Heuristic Analysis on Air Cargo Problem

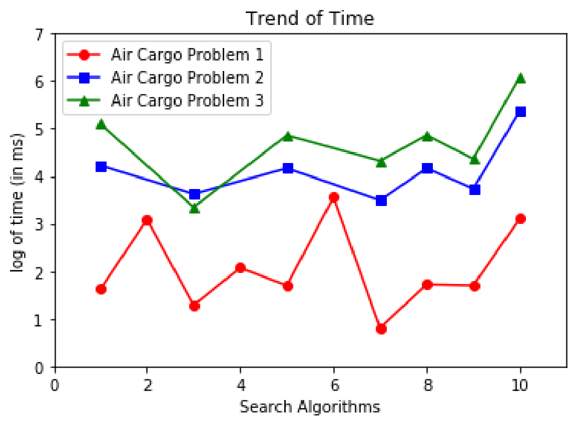
As part of fulfillment of project 3 in AIND, Udacity

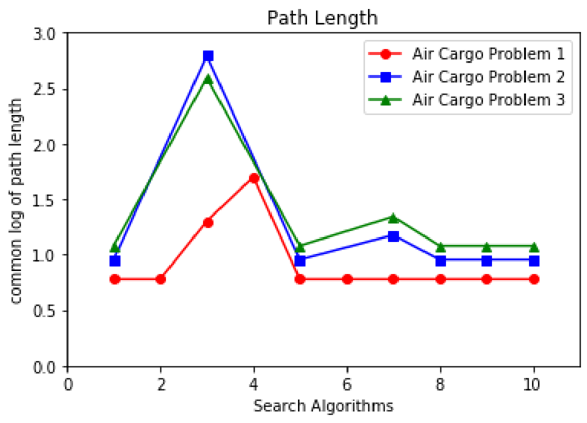
(A detailed introduction and template code can be found in https://github.com/udacity/AIND-Planning)

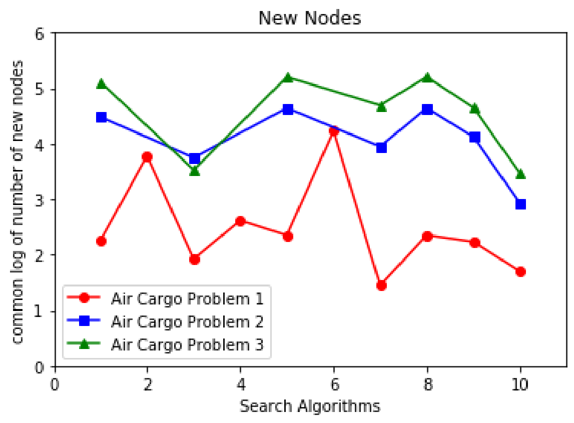
Introduction

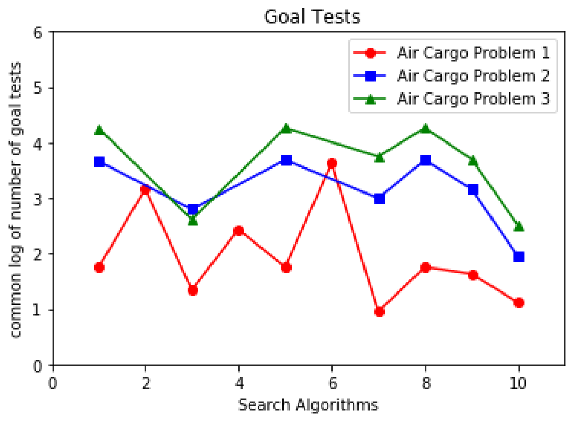
We define 3 problems, with increasing complexity, in classical PDDL (Planning Domain Definition Language) in the air cargo domain. We complete the skeleton code given in github to set up the problems for 10 different searches, with some equipped with automatically generated heuristics, to solve the problems. We recorded and plotted our experiment results in log scale in the following:

