Implement a Planning Search - heuristic analysis report

Guy Koren

guy.koren@gmail.com

Introduction

The goal of this project is to develop a planning search agent to solve deterministic logistics planning problems for an Air Cargo transport system. In the lectures we developed planning search agent where the heuristics were domain dependent (specific to navigation problems). In this project, the goal is to develop domain-independent heuristics.

We're provided with the Air Cargo domain action schema and 3 initial states that defined 3 planning problems.

Heuristics analysis

Optimal plan for problems 1,2,3

Following are the optimal plans (in terms of plan length) for the three problems.

The plans were constructed using Breadth First Search (search index 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: 0.20419625600334257

Load(C2, P2, JFK)

Load(C1, P1, SFO)

Fly(P2, JFK, SFO)

Unload(C2, P2, SFO)

Fly(P1, SFO, JFK)

Unload(C1, P1, JFK)

Solving Air Cargo Problem 2: using breadth_first_search...

Expansions Goal Tests New Nodes

3343 4609 30509

Plan length: 9 Time elapsed in seconds: 54.64369069796521

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)

Solving Air Cargo Problem 3: using breadth_first_search...

Expansions Goal Tests New Nodes

14663 18098 129631

Plan length: 12 Time elapsed in seconds: 335.9418767300085

Load(C2, P2, JFK)

Load(C1, P1, SFO)

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)

Non-heuristic search result metrics comparison

Problem 1

Search Strategy	Path Length (optimal?)	Node Expansions	Time Elapsed [sec]
Breadth First Search (1)	6 (Y)	43	0.18
Depth First Graph Search (3)	12 (N)	12	0.05
Uniform Cost Search (5)	6 (Y)	55	0.22

Problem 2

Search Strategy	Path Length (optimal?)	Node Expansions	Time Elapsed [sec]
Breadth First Search (1)	9 (Y)	3343	54.9
Depth First Graph Search (3)	575 (N)	582	10.29
Uniform Cost Search (5)	9 (Y)	4852	71.92

Problem 3

Search Strategy	Path Length (optimal?)	Node Expansions	Time Elapsed [sec]
Breadth First Search (1)	12 (Y)	14663	341.49
Depth First Graph Search (3)	596 (N)	627	12.65
Uniform Cost Search (5)	12 (Y)	18234	334.04

The tables above shows the results of 3 search strategy on the 3 problems.

We can observe that both Breadth First Search (1) and Uniform Cost Search (5) are optimal in terms of path length.

In terms of compute resources, the DFS (3) is most efficient but yields poor results in terms of path length.

In problems where the costs of all steps are equal (as in our case), BFS and UCS has almost the same complexity (O(b^d) vs O(b^(d+1)). Thus, **BFS is the preferred search method in this case**.

Heuristic search result metrics comparison

Although h_1 is not a real heuristic and should have practically be presented in the non-heuristic approach comparison, when used with A* we have presented it here as well:

Problem 1

Search Strategy	Path Length (optimal?)	Node Expansions	Time Elapsed [sec]
A* with h_1	6 (Y)	55	0.24
A* with h_ignore_preconditions	6 (Y)	41	0.18
A* with h_pg_levelsum	6 (Y)	55	0.22

Problem 2

Search Strategy	Path Length (optimal?)	Node Expansions	Time Elapsed [sec]
A* with h_1	9 (Y)	4852	73.08
A* with h_ignore_preconditions	9 (Y)	1450	22.8
A* with h_pg_levelsum	9 (Y)	86	206.88

Problem 3

Search Strategy	Path Length (optimal?)	Node Expansions	Time Elapsed [sec]
A* with h_1	12 (Y)	18234	330.38
A* with h_ignore_preconditions	12 (Y)	5040	95.12
A* with h_pg_levelsum	12 (Y)	318	1133.06

The tables above shows the results of the A* search algorithm with 3 different heuristics tested on the same 3 problems. We can observe that A* yielded optimal path length result, with all 3 heuristics.

Except for the 3rd problem, where the h_pg_levelsum heuristic exceeded the time budget (10 min), all three heuristics found the optimal path within the time budget, with h_ignore_precondition being most time efficient. However, the h_pg_levelsum is most efficient with its Node expansions and its memory consumption.

If we need to choose one method to represent the heuristic search result, then **h_ignore_preconditions is the best**. (as meets time constraints in all problems)

Best Heuristic vs. Best Non-heuristic

As we chose the **Breadth First Search** to be the best among the non-heuristic search methods, and the **A* with h_ignore_preconditions** as the best among the heuristic search methods, the following tables compares the two on the 3 problems.

Problem 1

Search Strategy	Path Length (optimal?)	Node Expansions	Time Elapsed [sec]
Breadth First Search (1)	6 (Y)	43	0.18
A* with h_ignore_preconditions	6 (Y)	41	0.18

Problem 2

Search Strategy	Path Length (optimal?)	Node Expansions	Time Elapsed [sec]
Breadth First Search (1)	9 (Y)	3343	54.9
A* with h_ignore_preconditions	9 (Y)	1450	22.8

Problem 3

Search Strategy	Path Length (optimal?)	Node Expansions	Time Elapsed [sec]
Breadth First Search (1)	12 (Y)	14663	341.49
A* with h_ignore_preconditions	12 (Y)	5040	95.12

The tables above clearly shows that the A* with h_ignore_preconditions dominates the BFS in all 3 problems , with respect to the time efficiency and Nodes expansions.

They both are optimal in terms of path length.

Therefore the A* with h_ignore_preconditions is our method of choice to solve these 3 problems

Conclusion

This project compared several non-informed (non-heuristic) search methods to the informed (heuristic) search methods. The results show that informed search methods yield better results and consume less compute resources. In addition, they allow custom design of heuristics that provides flexibility in solving various problems.