Airport solution analyses

By Benjamin Schultz Larsen

I will start by showing all my results, and thereafter compare the different algorithms by using data from all of the problems. The reason I am doing it this way is the pattern is the same (for time, planning length, expansions…) for all 3 problems.

Results

Problem 1

	Expansions	Goal Tests	New Nodes	Plan length	Time in sec
Breadth first search	43	56	180	6	0,0147
Depth first graph search	21	22	84	20	0,0062
Uniform cost search	55	57	224	6	0,1622
A* planning graph	11	13	50	6	0,3386
A* ignore precondition	41	43	170	6	0,0165
Bonus (domain dependent)	11	13	46	6	0,0038

Optimal plan
Length 6

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

Problem 2

	Expansions	Goal Tests	New Nodes	Plan length	Time in sec
Breadth first search	3343	4609	30509	9	4,6677
Depth first graph search	624	625	5602	619	2,0612
Uniform cost search	4852	4854	44030	9	5,3622
A* planning graph	86	88	841	9	29,2466
A* ignore precondition	1450	1452	13303	9	1,6748
Bonus (domain dependent)	112	114	1040	9	0,1349

Optimal plan
Length 9

```
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)
```

Problem 3

	Expansions	Goal Tests	New Nodes	Plan length	Time in sec
Breadth first search	14663	18098	129631	12	26,7796
Depth first graph search	408	409	3364	392	1,0852
Uniform cost search	18235	18237	159716	12	23,7633
A* planning graph	315	317	2902	12	146,732
A* ignore precondition	5040	5042	44944	12	6,9197
Bonus (domain dependent)	195	197	1883	12	0,3967

Optimal plan
Length 12

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

Analyses

Non-heuristic search (breadth-first, depth-first and uniform cost search)

Optimality

Uniform cost search will always find the best solution, and it did. Breadth-first search will also find the best solution for this problem, but only because every path cost is 1, because of the goal test is executed when a node is expanded.

Depth-first graph search is unusable in this kind of problem, it is more luck then everything else it found the goal. Planes will be flown back and forth for no reason.

Number of nodes (expansions, goal tests, and new nodes)

Breadth-first and uniform cost search searches a lot of nodes because it expands the node with the smallest path cost every time, it expands all levels of the tree until the solution is found. Uniformed cost search cost searches more nodes than breadth-first search because of the placement of the goal test.

Depth-first search does not expand that many nodes because it only expands nodes there will be in the final solution.

Time elapsed

Depth-first search can be fast because of it does not expand that many nodes.

Breadth-first and uniform cost search is both slow because it expands a lot of nodes. Breadth-first will be faster than uniform cost search because it expands fewer nodes (breadth-first search was actually slower then uniform cost search in problem 3).

Conclusion

Breadth-first search is a better choice then uniformed cost search when the path cost is 1, there is no reason to search a layer deeper, there won't be any nodes in the layer with a lower cost than the solution.

Heuristic search (A* planning graph, A* ignore precondition)

Optimality

Both heuristics found the optimal plan. As long as the heuristics underestimate the cost of finding the goal from a given node, the optimal solution will be found.

Time elapsed

The construction of a planning graph is a heavy process, there is a lot of things there has to be constructed. so finding the estimated cost takes time. I did profile the execution of the of the planning graph, and managed to shave 80% of the execution time of the heaviest function “contains_shared_item”.

Ignore precondition does only have to check how many of the goals are satisfied, and can be done fairly quickly, but the estimation is not near as good as the planning graph, so it has to expand many more nodes.

Number of nodes (expansions, goal tests, and new nodes)

The planning graph does a great job of estimating the path cost, with that good of a estimation, it does not have to expand many nodes.

The ignore precondition does not estimate the path cost that well and has to expand more nodes then the planning graph.

Conclusion

The 2 algorithms have their strength in different places.

If you have to found a solution in a memory limited system and time isn’t important, the planning graph is the way to go.

If you have to found a solution quickly, and memory isn’t the problem, ignore preconditions got you covered. If the expansion factor is high, it may not be the base because of the more nodes it has to search.

Best domain-independent heuristic

Overall, I think A* ignore precondition did the best job, it comes up with a usable heuristic fairly quickly. It expands a good amount of nodes, but not all of them like the none heuristics search algorithms. And it does it faster than all of the other optimal domain-independent heuristics.

Bonus - domain-dependent heuristic

I created an extra heuristic, it works the following way: if cargo is on a plane, it can be flown directly to the destination and has the cost of 1, it the cargo is in an airport, the cost will be 2 if there is a plane in the

airport (pick up, and fly to destination), if not, the cost will be 3 because a plane has to fly to the airport, pick it up, and fly to the destination. 0 if cargo is at goal.

This heuristic creates a good estimation, without the construction of a planning graph. It expanded fewest nodes, it did it quickest.