Search Algorithms Evaluation

Alex Staravoitau

alex.staravoitau@gmail.com

Air Cargo Problem 1

Evaluation

Search algorithm	Node expansions	Goal tests	New nodes	Elapsed (s)	Solution length
breadth_first_search	43	56	180	0.045	**6 (Optimal)**
depth_first_graph_search	12	**13**	**48**	**0.01**	12
uniform_cost_search	55	57	224	0.042	**6 (Optimal)**
A* using h_ignore_preconditions	41	43	170	0.049	**6 (Optimal)**
A* using h_pg_levelsum	**11**	**13**	50	1.363	**6 (Optimal)**

Table 1. Air Cargo Problem 1 evaluation.

Optimal solution

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

Air Cargo Problem 2

Evaluation

Search algorithm	Node expansions	Goal tests	New nodes	Elapsed (s)	Solution length
breadth_first_search	3346	4612	30534	15.768	**9 (Optimal)**
depth_first_graph_search	1124	1125	10017	**9.53**	1085
uniform_cost_search	4853	4855	44041	49.326	**9 (Optimal)**
A* using h_ignore_preconditions	1506	1508	13820	15.567	**9 (Optimal)**
A* using h_pg_levelsum	**86**	**88**	**841**	134.144	**9 (Optimal)**

Table 2. Air Cargo Problem 2 evaluation.

Optimal solution

```
Load(C3, P3, ATL)
Fly(P3, ATL, SFO)
Unload(C3, P3, SFO)
Load(C2, P2, JFK)
Fly(P2, JFK, SFO)
Unload(C2, P2, SFO)
Load(C1, P1, SFO)
Fly(P1, SFO, JFK)
Unload(C1, P1, JFK)
```

Air Cargo Problem 3

Evaluation

Search algorithm	Node expansions	Goal tests	New nodes	Elapsed (s)	Solution length
breadth_first_search	14120	17673	124926	113.24	**12 (Optimal)**
depth_first_graph_search	677	678	5608	**3.988**	660
uniform_cost_search	18223	18225	159618	432.26	**12 (Optimal)**
A* using h_ignore_preconditions	5118	5120	45650	102.342	**12 (Optimal)**
A* using h_pg_levelsum	**403**	**405**	**3708**	969.63	**12 (Optimal)**

Table 3. Air Cargo Problem 3 evaluation.

Optimal solution

```
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)
```

Summary

Having evaluated **5 search algorithms** (3 uninformed and 2 using A* search with different heuristics) on 3 problems of varying complexity we can see emerging patterns of each of the algorithm performance (Tables 1, 2 and 3).

Depth first search is the only one that didn't find an optimal solution, which is not surprising as it simply traverses the tree in the depth-first order, returning the first solution found. This is also why it was the first to return a solution on all three problems. As **depth first search failed to find an optimal solution**, we will not consider it in our further evaluations.

When it comes to **uninformed vs informed search algorithms**, informed ones seem to perform better or at least just as good on all three problems. Informed search algorithms were represented by A* search using two different heuristics, and their performance is either on par or better on all three problems than any uninformed algorithms evaluated here — both when it comes to speed and exploration costs (depending on what you're after).

Although informed search algorithms perform better overall, their performance highly depends on the **choice of heuristic function**, as Stuart Russel and Peter Norvig highlight in their *Artificial Intelligence: A Modern Approach* textbook. We can see that here as well, as A* search performance varies significantly depending on one of the two heuristics we use: if we use heuristic ignoring preconditions we find an optimal solution about one order of magnitude faster than with the one calculating levels costs. However `levelsum` heuristic requires roughly an order of magnitude less in terms of exploration costs: number of node expansions, goal tests and nodes additions.

To summarise, informed search algorithms will perform better than uninformed ones with a carefully chosen heuristic. The choice of heuristic however highly depends on the problem domain and required **speed/memory tradeoff**: if graph exploration costs are relatively high, it would make sense to use a `levelsum` heuristic; and if speed is way more of a factor, then a heuristic which is ignoring preconditions will be a much better fit.