Udacity Artificial Intelligence Nanodegree

Project 2: Build A Forward-Planning Agent

Arka Ghosh

October 2024

Air Cargo Problem 1 ($ python run_search.py -p 1 -s 1 2 3 4 5 6 7 8 9 10 11)

__Table 2__ - Displays all possible combinations of search and heuristics

				GBFS	GBFS	GBFS	GBFS	A*	A*	A*	A*
	BFS	DFS	UCS	Unmet	LevelSum	MaxLevel	SetLevel	Unmet	LevelSum	MaxLevel	SetLevel
Actions	20	20	20	20	20	20	20	20	20	20	20
Expansions	43	21	60	7	6	6	6	50	28	43	33
Goal Tests	56	22	62	9	8	8	8	52	30	45	35
New Nodes	178	84	240	29	28	24	28	206	122	180	138
Plan Length	6	20	6	6	6	6	6	6	6	6	6
Nodes/Action	8.9	4.2	12	1.45	1.4	1.2	1.4	10.3	6.1	9	6.9
Time Elapsed (sec)	0.003472	0.00189	0.00552	0.000969	0.112904	0.0782241	0.3077867	0.005871	0.287134	0.291517	0.7829641
Time/Action	0.000174	0.00009485	0.00027	0.00004845	0.0056452	0.003911205	0.015389335	0.000294	0.0143567	0.01457585	0.0391482

Air Cargo Problem 2 ($ python run_search.py -p 2 -s 1 2 3 4 5 6 7 8 9 10 11)

__Table 2__ - Displays all possible combinations of search and heuristics

					GBFS	GBFS	GBFS	A*	A*	A*	A*
	BFS	DFS	UCS	GBFS Unmet	LevelSum	MaxLevel	SetLevel	Unmet	LevelSum	MaxLevel	SetLevel
Actions	72	72	72	72	72	72	72	72	72	72	72
Expansions	3343	624	5154	17	9	27	9	2467	357	2887	1037
Goal Tests	4609	625	5156	19	11	29	11	2469	359	2889	1039
New Nodes	30503	5602	46618	170	86	249	84	22522	3426	26594	9605
Plan Length	9	619	9	9	9	9	9	9	9	9	9
Nodes/Action	423.652	77.805555	647.472	2.36111111	1.19444444	3.458333333	1.16666666	312.805	47.583333	369.36111	133.4027
Time Elapsed (sec)	1.170615	1.7547914	2.003338	0.011123069	2.401258	3.862002104	7.760764065	1.336268	64.9680942	384.304662	701.5205
Time/Action	0.01625	0.0243721	0.02782	0.00015448	0.033350806	0.053638918	0.10778839	0.01855	0.9023346	5.3375647	9.743340

Air Cargo Problem 3 ($ python run_search.py -p 3 -s 1 2 3 4 5 6 7 8 9 10 11)

Table 3 *- Displays all possible combinations of search and heuristics [**timeout** means to much time is being taken to reach the output > 15*

					GBFS	GBFS	GBFS	A*	A*	A*	A*
	BFS	DFS	UCS	GBFS Unmet	LevelSum	MaxLevel	SetLevel	Unmet	LevelSum	MaxLevel	SetLevel
Actions	88	88	88	88	88	88	88	88	88		
Expansions	14663	408	25	14	9	21	35	7388	369		
Goal Tests	18098	409	27	16	11	23	37	7390	371		
New Nodes	129625	3364	230	126	86	195	345	65711	3403		
Plan Length	12	392	12	15	14	13	17	12	12		
Nodes/Action	1473.01	38.227272	2.61363	1.43181818	0.977272727	2.215909091	3.920454545	746.7159	38.6704545		
Time Elapsed (sec)	6.68710	0.7682746	0.02468	5.69744703	2.401258	5.631700975	42.7573238	5.06574	117.76242	timeout	timeout
Time/Action	0.07599	0.0087303	0.00028	0.06474371	0.027287023	0.063996602	0.48587868	0.05756	1.3382093		

Air Cargo Problem 4 ($ python run_search.py -p 4 -s 1 2 3 4 5 6 7 8 9 10 11)

Table 4 *- Displays all possible combinations of search and heuristics [**timeout** means to much time is being taken to reach the output > 15 minutes]*

