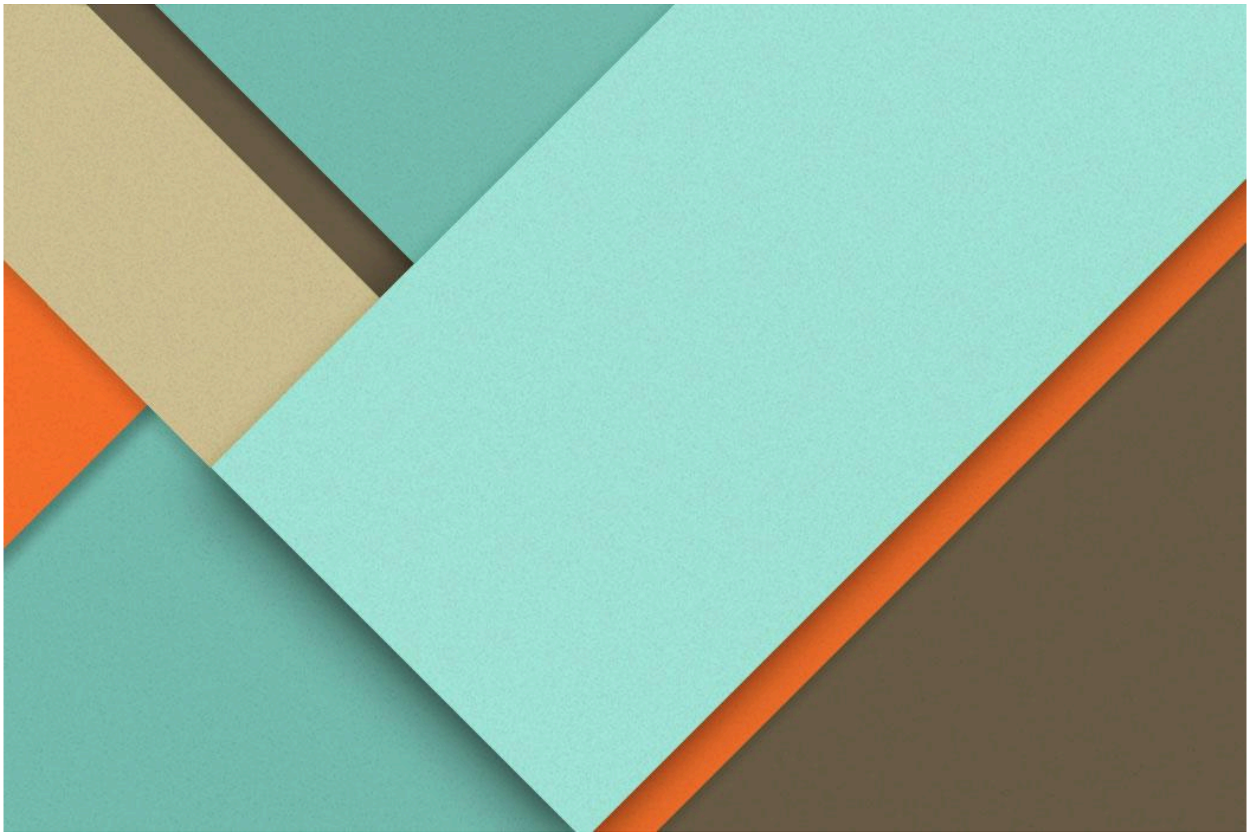

PROJECT 2: BUILD A FORWARD-PLANNING AGENT

July 2024

TUAN-ANH VO

EXPERIMENTS ANALYSIS

The project is to identify the optimal solution for each of the four cargo problems, arranged in increasing order of complexity, by analyzing and comparing various planning techniques below:

Breadth-First Search (BFS)
Depth-First Graph Search (DFS)
Uniform Cost Search (UCS)
Greedy Best-First Graph Search with h_unmet_goals
Greedy Best-First Graph Search with h_pg_levelsum
Greedy Best-First Graph Search with h_pg_maxlevel
Greedy Best-First Graph Search with h_pg_setlevel
A Search with h_unmet_goals*
A Search with h_pg_levelsum*
A Search with h_pg_maxlevel*
A Search with h_pg_setlevel*

An optimal solution is determined by evaluating specific factors and making trade-offs between space and time complexities.

