Implement a Planning a Search

Part 1 - Non-heuristic planning solution searches:

Air Cargo Problem 1	Expansions	Goal Tests	New Nodes	Plan length	Time elapsed in seconds
breadth_first_search	43	56	180	6	0.040307135
breadth_first_tree_search	1458	1459	5960	6	0.84606272
depth_first_graph_search	21	22	84	20	0.015712883
depth_limited_search	101	271	414	50	0.088999709
uniform_cost_search	55	57	224	6	0.04295138
recursive_best_first_search h_1	4229	4230	17023	6	2.393214557
greedy_best_first_graph_search h_1	7	9	28	6	0.006765491002
astar_search h_1	55	57	224	6	0.05009123201

The optimal plan for problem one is:
Load(C1, P1, SFO)
Load(C2, P2, JFK)
Fly(P1, SFO, JFK)
Fly(P2, JFK, SFO)
Unload(C1, P1, JFK)
Unload(C2, P2, SFO)

Compare and contrast non-heuristic search result metrics:

Breadth-first is our first non-heuristic search with expansions, goal tests, and new nodes generated of 43, 56, and 180 respectively. The algorithm takes roughly 0.040 seconds on a 20 16 Macbook Pro with a 2.9 GHz Intel Core i5 with plan length of 6.

Depth-first search has expansions, goal test, and new nodes of 21, 22, 84 respectively. It takes roughly 0.016 seconds to run with a plan length of 20. While the algorithm seems to run more efficiently with half the time to find a solution and less node expansions & nodes generated with ends up finding a less optimal solution plan of 20.

Greedy-best-first search seems to be the most optimal for problem one with expansion, goal test, and new nodes of 7, 9, and 28 respectively. More importantly it finds the minimal path of 6 just as breadth-first while outperforming depth-first in time execution of 0.007 seconds.

Air Cargo Problem 2	Expansions	Goal Tests	New Nodes	Plan length	Time elapsed in seconds
breadth_first_search	3343	4609	30509	9	12.28499329
breadth_first_tree_search	*	*	*	*	*
depth_first_graph_search	624	625	5602	619	3.102051679
depth_limited_search	222719	2053741	2054119	50	886.6685931
uniform_cost_search	4853	4855	44041	9	10.71036644
recursive_best_first_search h_1	*	*	*	*	*
greedy_best_first_graph_search h_1	998	1000	8982	21	2.112601148
astar_search h_1	4853	4855	44041	9	10.60759495

Note: * is placed for algorithms that did not provide an answer within 15 minutes of execution

The optimal plan for problem two is:
Load(C1, P1, SFO)
Load(C2, P2, JFK)
Load(C3, P3, ATL)
Fly(P1, SFO, JFK)
Fly(P2, JFK, SFO)
Fly(P3, ATL, SFO)
Unload(C3, P3, SFO)
Unload(C2, P2, SFO)
Unload(C1, P1, JFK)

Compare and contrast non-heuristic search result metrics:

Breadth-first for problem two has expansions, goal test, and new nodes of 3343, 4609, and 30509 respectively. The algorithm takes roughly 12.285 seconds and plan length of 9.

Depth-first search has expansions, goal test, and new nodes of 624, 625, 5602 respectively. It takes 3.102 seconds to find a solution plan of length 619. Just like in problem one the algorithm is much faster in execution with less expansion, goals tested, and new nodes generated but with a very inefficient solution compared to breadth-first search.

A-star search h-1 (not a true heuristic) provides expansions, goal test, and new nodes of 4853, 4855, 44041 with a solution plath of 9 and execution time of 10.608 seconds.

Based on the metrics provided I believe A-star is the optimal solution since it provides the optimal path with the least amount of execution time.

