Air Cargo Planning: Heuristic Analysis

Kalai Ramea

March 19, 2017

1 Problem Statement

Three types of air cargo problems were given as part of this exercise. They are illustrated in Figures 1(a), 1(b) and 1(c) respectively. First air cargo problem is a simple switch of cargoes between two airports. The second problem is slightly more complex than the first–here a third airport is given, and two of the cargoes need to be shipped to SFO airport and one to JFK. The third problem has an another additional airport, and cargoes need to be shipped from each airport, and SFO and JFK will receive two cargoes each.

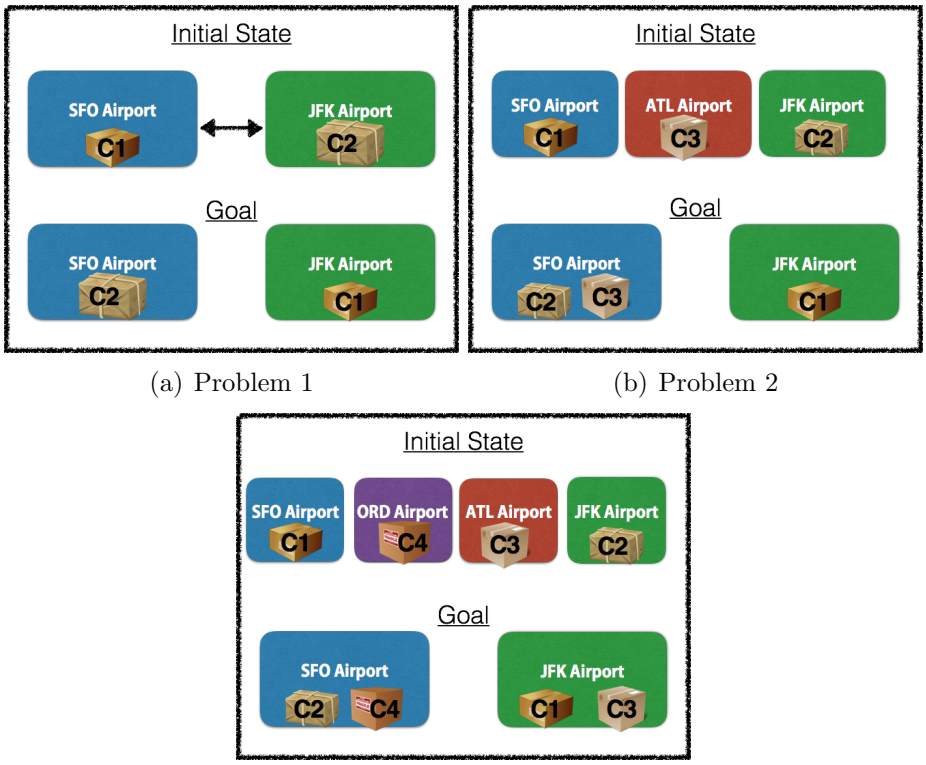
As part of this project, planning graphs were designed to solve these solutions under different search algorithms. This document analyses the effectiveness of various search techniques for these three problems, as well as the differences in heuristics analysis of A-star search algorithm for planning graphs.

2 Search Algorithms

The search algorithms considered for this heuristics analysis are (a) breath-first search, (b) depth-first graph search, and (c) greedy best first search approaches. Breath-first search explores the neighboring nodes in the state map before moving onto the next level, whereas depth-first algorithm searches all the levels from the root node first, and then backtracks to explore the neighboring nodes. Greedy best first search explores the most promising node at that specific point of time of searching.

These three searches will be analyzed for the following metrics:

- Nodes expanded in total during the search



(a) Problem 1 (b) Problem 2

(c) Problem 3

Figure 1: Illustration of Air Cargo Planning Problems

- Number of goal tests conducted
- Time elapsed
- Plan length–this will indicate whether the search resulted in optimality or not

Tables 1-4 show these attributes for the three types of air cargo planning problem. First, we can see that breath-first search traverses a lot of nodes to achieve a plan for the given problem amongst the three types of searches. Among the three depth-first does well if we are only considering node traversals. Interestingly, depth-first search algorithm traverses fewer number of nodes for problem 3 than problem 2, even though the complexity of problem 3 is higher than the other. This is because, since the search can from root to many different levels, if it finds a solution there, then it helps to have depth-first. Greedy best first search works well for simple problems, but as it gets more complex, we can see that it needs to traverse a lot of nodes to reach the result.

