AIND Implement a Planning Search Project - Analysis of the Results

Marco Zorzi

The first Air Cargo Problem is the simplest of the three problems to resolve by search algorithms.

I ran all 10 types of Search Algorithms with this result:

Solving Air Cargo Problem 1						
Search Algorithm	**Expansions**	**Goal Tests**	**New Nodes**	**Plan length**	**Time elapsed (s)**	**Optimal Plan**
Breadth First Search	43	56	180	6	0.119	YES
Breadth First Tree Search	1458	1459	5960	6	3.601	YES
Depth First Graph Search	21	22	84	20	0.056	NO
Depth Limited Search	101	271	414	50	0.333	NO
Uniform Cost Search	55	57	224	6	0.139	YES
Recursive Best First Search with h_1 Heuristic	4229	4230	17023	6	10.566	YES
Greedy Best First Graph Search with h_1 Heuristic	**7**	**9**	**28**	**6**	**0.019**	**YES**
A* Search with h_1 Heuristic	55	57	224	6	0.163	YES
A* Search with Ignore Preconditions Heuristic	41	43	170	6	0.142	YES
A* Search with Planning Graph Levelsum Cost Heuristic	**11**	**13**	**50**	**6**	**3.551**	**YES**

Depth First Graph Search and Depth Limited Search are not optimal.

The fastest optimal plan Search Algorithm is Greedy Best First Graph Search with h_1 Heuristic and it's also the lowest expansion.

Looking at A* Search algorithms we see that A* Search with Ignore Preconditions Heuristic is the fastest of the three A* Search Algorithm tested but A* Search with Planning Graph Level Sum Cost Heuristic has got a lower expansion.

An optimal sequence of actions provided for Problem 1 by Greedy Best First Graph Search with h_1 Heuristic:

Load(C1, P1, SFO)
Load(C2, P2, JFK)
Fly(P1, SFO, JFK)
Fly(P2, JFK, SFO)
Unload(C1, P1, JFK)
Unload(C2, P2, SFO)

For the second problem Breadth First Tree Search and Recursive Best First Search with h_1 Heuristic takes too much time to be completed so I stopped them.

Solving Air Cargo Problem 2						
Search Algorithm	**Expansions**	**Goal Tests**	**New Nodes**	**Plan length**	**Time elapsed (s)**	**Optimal Plan**
Breadth First Search	3401	4672	31049	9	25.281	YES
Breadth First Tree Search						
Depth First Graph Search	1192	1193	10606	1138	9.048	NO
Depth Limited Search	253158	2336904	2337280	50	1230.612	NO
Uniform Cost Search	4761	4763	43206	9	11.959	YES
Recursive Best First Search with h_1 Heuristic						
Greedy Best First Graph Search with h_1 Heuristic	**550**	**552**	**4950**	**9**	**1.350**	**YES**

A* Search with h_1 Heuristic	4761	4763	43206	9	11.855	YES
A* Search with Ignore Preconditions Heuristic	**1450**	**1452**	**13303**	**9**	**4.314**	**YES**
A* Search with Planning Graph Level sum Cost Heuristic	86	88	841	9	313.748	YES

In the second Air Cargo Problem, the fastest optimal plans are Greedy Best First Graph Search with h_1 Heuristic and A* Search with Ignore Preconditions Heuristic.

A* Search with Planning Graph Level Sum Cost Heuristic has the best Expansions due to the good algorithm, but it takes several times due to its complexity.

An optimal sequence of actions provided for Problem 2 by A* Search with Planning Graph Level sum Cost Heuristic:

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(C2, P2, SFO)
Unload(C1, P1, JFK)

For the third problem Breadth First Tree Search, Depth Limited Search and Recursive Best First Search with h_1 Heuristic takes too much time to be completed so I stopped them.