					GBFS	GBFS	GBFS	A*	A*	A*	A*
	BFS	DFS	UCS	GBFS Unmet	LevelSum	MaxLevel	SetLevel	Unmet	LevelSum	MaxLevel	SetLevel
Actions	104			104	104			104	104		
Expansions	99736			29	17			34330	1208		
Goal Tests	114953			31	19			34332	1210		
New Nodes	944130			280	165			328509	15		
Plan Length	14			18	18			14	15		
Nodes/Action	9078.173			2.692307692	1.586538462			3158.74	0.14423077		
Time Elapsed (sec)	54.75429	timeout	timeout	0.04124	17.674	timeout	timeout	51.831	248.13	timeout	timeout
Time/Action	0.526484			0.000396538	0.169942308			0.498375	2.38586538		

Analysis

I am trying to find the optimal solution for each of the four problems, by analysing and comparing various methods/algorithms. The problems are arranged in an increasing order of complexity. An optimal solution is based on certain factors, considering and selecting trade-offs between space or time complexities.

For the **first** problem, considering the 'time per action' metric, uninformed DFS performs the best amongst all, whereas considering 'nodes expanded per action' metric, Greedy Best First with 'MaxLevel' heuristic outshines all other methods,
with nodes expanded per action ratio of just 1:2. But considering both the metrics, the most fulfilling method is Greedy Best First along with 'LevelSum' heuristic.

For the **second** problem, considering the 'time per action' metric, Greedy best First performed the best out of all techniques, whereas considering 'nodes expanded per action' metric, Greedy Best First with the 'LevelSum' heuristic was the obvious winner, with a ratio of just 1:194. Another interesting thing to note is that DFS performs very poorly, with the 'plan length' showing an increase of
6777.77%, compared to all other plan lengths.

For the **third** problem, Greedy Best First with 'Unmet' heuristic performed outstandingly well, having a super low value of 'time per action', being only 0.000414. It's performance measure for the other metric, 'nodes expanded per action', was also comparable to the most optimal one (Greedy Best First with LevelSum - 1.431 & GBF Unmet - 2.613)

For the **fourth** problem, GBF with 'Unmet' heuristic took the least amount of time (0.06 sec), and performed the best under the criteria 'time per action' - 0.00058. A* with LevelSum performed the best under 'nodes expanded per action' - 0.1442. Uninformed Search (BFS) performed the worst, having the most (and considerably high) number of expansions and new nodes created. Throughout all problems, the Greedy Best First approach, along with 'LevelSum'
The heuristic has been a constant achiever, giving optimal results with very low trade-offs. Also, generally, DFS gives a very large number of plan lengths, as compared to all other techniques.

As the complexities of problems increased, a gradual pattern emerged amongst the different approaches -

- The uninformed search techniques start giving non-optimal results, having a very high number of expansions and created nodes, which implies they have high space complexities. Whereas, the Greedy and A* techniques hold good for increasing complexities in terms on space complexities.

- Another pattern that has emerged is that the A* technique with 'SetLevel' heuristic has a very high time complexity, since the time taken to get to the solution increases drastically, concerning the increase in the problem's complexities. Such is the case with uninformed searches, where too, the time complexities are very high.

- Considering the 'Plan lengths' of the solutions, there is a general increase as complexity increases, but are comparable amongst all other techniques except DFS. DFS's plan lengths increase very drastically as the complexity increases.

Q&A

1. Which algorithm or algorithms would be most appropriate for planning in a very restricted domain (i.e., one that has only a few actions) and needs to operate in real-time?

Greedy Best First with ‘LevelSum’ heuristic is the best choice for such situations, since having a low ‘time taken to find a solution’ and ‘time per action’ imply that it works very fast, which is ideal for programs that function in real-time. Also, having a low ‘nodes expanded per action’ implies that it is space efficient.

2. Which algorithm or algorithms would be most appropriate for planning in very large domains (e.g., planning delivery routes for all UPS drivers in the U.S. on a given day)

As seen in problem 4, when the domain is large, Greedy Best First with the ‘Unmet’ heuristic, since it takes the least time to arrive at a solution and has the lowest time per action value. It also has a very small amount of expansions and new nodes.

3. Which algorithm or algorithms would be most appropriate for planning problems where it is important to find only optimal plans?

According to me, an A* algorithm with ‘Unmet’ heuristics must be used, since it consistently shows a low value of plan length throughout the problems with increasing complexities, and it also has a very low ‘time per action’ value.