Heuristic Analysis

# Optimal Plans

|  |  |  |  |
| --- | --- | --- | --- |
| Problem | 1 | 2 | 3 |
| Plan | Length = 6  Load(C1, P1, SFO) Load(C2, P2, JFK) Fly(P2, JFK, SFO) Unload(C2, P2, SFO) Fly(P1, SFO, JFK) Unload(C1, P1, JFK) | Length = 9  Load(C1, P1, SFO) Load(C2, P2, JFK) Load(C3, P3, ATL) Fly(P2, JFK, SFO) Unload(C2, P2, SFO) Fly(P1, SFO, JFK) Unload(C1, P1, JFK) Fly(P3, ATL, SFO) Unload(C3, P3, SFO) | Length = 12  Load(C1, P1, SFO) Load(C2, P2, JFK) Fly(P2, JFK, ORD) Load(C4, P2, ORD) Fly(P1, SFO, ATL) Load(C3, P1, ATL) Fly(P1, ATL, JFK) Unload(C1, P1, JFK) Unload(C3, P1, JFK) Fly(P2, ORD, SFO) Unload(C2, P2, SFO) Unload(C4, P2, SFO) |

# Non-heuristic search comparison

|  |  |  |  |
| --- | --- | --- | --- |
| Problem | 1 | 2 | 3 |
| Breath-first (1) | * Optimal * 0.031s * 43 expansions | * Optimal * 16.986s * 3343 expansions | * Optimal * 138.886s * 14663 expansions |
| Depth-first (3) | * Non-optimal * 0.012s * 21 expansions | * Non-optimal * 4.69s * 624 expansions | * Non-optimal * 2.315s * 408 expansions |
| Uniform cost search (5) | * Optimal * 0.039s * 55 expansions | * Optimal * 10.77s * 4823 expansions | * Optimal * 53.416s * 18235 expansions |

For these specific problems, breath-first is guaranteed to yield an optimal plan. Depth first create a very long, non-optimal plan but yield the best run-time for problem 3 due to a solution being found early in the search. Uniform cost search also yield optimal solution for these specific problem, though it’s not a guarantee for more generic problems.

# Heuristic search comparison

|  |  |  |  |
| --- | --- | --- | --- |
| Problem | 1 | 2 | 3 |
| A\* with ignore\_preconditions | * Optimal * 0.036s * 41 expansions | * Optimal * 3.429s * 1421 expansions | * Optimal * 14.089s * 4859 expansions |
| A\* with level-sum | * Optimal * 0.181s * 39 expansions | * Optimal * 23.754s * 1111 expansions | * Non-optimal * 192.15s * 4295 expansions |

A\* with ignore\_preconditions are much faster than A\* with level-sum as it does not involve constructing a planning graph, which is an expensive operation. However, A\* with level-sum resulted in lower number of expansions.

For these specific problem, A\* with ignore\_preconditions is the better method due to its optimality and faster run-time. In other real-life situations, however, A\* with level-sum may be a more “efficient” heuristic where it is more expensive to “expand”, such as cases where a human actor is involved to find out the “distance” value (e.g. by making a phone call).

Both methods, being a heuristic search, provide no guarantee of finding an optimal path for all problems.

Please note that the implemented heuristic contains problem-specific optimization on the planning graph, such as ignoring mutex.