Air Cargo Problem 3	Expansions	Goal Tests	New Nodes	Plan length	Time elapsed in seconds
breadth_first_search	14663	18098	129631	12	89.94616196
breadth_first_tree_search	*	*	*	*	*
depth_first_graph_search	408	409	3364	392	1.75185688
depth_limited_search	*	*	*	*	*
uniform_cost_search	18223	18225	159618	12	43.31917354
recursive_best_first_search h_1	*	*	*	*	*
greedy_best_first_graph_search h_1	5578	5580	49150	22	13.6505
astar_search h_1	18223	18225	159618	12	46.61210832

Note: * is placed for algorithms that did not provide an answer within 15 minutes of execution

The optimal plan for problem three is:
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(C4, P2, SFO)
Unload(C3, P1, JFK)
Unload(C2, P2, SFO)
Unload(C1, P1, JFK)

Breadth-first for problem three has expansions, goal test, and new nodes of 14663, 18098, and 129631 respectively. The algorithm takes roughly 89.946 seconds and plan length of 12.

Depth-first search has expansions, goal test, and new nodes of 408, 409, 3364 respectively. It takes 1.752 seconds to find a solution plan of length 392. As seen in the past two problems the algorithm is much faster but has a significantly worse solution path.

Uniform cost search also arrives at the optimal path of 12 with expansions, goal test, and new nodes of 18223, 18225, 159618 respectively. Execution time is 43.319 seconds. Considering that uniform cost search finds the optimal path like breadth-first search but with half the execution time it seems like a reasonable trade-off compared to breadth or depth-first search.

Part 2 - Domain-independent heuristics:

Air Cargo Problem 1	Expansions	Goal Tests	New Nodes	Plan length	Time elapsed in seconds
astar_search h_ignore_preconditions	41	43	170	6	0.04195505199
astar_search h_pg_levelsum	45	47	188	6	1.123760569

A-star search (h ignore preconditions) has expansions, goal test, and new nodes of 41, 43, and 170 respectively. The algorithm finds the solution with plan length 6 and execution time of 0.042 seconds.

A-star search (h pg levelsum) has expansions, goal test, and new nodes of 45, 47, and 188 respectively. It however finds the same solution path of length six at 1.124 seconds.

Based on these preliminary finds I believe A-star search with h ignore preconditions to be the faster of the two heuristic. However I believe Greedy-best-first to be the best algorithm for finding the optimal solution since it computes the path in 0.007 seconds, a fraction of the time that A-star (h ignore preconditions) does.

Air Cargo Problem 2	Expansions	Goal Tests	New Nodes	Plan length	Time elapsed in seconds
astar_search h_ignore_preconditions	1450	1452	13303	9	3.500707348
astar_search h_pg_levelsum	1643	1645	15416	9	392.8285017

A-star search (h ignore preconditions) has expansions, goal test, and new nodes of 1450, 1452, and 13303 respectively. The algorithm and heuristic find a solution of length 9 within 3.5 seconds.

A-star search (h pg levelsum) has expansions, goal test, and new nodes of 1643, 1645, and 15416 respectively. It finds a path of length 9 with a execution time of 392.828 seconds.

Based of these finds I believe again A-star search (h ignore preconditions) the be the optimal algorithm + heuristic since (h pg levelsum) takes roughly 100 times more seconds to find the same path. A-start (h ignore preconditions) is still the fastest compared to the non-informed searches where A-star (h-1) executes in 10.6 seconds.

Air Cargo Problem 3	Expansions	Goal Tests	New Nodes	Plan length	Time elapsed in seconds
astar_search h_ignore_preconditions	5040	5042	44944	12	13.07161643
astar_search h_pg_levelsum	*	*	*	*	*

Note: * is placed for algorithms that did not provide an answer within 15 minutes of execution

A-star search (h ignore preconditions) has expansions, goal test, and new nodes of 5040, 5042, and 44944 respectively. The algorithm + heuristic find the optimal solution of lenght 12 in 13.072 seconds.

A-star search (h pg levelsum) did not execute in a timely manner (waited 15 instead of the designated 10 minuted) and therefore was omitted from analysis.

Based on my finds I conclude that A-star (h ignore preconditions) to be the better of the two heuristics since it finds a solution in a timely manner. The algorithm + heuristic also outperforms informed searched without any trade-offs by finding the optimal solution and the least execution time.

Conclusion:

After reviewing the performances of the each algorithm, the metrics seem to suggest that A-star (h ignore preconditions) is the overall best at finding an optimal solution in a relatively short period of time. Only in the first problem was greedy-best-first able to outperform the A-star heuristic which was a very simple solution that might not generalize to more variables and complex scheduling tasks.