Air Cargo Planning Heuristic Analysis

Nopthakorn Kutawan

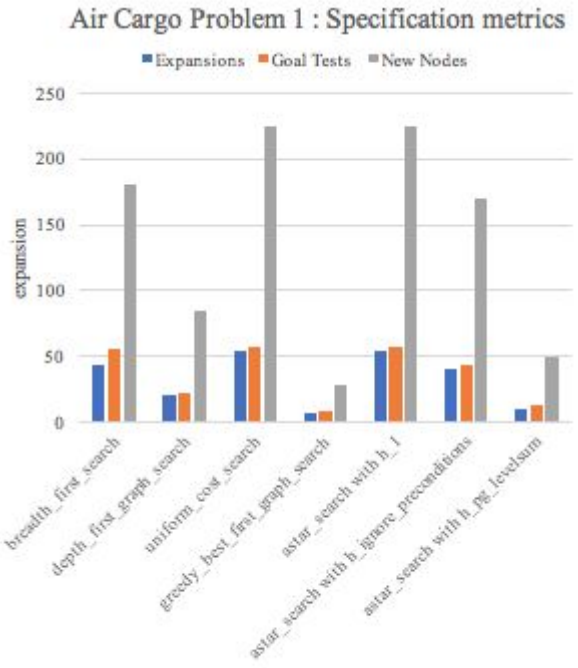
This heuristic analysis of various search algorithms with three planning problems to find an optimal path for the air cargo problem. the result of each problem as following.

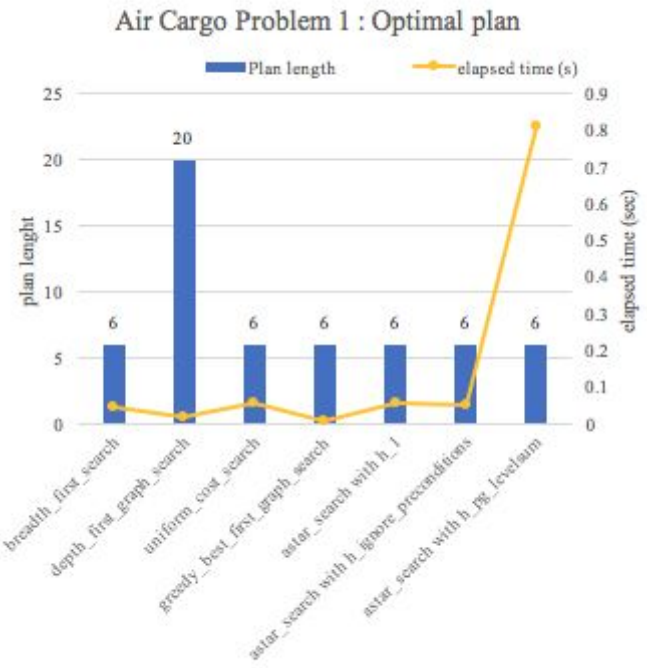
Summary of heuristic analysis

Air Cargo Problem 1

Initial and Goal state	Optimal Plan (Length: 6)
Init(At(C1, SFO) ∧ At(C2, JFK)	1.Load(C1, P1, SFO)
∧ At(P1, SFO) ∧ At(P2, JFK)	2.Load(C2, P2, JFK)
∧ Cargo(C1) ∧ Cargo(C2)	3.Fly(P1, SFO, JFK)
∧ Plane(P1) ∧ Plane(P2)	4.Fly(P2, JFK, SFO)
∧ Airport(JFK) ∧ Airport(SFO))	5.Unload(C2, P2, SFO)
Goal(At(C1, JFK) ∧ At(C2, SFO))	6.Unload(C1, P1, JFK)

The search planing result.





Summary of problem 1:

1. Totally 12 clauses, then search space contain 2^12 state.
2. The optimal search algorithm is “greedy_best_first_graph_search” with optimal plan length as 6, and spend the shortest time and also generates least new nodes and performs minimum expansions/goal tests, that means this algorithm consume a small memory and energy.
3. Another non-heuristic searches “breadth_first_search” and “uniform_cost_search” are comparable performances.
4. For heuristic searches, the “astar_search with h_1” and “astar_search with h_ignore_preconditions” show the fastest time to reach the goal state, but is don’t needed for limited state space problem.

