Written report – Juan Rios

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)
```

Air_cargo_p1

```
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)
```

Air_cargo_p2

```
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)
```

Air_cargo_p3

The following tables present the performance metrics of different search algorithms applied to all the planning problems of the project:

	Air_cargo_p1				
	Node expansions	**Goal tests**	**New nodes**	**Time elapsed(s)**	**Plan length**
BFS	43	56	180	0.027	6
DFGS	12	13	43	0.0072	12
Uniform cost	55	57	224	0.033	6
A* h1	55	57	224	0.033	6
A* h_ignore	41	43	170	0.04	6
A* h_pg_levelsum	11	13	50	1	6

	Air_cargo_p2				
	Node expansions	**Goal tests**	**New nodes**	**Time elapsed(s)**	**Plan length**
BFS	3343	4609	30509	11.8	9
DFGS	582	583	5211	2.65	575
Uniform cost	4853	4855	44041	37.15	9
A* h1	4853	4855	44041	37.2	9
A* h_ignore	1506	1508	13820	11.98	9
A* h_pg_levelsum	86	88	841	96.5	9

	Air_cargo_p3				
	Node expansions	**Goal tests**	**New nodes**	**Time elapsed(s)**	**Plan length**
BFS	14663	18098	129631	87.24	12
DFGS	627	628	5176	2.8	596
Uniform cost	18223	18225	159618	320	12
A* h1	18223	18225	159618	348	12
A* h_ignore	5118	5120	45650	78	12
A* h_pg_levelsum	403	405	3708	636	12

1) Experiments for non-heuristic planning solution searches

We can see from the tables that Breath First Search has a huge number of node expansions, goal tests and new nodes. This makes sense since it goes through the whole tree structure (expanding every node level by level). It takes longer to reach a solution than Depth First Search but reaches the optimal solution. In contrast Depth First Search reaches a feasible solution faster but at the cost of finding a sub-optimal solution that does not even make sense in a practical situation. This happens because the tree is

traversed from left to right, going deeper and deeper in the tree. This process creates a solution, but since it is only exploring one branch of the whole tree it finds a very bad one (it could be the case that the method finds an optimal one, but such a case would be just coincide). Uniform cost search also finds an optimal solution but the number of node expansions, node tests, new nodes and time elapsed is bigger for all the problems.

2) Experiment with A* searches with some heuristics

We can see that using A* with a dull heuristic creates the same metrics as uniform cost search. Now, when the two heuristics designed in the project are used (the relaxation of the problem when preconditions are ignored and planning graph with levelsum heuristic) the results are much better at the cost of more time, especially with planning graph/levelsum which is computationally more expensive than ignoring the preconditions. When both heuristics are compared with the non-heuristic search methods the number of expansion nodes and goal tests is dramatically reduced, but the heuristic levelsum with planning graph has much better metrics for expansion nodes, goal tests and new nodes reaching values way below the ones obtained by just ignoring the preconditions. This is expected since the planning graph is a polynomial size approximation of the exponential size of the tree that represents all possible states. However, the time it takes for the planning graph with levelsum to reach a solution is bigger by almost one order of magnitude or more than the time to reach a solution ignoring the preconditions. In this case an efficient implementation of the planning graph (mutex rules for actions and states) and the heuristic itself could help to improve the performance in the ‘time elapsed’ metric.

Given the previous discussion and the information provided in the tables, the A* methods with heuristics (excluding h1, which is not a heuristic) outperform the non-heuristic search methods. However, when comparing levelsum/planning graph with ignoring preconditions there is not a clear winner as best heuristic. In the former case we will need to expand less nodes, perform less goal tests and achieve less new nodes but the time it takes to reach a solution is much bigger than the one needed if we just ignore the preconditions. If the planning graph/levelsum can be improved to reach an optimal solution in times below the one by A*+ignore preconditions it would be the best search method, since it gets better results for all the other parameters. If we just stick with the results of this experiment, the best all-around method is A* + ignoring preconditions.