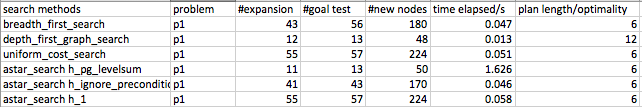
Problem 1



Optimal Path Length: 6

Optimal Plan: astar\_search h\_ignore\_precondition

Load(C1, P1, SFO)

Load(C2, P2, JFK)

Fly(P1, SFO, JFK)

Fly(P2, JFK, SFO)

Unload(C1, P1, JFK)

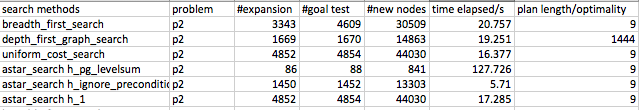
Unload(C2, P2, SFO), JFK)

Fly(P2, JFK, SFO)

Unload(C1, P1, JFK)

Unload(C2, P2, SFO)

Problem 2



Optimal Path Length: 9

Optimal Plan: astar\_search h\_ignore\_precondition

Load(C2, P2, JFK)

Load(C1, P1, SFO)

Load(C3, P3, ATL)

Fly(P2, JFK, SFO)

Unload(C2, P2, SFO)

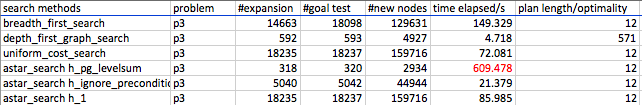
Fly(P1, SFO, JFK)

Unload(C1, P1, JFK)

Fly(P3, ATL, SFO)

Unload(C3, P3, SFO)

Problem 3



Optimal Path Length: 12

Optimal Plan: astar\_search h\_ignore\_precondition

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)

Fly(P1, ATL, JFK)

Unload(C4, P2, SFO)

Unload(C3, P1, JFK)

Unload(C1, P1, JFK)

Unload(C2, P2, SFO)

I will briefly evaluate all search methods here according to their performance.

1. Breadth-first-search will reach optimal solution (in the search section Video 10,11), but it’s time consuming due to exploring too many nodes for shortest path.
2. Depth-first-search in contrast, won’t search for optimal solution (AIMA 3.4.3) and therefore produced higher length path. So this one is always passed.
3. Uniform\_cost\_search and astar\_h1, both expand the same most number of nodes and both used best-first-graph search method. Compared to breadth-first-search, these search methods are not satisfied with reaching a goal, but also looking for best path (Search section, Video 16). Therefore, the search will be more thorough and time-consuming than BFS, but the result is guaranteed to be optimal.
4. A\* algorithm is implemented with 3 different heuristic functions. The essence of the algorithm is to expand the path with minimal function value f (AIMA 3.5.2) where:

f = g + h

g = path cost

h = estimated cost to goal

The best heuristics used is the h1 ignore\_precondition and it’s also the best algorithm in solving all the 3 problems, in that

1. Among all the search algorithm and heuristics that produce the optimal plan, h1 ignore\_precond explored the least number of nodes and reached the result with least amount of time-elapsed.
2. Compared to more advanced heuristic function h\_pg\_levelsum, h\_ignore\_precond is easier to compute and thus computationally less expensive, since h\_pg\_levelsum has to go through multiple levels for goal test and h\_ignore\_precond only need to deal with the current state level.
3. The advantage of h\_ignore\_precond over breadth-first search is not that obvious when dealing with problem 1 and search space is still small enough for brutal-force method. However, as the complexity explodes exponentially, using a heuristic function saves a lot of time.