang

Part 1 - Planning problems

Experiment and document metrics for non-heuristic planning solution searches

air_cargo_p1

	Node Expansion	Goal Tests	New Nodes	Plan length	Time lapsed in seconds	Optimality
breadth_first_search	43	56	180	6	0.0584	Y
depth_first_graph_search	21	22	84	20	0.0296	N
breadth_first_tree_search	1458	1459	5960	6	1.9173	Y
depth_limited_search	101	271	414	50	0.1659	N
uniform_cost_search	55	57	224	6	0.0756	Y
recursive_best_first_search with h_1	4229	4230	17023	6	5.4842	Y
greedy_best_first_graph_search with h_1	7	9	28	6	0.0111	Y

air_cargo_p2

	Node Expansion	Goal Tests	New Nodes	Plan length	Time lapsed in seconds	Optimality
breadth_first_search	3343	4609	30509	9	21.0645	Y
depth_first_graph_search	624	625	5602	619	5.0739	N
uniform_cost_search	4852	4854	44030	9	23.7945	Y
greedy_best_first_graph_search with h_1	990	992	8910	17	4.5948	N

air_cargo_p3

	Node Expansion	Goal Tests	New Nodes	Plan length	Time lapsed in seconds	Optimality
breadth_first_search	5621	7281	39000	12	41.1710	Y
depth_first_graph_search	1292	1293	5744	875	7.3071	N
uniform_cost_search	7304	7306	50711	12	40.4608	Y
greedy_best_first_graph_search with h_1	4014	4016	24387	30	19.9823	N

While breadth_first_search (BFS) is always optimal, depth_first_graph_search (DFS) is not optimal. This is DFS always expands the deepest node in the current frontier of the search graph, once it finds a goal, it terminates the searching regardless other possible paths. While BFS uses a FIFO queue, DFS uses a LIFO queue. A LIFO queue means that the most recently generated node is chosen for expansion. This must be the deepest unexpanded node because it is one deeper than its parent - which, in turn, was the deepest unexpanded node when it was selected (Russell and Norvig, Artificial Intelligence, 3rd Edition, page 87). Therefore, DFS will follow any chosen node to the end depending on the structure of the graph, without considering the length of the path.

Part 2 - Domain-independent heuristics

Experiment and document: metrics of A* searches with these heuristics

air_cargo_p1

	Node Expansion	Goal Tests	New Nodes	Plan length	Time lapsed in seconds	Optimality
astar_search with h_1	55	57	224	6	0.0741	Y
astar_search with h_ignore_preconditions	41	43	170	6	0.0678	Y
astar_search with h_pg_levelsum	11	13	50	6	0.5717	Y

air_cargo_p2

	Node Expansion	Goal Tests	New Nodes	Plan length	Time lapsed in seconds	Optimality
astar_search with h_1	4852	4854	44030	9	22.9485	Y
astar_search with h_ignore_preconditions	1450	1452	13303	9	8.0487	Y
astar_search with h_pg_levelsum	86	88	841	9	51.4904	Y

air_cargo_p3

	Node Expansion	Goal Tests	New Nodes	Plan length	Time lapsed in seconds	Optimality
astar_search with h_1	7304	7306	50711	12	39.7038	Y
astar_search with h_ignore_preconditions	2828	2830	20869	12	16.5671	Y
astar_search with h_pg_levelsum	94	96	841	12	53.1333	Y

A* with the "ignore preconditions" and "level-sum" heuristics perform better than astar_search with h_1 (similar to uniform cost search), reducing node expansion, goal tests, new nodes. "ignore preconditions" also reduces run time, while “level_sum” increases run time during the heuristic calculation process.

Part 3: Optimal plan and best algorithm

Why can DFS not find the optimal plan length?

Example why does ignore preconditions run faster than pg_levelsum? Please don't forget to site your references like you did in your research report

Problem	**Problem 1**	**Problem 3**	**Problem 2**
Algorithm	greedy_best_first_graph_search with h_1	astar_search with h_ignore_preconditions	astar_search with h_pg_levelsum

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)
		Load(C3, P3, ATL)
		Fly(P3, ATL, SFO)
		Unload(C3, P3, SFO)
		Load(C1, P1, SFO)
		Fly(P1, SFO, JFK)
		Unload(C1, P1, JFK)
		Load(C2, P2, JFK)
		Fly(P2, JFK, SFO)
		Unload(C2, P2, SFO)
			Load(C1, P1, SFO)
			Fly(P1, SFO, ATL)
			Load(C3, P1, ATL)
			Fly(P1, ATL, JFK)
			Unload(C3, P1, JFK)
			Unload(C1, P1, JFK)
			Load(C2, P1, JFK)
			Fly(P1, JFK, ORD)
			Load(C4, P1, ORD)
			Fly(P1, ORD, SFO)
			Unload(C4, P1, SFO)
			Unload(C2, P1, SFO)

Reason	Least node expansion, least goal test, lest new nodes, and shortest time lapsed, and optimal plan length. We just use h_1 uniform heuristic, rather than more complex heuristics in this case. The search space is relatively small and we can either use breadth_first_search or the greedy_best_first_graph here. Although greedy_best_first does not always provide optimal plan, it is optimal with fewer nodes here.	Less node expansion, less goal test, less new nodes, and faster than breadth_first_search, while providing an optimal solution. It is also much faster than the “level_sum” heuristic. Although the latter largely reduces node expansion, the run time is 6 times longer.	Least node expansion, least goal test, least new nodes, and optimal. Although it takes slightly longer time than breadth_first_search (53 seconds to 42 seconds), it largely reduces the state space from 5621 to 94 nodes (60 times!). I notice that as the number of fluent increases, breadth_first_search run time increases exponentially, while astar search increases only slightly.