Analysis:

In this article, I will summarize the performance of different search algorithms and compare them with each other in different problem contexts.

# 1. The optimal plans:

|  |  |  |
| --- | --- | --- |
| Problem #1 | Problem #2 | Problem #2 |
| Load(C1, P1, SFO)  Load(C2, P2, JFK)  Fly(P2, JFK, SFO)  Unload(C2, P2, SFO)  Fly(P1, SFO, JFK)  Unload(C1, P1, JFK) | Load(C1, P1, SFO)  Fly(P1, SFO, JFK)  Load(C2, P2, JFK)  Fly(P2, JFK, SFO)  Load(C3, P3, ATL)  Fly(P3, ATL, SFO)  Unload(C3, P3, SFO)  Unload(C1, P1, JFK)  Unload(C2, P2, SFO) | Load(C2, P2, JFK)  Fly(P2, JFK, ORD)  Load(C4, P2, ORD)  Fly(P2, ORD, SFO)  Load(C1, P1, SFO)  Fly(P1, SFO, ATL)  Load(C3, P1, ATL)  Fly(P1, ATL, JFK)  Unload(C4, P2, SFO)  Unload(C3, P1, JFK)  Unload(C1, P1, JFK)  Unload(C2, P2, SFO) |

# 2. Comparison of the non-heuristics search:

I chose to compare only algorithms #1, #3 and #5, namely breadth\_first\_search, depth\_first\_graph search and uniform\_cost\_search.

And the performance of them are summarized as follows:

Problem #1:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Breadth first search | Depth first search | Uniform cost search |
| Plan length | 6 | 20 | 6 |
| Expansion nodes | 43 | 21 | 55 |
| Time | 0.067 | 0.032 | 0.067 |

Problem #2:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Breadth first search | Depth first search | Uniform cost search |
| Plan length | 9 | 619 | 9 |
| Expansion nodes | 3343 | 624 | 4853 |
| Time | 21.57 | 5.27 | 20.49 |

Problem #3:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Breadth first search | Depth first search | Uniform cost search |
| Plan length | 12 | 392 | 12 |
| Expansion nodes | 14663 | 403 | 18223 |
| Time | 159.01 | 2.90 | 85.47 |

Obviously depth\_first\_graph\_search fails to find the optimal solution, due to the fact that it is not guaranteed to find an optimal solution in its nature. But the “Depth first search” is very fast and only consumes very little memory, if optimality is not important, it could be a feasible choice.

# 3. Comparison of the non-heuristics search:

And the performance of A\* with the “ignore preconditions” and “level-sum” heuristics are summarized as follows:

Problem #1:

|  |  |  |
| --- | --- | --- |
|  | Ignore Precondition | Level-Sum |
| Plan length | 6 | 6 |
| Expansion nodes | 41 | 11 |
| Time | 0.12 | 1.07 |

Problem #2:

|  |  |  |
| --- | --- | --- |
|  | Ignore Precondition | Level-Sum |
| Plan length | 9 | 9 |
| Expansion nodes | 1450 | 86 |
| Time | 12.04 | 101.53 |

Problem #3:

|  |  |  |
| --- | --- | --- |
|  | Ignore Precondition | Level-Sum |
| Plan length | 12 | 12 |
| Expansion nodes | 5040 | 318 |
| Time | 45.86 | 479.55 |

The two A\* algorithms variants all find optimal solutions, but it works much faster with the “ignore preconditions” heuristics. However, the expansion nodes of “level-sum” heuristics is much lower.

# 4. Comparison of non-heuristic search and heuristic search algorithm

For problem #1, the best heuristic is A\* with “ignore preconditions”, but it is still slower than “breadth first search”. I think that is because the search space is small enough, heuristics doesn’t matter much at all. And the calculation of heuristic causes extra overhead actually.

For problem #2 and problem #3, the best heuristics are still A\* with “ignore preconditions”, and they are better than the non-heuristics ones. I believe that is because the search space is big enough, simple brute force search is not efficient any more. And appropriate heuristics may lead to correct direction earlier.

Finally, I want to give an explanation of why the “ignore precondition” heuristic is much faster that “level-sum”. I think that’s because the overhead caused by constructing planning graph, which is itself a time-consuming process.