Heuristic Analysis

# OPTIMAL SOLUTION FOR PROBLEMS

Problem 1:

1. Load(C1, P1, SFO)
2. Fly(P1, SFO, JFK)
3. Load(C2, P2, JFK)
4. Fly(P2, JFK, SFO)
5. Unload(C1, P1, JFK)
6. Unload(C2, P2, SFO)

Problem 2:

1. Load(C1, P1, SFO)
2. Load(C2, P2, JFK)
3. Load(C3, P3, ATL)
4. Fly(P1, SFO, JFK)
5. Fly(P2, JFK, SFO)
6. Fly(P3, ATL, SFO)
7. Unload(C3, P3, SFO)
8. Unload(C2, P2, SFO)
9. Unload(C1, P1, JFK)

Problem 3:

1. Load(C1, P1, SFO)
2. Load(C2, P2, JFK)
3. Fly(P1, SFO, ATL)
4. Load(C3, P1, ATL)
5. Fly(P2, JFK, ORD)
6. Load(C4, P2, ORD)
7. Fly(P2, ORD, SFO)
8. Fly(P1, ATL, JFK)
9. Unload(C4, P2, SFO)
10. Unload(C3, P1, JFK)
11. Unload(C1, P1, JFK)
12. Unload(C2, P2, SFO)

# SEARCH RESULTS

**Problem 1:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Method | Expansions | Goal Tests | New Nodes | Time | Plan length |
| breadth\_first\_search | 43 | 56 | 180 | 0.041 | 6 |
| breadth\_first\_tree\_search | 1458 | 1459 | 5960 | 1.132 | 6 |
| depth\_first\_graph\_search | 21 | 22 | 84 | 0.018 | 20 |
| depth\_limited\_search | 101 | 271 | 414 | 0.115 | 50 |
| uniform\_cost\_search | 55 | 57 | 224 | 0.061 | 6 |
| recursive\_best\_first\_search with h\_1 | 4229 | 4230 | 17023 | 3.750 | 6 |
| greedy\_best\_first\_graph\_search with h\_1 | 7 | 9 | 28 | 0.007 | 6 |
| astar\_search with h\_1 | 55 | 57 | 224 | 0.057 | 6 |
| astar\_search with h\_ignore\_preconditions | 41 | 43 | 170 | 0.050 | 6 |
| astar\_search with h\_pg\_levelsum | 11 | 13 | 50 | 0.637 | 6 |

For problem 1, the best method is “greedy\_best\_first\_graph\_search with h\_1” as is spend the least time and use the least expansion, goal test and new nodes. Besides this method, “breadth\_first\_search” and “astar\_search” with heuristics also did a good job to find solutions. “astar\_search with h\_pg\_levelsum” save the number of new nodes efficiently, but takes longer time as it takes longer time to get the level sum. “depth\_first\_graph\_search” also saves time and number of new nodes, but it didn’t find the optimal result.

**Problem 2:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Method | Expansions | Goal Tests | New Nodes | Time | Plan length |
| breadth\_first\_search | 3346 | 4612 | 30534 | 18.049 | 9 |
| breadth\_first\_tree\_search | 10 mins + | | | | |
| depth\_first\_graph\_search | 107 | 108 | 959 | 0.389 | 105 |
| depth\_limited\_search | 10 mins + | | | | |
| uniform\_cost\_search | 4853 | 4855 | 44041 | 16.140 | 9 |
| recursive\_best\_first\_search with h\_1 | 10 mins+ | | | | |
| greedy\_best\_first\_graph\_search with h\_1 | 998 | 1000 | 8982 | 3.071 | 21 |
| astar\_search with h\_1 | 4853 | 4855 | 44041 | 16.330 | 9 |
| astar\_search with h\_ignore\_preconditions | 1450 | 1452 | 13303 | 4.655 | 9 |
| astar\_search with h\_pg\_levelsum | 86 | 88 | 841 | 58.762 | 9 |

For Problem 2, the best method is “astar\_search with h\_ignore\_preconditions” as it found the optimal result at with the least time. Although it has more expansions, goal tests and new nodes than “astar\_search with h\_pg\_levelsum”, but it saves the time very efficiently by more than 1/10. Same as problem 1, “depth\_first\_graph\_search” saves expansions, goal test, new nodes and time, but it didn’t find the optimal result.

**Problem 3:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Method | Expansions | Goal Tests | New Nodes | Time | Plan length |
| breadth\_first\_search | 14663 | 18098 | 129631 | 144.748 | 12 |
| breadth\_first\_tree\_search | 10 mins+ | | | | |
| depth\_first\_graph\_search | 408 | 409 | 3364 | 2.150 | 392 |
| depth\_limited\_search | 10 mins+ | | | | |
| uniform\_cost\_search | 18235 | 18237 | 159716 | 71.388 | 12 |
| recursive\_best\_first\_search with h\_1 | 10 mins+ | | | | |
| greedy\_best\_first\_graph\_search with h\_1 | 5614 | 5616 | 49429 | 21.258 | 22 |
| astar\_search with h\_1 | 18235 | 18237 | 159716 | 70.467 | 12 |
| astar\_search with h\_ignore\_preconditions | 5040 | 5042 | 44944 | 19.122 | 12 |
| astar\_search with h\_pg\_levelsum | 318 | 320 | 2934 | 171.532 | 12 |

For problem 3: The best method is astar\_search with h\_ignore\_preconditions” as it found the optimal result at with the least time. Although it has more expansions, goal tests and new nodes than “astar\_search with h\_pg\_levelsum”, but it saves the time very efficiently by more than 1/10. Same as problem 1 & 2, “depth\_first\_graph\_search” saves expansions, goal test, new nodes and time, but it didn’t find the optimal

# SEARCHING METHODS PERFORMANCE ANALYSIS

**Breadth First Searches & Depth First Searches:**

BFS will always find the optimal answer while take longer time and have more expansions, goal tests and number of new nodes compared with DFS. DFS generally get an answer very fast but rarely find the optimal plan as the result of the search method.

BFS’s searching time will increase sharply as the problem becomes more complex.

**Breadth First Searches & uniform\_cost\_search:**

Uniform\_cost\_search has slightly more expansions, goal tests and number of new nodes compared with DFS. But the time used will increase slower compared with BFS as problem became more complex.

**Astar Searches:**

Astar searches with difference heuristics will always find the optimal results, but the performance depends on the heuristics a lot. “h\_1” heuristic value is easy to get but not powerful enough to guide the search to get optimal result faster. It works just like the “uniform\_cost\_search”. “h\_ignore\_preconditions” heuristics is also easy to get and won’t expanse every node of the space as the cost can be estimated. In this way, the search space and time is reduced, which means less expansions, goal tests and number of new nodes compared with “h\_1” heuristic. “h\_pg\_levelsum” is the more efficient in guiding the search in the right direction, that is to narrow down the search space. However, to get the heuristic value will take more time.

# CONCLUSION

To sum up. If not using heuristics, “uniform\_cost\_search” will be the best choice as it will get the optimal results with less time. And it is actually working like “astar\_search with h\_1”. If use heuristics, among all the choices above, “astar\_search with h\_ignore\_preconditions” give the optimal results with less time, which should be the best choice. However, if we value less search space more, instead of less time more, “astar\_search with h\_pg\_levelsum” will be a better choice.