Forward Planning Agent

Michal Janczyk

May 10, 2020

1 Introduction

1.1 Abstract

Researchers in the field of Artificial Intelligence has made significant progress in the past several years. When we concentrate on the complicated nature of "Forward Planning Agent" as one of a major field of studies has grown strongly in terms of the size and the difficulty of problems that can be solved. In this report, I give an overview of the techniques that have been part of Project 2. I will be concentrating on the individual planning system, which partially was developed during this project by reviewing the underlying principles behind many of the successful planners, algorithms. Most of the overview is dedicated to combining symbolic logic and classical search to implement an agent which shows progression search to solve planning problems

2 Research Design

In Computing Science, formal methodologies are frequently used to prove facts about algorithms and systems. Today we will be focusing on the time or space complexity of an algorithm, or the correctness or the quality of the solutions generated by the algorithm. These experimental methodologies are broadly used in CS to evaluate new solutions for problems.

2.1 Space complexity

In Basic terms, space complexity is the amount of computer memory required to implement the algorithm when input values are of a specified size Analyze the search complexity as a function of domain size, search algorithm, and heuristic. An analysis of the computer memory required involves

the space complexity of the algorithm. There are three types of time complexity — Best, average, and worst case, But today we will focus on the overall complexity of the problem. The first task of this project is to visualize data for all search and heuristic combinations for air cargo problems 1 and 2. Results should include data at least one uninformed search, two heuristics with greedy best-first search, and two heuristics with A* on air cargo problems 3 and 4.

2.2 Time complexity

One measure of efficiency is the time used by a computer to solve a problem using the algorithm when input values are of a specified size. An analysis of the time required to solve the problem of a particular size involves the time complexity of the algorithm.We can observe a similar situation as an example above, in simple words for an algorithm, if we could perform and get what we want in just one computational approach, then that is said as O(1). Time complexity here comes into the 'Best case' category. The second task of this project is to visualize data for all search and heuristic combinations for air cargo problems 1 and 2. Results should include data at least one uninformed search, two heuristics with greedy best first search, and two heuristics with A* on air cargo problems 3 and 4

3 Analysis

Algorithm analysis is a way of determining the resources that are needed to execute an algorithm: the amount of time and space in memory, bandwidth, or the number of logic circuits.In algorithm analysis, the algorithm's running time plays an important role, because some simple problems can cause unusually long calculations. In this analysis, we will consider three cases of the longest-running time for each input of a given size and a case of average waiting time for the operation of a given algorithm, assuming that all input data of a given size are equally probable.
All calculations were done on my private workstation. All the tables below above for 4 problems present the numbers of nodes that ware expanded during the search process for each of the search functions.

Search functions	Actions	Expansions	Goal Tests	New Nodes	Plan length	Time elapsed
breadth_first_search	40	43	56	176	6	0.003194629
depth_first_graph_search	20	21	22	84	20	0.001534183
uniform_cost_search	20	60	62	240	6	0.004477719
greedy_best_first_graph_search h_unmet_goals	20	7	9	29	6	0.000780218
greedy_best_first_graph_search h_pg_levelsum	20	6	8	28	6	0.132784855
greedy_best_first_graph_search h_pg_maxlevel	20	6	8	24	6	0.105378566
greedy_best_first_graph_search h_pg_setlevel	20	6	8	28	6	0.380735642
astar_search h_unmet_goals	20	50	52	206	6	0.006181065
astar_search h_pg_levelsum	20	28	30	122	6	0.311943397
astar_search h_pg_maxlevel	20	43	45	180	6	0.336274646
astar_search h_pg_setlevel	20	33	35	138	6	0.820930595

Table 1: Air Cargo Problem 1

Search functions	Actions	Expansions	Goal Tests	New Nodes	Plan length	Time elapsed
breadth_first_search	72	3343	4609	30503	9	0.970018086
depth_first_graph_search	72	624	625	5602	619	1.423031775
uniform_cost_search	72	5154	5156	46618	9	1.587672668
greedy_best_first_graph_search h_unmet_goals	72	17	19	170	9	0.008895331
greedy_best_first_graph_search h_pg_levelsum	72	9	11	86	9	2.697451575
greedy_best_first_graph_search h_pg_maxlevel	72	27	29	249	9	4.922207431
greedy_best_first_graph_search h_pg_setlevel	72	9	11	84	9	12.0020902
astar_search h_unmet_goals	72	2467	2469	22522	9	1.064249373
astar_search h_pg_levelsum	72	357	359	3426	9	87.33677683
astar_search h_pg_maxlevel	72	2887	2889	26594	9	510.2079132
astar_search h_pg_setlevel	72	1037	1039	9605	9	1321.744459

Table 2: Air Cargo Problem 2

Search functions	Actions	Expansions	Goal Tests	New Nodes	Plan length	Time elapsed
breadth_first_search	88	14663	18098	129625	12	6.57455117
depth_first_graph_search	88	408	409	3364	392	0.615493155
uniform_cost_search	88	18510	18512	161936	12	11.82547527
greedy_best_first_graph_search h_unmet_goals	88	25	27	230	15	0.023863486
greedy_best_first_graph_search h_pg_levelsum	88	14	16	126	14	7.241973437
greedy_best_first_graph_search h_pg_maxlevel	88	21	23	195	13	8.483266251
greedy_best_first_graph_search h_pg_setlevel	88	35	37	345	17	116.5050212
astar_search h_unmet_goals	88	7388	7390	65711	12	5.701459599
astar_search h_pg_levelsum	88	369	371	3403	12	156.7433414
astar_search h_pg_maxlevel	88	9580	9582	86312	12	2617.825208
astar_search h_pg_setlevel	88	3423	3425	31596	12	6902.38818

