# Heuristic Analysis

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

```
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))
```

## Part I: Planning problems

For each of the 3 Planning Domain Definition Language (PDDL) problems, 8 uninformed planning searches were performed. The results are presented in the tables below.

### 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))
```

*Table 1: Non-Heuristic search result for Problem I*

| Algorithms | Expansions | Goal Tests | New Nodes | Plan Length | Time elapsed [s] |
|---|---|---|---|---|---|
| breadth_first_search | 43 | 56 | 180 | 6 | 0.03933 |
| depth_first_graph_search | 21 | 22 | 84 | 20 | 0.01986 |
| breadth_first_tree_search | 1458 | 1459 | 5960 | 6 | 0.92602 |
| depth_limit_search | 101 | 271 | 414 | 50 | 0.08982 |
| uniform_cost_search | 55 | 57 | 224 | 6 | 0.06457 |
| recursive_best_first_search h_1 | 4229 | 4230 | 17023 | 6 | 3.01592 |
| **greedy_best_first_graph_search h_1** | **7** | **9** | **28** | **6** | **0.00587** |
| A*_search h_1 | 55 | 57 | 224 | 6 | 0.04461 |

The optimal plan for problem I has 6 steps:

1. Load(C1, P1, SFO)
2. Load(C2, P2, JFK)
3. Fly(P1, SFO, JFK)
4. Fly(P2, JFK, SFO)
5. Unload(C1, P1, JFK)
6. Unload(C2, P2, SFO)

From the non-heuristic search algorithms used to solve the planning problem, Depth First Search and Depth limit search are the only 2 that did not generate an optimal plan. The Greedy Best First Graph Search did less node expansion and the execution time is smaller than any of the other algorithms.

## 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))
```

*Table 2: Non-Heuristic search result for Problem II*

| Algorithms | Expansions | Goal Tests | New Nodes | Plan Length | Time elapsed [s] |
|---|---|---|---|---|---|
| breadth_first_search | 3343 | 4609 | 30509 | 9 | 19.68177 |
| depth_first_graph_search | 624 | 625 | 5602 | 619 | 4.68433 |
| breadth_first_tree_search | Long time to calculate | | | | |
| depth_limit_search | Long time to calculate | | | | |
| **uniform_cost_search** | **4853** | **4855** | **44041** | **9** | **16.12372** |
| recursive_best_first_search h_1 | Long time to calculate | | | | |
| greedy_best_first_graph_search h_1 | 998 | 1000 | 8982 | 21 | 3.86005 |
| A*_search h_1 | 4853 | 4855 | 44041 | 9 | 17.12745 |

The optimal plan for problem II has 9 steps:

1. Load (C1, P1, SFO)
2. Load (C2, P2, JFK)
3. Load (C3, P3, ATL)
4. Fly (P1, SFO, JFK)
5. Fly (P2, JFK, SFO)
6. Fly (P3, ATL, SFO)
7. Unload (C3, P3, SFO)
8. Unload (C2, P2, SFO)
9. Unload (C1, P1, JFK)

From the non-heuristic search algorithms used to solve the planning problem, Depth First Search and Greedy Best First Graph Search are the only 2 that did not generate an optimal plan. Moreover, the Breadth First Tree Search, Depth Limit and Recursive Best First Search took too much time to calculate so it was terminated. The Uniform Cost Search did less node expansion and the execution time is slightly smaller than any of the other algorithms.

