Project: Air Cargo Planning

Results

The aim of this project was to employ uninformed and heuristics-based search in order to solve the problem of air cargo planning. Overall, 10 algorithms were compared:

- Breadth-First-Search (BFS)
- Breadth-First-Tree-Search (BFTS)
- Depth-First-Graph-Search (DFGS)
- Depth-Limited-Search (DLS)
- Uniform-Cost-Search (UCS)
- Recursive-Best-First-Search with dummy heuristic (RBFS_H1)
- Greedy-Best-First-Graph-Search with dummy heuristic (GBFGS_H1)
- A* with dummy heuristic (H1)
- A* with ignore preconditions heuristic (IP)
- A* with planning graph level-sum heuristic (HPG)

Table 1 depicts the overall results of the experiments. Within each column the color coding (ranging from green to red) gives a visual indication from the best to the worst measured results. For instance, for problem 1 the best (and optimal) plan length is 6, whereas the worst plan length is 50 (generated by DLS). BFTS, DLS and RBFS didn’t finish within 10 minutes of calculation for problems 2 and 3.

	Problem 1				Problem 2				Problem 3			
Algorithm	Expansions	Goal Tests	Plan length	Time	Expansions	Goal Tests	Plan length	Time	Expansions	Goal Tests	Plan length	Time
BFS	43	56	6	0.033	3343	4609	9	9.456	14663	18098	12	47.480
BFTS	1458	1459	6	1.031								
DFGS	21	22	20	0.015	624	625	619	3.732	408	409	392	1.891
DLS	101	271	50	0.091	222719	2053741	50	1006.319				
UCS	55	57	6	0.038	4853	4855	9	13.435	18223	18225	12	61.831
RBFS_H1	4229	4230	6	2.899								
GBFGS_H1	7	9	6	0.005	998	1000	21	2.745	5578	5580	22	18.859
A*_H1	55	57	6	0.040	4853	4855	9	13.241	18223	18225	12	60.133
A*_IP	41	43	6	0.041	1450	1452	9	4.647	5040	5042	12	20.377
A*_HPG	11	13	6	0.749	86	88	9	69.591	315	317	12	339.354

Table 1: Overall Results

When taking *optimality* (in terms of plan length) into consideration, the algorithms consistently performing optimal are BFS, UCS and A*. When further considering the search *speed* (in terms of seconds), the best trade-off is achieved by A* with the ignore preconditions heuristic. DFGS is overall the fastest algorithm but does not yield optimal plans.

	Problem 1		Problem 2		Problem 3	
Algorithm	Plan length	Time	Plan length	Time	Plan length	Time
BFS	6	0.033	9	9.456	12	47.480
BFTS	6	1.031				

DFGS	20	0.015	619	3.732	392	1.891
DLS	50	0.091	50	1006.319		
UCS	6	0.038	9	13.435	12	61.831
RBFS_H1	6	2.899				
GBFGS_H1	6	0.005	21	2.745	22	18.859
A*_H1	6	0.040	9	13.241	12	60.133
A*_IP	6	0.041	9	4.647	12	20.377
A*_HPG	6	0.749	9	69.591	12	339.354

Table 2: Optimality and Speed

Lastly, I'd like to take a look into the number of expansions necessary, correlating with the space requirements of the algorithms.
Not surprisingly, one of the algorithms requiring the fewest expansions is also the fastest: DFGS. The other algorithm, A* with the level-sum heuristic, needs even fewer expansions but is considerably slower. Whereas Depth-First-Graph-Search solves problem 3 in 1.891 seconds, A* requires almost 340 seconds. Why so? Because the level-sum heuristic - in its current implementation - consistently recalculates the heuristic for the graph, i.e. it determines for each goal when it is first seen in the planning graph, which is very costly.

	Problem 1		**Problem 2**		**Problem 3**	
Algorithm	Expansions	Time	Expansions	Time	Expansions	Time
BFS	43	0.033	3343	9.456	14663	47.480
BFTS	1458	1.031				
DFGS	21	0.015	624	3.732	408	1.891
DLS	101	0.091	222719	1006.319		
UCS	55	0.038	4853	13.435	18223	61.831

RBFS_H1	4229	2.899				
GBFGS_H1	7	0.005	998	2.745	5578	18.859
A*_H1	55	0.040	4853	13.241	18223	60.133
A*_IP	41	0.041	1450	4.647	5040	20.377
A*_HPG	11	0.749	86	69.591	315	339.354

Table 3: Space requirements and Speed

Analysis

Depending on the requirements it might make sense to favour a certain algorithm over another in a situation. If memory requirements are an issue and the primary goal is to solve for a solution fast (regardless of the optimality of the solution), Depth-First-Graph-Search seems - in the given problem context - the algorithm of choice. It is almost 9 respectively 10 times faster than Greedy-Best-First-Graph-Search or A* with the ignore preconditions heuristic on problem 3. However, if - what is more likely - the aim is to reduce the path length and number of changes of cargo from one plane to another, i.e. finding an optimal solution, A* with ignoring preconditions yields the best compromise on time.

In summary, even though uninformed searches like BFS and UCS yield optimal solutions their time intensiveness increases strongly with a growing solution space because they do not take the structure of the goal state into consideration (only yes or no). DFGS is fast but doesn't yield optimal solutions. A* with ignoring preconditions as a heuristic is optimal because the heuristic never overestimates the actual distance to a goal state since from every given state the number of actions required to a goal state can never be larger than the number of unsatisfied goal states since every action in the air cargo problem can maximally achieve one goal state in our problem statement.[1] It thereby searches through a smaller part of the solution space (compare node expansions) as BFS or basically all other algorithms. As described before, A* with the

[1] See: Stuart J. Russell, Peter Norvig (2010), Artificial Intelligence: A Modern Approach (3rd Edition)

level-sum heuristic does minimal expansions but is very compute- and memory-intensive due to its elaborative search through the graph and takes significantly longer to compute than the simple IP heuristic.

Optimal Sequence of Actions

As argued in AIMA, BFS yields optimal solutions when able to finish a search. Afterwards I list these plans exemplarily for optimal solutions to all three search problems.

Problem 1

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

Problem 2

```
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)
```

Problem 3

```
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)
```