I try to run as many techniques as possible per problem and deriving from the standard metrics the two important metrics “time per action” and “new nodes per action” to indicate the efficiency of each algorithm.

Air Cargo Problem 1:

	Actions	Expansions	Goal Tests	New Nodes	Time Elapse	Plan Length	Time per action	New nodes per action
Breadth-First Search (BFS)	20	43	56	178	0.00758041	6	0.0003790	2.15
Depth-First Graph Search (DFS)	20	21	22	84	0.00431486	6	0.0002157	1.05
Uniform Cost Search (UCS)	20	60	62	240	0.00976041	6	0.0004880	3
Greedy Best-First Graph Search with h_unmet_goals	20	7	9	29	0.00194746	6	0.0000974	0.35
Greedy Best-First Graph Search with h_pg_levelsum	20	6	8	28	0.17927744	6	0.0089639	0.3
Greedy Best-First Graph Search with h_pg_maxlevel	20	6	8	24	0.12834727	6	0.0064174	0.3
Greedy Best-First Graph Search with h_pg_setlevel	20	6	8	28	0.59701787	6	0.0298509	0.3
A* Search with h_unmet_goals	20	50	52	206	0.00972535	6	0.0004863	2.5
A* Search with h_pg_levelsum	20	28	30	122	0.53735519	6	0.0268678	1.4
A* Search with h_pg_maxlevel	20	43	45	180	0.53139829	6	0.0265699	2.15
A* Search with h_pg_setlevel	20	33	35	138	1.26345363	6	0.0631727	1.65

For the first problem, I run all the 11 algorithms:

+ For ‘time per action’ metric, the Greedy Search with unmet goal heuristic is optimal with smallest amount.
+ For ‘new node per action’ metric, all the Greedy Search with different heuristic outperform the other with ratio of or around 0.3

+ So considering both the metrics, the most fulfilling method is the Greedy ones

Air Cargo Problem 2:

	Actions	Expansions	Goal Tests	New Nodes	Time Elapse	Plan Length	Time per action	New nodes per action
Breadth-First Search (BFS)	72	3343	4609	30503	1.891848615	9	0.02627568	423.652778
Depth-First Graph Search (DFS)	72	624	625	5602	2.807411966	9	0.03899183	77.8055556
Uniform Cost Search (UCS)	72	5154	5156	46618	3.135873764	9	0.0435538	647.472222
Greedy Best-First Graph Search with h_unmet_goals	72	17	19	170	0.017081988	9	0.00023725	2.36111111
Greedy Best-First Graph Search with h_pg_levelsum	72	9	11	86	3.89766888	9	0.05413429	1.19444444
Greedy Best-First Graph Search with h_pg_maxlevel	72	27	29	249	6.206077794	9	0.08619552	3.45833333
Greedy Best-First Graph Search with h_pg_setlevel	72	9	11	84	13.73709424	9	0.19079298	1.16666667
A* Search with h_unmet_goals	72	2467	2469	22522	2.101877665	9	0.02919275	312.805556

For the second problem,

+ Considering the ‘time per action’ metric, again the Greedy best First Search with unmet goal heuristic performed the best out of all techniques.
+ Considering ‘new nodes per action’ metric, Greedy also substantially smaller than other techniques

+ One thing to notice is the BFS and DFS perform rather well in term of time but the new nodes just inflating so badly in this second problem.

