Artificial Intelligence Nanodegree Program

Project#2: Build a Forward-Planning Agent

March 11th, 2022

(1) Planning Graph Implementation

➔ **my_planning_graph.py**

(2) Heuristic Implementation

➔ **my_planning_graph.py**

(3) Experimental Results & Report

(a) Analyze the results as a function of domain size, search algorithm, and heuristic.

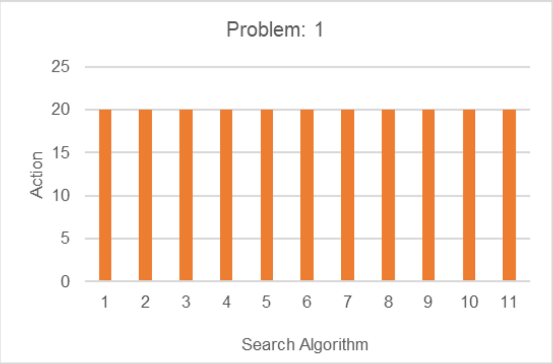
The number of Search Algorithm is defined as below:

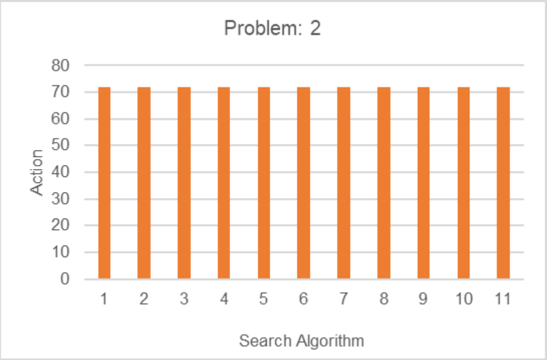
1 [["breadth_first_search", breadth_first_search, ""],
2 ['depth_first_graph_search', depth_first_graph_search, ""],
3 ['uniform_cost_search', uniform_cost_search, ""],
4 ['greedy_best_first_graph_search',greedy_best_first_graph_search, 'h_unmet_goals'],
5 ['greedy_best_first_graph_search',greedy_best_first_graph_search, 'h_pg_levelsum'],
6 ['greedy_best_first_graph_search',greedy_best_first_graph_search, 'h_pg_maxlevel'],
7 ['greedy_best_first_graph_search',greedy_best_first_graph_search, 'h_pg_setlevel'],
8 ['astar_search', astar_search, 'h_unmet_goals'],
9 ['astar_search', astar_search, 'h_pg_levelsum'],
10 ['astar_search', astar_search, 'h_pg_maxlevel'],
11 ['astar_search', astar_search, 'h_pg_setlevel']

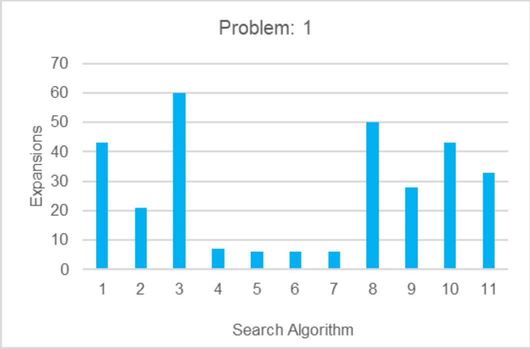
The search complexity, search time and the optimality as a function of domain size, search algorithm, and heuristic are shown below.

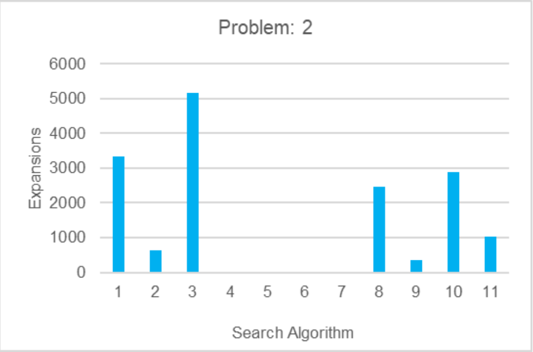
At first, the chart and table includes data for all search & heuristic combinations for air cargo problems 1 and 2 are shown in the following table and graphs.