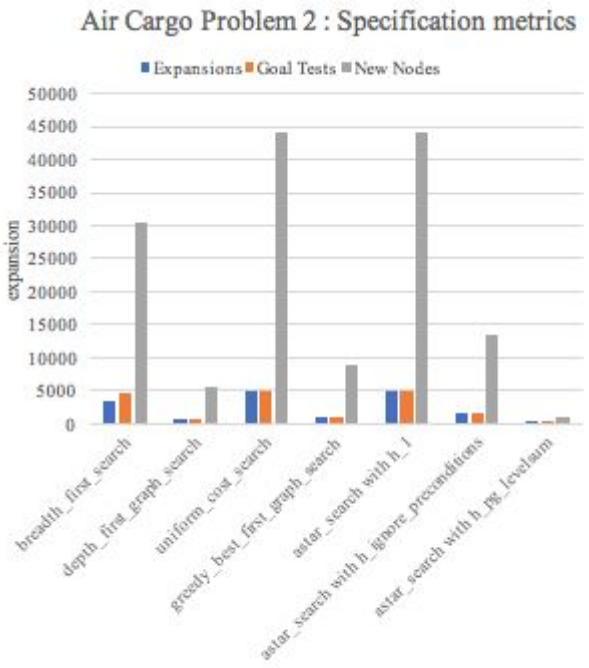
Air Cargo Problem 2

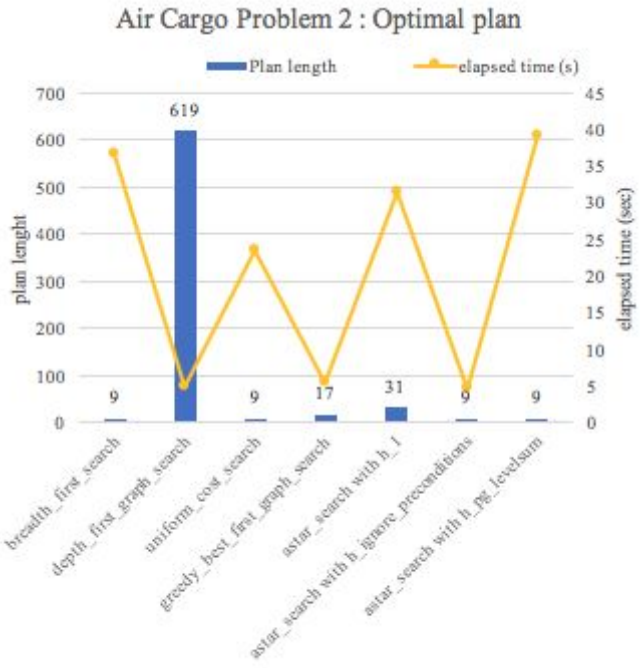
Initial and Goal state

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

The search planing result.





Summary of problem 2 :

1. Totally 27 clauses, then search space contain 2^27 state.
2. The best optimal search algorithm is “astar_search with h_ignore_preconditions” with plan length as 9, and spend shortest time and also generates minimum new nodes and expansions/goal tests.
3. For another heuristic searches, the “astar_search with h_pg_levelsum” generates very few number of new nodes and expansions/goal tests, but spend more time to reach goal state, that not make sense for computation.
4. For non-heuristic searches “breadth_first_search” and “uniform_cost_search” are comparable plan length anyway there are generate high number of new nodes and expansions/goal tests, that may good for run on high performance machine.
5 For “breadth_first_tree_search”, “depth_limited_search”, “recursive_best_first_search” I had about testing because spend time overday on my machine.

