# Air Cargo Planning Heuristic Analysis

Tilak
Udacity AIND – Implementation of Planning Search

Search strategies that come under **uninformed** search have no additional information about states beyond that provided in the problem definition. All they can do is generate successors and distinguish a goal state from a non-goal state, for example Breadth First, Depth First, Uniform Searches. **Informed** search strategy are one that uses problem-specific knowledge beyond the definition of the problem itself, they can find solutions more efficiently than an uninformed strategy.

The following tables show the results gathered after solving the air cargo problems with both uninformed and informed based search. The goal of this analysis is to document the results obtained from each search type and find an optimal solution for each air cargo problem, that is; a search algorithm that finds the lowest path among all possible paths from start to goal.

For each set of problems, the best solution has been highlighted with *green* color. *Red* color indicated that the test was stopped as it was taking longer time to conclude the search and to produce any optimal path.

Below is the 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))

Optimal plan for Problems 1, 2, and 3 are given below along with comparison of the performance in terms of **speed** (execution time, measured in seconds), **memory usage** (measured in search node expansions) and **optimality** (Yes, if a solution of optimal length is found; No, otherwise).

1) ***Problem 1***:

Below are the initial and goal states,

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))

Optimal path 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)

| Search Strategy | Search Algorithm | Time Elapsed | Path Length | Node Expansions | Goal Tests | New Nodes | Optimality |
|---|---|---|---|---|---|---|---|
| **Uninformed** | Breadth First Search | 0.132 | 6 | 43 | 56 | 180 | Yes |
| | Breadth First Tree Search | 2.605 | 6 | 1458 | 1459 | 5960 | No |
| | Depth First Graph Search | 0.037 | 20 | 21 | 22 | 84 | Yes |
| | Depth Limited Search | 0.199 | 50 | 101 | 271 | 414 | Yes |
| | Uniform Cost Search | 0.092 | 6 | 55 | 57 | 224 | Yes |
| | Recursive Best First Search with h_1 | 5.23 | 6 | 4229 | 4230 | 17023 | No |
| | Greedy Best First Graph search with h_1 | 0.006 | 6 | 7 | 9 | 28 | Yes |
| **Informed** | A* Search with h_1 | 0.052 | 6 | 55 | 57 | 224 | Yes |
| | A* Search with h_ignore_preconditions | 0.061 | 6 | 41 | 43 | 170 | Yes |
| | A* Search with h_pg_levelsum | 0.874 | 6 | 11 | 13 | 50 | Yes |

Analysis: All three *uninformed* search strategies, i.e. breadth first search, depth first graph search, uniform cost search and Best first, find a solution to all air cargo problems. Breadth first search always considers the shortest path first and a result of it it finds a solution to the problem in a reasonable amount of time and in an optimal way.

Depth first graph search does find a quick solution and requires a small amount of memory, but it lacks optimality. It is not optimal because it does not consider if a node is better than another, it simply explores the nodes that take it as deep as possible in the graph even if the goal is to its right.

*Informed* search did perform better, which suggest that when working with simple problems using a more elaborated approach, such as A* search with heuristics, is not worth the increase in the solution complexity.

At the end from uninformed "Greedy Best First Graph search with h_1" and from informed "A* Search with h_1" gave the best results based on time, path length, numbers of node expanded.

2) *Problem 2*:

Below are the initial and goal states,

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))

Optimal path is,
Load(C3, P3, ATL)
Fly(P3, ATL, SFO)
Unload(C3, P3, SFO)
Load(C1, P1, SFO)
Fly(P1, SFO, JFK)
Unload(C1, P1, JFK)
Load(C2, P2, JFK)
Fly(P2, JFK, SFO)
Unload(C2, P2, SFO)

| Search Strategy | Search Algorithm | Time Elapsed | Path Length | Node Expansions | Goal Tests | New Nodes | Optimality |
|---|---|---|---|---|---|---|---|
| Uninformed | Breadth First Search | 40.392 | 9 | 3343 | 4609 | 30509 | No |
| | Breadth First Tree Search | NA | NA | NA | NA | NA | NA |
| | Depth First Graph Search | 4.257 | 619 | 624 | 625 | 5602 | No |
| | Depth Limited Search | NA | NA | NA | NA | NA | NA |
| | Uniform Cost Search | 14.981 | 9 | 4849 | 4851 | 44001 | No |
| | Recursive Best First Search with h_1 | NA | NA | NA | NA | NA | NA |
| | Greedy Best First Graph search with h_1 | 2.358 | 16 | 966 | 968 | 8694 | Yes |
| Informed | A*Search with h_1 | 11.917 | 9 | 4849 | 4851 | 44001 | No |
| | A* Search with h_ignore_preconditions | 4.239 | 9 | 1443 | 1445 | 13234 | Yes |
| | A* Search with h_pg_levelsum | 48.441 | 9 | 85 | 87 | 831 | No |

**Analysis**: Breadth First Tree Search, Depth Limited Search and Recursive Best First Search with h_1 took more than 10 mins, so those tests had to be stopped.
At the end from uninformed "Greedy Best First Graph search with h_1" and from informed "A* Search with h_ignore_preconditions" gave the best results based on time, path length, numbers of node expanded.

3) *Problem 3*:

Below are the initial and goal states,

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))

Optimal path is,

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(C2, P2, SFO)
Unload(C1, P1, JFK)

| Search Strategy | Search Algorithm | Time Elapsed | Path Length | Node Expansions | Goal Tests | New Nodes | Optimality |
|---|---|---|---|---|---|---|---|
| Uninformed | Breadth First Search | 138.25 | 12 | 14663 | 18098 | 129631 | No |
| | Breadth First Tree Search | NA | NA | NA | NA | NA | NA |
| | Depth First Graph Search | 1.729 | 392 | 408 | 409 | 3364 | Yes |
| | Depth Limited Search | NA | NA | NA | NA | NA | NA |
| | Uniform Cost Search | 57.145 | 12 | 18235 | 18237 | 159716 | No |
| | Recursive Best First Search with h_1 | NA | NA | NA | NA | NA | NA |
| | Greedy Best First Graph search with h_1 | 18.227 | 21 | 5462 | 5464 | 48176 | No |
| Informed | A* Search with h_1 | 53.787 | 12 | 18235 | 18237 | 159716 | No |
| | A* Search with h_ignore_preconditions | 16.496 | 12 | 4945 | 4847 | 43991 | Yes |
| | A* Search with h_pg_levelsum | NA | NA | NA | NA | NA | NA |

**Analysis**: Breadth First Tree Search, Depth Limited Search, Recursive Best First Search with h_1 and A* Search with h_pg_levelsum took more than 10 mins, so those tests had to be stopped. Informed Search did perform better as the problem complexity increased. This is more evident in the air cargo problem 3, where the "A* Search with h_ignore_preconditions" performance was optimal and the fastest amongst those that were optimal. It's also worth noting that the 'h_pg_levelsum' heuristic did in overall perform poorly, most likely due to the heuristic being too complex.

**Conclusion**: According to the results obtained in this analysis, the breadth first search strategy can solve planning problems both fast and optimality, which makes it a good candidate to start off an analysis when dealing with search planning problems. As the complexity of the problems increase, it might be worth to consider if a heuristic based approach such as "A* Search with 'h_ignore_preconditions'" can outperform breadth first search and thus be used instead.