Udacity Artificial Intelligence Nanodegree May 2017 Cohort
Kelvin Kong. E:kelvin86@gmail.com

# AI Logic and Planning Research Review

## Air Cargo Problem 1

*Goals: At(C1, JFK) ^ At(C2, SFO)*

| | | Air Cargo Problem 1 | | | | |
|---|---|---|---|---|---|---|
| **No.** | **Search** | **Expansions** | **Goal Tests** | **New Nodes** | **Time (s)** | **Plan Length** |
| 1 | Breadth First Search | 43 | 56 | 180 | 0.069 | 6 |
| 2 | Breadth First Tree Search | 1458 | 1459 | 5960 | 2.211 | 6 |
| 3 | Depth First Graph Search | 12 | 13 | 48 | 0.01659 | 12 |
| 4 | Depth Limited Search | 101 | 271 | 414 | 0.196 | 50 |
| 5 | Uniform cost search | 55 | 57 | 224 | 0.097 | 6 |
| 6 | Recursive best first search h1 | 4229 | 4230 | 17029 | 5.94 | 6 |
| 7 | Greedy best first graph search h1 | 7 | 9 | 28 | 0.0063 | 6 |
| 8 | A* search h1 | 55 | 57 | 224 | 0.088 | 6 |
| 9 | A* search ignore preconditions heuristic | 41 | 43 | 170 | 0.057 | 6 |
| 10 | A* search level sum heuristic | 11 | 13 | 50 | 1.085 | 6 |

Optimal Solution:

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

Problem 1 is solved optimally using BFS with 43 expansions, 56 Goal Tests and 180 nodes created in 0.069 seconds. The solution is optimal at 6 steps.

Using depth first graph search, problem 1 is solved correctly with only 12 expansions, 13 goal tests and 48 nodes added to the search. The time taken to locate the solution is also faster than BFS. Although the final goal state is reached with less expansions and node creations, the suggested path will take 12 steps to reach the goal state, which is 6 steps more than the path obtained from BFS, thus not optimal.

Greedy best first graph search returns a correct solution for problem 1, with only 7 expansions, 9 goal tests and 28 nodes added. The solution is also optimal where the plan length is 6. This appears to be an improvement over breadth first search where the solution is similar but efficiency is observed in all metrics and runtime of only 0.0063s.

A*Search with ignore preconditions heuristics produced an optimal solution with a plan length of 6 with 41 expansions, 43 goal tests and 170 node creations. Runtime appears to be fast for this small problem at 0.05s. Compared to GBFS, all of the metrics is larger and more resource-consuming.

A* Search with level sum heuristics returns an optimal solution similar to others, with some slight variations on individual sequence. The number of expansion (11), goal tests (13) and nodes (50) is significantly less than the A* variant using ignore preconditions heuristics.

## Conclusion

For a simple problem such as Problem 1, the greedy best first graph search appears to be the best search algorithm among all non-heuristic searches considering that it performed best in all metrics while producing optimal results and also being the fastest, at only 0.0063s. Performance wise, it also beats all variants of A* search (h1, levelsum, ignore preconditions heuristic) in all metrics as well, expanding only 7 times, 9 goal tests and 28 node creations.

Between heuristic searches, A* with level sum heuristic managed to produce the lowest metrics at 11 expansions, 13 goal tests and 50 new nodes. It is almost 4 times more optimized compared to the A* ignore preconditions heuristic version (41 expansions, 43 goal tests, 170 new nodes), however it is slower, clocking at 1.08s as compared to 0.057s (ignore preconditions h). In a simple problem like this, this miniscule difference won't likely appear as a noticeable difference to a human user.

We conclude that greedy best first graph search is the best performing search algorithm for Problem 1.

## Air Cargo Problem 2

*Goals: At(C1, JFK) ^ At(C2, SFO) ^ At(C3, SFO)*

| | | Air Cargo Problem 2 | | | | |
|---|---|---|---|---|---|---|
| **No.** | **Search** | **Expansions** | **Goal Tests** | **New Nodes** | **Time (s)** | **Plan Length** |
| 1 | Breadth First Search | 3346 | 4612 | 30534 | 42.47 | 9 |
| 2 | Breadth First Tree Search | Timeout | | | | |
| 3 | Depth First Graph Search | 1124 | 1125 | 10017 | 19.56 | 1085 |
| 4 | Depth Limited Search | 213491 | 1967093 | 1967471 | 1916.17 | 50 |
| 5 | Uniform cost search | 4853 | 4855 | 44041 | 27.17 | 9 |
| 6 | Recursive best first search h1 | Timeout | | | | |
| 7 | Greedy best first graph search h1 | 895 | 897 | 8008 | 4.64 | 15 |
| 8 | A* search h1 | 4853 | 4855 | 44041 | 30.3 | 9 |
| 9 | A* search ignore preconditions heuristic | 1450 | 1452 | 13303 | 8.34 | 9 |
| 10 | A* search level sum heuristic | 86 | 88 | 841 | 100.46 | 9 |

Solution:

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

Breadth first search produced an optimal 9 step solution with 3346 total expansions, 4612 goal tests and 30534 nodes added. The goal is reached in ~42 seconds.