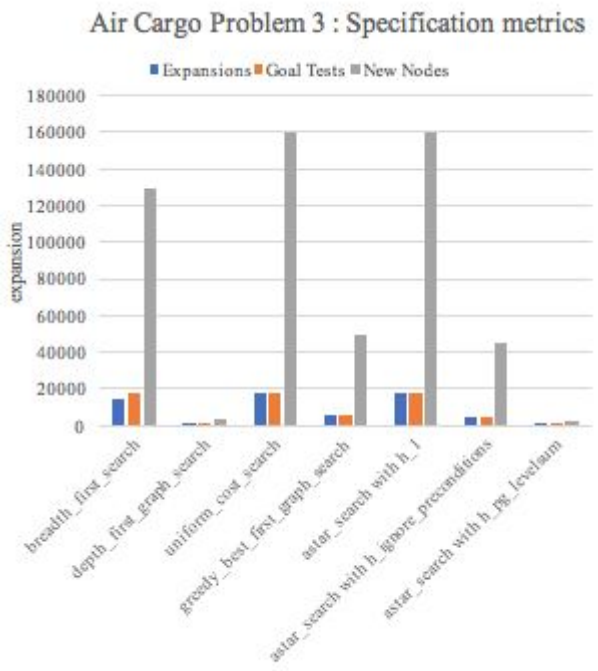
Air Cargo Problem 3

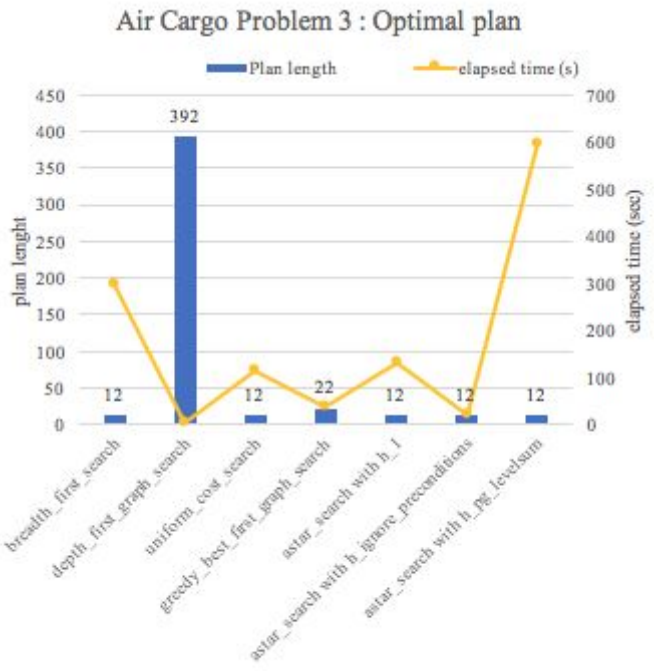
Initial and Goal state

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

The search planing result.





Summary of problem 3:
1. Totally 32 clauses, then search space contain 2^32 state.
2. The best optimal search algorithm is “astar_search with h_ignore_preconditions” with plan length as 12, which consume minimum time also generates the least number of new nodes and expansions/goal tests.
3. For another heuristic searches, the “astar_search with h_pg_levelsum” also generates very few number of new nodes and expansions/goal tests, but computation time very high for reach the goal state, on the other hand the “astar_search with h_1” spend smaller computation time but generates huge number of new nodes and expansions/goal tests.
4. For non-heuristic searches “uniform_cost_search” yield compatible comparable performance with “astar_search with h_1”.
5 For “breadth_first_tree_search”, “depth_limited_search”, “recursive_best_first_search” I had about testing because spend time over the day on my machine.

Conclusion and Justification

According to the results, in non-complex problems, the non-heuristic search algorithm is the optimal strategy to solve problems, as in this case as problem-1 the “greedy_best_first_graph_search” is optimal search strategy to solve problems in term of shortest path and time [1].

As the complexity increase more and more, the heuristic approach makes sense and perform better performance than the non-heuristic approach, based on problem 2 and 3, the “A* Search with h_ignore_preconditions” perform faster by ignoring the preconditions required in order to estimate the minimum actions that must be executed from the current state in order to satisfy all of the goal conditions [2].

Base on the result, for my opinion by consider in term of fastest computation time and shortest length (by don’t worry about hardware performance) I prefer to choose an A* Search with h_ignore_preconditions as better approach for used in the air cargo problem, since it shows a better performance and scalability when problem complexity is increased.

References

[1]. Stuart J. Russell, Peter Norvig (2010), Artificial Intelligence: A Modern Approach (3rd Edition).
[2] AIND Lesson 8 Search