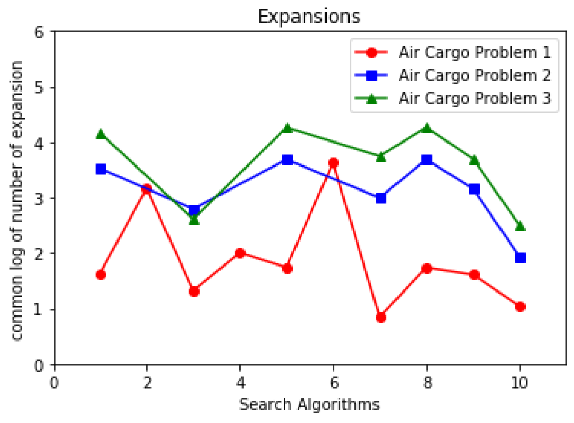
```
Problems
-----------------
    1. Air Cargo Problem 1
    2. Air Cargo Problem 2
    3. Air Cargo Problem 3

Search Algorithms
-----------------
    1. breadth_first_search
    2. breadth_first_tree_search
    3. depth_first_graph_search
    4. depth_limited_search
    5. uniform_cost_search
    6. recursive_best_first_search h_1
    7. greedy_best_first_graph_search h_1
    8. astar_search h_1
    9. astar_search h_ignore_preconditions
    10. astar_search h_pg_levelsum
```











Except for breadth-first tree search (algorithm 2), depth-limited search (algorithm 4), and recursive best-first search (algorithm 6) on solving problem 2 and 3, each of the other agent-problem pairs is able to search to a solution within 25 mins, with performance recorded in time elapsed, optimality of the solutions, and the number of node creation, expansion, and goal testing. The chart on log of path length shows which agent-problem pair search to an optimal solution. The chart on log of number of new nodes and goal test differed only very slightly as expected, and both are merely a vertical translation of the chart on log of expansions. This is because the number of nodes is roughly a constant (the average number of possible action) times the number of expansions.

Uninformed Planning Algorithms

Three uninformed planning algorithms passing all problems are breath-first search (BFS, algorithm 1), depth-first search (DFS, algorithm 3) and uniform-cost search (algorithm 5). By solutions of breath-first search listing in the appendix, an optimal solution respectively to problem 1, 2, and 3 should have path length 6, 9, and 12 (see section 3.4.1 of [1] for more properties of BFS). Depth-first search had the shortest time elapsed in two out of three test, but does not obtain an optimal solution in any case, which is an expected result (DFS terminates once a solution is found and does not always give optimal solution, see for example third edition AIMA's section 3.4.3 [1]). By chance, depth-first search solve the more complex problem 3 more effectively than problem 2, with less time and less node expansion.

For breath-first search and uniform-cost search, both of them search to an optimal solution in each problem in similar time and with similar number of node expansion. Probably, the problems are not complex enough to show advantages of uniform-cost search over breath-first search, nor to show the opposite.

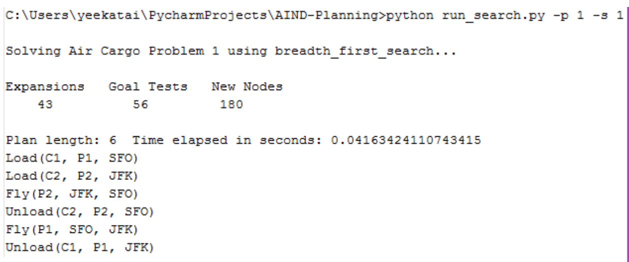
Automatic Heuristics

Three automatic heuristics with A*-search for planning passing all problems are a constant heuristic h1 serving as a control experiment (algorithm 8), ignore-preconditions heuristic (algorithm 9) and level-sum heuristic from the planning graph (algorithm 10). All experimental results found by A*-search are optimal solutions, which is natural if heuristic is not over-estimating. (Note that it is not guarantee that level-sum heuristic is not over-estimating, see section 10.3.1 in [1] that level-sum could be sometimes inadmissible.) The time lapse for these 3 problems are generally longer for automatic heuristics than uninformed planning algorithms, possibly due to more computation involved. However the growth rates with respect to problem complexity for these three automatic heuristics is generally slightly lower than that of uniform-cost search. For real problem with complexity orders of magnitude higher than our experiment, it is unsurprising to see (indeed we could expect) automatic heuristics with A*-search outperform uninformed planning algorithms.

