Performance of search algorithms

See table on last page.

Optimal Plans

Optimal plan for Air Cargo 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)

Optimal plan for Air Cargo Problem 2
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)

Optimal plan for Air Cargo 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)

Comparison of non-heuristic searches

The three algorithms I tested were depth-first search (DFS), breadth-first search (BFS), and uniform-cost search (UCS). DFS expands successive, unexplored nodes along a single path until reaching a leaf node, BFS expands all nodes in the frontier and explores each one before adding new nodes to the frontier, and UCS is like BFS except it factors in the cost of each graph edge when expanding nodes to guarantee it finds the shortest path to the goal. However, I in the Air Cargo problems all edges had an equal cost of 1, which makes the cost function equivalent to the depth of the node, and which makes UCS equivalent to BFS. However, with that said, the performance of UCS was *not* identical to BFS for the three problems given, merely similar for some reason. Both searches always returned the same, optimal plans but BFS was more efficient in all regards (expansions, goal tests, new nodes, search time) on all three problems, except problem 3 where UCS somehow finished the search faster even while doing more computations.

The difference in performance between DFS and BFS is more obvious. DFS searched through much less of the problem space to find a solution, and as a result returned a solution much more quickly than BFS. This is expected if there are multiple solutions to the problem and when brute-force application of actions is effective in yielding a solution. Otherwise it would be easy for DFS to get trapped in a rabbit-hole, needlessly exploring the depths of the problem space when there exists an efficient solution to the problem elsewhere. And since DFS returns the first solution found and does not guarantee to find the most efficient solution, all of the DFS solutions were significantly less optimal than the optimal solutions found by BFS.

Comparison of heuristic searches

Three “heuristic” A* searches were tested: h1, ignore preconditions (IP), and planning graph level sum (PGLS). I put heuristic in quotes because the h1 heuristic returns 1 no matter what the problem state, which isn’t a real heuristic or at least a useful heuristic. This makes h1 exactly equivalent to UCS, which you can see in the results table.

The two interesting heuristics tested were IP and PGLS. IP estimates the distance from the goal state by calculating how many actions would be required to reach the goal state if none of the actions had any preconditions. Since the Air Cargo problem preserves the subgoal independence assumption (i.e., solving each subgoal independently will achieve main goal), IP is equivalent in this case to the current number of unsatisfied goals. PGLS uses a planning graph to represent potential actions and states during the problem, and finds the sum of the number of further levels required to reach each subgoal in the planning graph.

Since all three heuristics were admissible all three search algorithms returned the same optimal plans. First I’ll compare h1 and IP. IP was more efficient than h1 in all respects, including time.

This makes sense because both were A* algorithms, but IP used a real heuristic, that was also very efficient to calculate and had little impact on total search time.

The differences between IP and PGLS are more interesting. IP was consistently much faster in terms of search time (in fact, was the fastest of all tested algorithms), meanwhile PGLS was actually the slowest algorithm tested. That’s only in terms of search time in seconds though; in terms of problem space exploration (expansions, goal tests, new nodes) it was the most efficient algorithm by far. That’s because the heuristic used was doing it’s own search for the goal state, albeit through the simplified planning graph instead of the full problem graph. This allowed PGLS to find the optimal goal in far fewer steps, with less guessing, than the other algorithms. On the other hand, this mini-search of the planning graph from every node checked clearly took a lot of time, based on how long it took the search to finish for each problem.

Best heuristic used

The best heuristic used for the three Air Cargo problems was the ignore preconditions heuristic (IP). Although it was not as fast as DFS, it was faster than any other search, and returned an optimal solution to each problem, unlike DFS. Also, for the problems tested it was more efficient than PGLS (in terms of runtime). Perhaps for more complex problems using a planning graph would have paid off, but for these relatively simple planning problems PGLS had too much overhead as a heuristic, and IP was just good enough to find an optimal solution quickly.

Problem	Search Algorithm	Expansions	Goal Tests	New Nodes	Plan Length (Optimality	Duration (sec)
Air Cargo Problem 1	breadth_first_search	43	56	180	6	0.111831695
	breadth_first_tree_search	1458	1459	5960	6	3.43289738
	depth_first_graph_search	21	22	84	20	0.05255943805
	depth_limited_search	101	271	414	50	0.263974078
	uniform_cost_search	55	57	224	6	0.1410058601
Air Cargo Problem 2	breadth_first_search	3343	4609	30509	9	41.18731691
	depth_first_graph_search	624	625	5602	619	8.493863737
	uniform_cost_search	4853	4855	44041	9	56.72848407
Air Cargo Problem 3	breadth_first_search	14663	18098	129631	12	255.662126
	depth_first_graph_search	408	409	3364	392	6.201228446
	uniform_cost_search	18151	18153	159038	12	250.7103607
Air Cargo Problem 1	astar_search with h1	55	57	224	6	0.145031151
	astar_search with h_ignore_precor	41	43	170	6	0.1190753849
	astar_search with h_pg_levelsum	11	13	50	6	1.157601061
Air Cargo Problem 2	astar_search with h1	4853	4855	44041	9	58.80334743
	astar_search with h_ignore_precor	1450	1452	13303	9	18.76081928
	astar_search with h_pg_levelsum	86	88	841	9	81.12546534
Air Cargo Problem 3	astar_search with h1	18151	18153	159038	12	263.3372349
	astar_search with h_ignore_precor	5038	5040	44926	12	78.00827292
	astar_search with h_pg_levelsum	314	316	2894	12	386.8985973