Heuristic Review

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

For Problem 1, an optimal plan would be:
Load(C1, P1, SFO)
Load(C2, P2, JFK)
Fly(P1, SFO, JFK)
Fly(P2, JFK, SFO)
Unload(C1, P1, JFK)
Unload(C2, P2, SFO)

For Problem 2, an optimal plan would be:
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)

For Problem 3, an optimal plan would be:
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)

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.

	S1(BFS) Exp/ GoalTest/ NewNode – Time elapsed	S3(DFS) Exp/ GoalTest/ NewNode – Time elapsed	S5(Uniform) Exp/ GoalTest/ NewNode – Time elapsed
P1	43/56/180 – 0.062s	12/13/48 – 0.0092s	55/57/224 – 0.062s
P2	3343/4609/30509 – 25.67s	582/583/5211 – 43.04s	4853/4855/44041 – 17.74s
P3	14663/18098/129631 – 115.57s	627/628/5176 – 43.66s	18223/18225/159618 – 164.38s

With the information table, we can review the results from 3 perspectives:

- Optimal plan
  In all 3 problems, both BFS and Uniform search can find the optimal plan. DFS is not even close.
- Memory consumed
  DFS consumed the least memory resource since it covers a minimal node during the calculation and invoke the least times of GoalTest. Uniform search performed the worst in this case: when the problem set is big(P3), it consumed 30 times of memory resource than DFS did. BFS is somewhere in between.
- Time consumed
  DFS consumed the least time in all 3 problems. Uniform search used 4 times of time resource than DFS did. BFS is somewhere in between.

The explanation of BFS, DFS and Uniform Search [1] from the video lessons states the reason of these differences of performance among them. The BFS and Uniform Search expand new nodes horizontally in the search tree so that they can guarantee the result plan is optimal (require least step) with too many redundant nodes expanded. While DFS expands nodes vertically so that it can reach a possible solution fast without consuming too much memory resource. But at the same time DFS risks the possibility that no plan is found when the problem is infinite.

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

	S9(A* w/ ignore) Exp/ GoalTest/ NewNode – Time elapsed	S10(A* w/ levelsum) Exp/ GoalTest/ NewNode- Time elapsed
P1	41/43/170 – 0.042s	11/13/50 – 0.70s
P2	1450/1452/13303 – 4.77s	86/88/841 – 63.09s

P3	5040/5042/44944 – 19.25s	325/327/3002 – 322.48s

We still review the results from 3 perspectives below:

- Optimal plan
  In all 3 problems, both heuristics search found the optimal plan
- Memory consumed
  The ignore_precondition heuristics consumed huge memory resource compare to level_sum heuristics. In P3, the ignore_precondition heuristics consumed 15 times of resource than its competitor.
- Time consumed
  The ignore_precondition heuristics made an impressive performance in terms of time consumption. In P3, it only took 6% of its competitor's time consumption to finish the task.

According to the definition of ignore pre-conditions heuristic function [2] and level sum heuristic function [3], we understand that the former simply uses the number of unsatisfied goals as its heuristic [4] so that it computes fast but the heuristic function is not accurate enough thus leads to a redundant consumption of nodes, while the latter is more complicated, it takes longer to calculation but it was more accurate so that it doesn't expand most of the unwanted nodes.

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 ignore precondition heuristics performs the best in speed, while level-sum heuristics saves the most space complexity. Since ignore precondition heuristics still solve the problem with acceptable memory resource, I think it is a better solution for general problems.

Besides, Depth First Search also archived a great balance among time performance and space performance. It is a great choice if the problem context does not require an optimal plan.

References:

[1] Udacity [Artificial Intelligence Nanodegree: online lesson Chapter Search]

[2] S. J. Russel and P. Norvig. [Artificial Intelligence: A modern approach third edition ch. 10.2.3]

[3] S. J. Russel and P. Norvig. [Artificial Intelligence: A modern approach third edition ch. 10.3.1]

[4] In Russel and Norvig's book, it mentioned that in order to make ignore pre-condition heuristic simple enough to be implemented, we can ignore the situations that some action achieve multiple goals and some actions may undo the effects of others.