Optimal plan for problem air_cargo_p1 (length of 6):
Load(C1, P1, SFO)
Fly(P1, SFO, JFK)
Load(C2, P2, JFK)
Fly(P2, JFK, SFO)
Unload(C1, P1, JFK)
Unload(C2, P2, SFO)

Optimal plan for problem air_cargo_p2 (length of 9):
Load(C1, P1, SFO)
Fly(P1, SFO, JFK)
Load(C2, P2, JFK)
Fly(P2, JFK, SFO)
Load(C3, P3, ATL)
Fly(P3, ATL, SFO)
Unload(C3, P3, SFO)
Unload(C2, P2, SFO)
Unload(C1, P1, JFK)

Optimal plan for problem air_cargo_p3 (length of 12):
Load(C2, P2, JFK)
Fly(P2, JFK, ORD)
Load(C4, P2, ORD)
Fly(P2, ORD, SFO)
Load(C1, P1, SFO)
Fly(P1, SFO, ATL)
Load(C3, P1, ATL)
Fly(P1, ATL, JFK)
Unload(C4, P2, SFO)
Unload(C3, P1, JFK)
Unload(C2, P2, SFO)
Unload(C1, P1, JFK)

Non-heuristic search result metrics:

Breadth First Search						
Problem	Expansions	Goal Tests	New Nodes	Time Elapsed [s]	Plan Length	Optimal
Air Cargo Problem 1	43	56	180	0,03	6	yes
Air Cargo Problem 2	3343	4609	30509	12,88	9	yes
Air Cargo Problem 3	14663	18098	129631	97,58	12	yes

Depth First Graph Search						
Problem	Expansions	Goal Tests	New Nodes	Time Elapsed [s]	Plan Length	Optimal
Air Cargo Problem 1	21	22	84	0,013	20	no
Air Cargo Problem 2	624	625	5602	3,34	619	no
Air Cargo Problem 3	408	409	3364	1,65	392	no

Uniform Cost Search						
Problem	Expansions	Goal Tests	New Nodes	Time Elapsed [s]	Plan Length	Optimal
Air Cargo Problem 1	55	57	224	0,04	6	yes
Air Cargo Problem 2	4853	4855	44041	10,84	9	yes
Air Cargo Problem 3	18233	18235	159697	54,92	12	yes

Analysis:

First thing to note is that Depth First Graph Search can't find optimal solution. It simply dives as deep as possible into the first node and its children. Because of nature of DFGS it can provide optimal solution only by accident, so it's not reliable. Breadth First Search does better job at finding optimal route. By definition it'll come up with optimal solution (the one that takes the least steps) but doesn't take into account the cost of those steps. Because of the need to search all nodes on a current graph level (go through entire breadth on a current level) it takes the longest to complete. Uniform Cost Search is the best of those three. It produces optimal solution about 40 seconds faster than BFS for hardest problem in the given problem set. We can relate to that algorithm as best-first search since every next node is chosen according to a function g(x) that returns the real distance from goal. It takes step cost into consideration.

Heuristic search result metrics:

A* search, heuristic: const						
Problem	Expansions	Goal Tests	New Nodes	Time Elapsed [s]	Plan Length	Optimal
Air Cargo Problem 1	55	57	224	0,036	6	yes
Air Cargo Problem 2	4853	4855	44041	11,17	9	yes
Air Cargo Problem 3	18233	18235	159697	47,65	12	yes

A* search, heuristic: ignore preconditions						
Problem	Expansions	Goal Tests	New Nodes	Time Elapsed [s]	Plan Length	Optimal
Air Cargo Problem 1	41	43	170	0,04	6	yes
Air Cargo Problem 2	1428	1430	13085	4,12	9	yes
Air Cargo Problem 3	4859	4861	43129	14,76	12	yes

A* search, heuristic: level sum						
Problem	Expansions	Goal Tests	New Nodes	Time Elapsed [s]	Plan Length	Optimal
Air Cargo Problem 1	11	13	50	0,85	6	yes
Air Cargo Problem 2	114	116	1120	199,96	9	yes
Air Cargo Problem 3	306	308	2821	901,3	12	yes

Analysis:

The heuristic provided as a constant number performs well only for the easiest problems and can't keep up with problem's complexity. Although it may be a good idea to come up with a fancy heuristic (such as level sum) as it radically decreases number of expansions, goal tests and new nodes, it's too computationally expensive and results in time spans even higher than in non-heuristic searches. My choice for the best path finding algorithm in project is **A* search, heuristic: ignore preconditions**. If we compare its expansions, goal tests and new nodes to every other search (heuristic and non-heuristic), we conclude that the only one performing less operations is A* search, heuristic: level sum, but its performance time is too high due to complexity of heuristic function.