Planning Search Heuristic Analysis

In the Air Cargo problem domain, we have three possible action types: Load, Unload, and Fly. The preconditions and effects are established such that, for a given cargo C to get from airport A1 to airport A2, C must first be loaded onto a plane P at A1, plane P flown to airport A2, and then C unloaded from P. Since the problem does not specify a distinct cost for each type of action or using additional resources, for example using only one plane instead of two, solution optimality is defined as the minimum number of actions that can move the problem from the initial state to its goal. As implemented, there is no precondition barring multiple cargo objects being loaded on the same plane simultaneously. So the minimum number of actions for getting C from A1 to A2 is 3, with the possibility of reducing this number only if multiple cargo objects share the same origin and destination airports. However, because none of the three problems examined have multiple cargo objects sharing the same origin and destination, the minimum number of actions for each cargo movement will be 3, with a possibility of being higher only if a plane needs to fly to an airport without cargo, due to the initial state. Problem 1's initial state and goal is defined as:

```
Init(At(C1, SFO) ∧ At(C2, JFK)
	∧ At(P1, SFO) ∧ At(P2, JFK)
	∧ Cargo(C1) ∧ Cargo(C2)
	∧ Plane(P1) ∧ Plane(P2)
	∧ Airport(JFK) ∧ Airport(SFO))
Goal(At(C1, JFK) ∧ At(C2, SFO))
```

Problem 1 can be achieved with 6 actions, but not less, given the action schema and initial state. For example:

Load(C1, P1, SFO)
Load(C2, P2, JFK)
Fly(P1, SFO, JFK)
Unload(C1, P1, JFK)
Fly(P2, JFK, SFO)
Unload(C2, P2, SFO)

Problem 2, with initial state and goal defined below, can be achieved in 9 actions.

```
Init(At(C1, SFO) ∧ At(C2, JFK) ∧ At(C3, ATL)
	∧ At(P1, SFO) ∧ At(P2, JFK) ∧ At(P3, ATL)
	∧ Cargo(C1) ∧ Cargo(C2) ∧ Cargo(C3)
	∧ Plane(P1) ∧ Plane(P2) ∧ Plane(P3)
```

∧ Airport(JFK) ∧ Airport(SFO) ∧ Airport(ATL))
Goal(At(C1, JFK) ∧ At(C2, SFO) ∧ At(C3, SFO))

An optimal solution to Problem 2 would be:

Load(C1, P1, SFO)
Fly(P1, SFO, JFK)
Unload(C1, P1, JFK)
Load(C2, P2, JFK)
Fly(P2, JFK, SFO)
Unload(C2, P2, SFO)
Load(C3, P3, ATL)
Fly(P3, ATL, SFO)
Unload(C3, P3, SFO)

Problem 3, although having a lower plane-to-cargo ratio than the first two problems, can still be achieved in 3 actions per cargo, or 12 actions total. Problem 3's definition is described on the top of the next page:

Init(At(C1, SFO) ∧ At(C2, JFK) ∧ At(C3, ATL) ∧ At(C4, ORD)
∧ At(P1, SFO) ∧ At(P2, JFK)
∧ Cargo(C1) ∧ Cargo(C2) ∧ Cargo(C3) ∧ Cargo(C4)
∧ Plane(P1) ∧ Plane(P2)
∧ Airport(JFK) ∧ Airport(SFO) ∧ Airport(ATL) ∧ Airport(ORD))
Goal(At(C1, JFK) ∧ At(C3, JFK) ∧ At(C2, SFO) ∧ At(C4, SFO))

And here is an optimal solution for Problem 3:

Load(C1, P1, SFO)
Fly(P1, SFO, ATL)
Load(C3, P1, ATL)
Fly(P1, ATL, JFK)
Unload(C1, P1, JFK)
Unload(C3, P1, JFK)
Load(C2, P2, JFK)
Fly(P2, JFK, ORD)
Load(C4, P2, ORD)
Fly(P2, ORD, SFO)
Unload(C2, P2, SFO)
Unload(C4, P2, SFO)

