Heuristic analysis for the Air Cargo planning problem

Abderrahim Kitouni

August 9, 2017

1 Optimal plans

For problem 1, the following sequence of actions is an optimal plan: `Load(C1, P1, SFO) -> Fly(P1, SFO, JFK) -> Load(C2, P2, JFK) -> Fly(P2, JFK, SFO) -> Unload(C1, P1, JFK) -> Unload(C2, P2, SFO)`.

For problem 2, this is an optimal plan: `Load(C1, P1, SFO) -> Fly(P1, SFO, JFK) -> Load(C2, P2, JFK) -> Fly(P2, JFK, SFO) -> Load(C3, P3, ATL) -> Fly(P3, ATL, SFO) -> Unload(C3, P3, SFO) -> Unload(C2, P2, SFO) -> Unload(C1, P1, JFK)`.

For problem 3, this is an optimal plan: `Load(C2, P2, JFK) -> Fly(P2, JFK, ORD) -> Load(C4, P2, ORD) -> Fly(P2, ORD, SFO) -> Load(C1, P1, SFO) -> Fly(P1, SFO, ATL) -> Load(C3, P1, ATL) -> Fly(P1, ATL, JFK) -> Unload(C4, P2, SFO) -> Unload(C3, P1, JFK) -> Unload(C2, P2, SFO) -> Unload(C1, P1, JFK)`.

2 Non-heuristic search comparison

Search method	Expansions	GoalTests	New nodes	Time	Path length
BFS	43	56	180	0:00.04	6
BFTS	1458	1459	5960	0:01.25	6
DFS	21	22	84	0:00.02	20
Depth-limited search	101	271	414	0:00.13	50
UCS	55	57	224	0:00.05	6

Table 1: Non-heuristic search algorithms for problem 1

Let's consider first non-heuristic search algorithms. The algorithms considered are breadth-first graph search (BFS), breadth-first tree search (BFTS), depth-first search (DFS), depth-limited search (depth-first search with depth limited to 50) and uniform-cost search (UCS).

Search method	Expansions	GoalTests	New nodes	Time	Path length
BFS	3343	4609	30509	0:21.45	9
BFTS				$> 10{:}00$	
DFS	624	625	5602	0:05.22	619
Depth-limited search				$> 10{:}00$	
UCS	4852	4854	44030	0:20.90	9

Table 2: Non-heuristic search algorithms for problem 2

The results obtained are in Tables 1, 2 and 3. The algorithms that give an optimal plan are breadth-first search and uniform cost search. Breadth-first tree search is very inefficient because it considers nodes multiple times, and even though it gives an optimal plan, it times out on problems 2 and 3.

Depth-first search is not optimal, and one can easily see that in the length of the plans it finds. Depth-limited search isn't optimal either, and while it can be used with iterative-deepening to find an optimal plan, it is very inefficient.

Notice also that since the costs are all equal to 1, breadth-first graph search and uniform-cost search are supposed to be equivalent. However there is some difference both in the number of nodes explored and the time taken. This is due to how the algorithms are implemented: breadth-first search is implemented using a FIFO queue (implemented on top of a python list), while uniform-cost search is implemented using a priority queue and a hash table for lookup. This explains why UCS is faster even though it explores more nodes (due to difference in breaking ties).

Search method	Expansions	GoalTests	New nodes	Time	Path length
BFS	14663	18098	129631	2:39.21	12
BFTS				> 10:00	
DFS	408	409	3364	0:03.08	392
Depth-limited search				> 10:00	
UCS	18235	18235	159716	1:48.11	12

Table 3: Non-heuristic search algorithms for problem 3

3 Heuristic comparison

Heuristic	Expansions	GoalTests	New nodes	Time
None (UCS)	55	57	224	0:00.05
Ignore precond.	41	43	170	0:00.04
PG levelsum	11	13	50	0:00.85

Table 4: A* search with different heuristics for problem 1

We now consider A* search with different heuristics. A* search always gives an optimal path (given that the heuristic is admissible), so the comparison here is about performance. The heuristics considered are the ignore-preconditions heuristic and the planning-graph levelsum heuristic (along with using no heuristic for comparison). The results are in Tables 4, 5 and 6.

Heuristic	Expansions	GoalTests	New nodes	Time
None (UCS)	4852	4854	44030	0:20.90
Ignore precond.	1450	1452	13303	0:06.51
PG levelsum	86	88	841	1:11.94

Table 5: A* search with different heuristics for problem 2

The ignore-preconditions heuristic gives a nice reduction in the number of node expansions, especially with the larger problem 3, and a nice speed-up compared to a non-heuristic search. The levelsum heuristic gives a drastic reduction in the number of node expansions (expanding

only 325 nodes in problem 3), however it takes more time than UCS. This is due to the fact that computing the planning graph is relatively expensive.

The best heuristic for this problem is the simpler ignore-preconditions heuristic. It is simple to implement, fast to compute and provides a good speed-up for the search.

Heuristic	Expansions	GoalTests	New nodes	Time
None (UCS)	18235	18235	159716	1:48.11
Ignore precond.	5040	5042	44944	0:29.61
PG levelsum	325	327	3002	5:58.49

Table 6: A* search with different heuristics for problem 3