Forward Planning Agent Search Heuristic Analysis

By Dharmanandana Reddy

I have implemented a classical planning search agent to solve planning search problems for air cargo system. I used uninformed (breadth-first, depth-first etc.,) and heuristic based search methods (A* search etc.,)

the goal of this analysis is to find optimal solution for each air cargo problem from different search techniques. search algorithm to find the lowest path and fastest among all possible paths from start to goal.

We compare the performance of eleven strategies in terms of execution time, memory usage (search node expansions) and optimal (plan length). Number of goal tests and number of new nodes are not considered as they don't change the results of our analysis

Performance measures are collected using following commands:

```
python run_search.py -p 1 -s 1 2 3 4 5 6 7 8 9 10 11 > run_search_results_p1.txt
python run_search.py -p 2 -s 1 2 3 4 5 6 7 8 9 10 11 > run_search_results_p2.txt
python run_search.py -p 3 -s 1 4 5 6 7 8 9 > run_search_results_p3.txt
```

Problem1 results: (Actions 20)

Search Technique	Actions	Expansions	Goal Tests	New nodes	Plan length	Time	Optimal
breadth_first_search	20	43	56	178	6	0.0062	Yes
depth_first_graph_search	20	21	22	84	20	0.0033	No
uniform_cost_search	20	60	62	240	6	0.0096	Yes
greedy_best_first_graph_search with h_unmet_goals	20	7	9	29	6	0.0015	Yes
greedy_best_first_graph_search with h_pg_levelsum	20	**6**	8	28	**6**	0.47	Yes
greedy_best_first_graph_search with h_pg_maxlevel	20	**6**	8	24	**6**	0.35	Yes
greedy_best_first_graph_search with h_pg_setlevel	20	**6**	8	28	**6**	0.65	Yes
astar_search with h_unmet_goals	20	50	52	206	6	0.0095	Yes
astar_search with h_pg_levelsum	20	28	30	122	6	1.2	Yes
astar_search with h_pg_maxlevel	20	43	45	180	6	1.24	Yes
astar_search with h_pg_setlevel	20	33	35	138	6	1.522	Yes

Problem2 results: (Actions 72)

Search Technique	Actions	Expansions	Goal Tests	New nodes	Plan length	Time	Optimal
breadth_first_search	72	3343	4609	30503	9	2.1	Yes
depth_first_graph_search	72	624	625	5602	619	3.16	No
uniform_cost_search	72	5154	5156	46618	9	3.58	Yes
greedy_best_first_graph_search with h_unmet_goals	72	17	19	170	9	0.02	Yes
greedy_best_first_graph_search with h_pg_levelsum	72	**9**	11	86	**9**	10.78	Yes
greedy_best_first_graph_search with h_pg_maxlevel	72	27	29	249	9	21.75	Yes
greedy_best_first_graph_search with h_pg_setlevel	72	9	11	84	**9**	15.9	Yes
astar_search with h_unmet_goals	72	2467	2469	22522	9	2.36	Yes
astar_search with h_pg_levelsum	72	357	359	3426	9	272.2	Yes
astar_search with h_pg_maxlevel	72	2887	2889	26594	9	1566	Yes
astar_search with h_pg_setlevel	72	1037	1039	9605	9	1439.8	Yes

I did not collect data for 'uniform_cost_search', 'depth_first_graph_search', 'astar_search with h_pg_maxlevel' and 'astar_search with h_pg_setlevel' for problem 3, they are excluded due to the reason is that the higher plan length, longer execution time(more than 10 mins) and higher memory consumption

Problem3 results: (Actions 88)

Search Technique	Actions	Expansions	Goal Tests	New nodes	Plan length	Time	Optimal
breadth_first_search	88	14663	18098	129625	12	10.73	Yes
greedy_best_first_graph_search with h_unmet_goals	88	25	27	230	15	0.036	No
greedy_best_first_graph_search with h_pg_levelsum	88	14	16	126	14	24.03	No
greedy_best_first_graph_search with h_pg_maxlevel	88	21	23	195	13	29.47	No
greedy_best_first_graph_search with h_pg_setlevel	88	35	37	345	17	87.53	No
astar_search with h_unmet_goals	88	7388	7390	65711	12	8.612	Yes
astar_search with h_pg_levelsum	88	369	371	3403	12	431.9	Yes

For problem 3, ‘astar_search with h_unmet_goals’ worked better than ‘breadth_first_search’ as ‘astar_search with h_unmet_goals’ took lesser memory space, execution time though plan length is same comparing to breadth_first_search.

Informed and Uninformed search strategies analysis conclusion and summary:

Breadth first search always considers the shortest path first and yields the results in an optimal way within reasonable time, which is recommended search strategy in handling simple problems and we don't have to use A* search for simple problems as A* increase solution complexity.

Considering nodes expansion for the search i.e. search complexity, greedy best first search consistently performed better than other algorithms even as the problem domain size is increased where number of nodes expansion and number of actions i.e. path length are lesser. The uniform cost search expands the more nodes followed by breadth first search.

Considering search time i.e. execution speed, the execution time is very close for different algorithms with uninformed searches doing better in simple problem domain area. The greedy best first searches better with increasing actions, the uninformed search execution time tend to worsen as the actions increases and the A* star algorithms are the slowest except for astar_search_h_unmet_goals.

Heuristic based search such as A* did performed better and handles well in case of problem complexity increases. This is more evident in problem 3 where 'astar_search with h_unmet_goals' performance shown optimal and fastest and lesser memory space (expansion nodes). And also worth note that 'astar_search with h_pg_levelsum' heuristic did not do so well for problem 3 as it took more execution time ofcourse number of expansion nodes lesser than 'h_unmet_goals', is it because of heuristic being too complex, may be.

According to the results, breadth first search strategy can solve simple planning problems in efficient, optimal and fastest way, so this is good candidate. As problem complexity increase, it may be worth considering heuristic based search (astar_search with h_unmet_goals) to deal with complex problems.

Questions: (Answering following questions based on above results)

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?

   Execution time is critical here to find the goal, so we need to choose the algorithm that runs fast. Given few actions and execution time criteria, Breadth first search would be the one appropriate to find optimal plan where planning in a very restricted domain.

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)

   Here the requirement is to find goal in very large domains with time consideration, so given this, A* star algorithm would be appropriate choice to find a goal with good heuristic and fewer node expansion.

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

   Breadth first search and uniform cost search are the choices to find optimal plans, but considering speed and node expansion, Breadth first search would be the better choice, which is more efficient.

Optimal Actions Sequence

The following table describes an optimal sequence of actions to solve air cargo problems:

Problem	Search Type	Optimal Sequence of actions
Air cargo problem1	breadth_first_search	Load(C1, P1, SFO) Load(C2, P2, JFK) Fly(P2, JFK, SFO) Unload(C2, P2, SFO) Fly(P1, SFO, JFK) Unload(C1, P1, JFK)
Air cargo problem2	breadth_first_search	Load(C1, P1, SFO) Load(C2, P2, JFK) 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)

Air cargo problem3	astar_search with h_unmet_goals	Load(C2, P2, JFK) Fly(P2, JFK, ATL) Load(C3, P2, ATL) Fly(P2, ATL, ORD) Load(C4, P2, ORD) Fly(P2, ORD, SFO) Unload(C4, P2, SFO) Unload(C2, P2, SFO) Load(C1, P2, SFO) Fly(P2, SFO, JFK) Unload(C3, P2, JFK) Unload(C1, P2, JFK)