Initially, three non-heuristic searches were used to solve Problems 1-3: breadth-first search (BFS), depth-first search (DFS), and A* with the heuristic being equal to a constant value of 1, which is not a true heuristic. The results obtained can be seen in Table 1 below:

TABLE 1				
Problem	**Search**	**Optimality (# actions)**	**Time Elapsed (seconds, 3 decimals)**	**Nodes Expanded**
1	BFS	6	0.126	43
	DFS	12	0.041	12
	A*, h=1	6	0.141	55
2	BFS	9	48.800	3401
	DFS	346	5.061	350
	A*, h=1	9	41.800	4780
3	BFS	12	347.508	14491
	DFS	3335	177.388	3491
	A*, h=1	12	173.145	17531

For all three Problems, BFS and A* found an optimal solution. BFS was able to find the optimal solution with less node expansions than A*, though. In Problem 1, BFS even found its solution in less time than A*, with A* being more time efficient in Problems 2 and 3. DFS was the fastest in the first two problems and expanded the least nodes in all three problems, but its solutions were not optimal. Based on the results of the non-heuristic searches, it appears that A* would be the best choice of the three, when the problems have more complexity. Although it expands more nodes, the time efficiency surpasses BFS in the more complicated problems. For Problem 1, BFS provides the best time efficiency while still finding an optimal solution.

Next, two heuristics were tested on Problems 1-3, using A* search. One of the heuristics was the “ignore preconditions” heuristic, in which the unsatisfied literals from the goal are counted, based on the given state. The other heuristic is the “level-sum” heuristic - a heuristic which requires a planning graph constructed using the problem and current state. For each literal in a problem’s goal, the level in the planning graph where that goal can be satisfied is added to a total sum. So for example, if a goal is already satisfied in the current state, it will appear on level 0 of the planning graph, so it will not increase the sum. But if the goal appears on level 1, then 1 will be added to the sum. Results from the two heuristics are depicted in Table 2:

TABLE 2				
Problem	**Heuristic**	**Optimality (# actions)**	**Time Elapsed (seconds, 3 decimals)**	**Nodes Expanded**
1	Ignore preconditions	6	0.144	41
	Level-sum	6	2.356	11
2	Ignore preconditions	9	15.480	1450
	Level-sum	9	200.375	86
3	Ignore preconditions	12	61.194	5022
	Level-sum	12	1040.699	312

In these particular problems, the "ignore preconditions" heuristic seems to be a better choice. We can see that it finds an optimal solution in substantially less time than the "level-sum" heuristic, in all three problems, even though the "level-sum" heuristic always expands far fewer nodes. The computational cost of generating the planning graphs seems to offset its efficiency in node expansions, in these specific cases and with this particular implementation. To be more specific, the computational cost of generating a single planning graph is described by Russell and Norvig (2009):

> *"A planning graph is polynomial in the size of the planning problem. For a planning problem with l literals and a actions, each S_i has no more than l nodes and l^2 mutex links, and each A_i has no more than a + l nodes (including the no-ops), $(a + l)^2$ mutex links, and 2(al + l) precondition and effect links. Thus, an entire graph with n levels has a size of $O(n(a + l)^2)$. The time to build the graph has the same complexity."*

It is possible that a more efficient implementation would be superior to the "ignore preconditions" heuristic, even for these specific air cargo problems. Because of the node expansion efficiency of "level-sum", we can infer that for a problem wherein node expansion is significantly more computationally costly, the benefits of using a planning graph would then outweigh its overhead. When compared to the non-heuristic searches, we can see that the "ignore preconditions" heuristic outperforms all optimal solutions in number of node expansions, and is faster in all cases except for BFS in Problem 1, by only tenths of a second. In conclusion, we can say that the "ignore preconditions" provides the right balance of simplicity and efficacy to provide fast, optimal solutions for the three simple air cargo planning problems in this exercise.

Reference

Russell, S., & Norvig, P. (2009). Artificial Intelligence: A Modern Approach, 3rd Edition.