Analysis

Implement a Planning Search

Problems

Below are 3 classical PDDL problems in the Air Cargo domain. All three problems share the same actions schema 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))

Problem 1:

Problem one involves 2 airports, JFK and SFO, 2 airplanes, P1 and P2, and 2 pieces of cargo, C1 and C2. We start the problem with C1 and P1 at SFO and C2 and P2 at JFK. Our goal is for C1 to be at JFK and C2 to be at SFO.
We can represent the initial conditions and goals of this problem with the logic below.

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))

Problem 2:

Problem one involves 3 airports, JFK, SFO and ATL, 2 airplanes, P1, P2 and P3, and 3 pieces of cargo, C1, C2, and C3. We start the problem with C1 and P1 at SFO, C2 and P2 at JFK, and C3 and P3 at ATL. Our goal is for C1 to be at JFK, and C2 and C3 to be at SFO.
We can represent the initial conditions and goals of this problem with the logic below.

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))

Problem 3:

Problem one involves 4 airports, JFK, SFO, ATL, and ORD, 2 airplanes, P1 and P2, and 4 pieces of cargo, C1, C2, C3, C4. We start the problem with C1 and P1 at SFO, C2 and P2 at JFK, C3 at ATL and C4 at ORD. Our goal is for C1 and C3 to be at JFK and C2 and C4 to be at SFO.
We can represent the initial conditions and goals of this problem with the logic below.

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 Plans

Before we hop into analysis of the algorithms used lets look at the optimal solutions for these plans. So for each problem there are multiple optimal plans. However the what makes a plan **optimal** is the number of step needed to achieve the goal.

Problem 1:

We can solve problem #1 in 6 steps. However there does exist plans that solve this problem with greater than 6 steps but we will ignore them since they are not optimal. The plans are as follows:

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

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

Problem 2:

We can solve problem #1 in 9 steps. However there does exist plans that solve this problem with greater than 6 steps but we will ignore them since they are not optimal. The plans are as follows:

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)

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)

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

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

Problem 3:

We can solve problem #1 in 12 steps. However there does exist plans that solve this problem with greater than 6 steps but we will ignore them since they are not optimal. The plans are as follows:

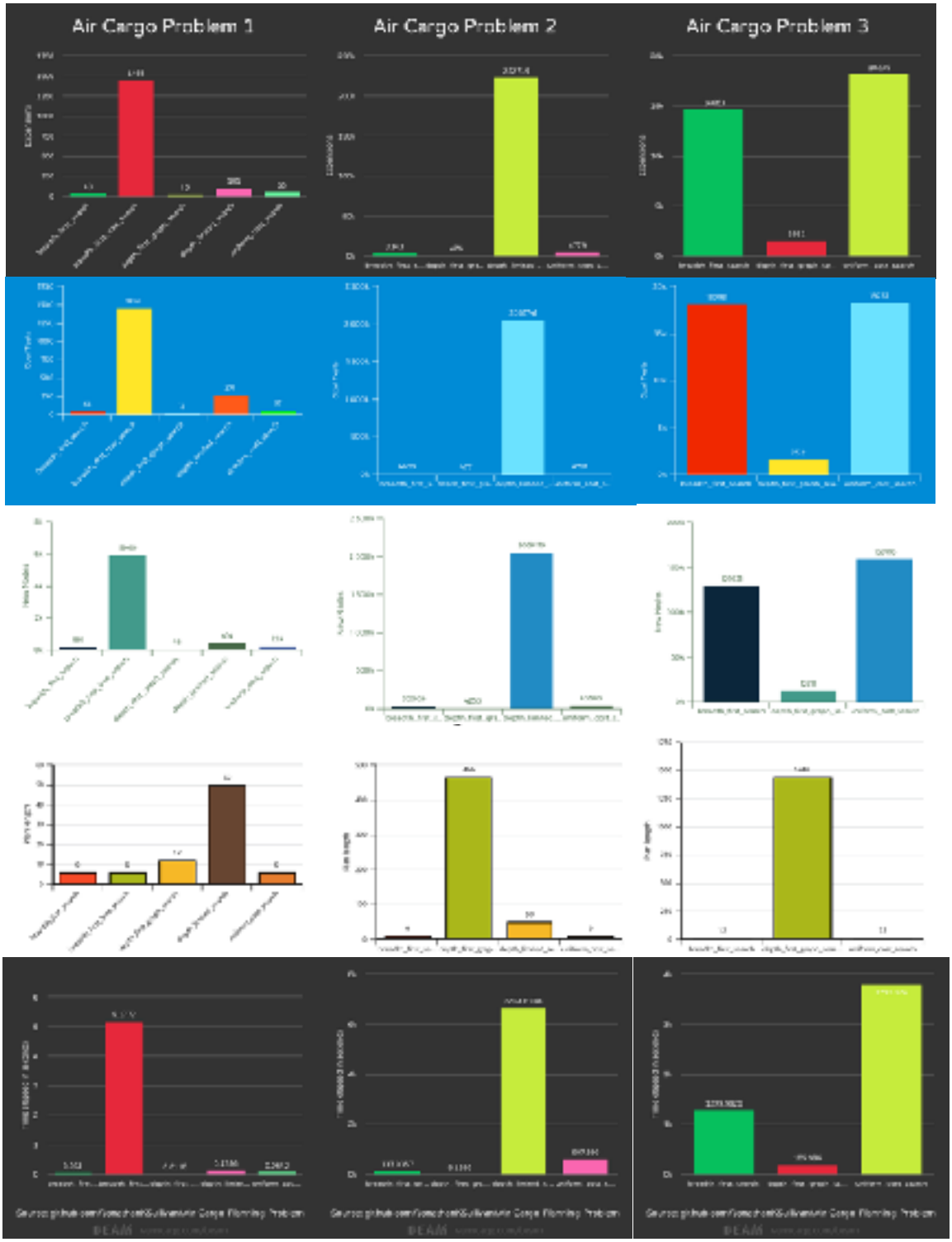
Load(C2, P2, JFK)
Fly(P2, JFK, ORD)
Load(C4, P2, ORD)
Fly(P2, ORD, SFO)
Load(C1, P1, SFO)
Fly(P1, SFO, ATL)
Load(C3, P1, ATL)
Fly(P1, ATL, JFK)
Unload(C4, P2, SFO)
Unload(C3, P1, JFK)
Unload(C2, P2, SFO)
Unload(C1, P1, JFK)

