Optimal plans for Problem 1, Problem 2 and Problem 3 are summarized in the table below:

Problem	optimal sequence of actions
air_cargo_p1	Load(C2, P2, JFK)-> Load(C1, P1, SFO) -> Fly(P2, JFK, SFO) -> Unload(C2, P2, SFO) -> Fly(P1, SFO, JFK) -> Unload(C1, P1, JFK)
air_cargo_p2	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(C1, P1, JFK) -> Unload(C2, P2, SFO)
air_cargo_p3	Load(C2, P2, JFK) -> Fly(P2, JFK, ATL) -> Load(C3, P2, ATL) -> Fly(P2, ATL, ORD) -> Load(C4, P2, ORD) -> Fly(P2, ORD, SFO) -> Load(C1, P2, SFO) -> Unload(C2, P2, SFO) -> Unload(C4, P2, SFO) -> Fly(P2, SFO, JFK) -> Unload(C1, P2, JFK) -> Unload(C3, P2, JFK)

Above are just examples of optimal plans as they are not unique in their domain. However, they all have the same plan length: 6 for Problem 1, 9 for Problem 2 and 12 for Problem 3.

Breadth First Search, Depth First Graph Search and Uniform Cost Search were being compared for analyzing non-heuristic search methods and results are summarized in the table below:

breadth_first_search	number of node expansions required	time elapsed (seconds)	optimal?	Plan length
Problem 1	43	0.046	Yes	6
Problem 2	3343	9.56	Yes	9
Problem 3	13124	36.8	Yes	12

depth_first_graph_search	number of node expansions required	time elapsed (seconds)	optimal?	Plan length
Problem 1	12	0.0089	No	12
Problem 2	582	3.36	No	575
Problem 3	1293	3.19	No	876

uniform_cost_search	number of node expansions required	time elapsed (seconds)	optimal?	Plan length
Problem 1	55	0.05	Yes	6
Problem 2	4853	13.55	Yes	9
Problem 3	17010	44.55	Yes	12

Only Breadth First Search and Uniform Cost Search could find optimal plan for all problems. Although Depth First Graph Search was the fastest and used the least memory, it didn't guarantee to find optimal solution. Breadth First Search and Uniform Cost Search are expanding all nodes in their proximity (either defined by depth or path cost) for each node from the beginning and therefore the optimal path is guaranteed to be reached. As depth increases, the number of nodes expanded increases exponentially by the order of the branching factor of the depth level and therefore, they consume much more memory and requires more time than Depth First Graph Search which expands the nodes by the order of depth and stops once a goal state is reached.

A* with "ignore preconditions" and "level-sum" heuristics were being compared for analyzing heuristic search methods and results are summarized in the table below:

astar_search h_ignore_preconditions	number of node expansions required	time elapsed (seconds)	optimal?	Plan length
Problem 1	41	0.046	Yes	6
Problem 2	1450	4.45	Yes	9
Problem 3	5040	18.94	Yes	12

astar_search h_pg_levelsum	number of node expansions required	time elapsed (seconds)	optimal?	Plan length
Problem 1	11	0.62	Yes	6
Problem 2	86	61.59	Yes	9
Problem 3	-	-	-	-

Both heuristics were able to find optimal plan for Problem 1 and 2. "Ignore preconditions" heuristic was able to find optimal plan for 3 but it took too long for "level-sum" heuristic to finish. "Level-sum" heuristic consumed much less memory but took longer time than "ignore preconditions". "Ignore preconditions" heuristic reduces forward search path during estimation by ignoring preconditions which leaded to fewer actions to take to reach the goal state and therefore, it took less time comparing to "Level-sum" heuristic which summed up all levels in a graph search and included all actions during estimation. However, "Level-sum" heuristic preserves the all information which leaded to a much better estimation which helps to prune the tree more effectively and hence it helped tree search to consume less memory. Although memory is an important computation resource, it is more important to get the answer in a reasonable amount of time as it is more practical in real world problems. Therefore, "Ignore preconditions" is the best heuristic to use in heuristic search methods.

As we can see from the tables above, A* with "ignore preconditions" strictly outperformed Breadth First Search and Uniform Cost Search in terms of time, memory and optimality. Although Depth First Graph Search took less time and memory, it failed to reach optimal solution which was critical. Hence, A* with "ignore preconditions" was the best search method for the problems.