**AIND-Planning**

**Shiv Shah**

**Heuristic Analysis**

An action schema and three air cargo problems were provided, which were solved using planning algorithm and tested using three uninformed search strategies (Breadth First, Depth First, Uniform Cost Search) and three informed search strategies that uses heuristics (h1 heuristic, ignore preconditions, level sum).

The optimal solution for problem 1, problem 2, and problem 3 were 6, 9, and 12. Below are the sample optimal sequence of actions for each problem.

**Problem 2:**

Load(C3, P3, ATL)

Fly(P3, ATL, SFO)

Unload(C3, P3, SFO)

Load(C2, P2, JFK)

Fly(P2, JFK, SFO)

Unload(C2, P2, SFO)

Load(C1, P1, SFO)

Fly(P1, SFO, JFK)

Unload(C1, P1, JFK)

**Problem 3:**

Load(C2, P2, JFK)

Fly(P2, JFK, ORD)

Load(C4, P2, ORD)

Fly(P2, ORD, SFO)

Unload(C4, P2, SFO)

Load(C1, P1, SFO)

Fly(P1, SFO, ATL)

Load(C3, P1, ATL)

Fly(P1, ATL, JFK)

Unload(C3, P1, JFK)

Unload(C2, P2, SFO)

Unload(C1, P1, JFK)

**Problem 1:**

Load(C1, P1, SFO)

Fly(P1, SFO, JFK)

Unload(C1, P1, JFK)

Load(C2, P2, JFK)

Fly(P2, JFK, SFO)

Unload(C2, P2, SFO)

**Uninformed Search Strategy:**

Test result:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Search Strategy** | **Expansions** | **Execution Time** | **Path Length** | **Optimal** |
| Problem 1 | Breadth First Search | 43 | 0.0563 | 6 | Yes |
| Depth First Search | 21 | 0.0263 | 20 | No |
| Uniform Cost Search | 55 | 0.0686 | 6 | Yes |
| Problem 2 | Breadth First Search | 3343 | 15.32 | 9 | Yes |
| Depth First Search | 624 | 4.4 | 619 | No |
| Uniform Cost Search | 4852 | 19.84 | 9 | Yes |
| Problem 3 | Breadth First Search | 14663 | 68.63 | 12 | Yes |
| Depth First Search | 408 | 2.32 | 392 | No |
| Uniform Cost Search | 18235 | 79.52 | 12 | Yes |

Here, the depth first search seems to beat Breadth First and Uniform Cost Search in execution time and number of node expansions in all cases. Even though, it is fastest and uses least memory of all, it is clearly not able to find optimal solution in any case. Its pathlength is way more than the optimal path length. On the other hand, both Breadth First and Uniform Cost search has found optimal solution for all three problems, but at the cost of more execution time and more memory usage. If we compare Breadth First and Uniform Cost search, we see that breadth first is performing better in space and execution time. Also, **the breadth first will always find the shortest path solution. Because breadth-first search explores the shallower nodes first, so as soon as the goal node is found, we know it is the shallowest goal node, and this will always be true because we have the same path cost (1 in this case) in the problem search space (Russell and Norvig, AIMA).**

Practically, we would want the solution to have less path length than the one provided by the depth first search, if not the optimal. Because, taking longer path will eventually end up costing more money and resources. So, even if we spend more on space and execution spend earlier, we will end up saving money in the end. So, Breadth first is the recommended strategy here.

**Informed Search Strategy:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Search Strategy** | **Expansions** | **Execution Time** | **Path Length** | **Optimal** |
| Problem 1 | A\* h\_1 | 55 | 0.0751 | 6 | Yes |
| A\* h\_ignore\_precondition | 41 | 0.0591 | 6 | Yes |
| A\* h\_pg\_levelsum | 11 | 0.4735 | 6 | Yes |
| Problem 2 | A\* h\_1 | 4852 | 20.23 | 9 | Yes |
| A\* h\_ignore\_precondition | 1450 | 6.13 | 9 | Yes |
| A\* h\_pg\_levelsum | 86 | 40.9 | 9 | Yes |
| Problem 3 | A\* h\_1 | 18235 | 81.04 | 12 | Yes |
| A\* h\_ignore\_precondition | 5040 | 22.6 | 12 | Yes |
| A\* h\_pg\_levelsum | 316 | 202.55 | 12 | Yes |

These searches are uniform cost search with a heuristic function. The h\_1 is not a true heuristic, so it will behave similar to the uniform cost search. Here, h\_pg\_levelsum is expands fewer nodes than h\_ignore\_precondition, however the latter performs much better in time, while providing optimum path length. So, h\_ignore\_precondition is the best here.

**Informed Search Strategy vs. Uninformed Search Strategy**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Search Strategy** | **Expansions** | **Execution Time** | **Path Length** | **Optimal** |
| Problem 1 | A\* h\_ignore\_precondition | 41 | 0.0591 | 6 | Yes |
| Breadth First Search | 43 | 0.0563 | 6 | Yes |
| Problem 2 | A\* h\_ignore\_precondition | 1450 | 6.13 | 9 | Yes |
| Breadth First Search | 3343 | 15.32 | 9 | Yes |
| Problem 3 | A\* h\_ignore\_precondition | 5040 | 22.6 | 12 | Yes |
| Breadth First Search | 14663 | 68.63 | 12 | Yes |

Comparing two of the best strategies from the two types of searches, we clearly see A\* search with ignore\_precondition heuristic the winner here. While both provides optimal solution, heuristic search has faster execution and requires less memory. So, A\* with ignore precondition would be the best choice for this problem. As discussed in the AIMA textbook as well as in the lecture, heuristic search does perform better by using that extra information about the goal, expanding those nodes first that are estimated to be closer to the goal.