# Project: Planning

By: Omar Kaab

## Part 3: Written Analysis

### Optimal plans for the problems 1, 2, and 3:

Using A* with one of the implemented admissible heuristics we have the following optimal solutions to the air cargo problems:

| ***Problem*** | **Length** | **Detailed solution** |
|---|---|---|
| *Problem 1* | 6 | Load(C1, P1, SFO)<br>Fly(P1, SFO, JFK)<br>Unload(C1, P1, JFK)<br>Load(C2, P2, JFK)<br>Fly(P2, JFK, SFO)<br>Unload(C2, P2, SFO) |
| *Problem 2* | 9 | Load(C3, P3, ATL)<br>Fly(P3, ATL, SFO)<br>Unload(C3, P3, SFO)<br>Load(C1, P1, SFO)<br>Fly(P1, SFO, JFK)<br>Unload(C1, P1, JFK)<br>Load(C2, P2, JFK)<br>Fly(P2, JFK, SFO)<br>Unload(C2, P2, SFO) |
| *Problem 3* | 12 | Load(C2, P2, JFK)<br>Fly(P2, JFK, ORD)<br>Load(C4, P2, ORD)<br>Fly(P2, ORD, SFO)<br>Unload(C4, P2, SFO)<br>Load(C1, P1, SFO)<br>Fly(P1, SFO, ATL)<br>Load(C3, P1, ATL)<br>Fly(P1, ATL, JFK)<br>Unload(C3, P1, JFK)<br>Unload(C1, P1, JFK)<br>Unload(C2, P2, SFO) |

## Comparing non-heuristic search:

First, the results:

| *Problem 1:* | Optimality (path length) | Time elapsed | Expansions | Goal Tests | New Nodes |
|---|---|---|---|---|---|
| *Breadth-first* | Optimal (6) | 0.03s | 43 | 56 | 180 |
| *Depth-first* | Non-optimal (20) | 0.01s | 21 | 22 | 84 |
| *Uniform cost* | Optimal (6) | 0.04s | 55 | 57 | 224 |

| *Problem 2:* | Optimality (path length) | Time elapsed | Expansions | Goal Tests | New Nodes |
|---|---|---|---|---|---|
| *Breadth-first* | Optimal (9) | 13.71s | 3343 | 4609 | 30509 |
| *Depth-first* | Non-optimal (619) | 3.50s | 624 | 625 | 5602 |
| *Uniform cost* | Optimal (9) | 43.91s | 4853 | 4855 | 44041 |

| *Problem 3:* | Optimality (path length) | Time elapsed | Expansions | Goal Tests | New Nodes |
|---|---|---|---|---|---|
| *Breadth-first* | Optimal (12) | 104.37s | 14663 | 18098 | 129631 |
| *Depth-first* | Non-optimal (392) | 1.78s | 408 | 409 | 3364 |
| *Uniform cost* | Optimal (12) | 439.79s | 18224 | 18226 | 159621 |

Depth-first is faster:

As expected depth first search is a much faster algorithm than breadth-first and uniform cost search. As it explores rapidly further nodes in the search tree it ends up missing the optimal solution.

Breadth-first and uniform cost search are "almost" equivalent:

Uniform cost search has one advantage over breadth-first, it can use any step cost, where the latter attributes the same cost to every new step of the path. This capacity is computationally expensive as it implies (AIMA 3rd edition, 3.4.2):

- More computations to manage the frontier as an ordered queue
- More tests are needed to check if we can't find a sorter path to one node in the frontier
- After discovering a solution, we have to wait until the moment we try to expand the goal node to conclude that it is an optimal solution.

## Comparing heuristic searches:

First, the results:

| *Problem 1:* | Time elapsed | Expansions | Goal Tests | New Nodes |
|---|---|---|---|---|
| *A* with "h_1"* | 0.04s | 55 | 57 | 224 |
| *A* with "ignore preconditions"* | 0.04s | 41 | 43 | 170 |
| *A* with "level-sum"* | 1.28s | 11 | 13 | 50 |

| *Problem 2:* | Time elapsed | Expansions | Goal Tests | New Nodes |
|---|---|---|---|---|
| *A* with "h_1"* | 46.32s | 4853 | 4855 | 44041 |
| *A* with "ignore preconditions"* | 16.05s | 1506 | 1508 | 13820 |
| *A* with "level-sum"* | 137.69s | 86 | 88 | 841 |

| *Problem 3:* | Time elapsed | Expansions | Goal Tests | New Nodes |
|---|---|---|---|---|
| *A* with "h_1"* | 400.29s | 18224 | 18226 | 159621 |
| *A* with "ignore preconditions"* | 91.01s | 5115 | 5117 | 45624 |
| *A* with "level-sum"* | 950.71s | 408 | 410 | 3758 |

*"ignore preconditions"* is better than *"h_1"*

Although *"h_1"* is not a real heuristic, it plays here the role of an admissible heuristic because it fits in the definition and never overestimates the distance to the goal. But it's easy to see that *"ignore preconditions"* gives a better estimate of the same distance. This translates to less exploration of the search tree and thus a drastic reduction of the computation time needed to reach an optimal solution.

Then why is *"level-sum"* taking more time?

The same reasoning would lead us to expect a better performance of *"level-sum"* but we clearly see that is not the case. Although *"level-sum"* carries valuable information about the goal and thus makes the algorithm explore a fraction of the nodes explored by the others, it is clearly computationally expensive (especially if the levelled graph is not reused)

## Conclusions: It all depends on what we are looking for

Uninformed searches like depth-first search are effective ways to find rapidly a solution regardless of optimality.

But as complexity of the problem increases, uninformed searches looking for an optimal solution get lost in a large space of useless branches (AIMA 3rd edition, 3.6.1). In this case, using better but easy to compute heuristics put more “intelligence” in our system. A clever trade-off should be found to reduce the number of explored nodes without spending too much time computing the cost of each node.