Heuristics for Planning agent

Oleg Medvedev

January, 2018

Introduction

This report summarizes the application of Classical Planning to solving the logistic planning problem for cargo transportation. We consider three problems of moving cargo between 2-4 airports with 2-3 planes. The problems are defined

Problems and optimal sequence

Problem 1 is very straightforward. It involves three actions per cargo - loading, flying and unloading. So a path of 6 actions is the optimal one.

Problem 2 is similar to problem 1 in that it requires three actions per each cargo, so a total of 9 actions.

Problem 3 is slightly more complicated as there are four cargoes and only two planes. There are only two destinations, so there are two pairs of two cargoes which need to be picked up in different places and delivered to the same airport. Each plane has to load a cargo, fly to another city, load again and then fly to destination and unload both cargoes, so (2+2+2) 6 actions per plane or 12 total.

Uninformed agents

As per the assignment we validated three different uninformed search algorithms for each of the three problems. We considered breadth-first search (BFS), depth-first search (DFS) and uniform cost search (UCS). The results are summarized below. Each cell contains the three numbers - first representing the length of the plan, second is the number of expanded nodes and third is the run time in seconds. Additionally all raw results are presented in the Appendix.

Problem	BFS	DFS	UCS
Problem 1	6 / 43 / 0.04s	20 / 21 / 0.02s	6 / 55 / 0.05s
Problem 2	9 / 3343 / 9.82s	619 / 624 / 4.06s	9 / 4852 / 13.16s

Problem	BFS	DFS	UCS
Problem 3	12 / 14663 / 48.3s	392 / 408 / 1.95s	12 / 18236 / 60.37s

In all cases both BFS and UCS managed to find the optimal solution, although BFS did it with slightly less node expansions in all cases. DFS didn't manage to converge on the optimal solution in neither of problems, it was however the fastest among all three. It agrees well with the lectures is that BFS and UCS are both guaranteed to find the best solution, although at the expense of more expansions and longer compute time. DFS on contrary returns the first path reaching a goal it manages to find, so it does so in only few expansions, however the path is not guaranteed to be the optimal one and we see it in all three cases.

Informed agents

The table below summarizes the results for A* agent with two different heuristics. One is with automatically generated ignore preconditions heuristics, which calculates the number of actions required to achieve all remaining goals, ignoring all positive and negative preconditions. The second one is the level sum heuristics using the planning graph.

Problem	Ignore preconditions	PG level sum
Problem 1	6 / 41 / 0.04s	6 / 11 / 0.91s
Problem 2	9 / 1450 / 4.35s	9 / 86 / 196.6s
Problem 3	12 / 5040 / 16.91s	12 / 315 / 934s

One may see that A* with ignore preconditions converges on the optimal path and does so in about two times less nodes extensions and two times faster for problems 2 and 3. The heuristics, albeit quite simple, helps to rate possible nodes and their distance from the goal state, thus allows reducing the number of expansions and computational time.

Planning graph level sum heuristics provides the smallest number of node expansions, at least one order of magnitude less than A* and more comparing to uninformed agents. The run time is quite long, however, which is the result of a fairly complex algorithm. It may likely be improved quite a bit.

Summary

Five different agents were considered for each of the three problems - three uninformed and two informed with heuristics. All agents except depth-first search managed to converge on the optimal solution, which is expected. DFS provided the fastest run time and one of the smallest number of node expansions, which was also expected. Breadth-first search had the best overall performance among uninformed agents.

Switching to informed agents with heuristics reduced the number of node expansions significantly. A* with relatively simple automatically generated "ignore preconditions" heuristics reduced the number of node expansions by three times. The level sum heuristics based on planning graph provided an order of magnitude improvement in node expansions at the expense of the run time, as it is spent on constructing the approximation of the search tree.