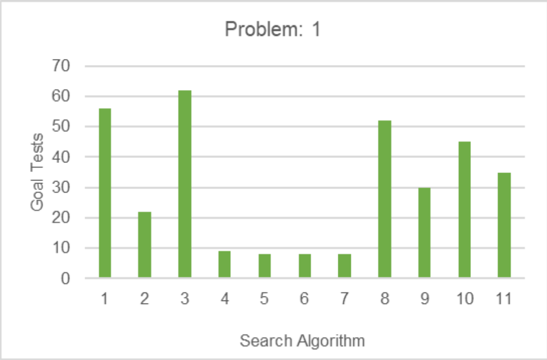
problem:	search:	Actions	Expansions	Goal Tests	New Nodes	Time[sec]
1	1	20	43	56	178	0.0019821
1	2	20	21	22	84	0.0014562
1	3	20	60	62	240	0.0031644
1	4	20	7	9	29	0.000539
1	5	20	6	8	28	0.067607
1	6	20	6	8	24	0.045845
1	7	20	6	8	28	0.1876847
1	8	20	50	52	206	0.0041256
1	9	20	28	30	122	0.1641107
1	10	20	43	45	180	0.1729627
1	11	20	33	35	138	0.4482229
2	1	72	3343	4609	30503	0.7303307
2	2	72	624	625	5602	0.891381
2	3	72	5154	5156	46618	1.0463211
2	4	72	17	19	170	0.0064692
2	5	72	9	11	86	1.3747275
2	6	72	27	29	249	2.0938031
2	7	72	9	11	84	5.8654908
2	8	72	2467	2469	22522	0.9359811
2	9	72	357	359	3426	45.6525362
2	10	72	2887	2889	26594	275.793292
2	11	72	1037	1039	9605	448.818569

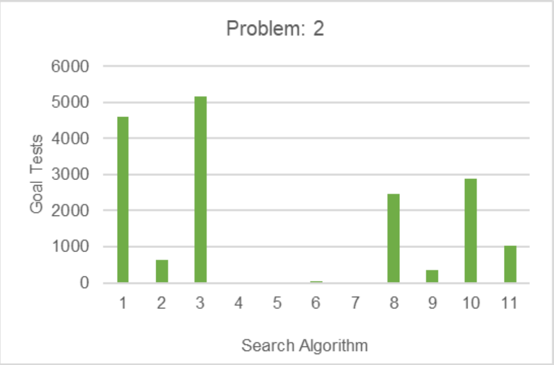


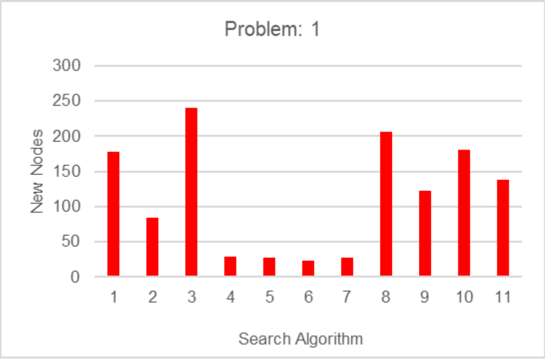


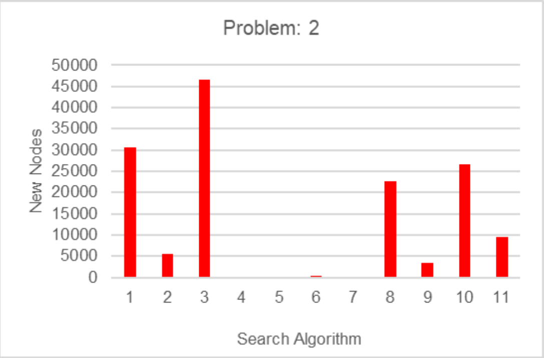

Problem: 1
Expansions
70
60
50
40
30
20
10
0
1 2 3 4 5 6 7 8 9 10 11
Search Algorithm