Air Cargo Problem 3:

	Actions	Expansions	Goal Tests	New Nodes	Time elapsed	Plan length	Time per action	New nodes per action
Breadth-First Search (BFS)	88	14663	18098	129625	10.12463632	12	0.11505269	1473.01136
Depth-First Graph Search (DFS)	88	408	409	3364	1.078231935	392	0.01225264	38.2272727
Uniform Cost Search (UCS)	88	18510	18512	161936	13.8122448	12	0.15695733	1840.18182
Greedy Best-First Graph Search with h_unmet_goals	88	25	27	230	0.034212427	15	0.00038878	2.61363636
Greedy Best-First Graph Search with h_pg_levelsum	88	14	16	126	8.924549528	14	0.10141534	1.43181818
Greedy Best-First Graph Search with h_pg_maxlevel	88	21	23	195	9.192624212	13	0.10446164	2.21590909
Greedy Best-First Graph Search with h_pg_setlevel	88	35	37	345	72.93299988	17	0.82878409	3.92045455
A* Search with h_unmet_goals	88	7388	7390	65711	8.217416387	12	0.09337973	746.715909
A* Search with h_pg_levelsum	88	369	371	3403	194.6306489	12	2.21171192	38.6704545

For the third problem,

+ Greedy Best First with ’Unmet’ heuristic again performed outstandingly well in term of time,
+ In term of node, the Greedy Best first with Level Sum is the best here

+ Notice that apart from the usual greedy family, the DFS also perform very well in the third problem

Air Cargo Problem 4:

	Actions	Expansions	Goal Tests	New Nodes	Time Elapse	Plan Length	Time per action	New nodes per action
Breadth-First Search (BFS)	104	99736	114953	944130	99.09563400	14	0.952842635	9078.17308
Greedy Best-First Graph Search with h_unmet_goals	104	29	31	280	0.06128450	18	0.000589274	2.69230769
Greedy Best-First Graph Search with h_pg_levelsum	104	17	19	165	16.34912580	17	0.157203133	1.58653846
A* Search with h_unmet_goals	104	34330	34332	328509	56.38125679	14	0.542127469	3158.74038
A* Search with h_pg_levelsum	104	1208	1210	15	254.63213214	15	2.448385886	0.14423077

Table showing one uninformed search (BFS), two heuristics with greedy best first search (Unmet and LevelSum), and two A heuristics (Unmet, LevelSum)*

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'
+ A* with LevelSum performed the best under 'new nodes per action'
+ Uninformed Search (BFS) performed the worst in the complex problem, both in term of time and nodes expanding.

CONCLUSION & OBSERVATION

+ Generally speaking, the GBF family outperform all other techniques consistently in problems disregarding complexity, with the variant of unmet goal heuristic arguably the best one.
+ BFS and A* with unmet goal have the tendency to inflate the new nodes extensively (general A* techniques perform rather well on this metric except this one)
+ DFS has a more plan lengths compared to all other techniques.
+ As the complexity increases, The uninformed search techniques start giving non optimal results with high number of expansions and created nodes, which implies they have high space complexities.

+ Whereas, the Greedy and A* techniques in contrast significantly in term of space complexity, stay consistent with small quantity of nodes expansion through 4 problems.

+ In term of time complexity , the uninformed techniques along with the outlier A* technique with 'SetLevel' heuristic perform rather poorly also as problems get more complex

QUESTIONS

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?

With emphasis on efficiency in real time, the GBF with 'LevelSum' heuristic is the best choice with very low goal test, and time to find solution as well as very low expansion as manifested in the problem one

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)

**A* Search with a well-chosen heuristic Generally the most robust choice for very large domains due to its balance of exploration and exploitation, ensuring optimal solutions.

GBF with it efficiency is a good alternative but since it doesn't work on the whole data, it could pose some problems when the data becomes massive

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

A* search is the best finding the optimal solution when the heuristic used is admissible (never overestimates the true cost) and consistent (satisfies the triangle inequality).It combines the cost to reach the current node and the estimated cost to reach the goal, providing a balance between exploring promising paths and ensuring optimality.