## 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))
```

*Table 3: Non-Heuristic search result for Problem III*

| Algorithms | Expansions | Goal Tests | New Nodes | Plan Length | Time elapsed [s] |
|---|---|---|---|---|---|
| breadth_first_search | 14663 | 18098 | 129631 | 12 | 152.96080 |
| depth_first_graph_search | 408 | 409 | 3364 | 392 | 2.52844 |
| breadth_first_tree_search | Long time to calculate | | | | |
| depth_limit_search | Long time to calculate | | | | |
| uniform_cost_search | 18223 | 18225 | 159618 | 12 | 70.60573 |
| recursive_best_first_search h_1 | Long time to calculate | | | | |
| greedy_best_first_graph_search h_1 | 5578 | 5580 | 49150 | 22 | 19.97673 |
| **A*_search h_1** | **18223** | **18225** | **159618** | **12** | **66.07622** |

The optimal plan for problem III has 12 steps:

1. Load (C1, P1, SFO)
2. Load (C2, P2, JFK)
3. Fly (P1, SFO, ATL)
4. Load (C3, P1, ATL)
5. Fly (P2, JFK, ORD)
6. Load (C4, P2, ORD)
7. Fly (P2, ORD, SFO)
8. Fly (P1, ATL, JFK)
9. Unload (C4, P2, SFO)
10. Unload (C3, P1, JFK)
11. Unload (C2, P2, SFO)
12. Unload (C1, P1, JFK)

From the non-heuristic search algorithms used to solve the planning problem, Depth First Search and Greedy Best First Graph Search are the only 2 that did not generate an optimal plan. Moreover, the Breadth First Tree Search, Depth Limit and Recursive Best First Search took too much time to calculate so it was terminated. The A* search has an execution time slightly smaller than any of the other algorithms.

# Part II: Domain-independent heuristics

For the 3 PDDL, 2 A* searches with domain-independent heuristics were performed. The first one called h_ignore_preconditions is taking the minimum number of actions from the current state in order to satisfy all the goal conditions by ignoring the preconditions of all the actions. The second one called h_pg_sum created and uses a planning graph to estimate the number of actions that must be undertaken from the current state in order to satisfy all the goal conditions. The results are presented in the tables below.

*Table 4: Non-Heuristic and heuristic search result for Problem I*

| Algorithms | Expansions | Goal Tests | New Nodes | Plan Length | Time elapsed [s] |
|---|---|---|---|---|---|
| breadth_first_search | 43 | 56 | 180 | 6 | 0.03933 |
| depth_first_graph_search | 21 | 22 | 84 | 20 | 0.01986 |
| breadth_first_tree_search | 1458 | 1459 | 5960 | 6 | 0.92602 |
| depth_limit_search | 101 | 271 | 414 | 50 | 0.08982 |
| uniform_cost_search | 55 | 57 | 224 | 6 | 0.06457 |
| recursive_best_first_search h_1 | 4229 | 4230 | 17023 | 6 | 3.01592 |
| greedy_best_first_graph_search h_1 | 7 | 9 | 28 | 6 | 0.00587 |
| A*_search h_1 | 55 | 57 | 224 | 6 | 0.04461 |
| **A*_search with h_ignore_preconditions** | 41 | 43 | 170 | 6 | 0.04650 |
| **A*_search with h_pg_levelsum** | 11 | 13 | 50 | 6 | 1.25329 |

Both heuristics generated the optimal plans. The A*_search with h_pg_levelsum had less goal tests and expanded the least amount of node, but had a longer execution time compare to the A*_search with h_ignore_preconditions algorithm.

Moreover, if non-heuristic algorithms are compared with the heuristic one, the greedy_best_first_search_h1 is still the better option since it has less node expansion and the execution time is smaller than any of the other algorithms.

*Table 5: Non-Heuristic and heuristic search result for Problem II*

| Algorithms | Expansions | Goal Tests | New Nodes | Plan Length | Time elapsed [s] |
|---|---|---|---|---|---|
| breadth_first_search | 3343 | 4609 | 30509 | 9 | 19.68177 |
| depth_first_graph_search | 624 | 625 | 5602 | 619 | 4.68433 |
| breadth_first_tree_search | Long time to calculate | | | | |
| depth_limit_search | Long time to calculate | | | | |
| uniform_cost_search | 4853 | 4855 | 44041 | 9 | 16.12372 |
| recursive_best_first_search h_1 | Long time to calculate | | | | |
| greedy_best_first_graph_search h_1 | 998 | 1000 | 8982 | 21 | 3.86005 |
| A*_search h_1 | 4853 | 4855 | 44041 | 9 | 17.12745 |
| **A*_search with h_ignore_preconditions** | 1450 | 1452 | 13303 | 9 | 5.03626 |
| **A*_search with h_pg_levelsum** | 86 | 88 | 841 | 9 | 238.42126 |

Both heuristics generated the optimal plans. The A*_search with h_pg_levelsum had less goal tests and expanded the least amount of node, but had a longer execution time compare to the A*_search with h_ignore_preconditions algorithm.

Moreover, if non-heuristic algorithms are compared with the heuristic one, the A*_search with h_ignore_preconditions is a better option since it has less node expansion and the execution time is smaller than any of the other algorithms.

*Table 6 :Non-Heuristic and heuristic search result for Problem III*

| Algorithms | Expansions | Goal Tests | New Nodes | Plan Length | Time elapsed [s] |
|---|---|---|---|---|---|
| breadth_first_search | 14663 | 18098 | 129631 | 12 | 152.96080 |
| depth_first_graph_search | 408 | 409 | 3364 | 392 | 2.52844 |
| breadth_first_tree_search | Long time to calculate | | | | |
| depth_limit_search | Long time to calculate | | | | |
| uniform_cost_search | 18223 | 18225 | 159618 | 12 | 70.60573 |
| recursive_best_first_search h_1 | Long time to calculate | | | | |
| greedy_best_first_graph_search h_1 | 5578 | 5580 | 49150 | 22 | 19.97673 |
| A*_search h_1 | 18223 | 18225 | 159618 | 12 | 66.07622 |
| **A*_search with h_ignore_preconditions** | 5040 | 5042 | 44944 | 12 | 26.07189 |
| **A*_search with h_pg_levelsum** | 325 | 327 | 3002 | 12 | 1387.94056 |

Both heuristics generated the optimal plans. The A*_search with h_pg_levelsum had less goal tests and expanded the least amount of node, but had a longer execution time compare to the A*_search with h_ignore_preconditions algorithm.

Moreover, if non-heuristic algorithms are compared with the heuristic one, the A*_search with h_ignore_preconditions is a better option since it has less node expansion and the execution time is smaller than any of the other algorithms.

# Conclusion

To conclude, both heuristic algorithm generated the optimal plan for all the problem. Also, except for problem I, the the A*_search with h_ignore_preconditions is a better option since it has less node expansion and the execution time is smaller than any of the other algorithms. Moreover, A*_search with h_pg_levelsum did the less evaluations and expansions but the execution time is long.