AIND- Planning analysis

This project introduce the way of solving the planning problem which convert the problem into propositional or relational representations of states and actions and search with domain independent heuristic. It is powerful and flexible algorithm for solving large scale of problem.
In the first part , we learned how to break down the problem into PDDL. The PDDL is a language representation problem which a state of the world is represented by a collection of variable. It primarily describes a system using a set of preconditions and post-conditions. In this project, we break down the air cargo planning problem into PDDL.
In the second part, we implement the core part of the planning graph and ignore precondition heuristic.
The benefit of the planning graph is able to apply to any of the search techniques we have seen so far. In this project, we combine with A* search. The biggest benefit is polynomial size approximation instead of exponential size like tree.
The following section is to answer the question from the part 3.

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

P1	P2	P3
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(C2, P2, SFO) Unload(C1, P1, JFK)	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)

- **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.**

For the first problem, this problem is not too complex and all of the search algorithms are able to come up with the solution at least. Here is the data report below.

	Problem 1				
SearchAlgorithmNo	NodeExpansions	GoalTests	NewNodes	Plan length	Time Elapsed
1.breadth_first_search	43	56	108	6	0.028333
2.breadth_first_tree_search	1458	1459	5960	6	0.931226071
3.depth_first_graph_search	12	13	48	12	0.00828451
4.depth_limited_search	101	271	414	50	0.085384982
5.uniform_cost_search	55	57	224	6	0.032787009

Algorithm 1,2 and 5 which are breadth_first_search, breadth_first_tree_search and uniform_cost_search gave the optimal solutions. The breadth_first_search is best one for all of the metrics.

For the second problem, 1 and 5 are able to get the optimal solution and the other several metric is pretty close. For the problem 2, the breadth_first_search is also the best one for non-heristic method. For the empty value is too long to get the result for 2 and 4 algorithm.

	Problem 2				
SearchAlgorithmNo	NodeExpansions	GoalTests	NewNodes	Plan length	Time Elapsed
1.breadth_first_search	2309	3366	17306	9	5.212
2.breadth_first_tree_search					
3.depth_first_graph_search	118	119	415	84	0.196312654
4.depth_limited_search					
5.uniform_cost_search	3588	3590	26334	9	7.577908374

For the third problem, this problem is much more complex than the other two. It also the breadth_first_search and uniform_cost_search gave the optimal solutions. The first one also the best performance for all metrics. Even the third algorithm depth_first_graph_search time elapsed is the lest one. But the plan length is way more to the optimal.

	Problem 3				
SearchAlgorithmNo	NodeExpansions	GoalTests	NewNodes	Plan length	Time Elapsed
1.breadth_first_search	11301	15053	88847	12	32.9066
2.breadth_first_tree_search					
3.depth_first_graph_search	118	119	415	788	2.709029266
4.depth_limited_search					
5.uniform_cost_search	15789	15791	122337	12	47.60609824

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

For the less complex problem, all of the algorithm with heuristic function are able to get the optimal solution. The greddy_best_frist_search h_1 is the best one for all of the metrics.

	Problem 1				
Search Algorithm No	Node Expansions	GoalTests	NewNodes	Plan length	Time Elapsed
6.recursive_best_first_searchh_1	4229	4230	17029	6	2.73694
7.greedy_best_first_graph_searchh_1	7	9	28	6	0.00534
8.astar_searchh_1	55	57	224	6	0.04028
9.astar_searchh_ignore_preconditions	41	43	107	6	0.034701
10.astar_searchh_pg_levelsum	11	13	50	6	1.49564

For the second problem, No 8, 9 and 10 are able to come out the optimal solution. The A star search with planning graph level sum heuristic is take less nodes for all algorithm, but it takes too long time to get the result. According the analysis from the book AIMA, the planning graph is polynomial increasing. For my understanding, it just because take too long time to create the planning graph. Once it built and cached in the memory. It should be much faster the others and achieved the best performance.

	Problem 2				
Search Algorithm No	Node Expansions	GoalTests	NewNodes	Plan length	Time Elapsed
6.recursive_best_first_searchh_1					
7.greedy_best_first_graph_searchh_1	449	451	2759	19	0.76626
8.astar_searchh_1	3588	3590	26334	9	7.09955
9.astar_searchh_ignore_preconditions	1078	1080	8310	9	2.81692
10.astar_searchh_pg_levelsum	225	227	1483	9	461.567

For the third problem, 7,9 and 10 algorithm are able to get the optimal solution and 9 takes less time to get the result and 10 take the less nodes. The reason of take so long time is as same as second problem that I guessed.

	Problem 3				
Search Algorithm No	Node Expansions	Goal Tests	New Nodes	Plan length	Time Elapsed
6.recursive_best_first_searchh_1					
7.greedy_best_first_graph_searchh_1	5786	5788	42177	31	15.07151
8.astar_searchh_1	15789	15791	122337	12	48.5253
9.astar_searchh_ignore_preconditions	3764	3766	30016	12	13.23296
10.astar_searchh_pg_levelsum	448	450	3191	12	2117.17

- **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 heuristic is the planning graph level sum, it take less node to explore. But it need to take pretty long time to create the graph. The ignore preconditions is the less optimal heuristic function. If we design a program and run it on the server and the problem is alway the same. We should take the No.10 to cache the heuristic result and if we run the program on local device and the problem always new. I recommend to use the No.9.

- **Provide tables or other visual aids as needed for clarity in your discussion.**

  The tables are on the above answer.

Conclusion

This experiment result clearly shows that the informed search that the search with heuristic is better than uniformed search algorithm for optimal result. And A* takes less time to get the result, planning graph takes less nodes to get the optimal result. If we don't care about the optimal result, the depth first graph search is the best one.