Depth first Graph search produced a result in ~19 seconds, however the solution is not optimal with a plan length of 1085 which is way above our threshold and therefore not acceptable in this case.

Problem 2 was solved with an optimal solution by breadth first search, uniform cost search, A* ignore preconditions heuristic search and A* level sum heuristic search. Greedy best first graph search h1 produced a reasonable result in only 4.6 seconds, however it is not optimal (plan length of 15). From the metrics we would conclude that A* search with ignore preconditions heuristic performed the best in terms of speed at 8.34 seconds as compared to the rest. A* with levelsum heuristic produced the least expansions (86) and nodes (841), however the time taken(100.46s) is almost 10 times longer compared to A* ignore preconditions variant.

## Conclusion

Based on the results, we conclude that A* search with ignore preconditions heuristic is the best option for Problem 2 factoring in the search time with our current setup, despite that it may not be the most economical in terms of resource usage.

In this relatively simple problem, we are able to see that heuristic searches produced solutions with much less branching factor. And a simple heuristic like ignoring preconditions managed to outperform uniform cost search (non-heuristic) by reducing the number of expansions from 4853 to 1450, while also cutting down the search time from 27s to 8s. This difference is amplified compared to the much simpler Problem 1.

## Air Cargo Problem 3

*Goals: At(C1, JFK) ^ At(C2, SFO) ^ At(C3, JFK) ^ At(C4, SFO)*

| | | Air Cargo Problem 3 | | | | |
|---|---|---|---|---|---|---|
| **No.** | **Search** | **Expansions** | **Goal Tests** | **New Nodes** | **Time (s)** | **Plan Length** |
| 1 | Breadth First Search | 14120 | 17673 | 124926 | 247.08 | 12 |
| 2 | Breadth First Tree Search | Timeout | | | | |
| 3 | Depth First Graph Search | 677 | 678 | 5608 | 9.153 | 660 |
| 4 | Depth Limited Search | Timeout | | | | |
| 5 | Uniform cost search | 18233 | 18235 | 159697 | 106.81 | 12 |
| 6 | Recursive best first search h1 | Timeout | | | | |
| 7 | Greedy best first graph search h1 | 5180 | 5182 | 45655 | 32.64 | 22 |
| 8 | A* search h1 | 18233 | 18235 | 159697 | 114.49 | 12 |
| 9 | A* search ignore preconditions heuristic | 4951 | 4953 | 44051 | 37.97 | 12 |
| 10 | A* search level sum heuristic | 305 | 307 | 2816 | 440.35 | 12 |

Optimal Solution:

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

## Conclusion

For problem 3, depth first graph search and greedy best first graph search both produced results in 9.15s and 32.64s, however the solution isn't optimal with a plan length of 660 and 22 respectively.

Searches that produced optimal solutions are breadth first search, uniform cost search, A* search h1, A* with ignore preconditions heuristic and A* with level sum heuristic. Among all searches, A* level sum heuristic expanded the least with only 305 expansions, 307 goal tests and 2816 nodes, the most economical in terms of resource requirements. However, the search time was 440.35s, which is orders of magnitude higher than A* search with ignore preconditions heuristic. The latter variant managed to obtain an optimal solution in 37.97s, although the other metrics – expansions (4951), Goal tests( 4953) and new nodes (44051) are much higher compared to the level sum variant.

Non heuristic searches such as breadth first search and uniform cost search fared badly in this problem, with very large expansions (14120), goal tests(17673) and 124926 node creations. They are also not as good in terms of speed, clocking at 247.08s and 106.81s respectively.

For problem 3, we conclude that A* search with ignore preconditions heuristic is the best performing search algorithm.

### Discussion

In general, heuristic searches performed much more efficiently compared to non-heuristic searches such as breadth first search and uniform cost search or A*search h1(which is essentially the same as uniform cost search). It is superior in all of the benchmarked metrics while still being faster at the same time. It appears that the ignore preconditions heuristic seemed to perform reasonably good in this set of problems. It works by dropping all preconditions from actions, which results in a relaxed problem. A set-cover problem can achieve a more accurate heuristic, however it is NP-hard. (Russell & Norvig, 2010)

When the problem complexity increases, the performance improvement becomes much more pronounced. As the search space becomes larger, heuristics will play a major role in reducing the search space and thus simplifying the problem significantly, allowing the discovery of an optimal solution within a reasonable amount of time.

## References

Russell, S., & Norvig, P. (2010). *Artificial Intelligence: A Modern Approach (3rd Edition).* Pearson.