Appendix A - Raw results for uninformed agents

Case P1-S1

```
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: 0.039851423003710806
Load(C1, P1, SFO)
Load(C2, P2, JFK)
Fly(P2, JFK, SFO)
Unload(C2, P2, SFO)
Fly(P1, SFO, JFK)
Unload(C1, P1, JFK)
```

Case P1-S3

```
python run_search.py -p 1 -s 3`

Solving Air Cargo Problem 1 using depth_first_graph_search...

Expansions   Goal Tests   New Nodes
    21          22          84

Plan length: 20  Time elapsed in seconds: 0.0190653299796395
Fly(P1, SFO, JFK)
Fly(P2, JFK, SFO)
Load(C2, P1, JFK)
Fly(P1, JFK, SFO)
Fly(P2, SFO, JFK)
Unload(C2, P1, SFO)
Fly(P1, SFO, JFK)
Fly(P2, JFK, SFO)
Load(C2, P2, SFO)
Fly(P1, JFK, SFO)
Load(C1, P2, SFO)
Fly(P2, SFO, JFK)
Fly(P1, SFO, JFK)
Unload(C2, P2, JFK)
Unload(C1, P2, JFK)
Fly(P2, JFK, SFO)
```

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

Case P1-S5

```
python run_search.py -p 1 -s 5

Solving Air Cargo Problem 1 using uniform_cost_search...

Expansions   Goal Tests   New Nodes
    55          57         224

Plan length: 6  Time elapsed in seconds: 0.04902327200397849
Omitted as it is the same as P1-S1
```

Case P2-S1

```
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: 9.823018399998546
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)
```

Case P2-S3

```
python run_search.py -p 2 -s 3

Solving Air Cargo Problem 2 using depth_first_graph_search...
```

```
Expansions   Goal Tests   New Nodes
   624         625         5602

Plan length: 619  Time elapsed in seconds: 4.058151098026428
Omitted as too long
```

Case P2-S5

```
python run_search.py -p 2 -s 5

Solving Air Cargo Problem 2 using uniform_cost_search...

Expansions   Goal Tests   New Nodes
   4852        4854       44030

Plan length: 9  Time elapsed in seconds: 13.163193234999198
Omitted as it is the same as P2-S1
```

Case P3-S1

```
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: 48.30484216596233
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)
```

Case P3-S3

```
python run_search.py -p 3 -s 3
```

```
Solving Air Cargo Problem 3 using depth_first_graph_search...

Expansions   Goal Tests   New Nodes
   408         409         3364

Plan length: 392  Time elapsed in seconds: 1.9534217299660668
Omitted as too long
```

Case P3-S5

```
python run_search.py -p 3 -s 5

Solving Air Cargo Problem 3 using uniform_cost_search...

Expansions   Goal Tests   New Nodes
  18236       18238       159726

Plan length: 12  Time elapsed in seconds: 60.371445766999386
Omitted as it is similar to P3-S1
```

Appendix B - Raw results for agents with heuristics

Case P1-S9

```
python run_search.py -p 1 -s 9

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

Expansions   Goal Tests   New Nodes
    41          43         170

Plan length: 6  Time elapsed in seconds: 0.036806308024097234
Load(C1, P1, SFO)
Fly(P1, SFO, JFK)
Unload(C1, P1, JFK)
Load(C2, P2, JFK)
Fly(P2, JFK, SFO)
Unload(C2, P2, SFO)
```

Case P2-S9

```
python run_search.py -p 2 -s 9
```

```
Solving Air Cargo Problem 2 using astar_search with h_ignore_preconditions...

Expansions   Goal Tests   New Nodes
   1450        1452       13303

Plan length: 9  Time elapsed in seconds: 4.347497334994841
Load(C3, P3, ATL)
Fly(P3, ATL, SFO)
Unload(C3, P3, SFO)
Load(C1, P1, SFO)
Fly(P1, SFO, JFK)
Unload(C1, P1, JFK)
Load(C2, P2, JFK)
Fly(P2, JFK, SFO)
Unload(C2, P2, SFO)
```

Case P3-S9

```
python run_search.py -p 3 -s 9

Solving Air Cargo Problem 3 using astar_search with h_ignore_preconditions...

Expansions   Goal Tests   New Nodes
   5040        5042       44944

Plan length: 12  Time elapsed in seconds: 16.913793624960817
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)
```

Case P1-S10

```
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: 0.9059329880401492
```

Case P2-S10

```
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: 196.56146006798372
```

Case P3-S10

```
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
   315         317         2902

Plan length: 12  Time elapsed in seconds: 934.1108886679867
```