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

Optimal Sequences Problem 1	Optimal Sequences Problem 2	Optimal Sequences Problem 3
• Load(C1, P1, SFO) • Load(C2, P2, JFK) • Fly(P1, SFO, JFK) • Fly(P2, JFK, SFO) • Unload(C1, P1, JFK) • Unload(C2, P2, SFO)	• 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(C1, P1, JFK) • Unload(C2, P2, SFO)	• 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(C4, P2, SFO) • Unload(C3, P1, JFK) • Unload(C2, P2, SFO) • Unload(C1, P1, JFK)

For planning exercise 1, 6 of the 7 planning algos had the optimal length of 6. It would be reasonable to select planning algo by time elapsed. Therefore, the winning algo was Greedy Best First Graph Search algo. We see that for this particular algo, the expansions, goal tests and new nodes were the fewest among all the algos.

P1	Planning Algo		Breadth First Search	Depth First Search	Uniform Cost Search	Greedy Best First Graph Search	A-star		
	Heuristic		N/A	N/A	N/A	null heuristic	null heuristic	ignore precond	levelsum heuristic
	Metrics	Expansions	43	12	55	7	55	55	43
		Goal Tests	56	13	57	9	57	57	45
		New Nodes	180	48	224	28	224	224	178
		Plan Length	6	12	6	6	6	6	6
		Time Elapsed	0.027	0.010	0.033	0.007	0.036	0.042	1.745

For the planning exercise 2, 5 of the 7 planning algos had the optimal length of 9. It would also be reasonable to select planning algo by time elapsed. Therefore, the winning algo were Uniform Cost Search, A-star with null or ignore precondition heuristics. They have very similar performances regarding time elapsed, and their expansions, goal tests, and new nodes are the same.

P2	Planning Algo		Breadth First Search	Depth First Search	Uniform Cost Search	Greedy Best First Graph Search	A-star		
	Heuristic		N/A	N/A	N/A	null heuristic	null heuristic	ignore precond	levelsum heuristic
	Metrics	Expansions	3,346	1,124	4,853	998	4,853	4,853	3,299
		Goal Tests	4,612	1,125	4,855	1,000	4,855	4,855	3,301
		New Nodes	30,534	10,017	44,041	8,982	44,041	44,041	29,901
		Plan Length	9	1,085	9	17	9	9	9
		Time Elapsed	10.680	6.192	9.502	1.963	9.488	9.467	982.031

For the planning exercise 3, 5 of the 7 planning algos had the optimal length of 12. It would also be reasonable to select planning algo by time elapsed. Therefore, the winning algo were also Uniform Cost Search, A-star with null or ignore precondition heuristics. They have very similar performances regarding time elapsed, and their expansions, goal tests, and new nodes are the same.

P3	Planning Algo	Breadth First Search	Depth First Search	Uniform Cost Search	Greedy Best First Graph Search	A-star		
	Heuristic	N/A	N/A	N/A	null heuristic	null heuristic	ignore precond	levelsum heuristic
Metrics	Expansions	14,120	677	18,223	5,578	18,223	18,223	11,357
	Goal Tests	17,673	678	18,225	5,580	18,225	18,225	11,359
	New Nodes	124,926	5,608	159,618	49,150	159,618	159,618	101,106
	Plan Length	12	660	12	22	12	12	12
	Time Elapsed	76.361	2.820	41.490	12.652	41.184	46.031	6,022.357

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.

I chose Breath First, Depth First and Uniform Cost searches for this exercise. As seen in the previous tables, we could quickly determine the optimal algos by first determine if the plan length is optimal. We can then select the optimal algo by looking at time elapsed and number of node expansions. For problem 1, because the environment was small, difference in time elapsed and node expansions weren't apparent. However, Breath First was the winner because it had both the minimal node expansion and time elapsed.

For problem 2, as the environment was larger, the differences in performance metrics were more apparent. I believe the winner was the Uniform Cost search, as it had 10% performance improvement over Breath First on the time elapsed metric.

For problem 3, we could really see the difference is performances. The winner was also the Uniform Cost search, as it had 45% performance improvement over Breath First on the time elapsed metric.

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

All the A* algos were able to produce the correct results. However, the metrics, especially time elapsed, varied wildly differently. For problem 1, ignore precondition performed 40x faster than level-sum. For problem 2, ignore precondition performed 100x faster than level-sum. For problem 3, ignore precondition performed 130x faster than level-sum. Even though level-sum has on average 36% less in expansions, goal tests and new nodes, I felt that ignore precondition came out on top. One important thing to note was that when the problem was small, the time elapsed difference between ignore precondition and level-sum was not apparent. However, when the problem grew, the difference in performance was quite large. For problem 3, A* with level-sum heuristics took 100+ minutes to complete.

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 best heuristics were null and ignore precondition. I was quite surprised that null heuristic, which just outputs constant, was very similar to ignore preconditions, which checked if actions would achieve goals. In addition, A* performance was very similar to uniform Cost Search. This is because Uniform Cost was similar to the null heuristic in that it used the pre-defined path costs, which were uniform. To improve on the performance of these algorithms, we needed to have real world ways to effectively measure costs. For example, path costs should be specific to operational expenses (fuel, personnel) and time;, ignore precondition should return cost such as distance between current state and goal state.

Some Interesting Links

http://stackoverflow.com/questions/3332947/when-is-it-practical-to-use-dfs-vs-bfs