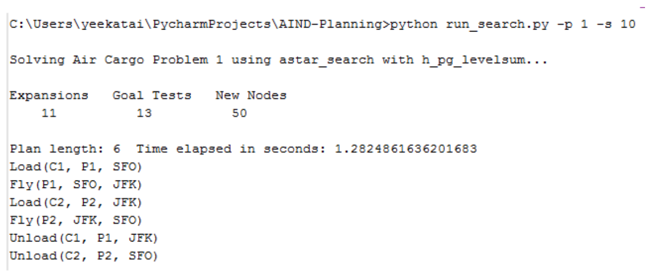
The superiority of automatic heuristics approaches can be better seen in the number of expansions involved. To start with, the constant heuristic had roughly the same number of expansion as breath-first search. Then, the number of expansion by using ignore-preconditions heuristic is reduced to only a fraction. Finally, by recording mutexes to point out difficult interactions, and by simplifying computation with level-sum in planning graph, level-sum heuristic is the most accurate heuristic in our experiment. Such approach was able to reduce the number of expansion by more than one order of magnitude (say in problem 3, it's one-sixtieth of that of the constant heuristic). While our results agree with some properties of planning graph stated in AIMA's section 10.3 [1], we urge interested reader to consult the cited book for a thorough study, or to take the program AIND in Udacity [2] if you are as well looking for a MOOC experience.

Appendix: Selected Raw Results

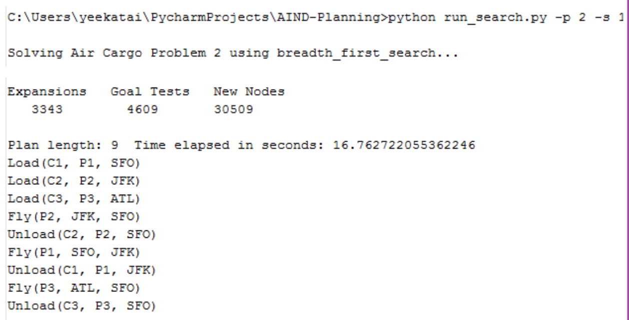
We state raw results of algorithms 1 and 10 in the following. In particular, two optimal sequences of actions are included for each of the problems.



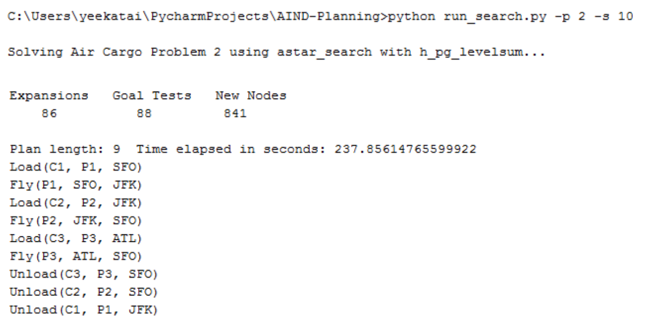
```
C:\Users\yeekatai\PycharmProjects\AIND-Planning>python run_search.py -p 1 -s 1

Solving Air Cargo Problem 1 using breadth_first_search...

Expansions   Goal Tests   New Nodes
    43          56         180

Plan length: 6  Time elapsed in seconds: 0.04163424110743415
Load(C1, P1, SFO)
Load(C2, P2, JFK)
Fly(P2, JFK, SFO)
Unload(C2, P2, SFO)
Fly(P1, SFO, JFK)
Unload(C1, P1, JFK)
```



