AirCargo planning problem - Heuristic Analysis

Wrote by: Luka Anicin

1. **OPTIMAL PLANS:**

If we want to determine which search is optimal and which is not, we will need the best case, which is also going to be our base line for comparison.

Note that all these results are taken with Breath-first search.

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(P3, ATL, SFO)

Unload(C3, P3, SFO)

Fly(P2, JFK, SFO)

Unload(C2, P2, SFO)

Fly(P1, SFO, JFK)

Unload(C1, P1, JFK)

Problem 3:

Load(C1, P1, SFO)

Load(C2, P2, JFK)

Fly(P2, JFK, ORD)

Load(C4, P2, ORD)

Fly(P2, ORD, SFO)

Unload(C2, P2, SFO)

Unload(C4, P2, SFO)

Fly(P1, SFO, ATL)

Load(C3, P1, ATL)

Fly(P1, ATL, JFK)

Unload(C1, P1, JFK)

Unload(C3, P1, JFK)

Abbreviations:

Full name	Abbreviations
Breadth – First Search	BFS
Breadth – First Tree Search	BFTS
Depth First Graph Search	DFS
Depth Limited Search	DLS
Uniform – Cost Search	UCS
Recursive Best First Search (with h1)	RBFS
Greedy Best First Search (with h1)	GBFGS
A star search – with h1	A*
A star search – with h_ignore_predictions	A* h_ignore
A star search – with h_levelsum	A* h_levelsum

2. NON-HEURISTIC ANALYSIS:

Problem 1:

	Expansions:	Goal Tests:	Plan length:	Time elapsed: (seconds)	Optimal:
BFS	43	56	6	0.027143	Yes
BFTS	1458	1459	6	0.715423	Yes
DFS	21	22	20	0.011126	No
DLS	101	271	50	0.070254	No
UCS	55	57	6	0.028739	Yes

Problem 1 results analysis: In the case of Problem 1, none of the search functions have run more than a second. The best run time had DFS, but also it wasn't optimal in results, because it has created a plan which is much longer than the optimal one. This results are showing that DFS will find the plan for reaching the goal, but it won't necessary be the optimal one. As we can see the speed of run time for DFS, it appears we can use it for checking if there is at least one plan for reaching our goal(s). BFS, DLS, UCS all reached goals in optimal number of actions. From these three BFS had the best run time and also the smallest number of expended nodes.

According to Problem 1: BFS showed the best performance from non-heuristic searches.

Problem 2:

	Expansions:	Goal Tests:	Plan length:	Time elapsed: (seconds)	Optimal:
BFS	3401	4672	9	10.961455	Yes
BFTS	-	-	-	-	-
DFS	1192	1193	1138	6.933216	No
DLS	-	-	-	-	-
UCS	4761	4763	9	8.981306	Yes

Problem 2 results analysis: In the Problem 2, the best performance we got from UCS this time. UCS reached the goal in the optimal number of moves, had really good time, the best time of those searches who run below 10 minutes mark. In the Problem 2, DFS approved our knowledge from Problem 1. It reached the goal but not in the optimal number of moves, but it had the best run time again.

In the case of Problem 2 we had a different situation, BFTS and DLS had run for over 20 minutes and didn't find any plan for our problem.

According to Problem 2: UCS was the best in time and optimal in number of actions taken to the goal. BFS was again up there, in this case with a bit worse time but less number of nodes expended.

Problem 3:

	Expansions:	Goal Tests:	Plan length:	Time elapsed: (seconds)	Optimal:
BFS	14491	17947	12	82.243215	Yes
BFTS	-	-	-	-	-
DFS	2099	2100	2014	15.965234	No
DLS	-	-	-	-	-
UCS	17783	17785	12	39.749318	Yes

Problem 3 result analysis: In the Problem 3, we had the same situation as we had in Problem 2. In this Problem as well we have situation that UCS runs faster and in optimal number of actions. BFS is better than UCS again in number of Expansions.

For Problem 3 BFTS and DLS run for more than 20 minutes with no feedback.