Table 3: Air Cargo Problem 3

Search functions	Actions	Expansions	Goal Tests	New Nodes	Plan length	Time elapsed
breadth_first_search	104	99736	114953	944130	14	52.4114437
depth_first_graph_search	104	25174	25175	228849	24132	2262.485053
uniform_cost_search	104	113339	113341	1066413	14	57.29364328
greedy_best_first_graph_search h_unmet_goals	104	29	31	280	18	0.035541602
greedy_best_first_graph_search h_pg_levelsum	104	17	19	165	17	12.73226341
greedy_best_first_graph_search h_pg_maxlevel	104	56	58	580	17	26.86883943
greedy_best_first_graph_search h_pg_setlevel	104	107	109	1164	23	485.7718317
astar_search h_unmet_goals	104	34330	34332	328509	14	33.33765512
astar_search h_pg_levelsum	104	1208	1210	12210	15	1008.955483
astar_search h_pg_maxlevel	104	62077	62079	599376	14	31422.47951
astar_search h_pg_setlevel	104	22606	22608	224229	14	86117.3038

Table 4: Air Cargo Problem 4

3.1 Analysis of the search complexity as a function of domain size, search algorithm, and heuristic

The problem gathers in Table 1 shows very precisely how for some search functions the number of expansions increases with a constant number of performed actions. We can observe that the smallest recorded expansion value is 6 for one greedy algorithm differing only in types of heuristics searching. The highest value of 60 was recorded for the uniform_cost_search algorithm.
Similar results were included in the second problem, which is shown in Table 2. The smallest values are noticeably found for delayed algorithms and their search functions. The range of expansion values ranges from 9 to a maximum of 27. Again the highest value of 5154 was recorded for the uniform_cost_search algorithm.
The situation is also repeated for the two main problems collected in Tables 3 and 4.
Based on the results presented, it can be concluded that the greedy search algorithms are characterized by the smallest increase in complexity when compared to other search algorithms. If we take a closer look at the results obtained for the A* search family, we can observe with the search heuristics h_pg_levelsum it performed very properly in terms of the complexity of the search.
If we consider algorithms that did worse in finding the right path. We may conclude that A* search with the h_pg_maxlevel heuristic worked very badly in terms of search complexity. The number of expansion exceeds 62000 nodes. Of all the considered algorithms, uniform_cost_search fared the worst. An algorithm that did not work well in terms of the complexity of the search was also depth_first_graph_search. The first search performed poorly in terms of the complexity of the search. The length of the plan that was observed during problem 4 also shows that the algorithm did not meet the basic requirements.

3.2 Analysis of search time as a function of domain size, search algorithm, and heuristic

The search time execution time varies considerably certainly across the different searches as the problem space expanded. We can observe the largest differences in the performance of algorithms for problem 4. The calculated time in seconds.
Here again, as in the case of the previous complexity of space, we can notice a significant advantage of greedy algorithms for which the goal was achieved.
On the other hand, the search for h_pg_maxlevel and h_pg_setlevel found the optimal solution, while the time spent on the process of finding a proper route was far too high. The total time for heuristic's h_pg_setlevel oscillates around 24 hours. Good example is depth_first_graph_search also where we

observe performance degraded significantly as the search space expanded.

3.3 Analysis of the optimal of solution as a function of domain size, search algorithm, and heuristic

If we try to find the optimal solution as a function of domain size, search algorithm, and heuristics, we must use the term - Admissibility. The only family of the algorithms presented that uses this property is A*. A search algorithm is said to be admissible if it is guaranteed to return an optimal solution. Assuming that the heuristics used in the algorithm is acceptable, we can take into account, and A is acceptable. The options are that he will always give us a solution to the solution if any. What's more, this algorithm in operation searches fewer nodes than any other acceptable search algorithm with the same heuristics. This is because A* creates an "optimistic" estimate of the cost of paths by each node that it takes into account - optimistic in the sense that the true cost of the path through a given node to the node must be as small as possible as the rating. We may also notice that the greedy algorithms performed very well in finding the most optimal paths to goals.

4 Conclusion

- 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?

Based on the conducted experience, we can create that the algorithms that would be most suitable for planning in a very limited domain and which would have to work in real-time belong to the family of greedy algorithms.

- 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)

Analyzing the results obtained, we can see that the algorithms that would be the most suitable for planning in very large domains are included in the A* family. Additionally using a very large domain we would like to have an opportunity to obtain the most optimal search paths. The exception can be only heuristics h_pg_setlevel for which the search times for complex problems are quite high.

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

By interpreting the obtained results, the algorithms that would be the most suitable for planning problems, in which it is important to find only optimal plans, should be the algorithms from the A* family. When A* finishes searching, it has a path found whose cost is less than the estimated cost of any other path (another, i.e., passing through at least one node not included in the path found). Because the estimates are optimistic, the A* algorithm can safely ignore these other paths. In other words, A* will never overlook a lower-cost path and is therefore acceptable.