heuristic_analysis

January 6, 2018

1 Air Cargo Planning Problem Heuristic Analysis

Author: Michael C. J. Kao

This report analysis three problems of various difficulty in the Airplane cargo planning setting as outlined in the README.md

We first analysis the problem one by one with all ten searches tested, we then provide a conclusion on the overall performance and the comparison between heuristic and non-heuristic methods.

```
In [1]: import heuristic_helper as hh
        all_search_indices = range(1, 11)
```

2 Search Comparison

In each of the subsection below, the optimal solution is shown along with the result of each search algorithm. Methods which fails to obtain a solution within the 10 minutes time constraint are not shown.

We compare the `optimality`, `memory consumption` and `speed` of each method for each problem.

2.1 Problem 1:

Aircraft Number: 2

Cargo Number: 2

Airport Number: 2

```
In [2]: hh.create_result_dataframe(problem_index=1, search_index=all_search_indices)
```

```
Optimal Path Length: 6

Optimal Plan:

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

```
Unload(C1, P1, JFK)
```

```
Out[2]:                                             Nodes Expanded  Path Length  \
        breadth_first_search                                  180            6
        breadth_first_tree_search                            5960            6
        depth_first_graph_search                               84           20
        depth_limited_search                                  414           50
        uniform_cost_search                                   224            6
        recursive_best_first_search (param=h_1)             17023            6
        greedy_best_first_graph_search (param=h_1)             28            6
        astar_search (param=h_1)                              224            6
        astar_search (param=h_ignore_preconditions)           170            6
        astar_search (param=h_pg_levelsum)                    158            6

                                                     Time (s)  optimal
        breadth_first_search                           0.0439     True
        breadth_first_tree_search                      0.8655     True
        depth_first_graph_search                       0.0112    False
        depth_limited_search                           0.0719    False
        uniform_cost_search                            0.0295     True
        recursive_best_first_search (param=h_1)        2.4794     True
        greedy_best_first_graph_search (param=h_1)     0.0042     True
        astar_search (param=h_1)                       0.0316     True
        astar_search (param=h_ignore_preconditions)    0.0302     True
        astar_search (param=h_pg_levelsum)             0.9108     True
```

In the simple task above, all algorithm were able to solve the problem under the 10 minutes time constraint.

- Optimality: All achieved optimal solution except `depth first graph search` and `depth limited search`.
- Time: `depth first graph search` had the lowest time, while `recursive best first search` had the longest.
- Memory: `recursive best first search` and `breadth first tree search` both has significant number of nodes than the other methods.

2.2 Problem 2:

Aircraft Number: 3
Cargo Number: 3
Airport Number: 3

```
In [3]: hh.create_result_dataframe(problem_index=2, search_index=all_search_indices)
```

```
Optimal Path Length: 9

Optimal Plan:
```

```
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)
```

```
Out[3]:                                           Nodes Expanded Path Length  \
        breadth_first_search                                30509           9
        breadth_first_tree_search                               -           -
        depth_first_graph_search                             5602         619
        depth_limited_search                                    -           -
        uniform_cost_search                                 44030           9
        recursive_best_first_search (param=h_1)                 -           -
        greedy_best_first_graph_search (param=h_1)           8910          27
        astar_search (param=h_1)                            44030           9
        astar_search (param=h_ignore_preconditions)         13303           9
        astar_search (param=h_pg_levelsum)                  10232           9

                                                       Time (s)  optimal
        breadth_first_search                             6.6884     True
        breadth_first_tree_search                             -    False
        depth_first_graph_search                         3.1632    False
        depth_limited_search                                  -    False
        uniform_cost_search                              9.4703     True
        recursive_best_first_search (param=h_1)               -    False
        greedy_best_first_graph_search (param=h_1)       1.9175    False
        astar_search (param=h_1)                         9.3552     True
        astar_search (param=h_ignore_preconditions)      3.3819     True
        astar_search (param=h_pg_levelsum)             259.5805     True
```

For this problem, the `breadth first tree search`, `depth limited search` and `recursive best first search` were unable to complete under the time constraint.

- Optimality: `depth first graph search` and `greedy best first graph search` did not obtain optimal plan.
- Time: `depth first graph search` had the lowest time again, while `A* star search with level sum heuristic` had the longest within the time constraint.
- Memory: `depth first graph search` uses the least memory followed by `greedy best first graph search`.

2.3 Problem 3:

Aircraft Number: 2

Cargo Number: 4
Airport Number: 4

```
In [4]: hh.create_result_dataframe(problem_index=3, search_index=all_search_indices)
```

```
Optimal Path Length: 12

Optimal Plan:

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)
```

```
Out[4]:                                            Nodes Expanded Path Length  \
        breadth_first_search                               128605          12
        breadth_first_tree_search                               -           -
        depth_first_graph_search                             3364         392
        depth_limited_search                                    -           -
        uniform_cost_search                                158272          12
        recursive_best_first_search (param=h_1)                 -           -
        greedy_best_first_graph_search (param=h_1)          48822          26
        astar_search (param=h_1)                           158272          12
        astar_search (param=h_ignore_preconditions)         44769          12
        astar_search (param=h_pg_levelsum)                      -           -

                                                      Time (s)  optimal
        breadth_first_search                           33.6191     True
        breadth_first_tree_search                            -    False
        depth_first_graph_search                        1.4277    False
        depth_limited_search                                 -    False
        uniform_cost_search                            41.0995     True
        recursive_best_first_search (param=h_1)              -    False
        greedy_best_first_graph_search (param=h_1)     13.1612    False
        astar_search (param=h_1)                       43.5781     True
        astar_search (param=h_ignore_preconditions)    14.4305     True
        astar_search (param=h_pg_levelsum)                   -    False
```

In this final problem, the `breadth first tree search`, `depth limited search`, `recursive best first search` and `A* search with level sum heuristic` did not finish.

- Optimality: All achieved optimal solution except `depth first graph search` and `greedy best firs graph search`.
- Time: `depth first graph search` has a significant less time in comparison to other problems.
- Memory: `depth first graph search` was the most memory efficient of all method completed.

3 Search Method Discussion

The question which method is better really depends on the context, although the analysis exposes the weaknesses of certain methods, there are several contenders when choosing the most applicable method.

In general, `depth first graph search` has been the fastest and also the most memory efficient method in the study. Yet, the solution is also far from optimal. In problem 3, it takes 392 steps in contrast to the optimal 12 steps.

If optimality is essential, than the `A* search with precondition heuristic` is a good alternative. It found optimal solution in all three problems with the lowest resource and time cost in comparison to other optimal methods such as `breadth first search`.