Eric Iacutone

Heuristic Analysis, Air Cargo Planning

**Problem 1**

**Initial State:**

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:**

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



**Problem 2**

**Initial 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 Path:**

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 3**

**Initial 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 Path:**

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)

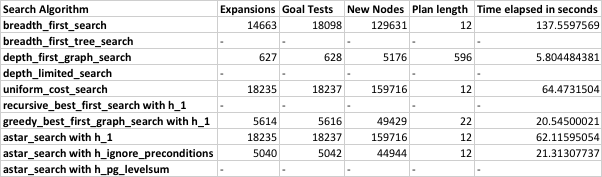
Load(C3, P1, ATL)

Fly(P1, ATL, JFK)

Unload(C3, P1, JFK)

Unload(C2, P2, SFO)

Unload(C1, P1, JFK)



**Results**

For simple problems, both breadth-first and A\* search both find a solution in a reasonable time period with an equal plan length. As the planning graph complexity increases, the increase in expansions, goal tests, new nodes and time elapsed varies drastically among the search algorithms. The results indicate that a depth-first graph search is the fastest for the problem 3 (contains the most complexity). However, this algorithm also has the longest plan length. Since plan length is import, we can discard depth-first search results. Both breadth-first search and A\* search find equal plan lengths for the problem set. However, as complexity grows, A\* search finds an optimal path much faster then breadth-first search. Another advantage to A\* search is the ability to code an informed heuristic, with the h\_ignore\_preconditions heuristic reducing the amount of time to find the optimal plan length.

The definition of h\_ignore\_preconditions is “this heuristic estimates the minimum number of actions that must be carried out from the current state in order to satisfy all of the goal conditions by ignoring the preconditions required for an action to be executed.” The definition of h\_pg\_levelsum is “this heuristic uses a planning graph representation of the problem state space to estimate the sum of all actions that must be carried out from the current state in order to satisfy each individual goal condition.” I believe the h\_pg\_levelsum is more a more complex heuristic to calculate; it takes into account all the actions in the planning graph. The heuristic becomes much more memory intensive to perform and therefore slower than h\_ignore\_preconditions.