As far as optimality is concerned (see Table 8), breath-first search finds optimal solutions at 6, 9 and 12 plan lengths for the three air cargo problems. Greedy best first finds optimal solution for simpler problems (here problem 1), but does not provide optimal solutions for the rest. Depth-first search fails to find optimal solutions for all the three problems.

3 A-star Search with different Heuristic Functions

In the previous section, the search algorithms analyzed are called non-heuristic algorithms. In this section, an informed search algorithm, also called A-star algorithm will be analyzed for the three heuristic functions: (a) h_1 (a constant number for heuristic), (b) ignore preconditions heuristic, and (c) a level-sum heuristic. For the ignore precondition heuristic, the minimum number of actions that must be carried out from the current state in order to satisfy all of the goal conditions by ignoring the preconditions is estimated. For the level-sum heuristic, sum of all actions in the planning graph state that must be carried out from the current state in order to satisfy each individual goal condition is estimated.

Table 1: Nodes Expanded

Search Algorithm	**Problem 1**	**Problem 2**	**Problem 3**
Breath-First search	43	3343	14663
Depth-first graph	21	624	408
Greedy best first	7	998	5578

Table 2: Goal Tests

Search Algorithm	**Problem 1**	**Problem 2**	**Problem 3**
Breath-First search	56	4609	18098
Depth-first graph	22	625	409
Greedy best first	9	1000	5580

Table 3: Time Elapsed (seconds)

Search Algorithm	**Problem 1**	**Problem 2**	**Problem 3**
Breath-First search	0.05	20.51	143.46
Depth-first-graph	0.02	4.38	2.45
Greedy best first	0.01	9.14	184.73

Table 4: Plan Length

Search Algorithm	**Problem 1**	**Problem 2**	**Problem 3**
Breath-First search	6	9	12
Depth-first graph	20	619	392
Greedy best first	6	17	22

Table 5: Nodes Expanded

Heuristic	**Problem 1**	**Problem 2**	**Problem 3**
h_1	55	4853	18223
Ignore Preconditions	41	1506	5118
Level-Sum	11	86	**

Table 6: Goal Tests

Heuristic	**Problem 1**	**Problem 2**	**Problem 3**
h_1	57	4855	18225
Ignore Preconditions	43	1508	5120
Level-Sum	13	88	**

Table 7: Time Elapsed (seconds)

Heuristic	**Problem 1**	**Problem 2**	**Problem 3**
h_1	0.060	55.371	629.305
Ignore Preconditions	0.065	17.756	138.480
Level-Sum	3.050	373.472	**

Table 8: Plan Length

Heuristic	**Problem 1**	**Problem 2**	**Problem 3**
h_1	6	9	12
Ignore Preconditions	6	9	12
Level-Sum	6	9	**

Tables 5-8 show the attributes for the three types of air cargo planning problem for the three heuristics. It has to be noted that planning problem 3 took a long time to run with level-sum heuristic, so it is not reported here.

We can observe that a reasonable heuristic can drastically reduce the number of nodes traversed in the problem. Here level-sum heuristic does very well to reduce the number of nodes traversed amongst the three. However, one has to be mindful that with added information in the search, the search time increases as it can be noticed in these three searches. Another finding is, all the three types of searches find optimal solutions for the cargo problems (the third cargo problem was not completed, but we can assume that level-sum is capable of finding an optimal solution). This is a significant improvement compared to the non-heuristic search functions.

4 Summary

As a summary, we can see that except for depth-first search algorithms, the rest of them are comparable in their optimality. Greedy first does well for simpler problems, but gets worse as the problem becomes more complex. A-star algorithms do well for all the heuristics, but as the heuristics get complicated, the time taken to solve the problem increases. Hence, a trade-off decision needs to be made between computational time and precision while choosing the heuristic function.