Final thoughts and non-heuristic metrics

As Mr. Peter Norvig said in his video on Depth First Search, it will run to the deepest part of the tree first and then run back and start over again. In the example that we have seen in the class (route finding problem), it will surely find the goal but in most cases that route won't be optimal. We saw that same thing in these AirCargo problems as well. Despite of its fastest time, the plan it finds is not optimized. Therefore, the logical use of DFS is simply checking at the beginning if there's at least one possible plan to the goal.

In terms of the best algorithm, the first place belongs to UCS. It always finds an optimal solution for a problem, runs faster than BFS, but the number of nodes that it expends is larger than BFS.

Heuristic searches

Problem 1

	Expansions:	Goal Tests:	Plan length:	Time elapsed: (seconds)	Optimal:
RBFS	4229	4230	6	2.080072	Yes
GBFGS	7	9	6	0.004255	Yes
A*	55	57	6	0.029999	Yes
A* h_ignore	41	43	6	0.028344	Yes
A* h_levelsum	55	57	6	1.109378	Yes

Problem 1 results heuristic analysis: At this problem set all heuristic searches reached goals in optimal number of actions. GBFGS was the fastest one by far and the number of nodes it has expended was the lowest as well. Right behind it, by its performance, was A* h_ignore seach. This one was pretty fast as well, and also this 'version' of A* search had the best performance if we look just at A* variations results.

According just to Problem 1 the best performance had GBFGS.

Problem 2

	Expansions:	Goal Tests:	Plan length:	Time elapsed: (seconds)	Optimal:
RBFS	-	-	-	-	-
GBFGS	550	552	9	1.037842	Yes
A*	4761	4763	9	9.123738	Yes
A* h_ignore	1450	1452	9	3.421618	Yes
A* h_levelsum	4761	4763	9	1303.646511	Yes

Problem 2 results heuristic analysis: In the case of the Problem 2, we didn't have many changes on the 'top'. GBFGS showed the best performance in time, optimality of search and number of nodes expended. Again, at the second place we have A* h_ignore and also it is the best of A* searches in this Problem as well. A* h_levelsum run for 1303.65 seconds and reached optimal number of nodes. The biggest changes we had from Problem 1: RBFS run for 15 minutes without any results.

According just to Problem 2 the best performance had GBFGS.

Problem 3

	Expansions:	Goal Tests:	Plan length:	Time elapsed: (seconds)	Optimal:
RBFS	-	-	-	-	-
GBFGS	4031	4033	12	8.697766	Yes
A*	17783	17785	12	39.227662	Yes
A* h_ignore	5003	5005	12	13.454447	Yes
A* h_levelsum	17783	17785	12	9668.054219	Yes

Problem 3 results heuristic analysis: Problem 3 didn't bring any differences in its results comparing to Problem 2.

According just to Problem 3 the best performance showed GBFGS.

Total performance of heuristic searches: GBFGS algorithm in all 3 problems expended the least nodes, always found the optimal plan (with the optimal number of actions) and the run time was the best as well.

The best algorithm to use according to all parameters is GBFGS.

As we used A* search with three different heuristics, the heuristic that showed the best performance was h_ignore_prerconditions in all 3 problems.

Comparing A* search algorithms with lelvesum and ignore_preconditions heuristics

The A* version with lelvesum heuristic runs the slowest of all informed searches, because of the nature of its heuristic function, which needs to traverse the graph and check where the goal is (on which layer/level).

Their performance on the node expansion is also different. The A* with ignore_preconditions heuristic expended less nodes in every of three problems. Its run time is drastically lower than A* with levelsum heuristic (in the problem set 3 A*h_levelsum took about 3 hours to execute, A*h_ignore just 13 seconds).

Based on A* searches, *ignore_preconditions* was the best heuristic.

Was non-heuristic planning better than heuristic panning approach?

As the results are showing, Greedy Best First Search (GBFS) has shown the best performance overall in the category of heuristic search functions. If we compare it with UCS and BFS, which have shown the best performance in the category of non-heuristic planning, we can clearly see that GBFS runs with the lowest time, expended noticeable less nodes than UCS and BFS and every time it reached goal with optimum number of actions.

In my case GBFS (or heuristic planning approach) has shown better results. This yields that it's is much better to used informed (heuristic) search functions then uninformed.