Implement a Planning Search – Heuristic Analysis

Artificial Intelligence Nanodegree Program. Project 3

Angel Martínez-Tenor

1: Optimal plans

Except for deep-first search, all the strategies led to optimal plans. You need at least 6, 9, and 12 steps to solve air_cargo_p1, air_cargo_p2 and air_cargo_p3 respectively.

Step	**air_cargo_p1**	**air_cargo_p2**	**air_cargo_p3**
1	Load(C1, P1, SFO)	Load(C1, P1, SFO)	Load(C1, P1, SFO)
2	Load(C2, P2, JFK)	Load(C2, P2, JFK)	Load(C2, P2, JFK)
3	Fly(P2, JFK, SFO)	Load(C3, P3, ATL)	Fly(P1, SFO, ATL)
4	Unload(C2, P2, SFO)	Fly(P1, SFO, JFK)	Load(C3, P1, ATL)
5	Fly(P1, SFO, JFK)	Unload(C1, P1, JFK)	Fly(P2, JFK, ORD)
6	Unload(C1, P1, JFK)	Fly(P2, JFK, SFO)	Load(C4, P2, ORD)
7		Unload(C2, P2, SFO)	Fly(P1, ATL, JFK)
8		Fly(P3, ATL, SFO)	Unload(C1, P1, JFK)
9		Unload(C3, P3, SFO)	Unload(C3, P1, JFK)
10			Fly(P2, ORD, SFO)
11			Unload(C2, P2, SFO)
12			Unload(C4, P2, SFO)

There are several optimal sequences. You can load several cargoes in different planes before any flight, or fly each plane just after loading. Note that the optimal solution in air_cargo_p3 takes advantage of loading more than one cargo in the same plane to save steps.

2: Non-heuristic search

We tested three uniformed strategies: breadth-first search (BFS), depth-first graph search (DFS), and uniform-cost search (UCS).

	air_cargo_p1			**air_cargo_p2**			**air_cargo_p3**		
	BFS	**DFS**	**UCS**	**BFS**	**DFS**	**UCS**	**BFS**	**DFS**	**UCS**
Expansion	43	21	55	3346	107	4853	14120	292	18235
Goal tests	56	22	57	4612	108	4855	17673	293	18237
New nodes	180	84	224	30534	959	44041	124926	2388	159716
Plan length	6	20	6	9	105	9	12	288	12
Time elapsed (s)	0.04	0.01	0.04	13.56	0.33	41.82	95.85	1.11	382.76

Strategies: BFS: breadth-first search, DFS: depth-first graph search, UCS: uniform-cost search

Comparing the result metrics we can state that BFS and UCS achieve the optimality criteria. The number of steps of the solutions returned by DFS grows quickly with the complexity of the problem (x3, x12, and x24 respectively). As seen in the lessons, DFS searches the deepest nodes first returning the first sequence of actions leading to the solution. This solution is usually far from optimal since it is easy to find one before exploring swallow nodes leading to optimal solutions. As opposite to DFS, BFS always finds the optimal sequence as all the nodes in each layer are explored before going a level deeper, i. e., no action leading to an optimal goal is left unexplored before going deeper.

On the other hand, DFS advantages BFS and UCS in running time (time elapsed) and running space (node expansions). DFS finishes as soon as the first (mostly non-optimal) solution is found, which is easy for the air cargo problems. The difference between running time and running space requirements is most noticeable in the third problem. There DFS only needs between 1-2% of the resources of the second best option (BFS).

(We marked UCS columns in grey since all the metrics of BSF results in better or equal metrics than UCS)

3: Domain-independent heuristic search

The informed searches tested were: A* with *ignore preconditions heuristic (A*IP)* and *level-sum* heuristic (*A*LS).*

	air_cargo_p1		air_cargo_p2		air_cargo_p3	
	A*IP	**A*LS**	**A* IP**	**A* LS**	**A* IP**	**A* LS**
Expansion	41	11	1506	86	5118	404
Goal tests	43	13	1508	88	5120	406
New nodes	170	50	13820	841	45650	3718
Plan length	6	6	9	9	12	12
Time elapsed (s)	0.04	2.03	13.69	214.55	92.24	1477.1

Search algorithms: A*IP: A* with ignore preconditions heuristic, A*LS: A* with level sum heuristic

A*IP and A*LS, both optimal, improve the above non-heuristic optimal strategies (BSF, UCS) by requiring less memory (lower running space). These results show the advantages of informed (or directed) strategies based on heuristics. Thus effective heuristics can be derived by relaxing the planning problem and assuming sub-goal independence. In particular, A*IP achieves the same results as BSF in the same time by using approximately a third of the memory.

On the other hand, A*LS is able to achieve the optimal solution which much less memory than A*IP. Hence, the level-sum strategy provides a better heuristic than ignoring the preconditions. Unfortunately, A*LS has a notable drawback: its running time. A*LS took 16 times longer than A*IP to find the optimal solution. We improved the level-sum heuristic function to minimize this problem without success. Level-sum needs to iterate over the graph to find the first level in which each literal of the goal appears, thus requiring more computing. Hence there is a trade-off with ignoring preconditions heuristic (lower running time, higher running space), vs level-sum heuristic (higher running time, lower running space).

The selection of the best heuristic used in these problems depends on the available memory. Among the strategies tested here, we suggest using A*IP for problems like this one. A*IP advantages the optimal non-heuristic strategies without increasing the execution time. For very large problems unsolvable by A*IP due to memory requirements, only A*LS could solve the problem effectively.