Project Review: Planning

Lee Honan, March 2018

This document reviews the methods used to solve the project's three planning problems in terms of:

- the performance and behaviour of uninformed, non-heuristic search methods;
- the performance and behaviour of heuristic search methods;
- relative differences between several of these methods;
- the optimal method for these problems (with a recommendation).

Optimal Plans

The breadth-first search was used to find optimal plans. These are given in the table below.

Problem 1	**Problem 2**	**Problem 3**
Length = 6	Length = 9	Length = 12
Load(C2, P2, JFK) Load(C1, P1, SFO) Fly(P2, JFK, SFO) Unload(C2, P2, SFO) Fly(P1, SFO, JFK) Unload(C1, P1, JFK)	Load(C2, P2, JFK) Load(C1, P1, SFO) Load(C3, P3, ATL) Fly(P2, JFK, SFO) Unload(C2, P2, SFO) Fly(P1, SFO, JFK) Unload(C1, P1, JFK) Fly(P3, ATL, SFO) Unload(C3, P3, SFO)	Load(C2, P2, JFK) Load(C1, P1, SFO) Fly(P2, JFK, ORD) Load(C4, P2, ORD) Fly(P1, SFO, ATL) Load(C3, P1, ATL) Fly(P1, ATL, JFK) Unload(C1, P1, JFK) Unload(C3, P1, JFK) Fly(P2, ORD, SFO) Unload(C2, P2, SFO) Unload(C4, P2, SFO)

Search Metric Results

Running the various search methods against the three problems gave the results shown in the table below. Note that methods 2, 4, and 6 were too costly to run against problems 2 and 3 (each taking over 10 minutes on a powerful workstation).

	Problem 1					Problem 2					Problem 3				
	Expansions	Goal Tests	New Nodes	Plan Length	Time (s)	Expansions	Goal Tests	New Nodes	Plan Length	Time (s)	Expansions	Goal Tests	New Nodes	Plan Length	Time (s)
1. breadth_first_search	43	56	180	6	0.03	3343	4609	30509	9	9.37	14663	18098	129631	12	50.03
2. breadth_first_tree_search	1458	1459	5960	6	1.01										
3. depth_first_graph_search	12	13	48	12	0.01	582	583	5211	575	3.34	627	628	5176	596	3.51
4. depth_limited_search	101	271	414	50	0.10										
5. uniform_cost_search	55	57	224	6	0.05	4853	4855	44041	9	13.14	18223	18225	159618	12	60.98
6. recursive_best_first_search h_1	4229	4230	17029	6	2.94										
7. greedy_best_first_graph_search h_1	7	9	28	6	0.01	998	1000	8982	21	2.71	5578	5580	49150	22	18.95
8. astar_search h_1	55	57	224	6	0.05	4853	4855	44041	9	14.06	18223	18225	159618	12	62.40
9. astar_search h_ignore_preconditions	41	43	170	6	0.04	1450	1452	13303	9	5.05	5040	5042	44944	12	19.73
10. astar_search h_pg_levelsum	32	34	138	6	1.20	1587	1589	14844	9	512.33	8532	8534	77857	12	4135.32

Analysis — Non-Heuristic Methods

Noteworthy observations for the performance and behaviour of three non-heuristic search methods are given below:

- Breadth-first search (1). As would be expected, BFS produced an optimal result, at the cost of a relatively high number of node expansions and evaluations— translating to one of the slower search methods. Given the high space complexity of BFS — $O(b^d)$, BFS may be suitable for smaller problem sets such as these but is likely to be a poor choice for larger and more complex planning problems.
- Depth-first search (3). This was the least optimal method applied to all three problems (although depth-limited was even worse for problem one). It produced convoluted (arguably nonsensical) plans, with other optimal plans (e.g. from UCS) being ~2% the length of those produced by DFS. As a DFS it was consistently fast (even with the 'bushier' graph of problem three), and expanded the fewest nodes as it found solutions that it considered to be satisfactory. That is, it had relatively low time and space complexity when compared with BFS and UCS.
- Uniform cost search (5). Like BFS, UCS produced an optimal result, but was even more expensive in terms of node expansion and evaluation. This is to be expected because where BFS will stop searching when it reaches a level of the tree where a goal is met, whereas UCS will continue searching for a less costly node that meets the goal. In the case of the implementation used the UCS cost determination is naive (simply adding 1 for each path step). As a result, to quote the AIMA text it did "more work by expanding nodes at depth *d* unnecessarily"[1]. One could see UCS performing better than BFS with a more effective cost evaluation.

Analysis — Heuristic Methods

Noteworthy observations for the performance and behaviour of three heuristic search methods are given below:

- A-star with ignore preconditions search (9). This method produced an optimal result for each problem, and had the best time and space complexity performance across the methods that consistently produced optimal results. For the more complex problems 2 and 3 it expanded roughly a third of the nodes that BFS did. Unlike UCS the use of the 'ignore preconditions' heuristic provides a more useful yet still fast to compute cost metric. That allowed the A* search algorithm to perform in line with its potential.
- A-star with level-sum (10). Like (9) this produced an optimal solution, and while not as efficient as (9), reasonable time and space complexity. However the level-sum heuristic was costly, growing exponentially worse with graph size (several hundred milliseconds per expansion for problem 3). That is to be expected when looking at the complexity of this heuristic, and the need to iterate through each level and s-node when calculating the level-sum.

[1] Russel, S. and Norvig, P. (2010). *Artificial Intelligence: A Modern Approach*, 3rd Edition: p85

Recommendation

Given the results seen, I make the following recommendations:

- A-star with ignore preconditions search (9) as the best informed search method and heuristic. This produced an optimal result with acceptable time and space complexity.
- For an uninformed search I would recommend breadth-first search (1), although I would consider uniform cost search (5) if a more discriminatory means of determining path cost could be found. Of course, the most effective means of determining path cost would be obtaining more information about the problem space — resulting in an *informed* search.