Part I Heuristic Analysis

In this first part of the project, I chose the following algorithms to analyse.

- Breadth First Search (BFS);
- Deep First Graph Search (DFGS);
- Uniform Cost Search (UFC);
- Greedy Best First Graph Search (GBFGS).

Problem 1

	Plan Length	Expansions	Goal Tests	New Nodes	Time Elapsed
BFS	6	43	56	180	0,05698
DFGS	20	21	22	84	0,02266
UCS	6	55	57	224	0,04550
GBFGS	6	7	9	28	0,00685

This problem's initial state has 12 fluents, meaning that our space have a size of 2^{12} = 4096.

Since it's a small problem, all the algorithms performed fast. However, the GBFGS was the fastest. Moreover, it was capable to provide an optimal plan with significantly less expansions, goal tests and new nodes compared with the others algorithms.

DFGS was an interesting one. Even though it was the second fastest, the algorithm provided a plan with a size of 20 steps. If we put it in practice, it would waste a lot of resource.

Optimal plan

This plan was provided by GBFGS algorithm.

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

Problem 2

	Plan Length	Expansions	Goal Tests	New Nodes	Time Elapsed
BFS	9	3343	4609	30509	30,85834
DFGS	619	624	625	5602	6,95145
UCS	9	4833	4835	43855	37,46547
GBFGS	25	325	327	2928	2,65884

This problem has 27 fluents, so the space has a size of 2^{27} = 134,217,728. Compared with problem 1, the space increased by 32768 times!
We can notice that the time elapsed by the algorithms significantly increased.

GBFGS was the fastest one again and ran with less expansions, goal tests and new nodes compared with the others algorithms. However, the plan had a lot of steps.
DFGS was similar to GBFGS. However, the plan had even more steps than GBFGS.

UFC and BFS were the ones that provided a feasible plan, but it lasted about 30 seconds to formulate the plan. Both of them had similar numbers for expansions, goal tests and new nodes, but BFS was slightly better than UFC.

Optimal plan

This plan was provided by UCS algorithm.

Load(C1, P1, SFO)
Load(C2, P2, JFK)
Load(C3, P3, ATL)
Fly(P1, SFO, JFK)
Fly(P2, JFK, SFO)
Fly(P3, ATL, SFO)
Unload(C3, P3, SFO)
Unload(C2, P2, SFO)
Unload(C1, P1, JFK)

Problem 3

	Plan Length	Expansions	Goal Tests	New Nodes	Time Elapsed
BFS	12	14663	18098	129631	154,42993
DFGS	392	408	409	3364	4,38706
UCS	12	18069	18071	158349	182,15915
GBFGS	27	2540	2542	22552	25,48254

This problem has 32 fluents, so the space is equal to 2^{32} = 4,294,967,296. The size increased by 32 times compared with the problem 2 and it achieved a billion decimal place.

Again, BFS and UCS were able to provide a feasible plan, however the time elapsed have considerably increased. This might be due the fact that the number of expansions, goal tests and new nodes are high.

Surprisingly, DFGS kept the performance in all problems, however the plan length wasn't feasible.

Optimal Plan

This plan was provided by UCS algorithm.

Load(C1, P1, SFO)
Load(C2, P2, JFK)
Fly(P1, SFO, ATL)

Load(C3, P1, ATL)
Fly(P2, JFK, ORD)
Load(C4, P2, ORD)
Fly(P2, ORD, SFO)
Fly(P1, ATL, JFK)
Unload(C1, P1, JFK)
Unload(C4, P2, SFO)
Unload(C2, P2, SFO)
Unload(C3, P1, JFK)

Part II Heuristic Analysis

In this part, it was used the following algorithms:

- A* Search Ignore Preconditions (A* IP);
- A* Search Level Sum (A* LS);

Problem 1

	Plan Length	Expansions	Goal Tests	New Nodes	Time Elapsed
A* IP	6	41	43	170	0,05623
A* LS	6	39	41	158	1,17010

Both heuristics were able to provide an optimal plan. The expansions, goal tests and new nodes of both heuristics were close to each other. However, A* LS had a time elapsed significantly higher than A* IP and the other algorithms from the Part I.

Optimal Plan

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

Problem 2

	Plan Length	Expansions	Goal Tests	New Nodes	Time Elapsed
A* IP	9	1435	1437	13159	15,66339
A* LS	9	1111	1113	10053	478,15850

Both heuristics provided an optimal plan. However, A* LS lasted for about 8 minutes. This heuristic might provide an optimal plan, but it's not efficient as A* IP.

Finally, comparing A* IP against BFS and UCS, A* IP was able to provide an optimal plan saving about 15 seconds and the expansions, goal tests and new nodes were about 10 times lower.

Optimal Plan

Load(C3, P3, ATL)
Fly(P3, ATL, SFO)
Unload(C3, P3, SFO)
Load(C1, P1, SFO)
Fly(P1, SFO, JFK)
Unload(C1, P1, JFK)
Load(C2, P2, JFK)
Fly(P2, JFK, SFO)
Unload(C2, P2, SFO)

Problem 3

	Plan Length	Expansions	Goal Tests	New Nodes	Time Elapsed
A* IP	12	4865	4867	43228	56,70828
A* LS	12	1916	1918	16892	1.662,27575

Both heuristics provided an optimal plan. However, even though A* LS use less expansions, goal tests and new nodes than A* IP, it took about 28 minutes to solve the problem, when A* IP took less than a minute.

Now, comparing A* IP against BFS and UCS, again, A* IP was able to solve the problem saving about 1 minute and using less expansions, goal test and new nodes.

Optimal Plan

Load(C2, P2, JFK)
Fly(P2, JFK, ORD)
Load(C4, P2, ORD)
Fly(P2, ORD, SFO)
Unload(C4, P2, SFO)
Unload(C2, P2, SFO)
Load(C1, P1, SFO)
Fly(P1, SFO, ATL)
Load(C3, P1, ATL)
Fly(P1, ATL, JFK)

Unload(C3, P1, JFK)
Unload(C1, P1, JFK)

Summary

We can see that, independently of the size of the problem, the BFS, UCS, A* IP and A* LS are guaranteed to provide an optimal plan as stated in “Artificial Intelligence: A Modern Approach” (AIMA), P. Norvig and S. Russel, 3rd edition, pp 108 and 109. However, the size of the problem drastically impacts on how long the algorithm will take to search the plan.

We also can see that DFGS and GBFGS are not guaranteed to provide an optimal solution. However, both are fast even if the size of the problem increases as stated in AIMA book, P. Norvig and S. Russel, 3rd edition, pp 108.

Finally, A* IP was more efficient than the other search algorithms. Comparing it against A* LS, A* IP might be faster because of the way it was implemented. We can see that A* IP just compare the current state against the goal, while A* LS instantiates a Planning Graph in order to calculate the level sum. Building this graph is expensive, and for each time we need to calculate the heuristic, we’re building this planning graph.