TODO: Include the following in your written analysis.

1. Provide an optimal plan for Problems 1, 2, and 3.

The optimal plan for Problem 1 is:
Load(C2, P2, JFK)
Load(C1, P1, SFO)
Fly(P2, JFK, SFO)
Unload(C2, P2, SFO)
Fly(P1, SFO, JFK)
Unload(C1, P1, JFK)

The optimal plan for Problem 2 is:
Load(C3, P3, ATL)
Load(C2, P2, JFK)
Load(C1, P1, SFO)
Fly(P3, ATL, SFO)
Unload(C3, P3, SFO)
Fly(P2, JFK, SFO)
Unload(C2, P2, SFO)
Fly(P1, SFO, JFK)
Unload(C1, P1, JFK)

The optimal plan for Problem 3 is:
Load(C2, P2, JFK)
Load(C1, P1, SFO)
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)

2. Compare and contrast non-heuristic search result metrics (optimality, time elapsed, number of node expansions) for Problems 1,2, and 3. Include breadth-first, depth-first, and at least one other uninformed non-heuristic search in your comparison; Your third choice of non-heuristic search may be skipped for Problem 3 if it takes longer than 10 minutes to run, but a note in this case should be included.

With this 3-problem sets, Breadth First Search and Uniform Cost Search are the only two uninformed search strategies that systematically yield an optimal action plan. When it comes to execution speed and memory usage, Depth First Graph Search is the fastest and uses the least memory. However, it does not generate an optimal action plan (problem 1: plan length of 12 instead of 6, problem 2: plan length of 346 instead of 9, problem 3: plan length of 1878 instead of 12).

Because it performs faster and uses less memory than Uniform Cost Search, Breadth First Search (BFS) is the recommended search strategy, as BFS is complete and optimal. Its only downside is memory usage, if the problem's branching factor is high.
For problems 2 and 3, the Depth First Graph Search plan lengths are so much longer than the optimal path length that it wouldn't make sense to use this search strategy. Greedy Best First Graph Search is the best alternative. In problems 1 and 2, it manages to find the optimal path. In problem 3, it does not find the optimal path but the path length it generates is 22 instead of 10, which is much better than Depth First Graph Search (1878 path length!). Moreover, it still provides execution time savings and uses less memory than the best search strategy for an optimal solution (Breadth First Search).
The following is the table of results I got:

		Expensions	Goal Test	Plan Length	Time elapsed (s)	Length	Optimal
Problem 1	breadth first search	43	56	180	0.031008	6	TRUE
	breadth first tree search	1458	1459	5960	0.915308	6	TRUE
	depth first graph search	12	13	48	0.007793	12	NO
	depth limited search	101	271	414	0.085062	50	NO
	uniform cost search	55	57	224	0.035496	6	TRUE
Problem 2	breadth first search	3401	4672	31049	13.572273	9	TRUE
	depth first graph search	350	351	3142	1.505550	346	NO
	uniform cost search	4761	4763	43206	11.305999	9	TRUE
Problem 3	breadth first search	14491	17947	128184	101.134100	12	TRUE
	depth first graph search	1948	1949	16353	20.160727	1878	NO
	uniform cost search	17783	17785	155920	49.404191	12	TRUE

3. Compare and contrast heuristic search result metrics using A* with the "ignore preconditions" and "level-sum" heuristics for Problems 1, 2, and 3.

Of the two strategies mentioned above, A* Search with Ignore Preconditions heuristic is the fastest. If we let search run to completion on my machine, A* Search with Level Sum heuristic uses the least memory*, but its execution time is much slower (469 seconds for problem 2 and I do not get a result for problem 3 as it run too long!).

The following is a table for the result I got:

		Expansions	Goal Test	Plan Length	Time elapsed (s)	Path Length	Optimal
Problem 1	A* search with ignore preconditions	41	43	170	0.042268	6	TRUE
	level_sum heuristics	11	13	50	2.007770	6	TRUE
Problem 2	A* search with ignore preconditions	1450	1452	13303	5.790103	9	TRUE
	level_sum heuristics	86	88	841	469.116006	9	TRUE
Problem 3	A* search with ignore preconditions	5003	5005	44586	19.83899	12	ture
	level_sum heuristics	xxx	xxx	xxx	xxx	xxx	xxx

4. What was the best heuristic used in these problems? Was it better than non-heuristic search planning methods for all problems? Why or why not?

The search strategies that generate optimal plans are: Breadth First Search, Uniform Cost Search, and A* Search with heuristics.

Depth first graph search does find a quick solution and requires a small amount of memory, but it lacks optimality. It is not optimal because it does not consider if a node is better than another, it simply explores the nodes that take it as deep as possible in the graph even if the goal ii to its right (ref. 1).

Non-heuristic based search performs better in problem 1 and 2, which suggest that when working with simple problems using a more elaborated approach, such as A* search with heuristics, is not with the increase in the solution complexity.

Heuristic based search performs better as the problem complexity increased. This is more evident in problem 3, where the A* search with 'h_ignore_preconditions' performance was optimal and the fastest amongst those that were optimal. It's also worth noting that the 'levelsum' heuristic did in overall perform poorly, most likely due to the heuristic being too complex.

According to the results obtained in this analysis, the breadth first search strategy can solve planning problems both fast and optimality, which makes it a good candidate to start off an analysis when dealing with search planning problems. As the complexity of the problems increase, it might be worth to consider if a heuristic based approach can outperform breadth first search and thus be used instead. The results above also clearly illustrate the benefits of using informed search strategies with custom heuristics over uninformed search techniques when searching for an optimal plan. One more subtle benefit is that one can customize the trade-off between speed and memory usage by using different heuristics, which is simply not possible with uninformed search strategies.

References

1, Stuart J. Russell, Peter Norvig(2010), Artificial Intelligence: A modern Approach (3rd Edition).