**Background**

Planning problems are a staple of classical AI and project 3, Implementing a Planning Search, requires that a program for performing planning search is completed (framework provided by Udacity) and that the results of running various types of search using the program are compared with each other. This report provides the results. resulting from my own testing and analysis of these results.

**Uninformed Search Strategies**

3 uninformed search strategies were compared for air cargo problems p1, p2 and p3:

* Breadth First Search
* Depth First Search
* Uniform Cost Search

The results are shown in tables 1,2 and 3 below.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Problem** | **Search** | **Node Expansions** | **Goal Tests** | **Time Elapsed (s)** | **Plan Length** | **Planning Path** |
| p1 | BFS | 43 | 56 | 0.19718860317315653 | 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) |
| p1 | DFS | 21 | 22 | 0.02392778599523538 | 20 | Fly(P1, SFO, JFK)  Fly(P2, JFK, SFO)  Load(C2, P1, JFK)  Fly(P1, JFK, SFO)  Fly(P2, SFO, JFK)  Unload(C2, P1, SFO)  Fly(P1, SFO, JFK)  Fly(P2, JFK, SFO)  Load(C2, P2, SFO)  Fly(P1, JFK, SFO)  Load(C1, P2, SFO)  Fly(P2, SFO, JFK)  Fly(P1, SFO, JFK)  Unload(C2, P2, JFK)  Unload(C1, P2, JFK)  Fly(P2, JFK, SFO)  Load(C2, P1, JFK)  Fly(P1, JFK, SFO)  Fly(P2, SFO, JFK)  Unload(C2, P1, SFO) |
| p1 | UCS | 55 | 57 | 0.05737865428834286 | 6 | Load(C1, P1, SFO)  Load(C2, P2, JFK)  Fly(P1, SFO, JFK)  Fly(P2, JFK, SFO)  Unload(C1, P1, JFK)  Unload(C2, P2, SFO) |

**Table 1** – Results of using uniformed search strategies on air cargo problem p1

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Problem** | **Search** | **Node Expansions** | **Goal Tests** | **Time Elapsed (s)** | **Plan Length** | **Plan Path** |
| p2 | BFS | 3343 | 4609 | 26.171465976646548 | 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) |
| p2 | DFS | 624 | 625 | 4.16513846587114 | 619 | Too large to fit here |
| p2 | UCS | 4853 | 4855 | 15.863818732601194 | 9 | 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(C1, P1, JFK)  Unload(C2, P2, SFO)  Unload(C3, P3, SFO) |

**Table 2** – Results of using uniformed search strategies on air cargo problem p2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Problem** | **Search** | **Node Expansions** | **Goal Tests** | **Time Elapsed (s)** | **Plan Length** | **Plan Path** |
| p3 | BFS | 14663 | 18098 | 164.43370933846663 | 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) |
| p3 | DFS | 408 | 409 | 2.242594127521531 | 392 | Too large to fit here |
| p3 | UCS | 18151 | 18153 | 67.93223655932519 | 12 | 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(C1, P1, JFK)  Unload(C2, P2, SFO)  Unload(C3, P1, JFK)  Unload(C4, P2, SFO) |

**Table 3** – Results of using uniformed search strategies on air cargo problem p3

From the above we can see that BFS and UCS both arrive at the solutions with the lowest number of path steps for all 3 problems. In all 3 problems BFS takes longest to calculate the solution. In all 3 cases UCS takes roughly 50% of the time BFS takes to arrive at a solution. In all 3 problems DFS takes a fraction of the time to produce a solution but the solution path is much longer and is not one that would be used in practice. UCS is therefore the best strategy of the uninformed search strategies tested.

**Searching Using Planning Graph Heuristics**

Logic was added to the search program to construct a Planning Graph data structure and to use this to obtain heuristics that could be employed in the A\* Search.

* Relaxed Constraint Heuristic
* Level Sum Heuristic