Solving Air Cargo Problem 3						
Search Algorithm	**Expansions**	**Goal Tests**	**New Nodes**	**Plan length**	**Time elapsed (s)**	**Optimal Plan**
Breadth First Search	14491	17947	128184	12	333.936	YES
Breadth First Tree Search						
Depth First Graph Search	2099	2100	17558	2014	82.536	NO
Depth Limited Search						
Uniform Cost Search	17783	17785	155920	12	190.177	YES
Recursive Best First Search with h_1 Heuristic						
Greedy Best First Graph Search with h_1 Heuristic	4031	4033	35794	22	42.837	NO
A* Search with h_1 Heuristic	17783	17785	155920	12	192.707	YES
A* Search with Ignore Preconditions Heuristic	**5003**	**5005**	**44586**	**12**	**62.395**	**YES**
A* Search with Planning Graph Levelsum Cost Heuristic	311	313	2863	12	1126.468	YES

In this case, Greedy Best First Graph Search with h_1 Heuristic is not an optimal plan, in fact in the problem 1 and 2 it was apparently a “optimal plan” but Best First Graph Search is not an optimal algorithm by design and the third problem show it.

Overall Uniform Cost Search is the one of the best search algorithm with less expansion and a fast response. A* Search with Planning Graph Level Sum Cost Heuristic is the best one of A* Search algorithm.

A* Search with Planning Graph Level Sum Cost Heuristic is a very good algorithm looking at the expansion nodes, but it takes too much time due its complexity.

An optimal sequence of actions for Problem 3 provided by A* Search with Ignore Preconditions Heuristic:

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)

Depth-first search always expands the deepest node in the current frontier of the search tree. The search proceeds immediately to the deepest level of the search tree, where the nodes have no successors. As those nodes are expanded, they are dropped from the frontier, so then the search "backs up" to the next deepest node that still has unexplored successors. Depth-first search can enter in infinite loops in finite state spaces and create a proliferation of redundant paths. In infinite state spaces fail if an infinite non-goal path is encountered. Depth-first search is nonoptimal.

The failure of depth-first search in infinite state spaces can be alleviated by supplying depth-first search with a predetermined depth limit. That is, nodes at depth are treated as if they have no successors. This approach is called depth-limited search. The depth limit solves the infinite-path problem. Unfortunately, it also introduces an additional source of incompleteness if we choose $l < d$, that is, the shallowest goal is beyond the depth limit. (This is likely when d is unknown.) Depth-limited search will also be nonoptimal if we choose $l > d$. Its time complexity is $O(b^l)$ and its space complexity is $O(b^l)$. Depth-first search can be viewed as a special case of depth-limited search with $l=\infty$.

Uniform-cost search expands the node n with the lowest path cost g(n). This is done by storing the frontier as a priority queue ordered by g. Uniform-cost search is guided by path costs rather than depths, so its complexity is not easily characterized in terms of b and d. Instead, let $C*$ be the cost of the optimal solution, and assume that every action costs at least. Then the algorithm's worst-case time and space complexity is much greater than b^d. This is because Uniform-cost search can explore large trees of small steps before exploring paths involving large and perhaps useful steps search can explore large trees of small steps before exploring paths involving large and perhaps useful steps. Uniform-cost search expands nodes in order of their optimal path cost. So, the first goal node selected for expansion must be the optimal solution.

Greedy Best First Graf Search is the fastest algorithm tested in this project because goes for the goal, but it can be easily find a plan that is not optimal.

Breadth-first search is a simple strategy in which the root node is expanded first, then all the successors of the root node are expanded next, then their successors, and so on. In general, all the nodes are expanded at a given depth in the search tree before any nodes at the next level are expanded. Breadth First Search is then optimal but slow.

A* Search algorithm combines the strengths of Breadth First Search (it finds the shortest path) and Greedy Best First (it's fast) and it is optimal because the optimistic heuristic: in fact, all good paths are explored and never ignores an optimal plan.

A* Search algorithm with ignore-preconditions heuristic is significantly faster than the level-sum heuristic because has a simple heuristic than level-sum heuristic. It drops all preconditions from actions and every action becomes applicable in every state and any single goal fluent can be achieved in one step.

A* Search with Planning Graph Level Sum Cost Heuristic returns the sum of the level costs of the goals with more accuracy but requiring greater computing ability.

Thanks to the video lessons of Udacity AIND about game agents where were explained to prefer an easy algorithm to a complex one to reach best performances and looking at the result of the various tests it can be concluded that the algorithm such as A* Search with Ignore Preconditions Heuristic is preferable to the other ones in this case.

Reference:

Book:
Artificial Intelligence a Modern Approach (3rd Edition), RUSSELL & NORVIG

Site:
http://cs.stanford.edu/people/abisee/gs.pdf