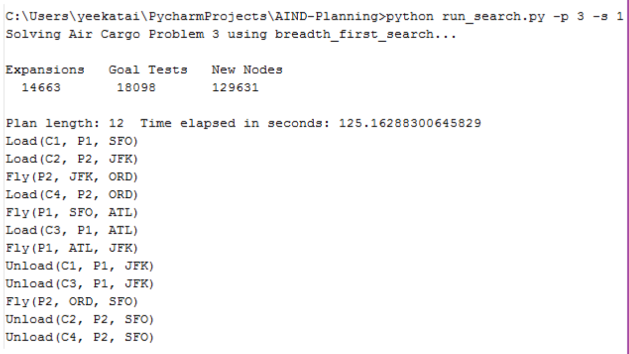
```
C:\Users\yeekatai\PycharmProjects\AIND-Planning>python run_search.py -p 1 -s 10

Solving Air Cargo Problem 1 using astar_search with h_pg_levelsum...

Expansions   Goal Tests   New Nodes
    11          13          50

Plan length: 6  Time elapsed in seconds: 1.2824861636201683
Load(C1, P1, SFO)
Fly(P1, SFO, JFK)
Load(C2, P2, JFK)
Fly(P2, JFK, SFO)
Unload(C1, P1, JFK)
Unload(C2, P2, SFO)
```



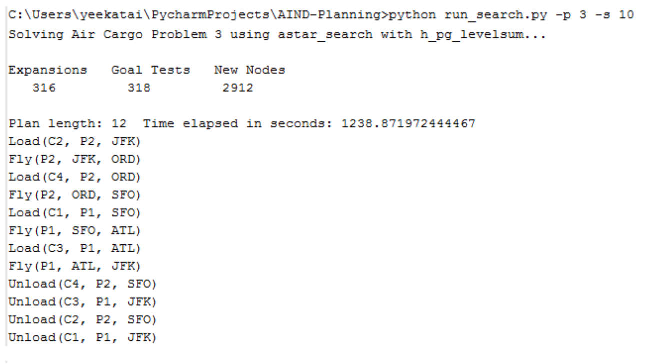
```
C:\Users\yeekatai\PycharmProjects\AIND-Planning>python run_search.py -p 2 -s 1

Solving Air Cargo Problem 2 using breadth_first_search...

Expansions   Goal Tests   New Nodes
   3343        4609       30509

Plan length: 9  Time elapsed in seconds: 16.762722055362246
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)
```



```
C:\Users\yeekatai\PycharmProjects\AIND-Planning>python run_search.py -p 2 -s 10

Solving Air Cargo Problem 2 using astar_search with h_pg_levelsum...

Expansions   Goal Tests   New Nodes
    86          88         841

Plan length: 9  Time elapsed in seconds: 237.85614765599922
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)
```



```
C:\Users\yeekatai\PycharmProjects\AIND-Planning>python run_search.py -p 3 -s 1
Solving Air Cargo Problem 3 using breadth_first_search...

Expansions   Goal Tests   New Nodes
  14663       18098       129631

Plan length: 12  Time elapsed in seconds: 125.16288300645829
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)
```



```
C:\Users\yeekatai\PycharmProjects\AIND-Planning>python run_search.py -p 3 -s 10
Solving Air Cargo Problem 3 using astar_search with h_pg_levelsum...

Expansions   Goal Tests   New Nodes
   316         318        2912

Plan length: 12  Time elapsed in seconds: 1238.871972444467
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)
```

References

[1] Russell, Stuart; Norvig, Peter. Artificial Intelligence: A Modern Approach. Prentice Hall.

[2] Artificial Intelligence Nanodegree, Udacity. Retrieved Oct 25, 2017, from url: https://www.udacity.com/course/artificial-intelligence-nanodegree–nd889