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)

Load(C1, P1, SFO)
Load(C2, P2, JFK)
Fly(P1, SFO, ATL)
Load(C3, P1, ATL)
Fly(P2, JFK, ORD)
Load(C4, P2, ORD)
Fly(P2, ORD, SFO)
Unload(C4, P2, SFO)
Fly(P1, ATL, JFK)
Unload(C3, P1, JFK)
Unload(C2, P2, SFO)
Unload(C1, P1, JFK)

Load(C1, P1, SFO)
Load(C2, P2, JFK)
Fly(P2, JFK, ORD)
Load(C4, P2, ORD)
Fly(P1, SFO, ATL)
Load(C3, P1, ATL)
Fly(P1, ATL, JFK)
Unload(C1, P1, JFK)
Unload(C3, P1, JFK)
Fly(P2, ORD, SFO)
Unload(C2, P2, SFO)
Unload(C4, P2, SFO)

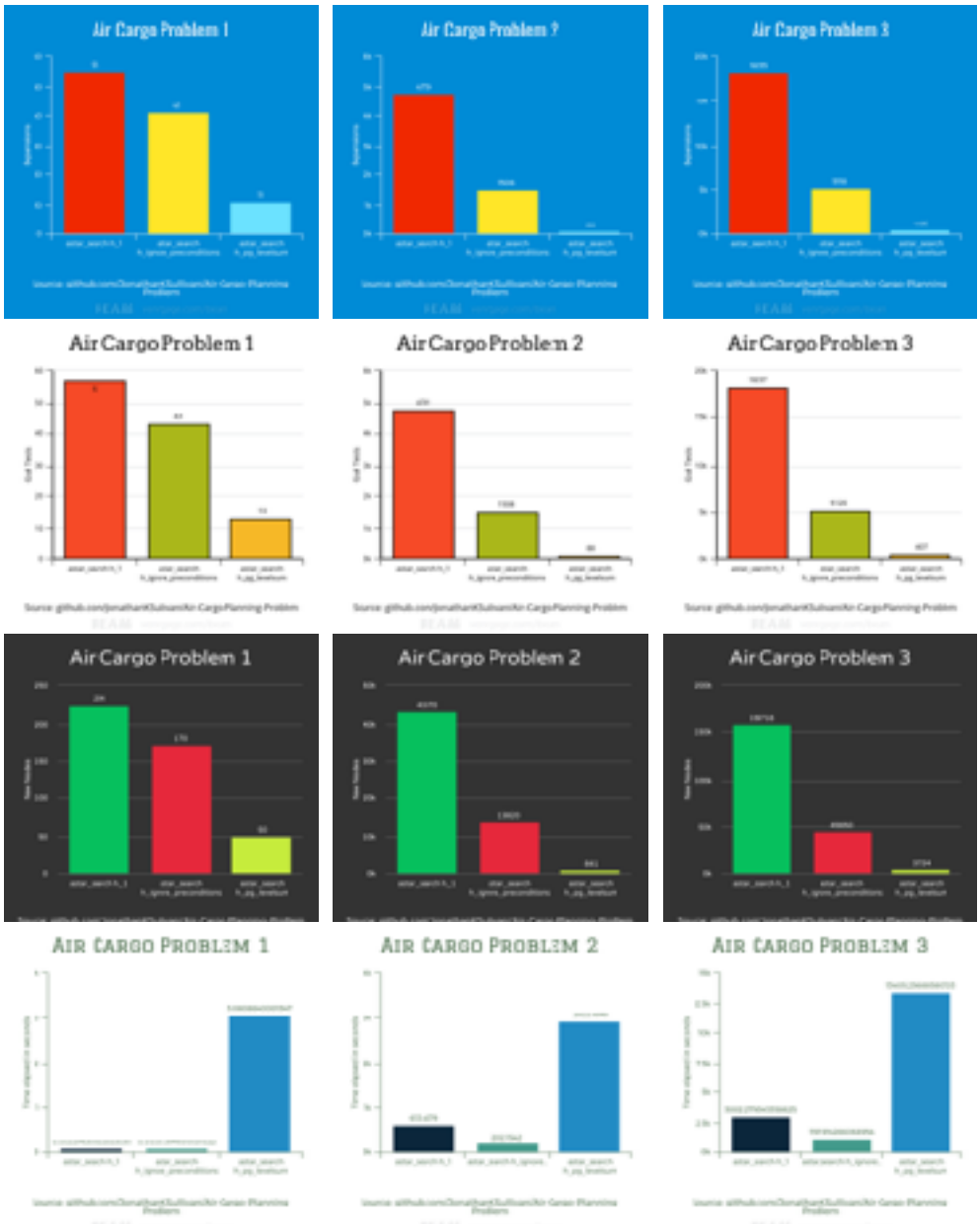
Non-Heuristic Search Result Metrics



Above is a visualization of the metrics taken during our experiments of planning algorithms done without heuristics. The first thing I think one should note is that is the row with plan lengths.

Notice that only breadth_first_search, breadth_first_tree_search, and uniform_cost_search find the optimal plan. Out of these three only breadth_first_search and uniform_cost_search are fast enough to complete on more complex problems. In fact on problem 1 breadth_first_search took 5 seconds to complete while the average time for the other planning algorithm was around 0.06665 seconds. This leaves us with 2 viable algorithms that we could implement to solve this problem. Out of these 2 breadth_first_search seems to be the best choice, because it expands less nodes, creates less new nodes, runs more goal tests and completes in a fraction of the time. Also the depth_limited_search took to long to complete on problem 3.

Heuristic Search Result Metrics using A*



Above is a visualization of the metrics taken during our experiments of planning algorithms implementing an a* search using 3 different heuristics. We the h1 heuristic, the "ignore preconditions" heuristic and the "level-sum" heuristic. Implementing these with A* search always yielded an optimal solution. When it comes to speed using a level sum heuristic is too slow however it is very efficient in how it uses memory. astar_search h_ignore_preconditions seems to be the fast to of the bunch and also more memory efficient than h_pg_levelsum. Out of these two I think it would definitely be matter of hardware on choosing which to implement. If speed is less of a concern than memory astar_search with h_pg_levelsum would be the way to go however If memory is less of a concern than speed astar_search with h_ignore_preconditions would be the way to go

Best Heuristic

In my opinion h_ignore_preconditions is the best heuristic to use because it is not computationally intensive and can be derived from relaxing the problem. I do think better heuristics could be used. I mainly state this because it only performed slightly better than the breath_first_search without heuristics.