Heuristic Analysis

All problems are in the Air Cargo domain. They have the same action schema defined, but different initial states and goals.

Air Cargo Action Schema:

```
Action(Load(c, p, a),
    PRECOND: At(c, a) ∧ At(p, a) ∧ Cargo(c) ∧ Plane(p) ∧ Airport(a)
    EFFECT: ¬ At(c, a) ∧ In(c, p))
Action(Unload(c, p, a),
    PRECOND: In(c, p) ∧ At(p, a) ∧ Cargo(c) ∧ Plane(p) ∧ Airport(a)
    EFFECT: At(c, a) ∧ ¬ In(c, p))
Action(Fly(p, from, to),
    PRECOND: At(p, from) ∧ Plane(p) ∧ Airport(from) ∧ Airport(to)
    EFFECT: ¬ At(p, from) ∧ At(p, to))
```

Problem I:

Initial State and Goal

```
Init(At(C1, SFO) ∧ At(C2, JFK)
    ∧ At(P1, SFO) ∧ At(P2, JFK)
    ∧ Cargo(C1) ∧ Cargo(C2)
    ∧ Plane(P1) ∧ Plane(P2)
    ∧ Airport(JFK) ∧ Airport(SFO))
Goal(At(C1, JFK) ∧ At(C2, SFO))
```

Metrics

Algorithm	Node Expansion	Goal Tests	New Nodes	Plan Length	Time Elapsed [seconds]
breadth_first_search	43	56	180	6	0.0211
breadth_first_tree_search	1458	1459	5960	6	0.5877
depth_first_graph_search	21	22	84	20	0.0103
depth_limited_search	101	271	414	50	0.0602
uniform_cost_search	55	57	224	6	0.0258
recursive_best_first_search h_1	4229	4230	17023	6	1.6969
greedy_best_first_graph_search h_1	7	9	28	6	0.0047
astar_search h_1	55	57	224	6	0.0268
astar_search h_ignore_preconditions	41	43	170	6	0.0243
astar_search h_pg_levelsum	11	13	50	6	0.6113

Optimal 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)
```

Problem II

Initial State and Goal

```
Init(At(C1, SFO) ∧ At(C2, JFK) ∧ At(C3, ATL)
    ∧ At(P1, SFO) ∧ At(P2, JFK) ∧ At(P3, ATL)
    ∧ Cargo(C1) ∧ Cargo(C2) ∧ Cargo(C3)
    ∧ Plane(P1) ∧ Plane(P2) ∧ Plane(P3)
    ∧ Airport(JFK) ∧ Airport(SFO) ∧ Airport(ATL))
Goal(At(C1, JFK) ∧ At(C2, SFO) ∧ At(C3, SFO))
```

Metrics

Algorithm	Node Expansion	Goal Tests	New Nodes	Plan Length	Time Elapsed [seconds]
breadth_first_search	3343	4609	30509	9	5.4061
breadth_first_tree_search	Timed Out				
depth_first_graph_search	624	625	5602	619	2.1287
depth_limited_search	222719	2053741	2054119	50	618.0923
uniform_cost_search	4853	4855	44041	9	7.5371
recursive_best_first_search h_1	Timed Out				
greedy_best_first_graph_search h_1	998	1000	8982	21	1.5554
astar_search h_1	4853	4855	44041	9	7.6248
astar_search h_ignore_preconditions	1450	1452	13303	9	2.7769
astar_search h_pg_levelsum	86	88	841	9	133.0823

Optimal Plan

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

Problem III

Initial State and Goal

```
Init(At(C1, SFO) ∧ At(C2, JFK) ∧ At(C3, ATL) ∧ At(C4, ORD)
    ∧ At(P1, SFO) ∧ At(P2, JFK)
    ∧ Cargo(C1) ∧ Cargo(C2) ∧ Cargo(C3) ∧ Cargo(C4)
    ∧ Plane(P1) ∧ Plane(P2)
    ∧ Airport(JFK) ∧ Airport(SFO) ∧ Airport(ATL) ∧ Airport(ORD))
Goal(At(C1, JFK) ∧ At(C3, JFK) ∧ At(C2, SFO) ∧ At(C4, SFO))
```

Metrics

Algorithm	Node Expansion	Goal Tests	New Nodes	Plan Length	Time Elapsed [seconds]
breadth_first_search	14663	18098	129631	12	28.1912
breadth_first_tree_search	Timed Out				
depth_first_graph_search	408	409	3364	392	1.1158
depth_limited_search	Timed Out				
uniform_cost_search	18223	18225	159618	12	39.6525
recursive_best_first_search h_1	Timed Out				
greedy_best_first_graph_search h_1	5578	5580	49150	22	10.6234
astar_search h_1	18223	18225	159618	12	33.6784
astar_search h_ignore_preconditions	5040	5042	44944	12	11.0498
astar_search h_pg_levelsum	318	320	2934	12	770.2634

Optimal Plan

```
Load(C2, P2, JFK)
Fly(P2, JFK, ORD)
Load(C4, P2, ORD)
Fly(P2, ORD, SFO)
Unload(C4, 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)
Unload(C2, P2, SFO)
```

Observations

The initial problem is simple enough for most algorithms to have a good performance. The worst performer is `recursive_best_first_search h_1` at 1.6969 seconds and expanding 4229 nodes. Although the plan length is still good enough with 6 paths. The increasing complexity has a positive correlation with the time it takes for each algorithm to find a solution.

Its also surprising to see how breadth first search takes longer than depth first search considering that depth first search would expand much more nodes. Also important to note that both have very bad performance making them timeout during the third problem or even the second one for breadth first search.

Heuristics where not as useful as expected, the results where not the fastest ones but all of them did provide the minimum amount of steps.

Overall the `depth_first_graph_search` would be my method of choice followed by `astar_search_h_ignore_preconditions`.

Comparisons with References

Breadth first search is optimal and would always find the shortest path but might not always find the cheapest (less number of steps), for this we have Uniform Cost Search which we can see expands more nodes as its trying to find a more cost effective path. We can also appreciate how Depth First Graph Search is not optimal since there is more than one state (AIND Lesson 7 on Search).

A* Search algorithm is not guaranteed to be optimal but when combined with `h_ignore_preconditions` and `h_pg_levelsum` heuristics optimal paths are achieved based on the newly applied constraints (AIND Lesson 4 Introduction).

When comparing `astar_search h_pg_levelsum` with `astar_search h_ignore_preconditions` we can see how heavy calculating heuristics can be, in particular against `ignoring preconditions` which is frequently used in planning search (Artifical Intelligence: A Modern Approach, 3rd ed., Russell and Norvig, p. 376). While this is the case levelsum is also used in planning (Artificial Intelligence: A Modern Approach, 3rd ed., Russell and Norvig, p. 382) but it comes down to the use case considered given that from the Game Playing project in AIND we were able to appreciate how sometimes its best to choose the cheapest options over the smartest one.