Air Cargo Planning Heuristic Analysis

Date: Oct. 8, 2017

Name: Hiromasa Sakata

Project Name: Implement a Planning Search

1. Non-heuristic Search Result Metrics

The following tables describes the scores of non-heuristic search methods.

Search Method	Expansions	Goal Tests	New Nodes	Plan Length	Time (s)	Optimal
Breadth First Search	43	56	180	6	0.020	optimal
Depth First Graph Search	12	13	48	12	0.006	not optimal
Uniform Cost Search	55	57	224	6	0.025	optimal

Figure 1. Problem 1 Results (Non-heuristic Searches)

Search Method	Expansions	Goal Tests	New Nodes	Plan Length	Time (s)	Optimal
Breadth First Search	3343	4609	30509	9	9.138	optimal
Depth First Graph Search	1527	1528	13620	1357	7.764	not optimal
Uniform Cost Search	4853	4855	44041	9	7.871	optimal

Figure 2. Problem 2 Results (Non-heuristic Searches)

Search Method	Expansions	Goal Tests	New Nodes	Plan Length	Time (s)	Optimal
Breadth First Search	14663	18098	129631	12	65.330	optimal
Depth First Graph Search	4189	4190	35475	3955	45.508	not optimal
Uniform Cost Search	18223	18225	159618	12	34.739	optimal

Figure 3. Problem 3 Results (Non-heuristic Searches)

Figure 1-3 show that Breadth First Search and Uniform Cost Search are always optimal, and Depth First Search is always not optimal. This is because that Depth First Search expands the deepest node. It means that the search proceeds to the deepest level of the search tree. Therefore, this search takes more time than other methods and cannot reach the optimal solution. Comparing the two methods, BFS and UCS, BFS performed better little in problem 1 but UCS outperformed in problem 2 and 3. It means that UCS is the best method in the three non-heuristic searches.

2. Heuristic Search Result Metrics

The following tables describes the scores of heuristic search methods.

Search Method	Expansions	Goal Tests	New Nodes	Plan Length	Time (s)	Optimal
A* Search h_1	55	57	224	6	0.024	optimal
A* Search h_ignore_preconditions	41	43	170	6	0.024	optimal
A* Search h_pg_levelsum	11	13	50	6	0.468	optimal

Figure 4. Problem 1 Results (Heuristic Searches)

Search Method	Expansions	Goal Tests	New Nodes	Plan Length	Time (s)	Optimal
A* Search h_1	4853	4855	44041	9	7.542	optimal
A* Search h_ignore_preconditions	1450	1452	13303	9	2.863	optimal
A* Search h_pg_levelsum	86	88	841	9	42.862	optimal

Figure 5. Problem 2 Results (Heuristic Searches)

Search Method	Expansions	Goal Tests	New Nodes	Plan Length	Time (s)	Optimal
A* Search h_1	18223	18225	159618	12	33.983	optimal
A* Search h_ignore_preconditions	5040	5042	44944	12	11.153	optimal
A* Search h_pg_levelsum	325	327	3002	12	213.249	optimal

Figure 6. Problem 3 Results (Heuristic Searches)

Figure 4-6 show that all the heuristic search methods reached the optimal solutions in all problems. When it comes to required time, A* Search h_ignore_preconditons requires the shortest time, and A* Search h_pg_levelsum requires the longest time in all problems. Therefore, A* Search h_pg_levelsum is the best method among these heuristic methods.

3. Optimal Solutions

The optimal solutions are as below.

- **Problem 1**

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

- **Problem 2**

  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**

  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)

4. Discussion

By comparing Figure 1 and 4, 2 and 5, 3 and 6, it is turned out that the more difficult a problem is, the better the heuristic methods work. In problem 1, Breadth First Search (non-heuristic method) reached the solution fast. However, In problem 2 and 3, A* Search h_ignore_precondition(heuristic method) solved the problem faster.

According to Russel and Norvig (2010), The space complexity of BFS is $O(b^d)$ (b is the number of branches and d is the number of depth.) Therefore, the complexity increases exponentially as the number of depth increases.

When it comes to the optimality, BFS is always optimal. However, the tree-search version of A* is optimal if the heuristic function is admissible, while the graph-search version is optimal if the function is consistent.

In this experiment, it seems that you should use BFS for simple problems and A* Search for more complex problems.

5. Reference

Stuart J. Russel, Peter Norvig (2010), Artificial Intelligence: A Modern Approach (3rd Edition)