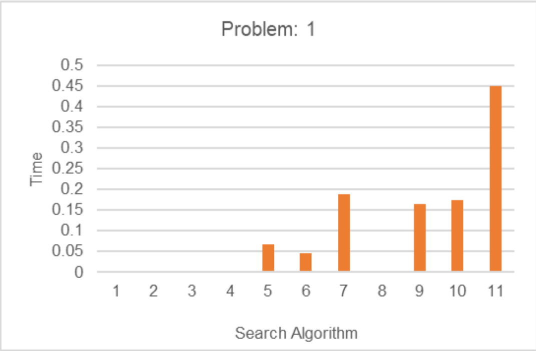

Problem: 2
Expansions
6000
5000
4000
3000
2000
1000
0
1 2 3 4 5 6 7 8 9 10 11
Search Algorithm

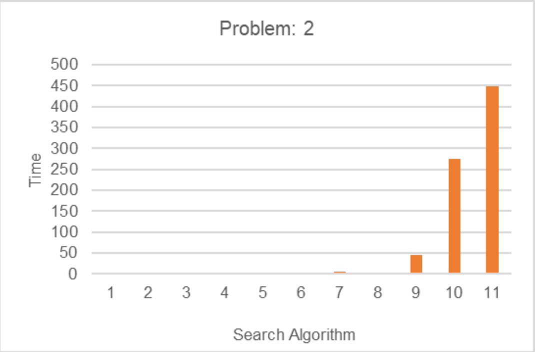

Problem: 1
Goal Tests
70
60
50
40
30
20
10
0
1 2 3 4 5 6 7 8 9 10 11
Search Algorithm


Problem: 2
Goal Tests
6000
5000
4000
3000
2000
1000
0
1 2 3 4 5 6 7 8 9 10 11
Search Algorithm


Problem: 1
New Nodes
300
250
200
150
100
50
0
1 2 3 4 5 6 7 8 9 10 11
Search Algorithm


Problem: 2
New Nodes
50000
45000
40000
35000
30000
25000
20000
15000
10000
5000
0
1 2 3 4 5 6 7 8 9 10 11
Search Algorithm


Problem: 1
Time
0.5
0.45
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
1 2 3 4 5 6 7 8 9 10 11
Search Algorithm


Problem: 2
Time
500
450
400
350
300
250
200
150
100
50
0
1 2 3 4 5 6 7 8 9 10 11
Search Algorithm

Before I go to the problem 3 and 4, I selected the following search algorithms:

- One uninformed search:

  I should select one from No.1, 2, 3.

  Action number and Time are not so different, but No.2 has the least Expansions, Goal Tests and New Nodes. It will be better to run with more complicated problems, so **I chose No.2 from uninformed search algorithm**.
- Two heuristics with greedy best first search:

  I should select two from No.4, 5, 6, 7.

  No.4 is much faster than others in problem 1 and 2, so **I chose No.4**.

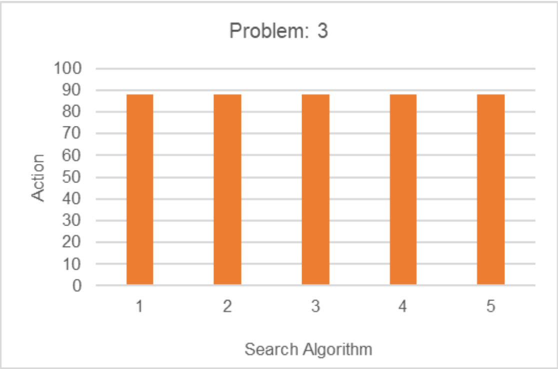
  There are not so difference between No. 5, 6, 7 in problem 1, but No.5 is faster than No.6, 7 in problem 2, so **I chose No.5**.
- Two heuristics with A*:

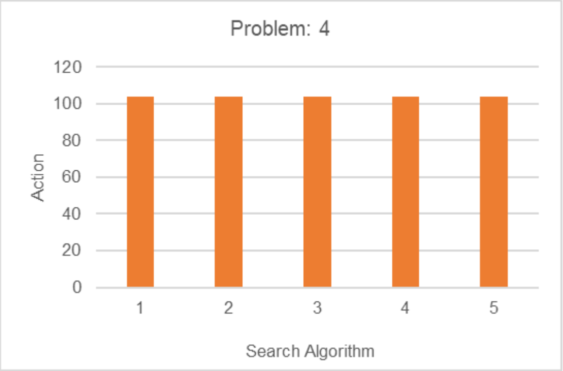
  I should select two from No.8, 9, 10, 11.

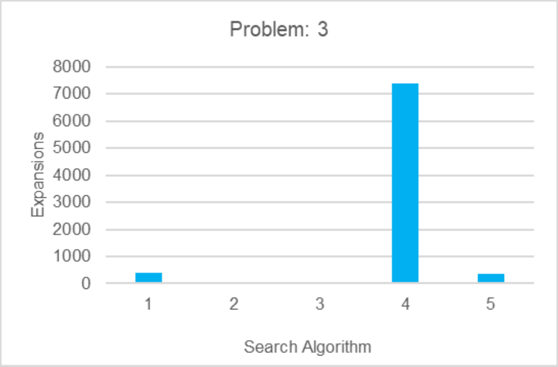
  No.10, 11 are much slower than No.8, 9, so **I chose No.8, 9**.

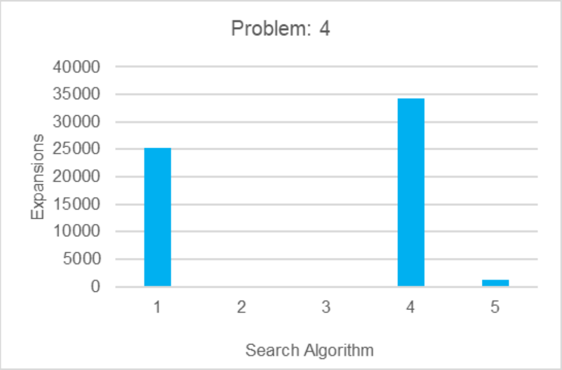
The chart and table includes data for selected search & heuristic combinations for air cargo problems 3 and 4 are shown in the following table and graphs.

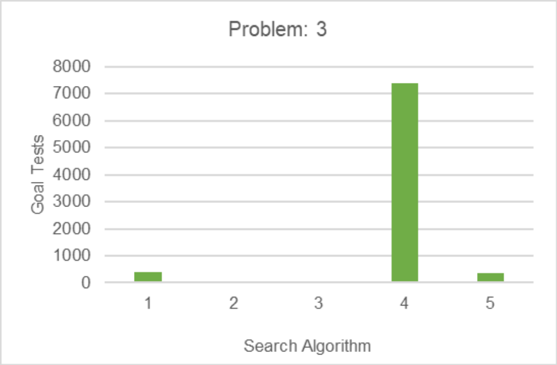
problem:	search:	Actions	Expansions	Goal Tests	New Nodes	Time
3	2	88	408	409	3364	0.301508
3	4	88	25	27	230	0.01062
3	5	88	14	16	126	3.462759
3	8	88	7388	7390	65711	2.5370707
3	9	88	369	371	3403	102.795644
4	2	104	25174	25175	228849	1541.16226
4	4	104	29	31	280	0.0312994
4	5	104	17	19	165	7.9906463
4	8	104	34330	34332	328509	22.5568907
4	9	104	1208	1210	12210	465.553544

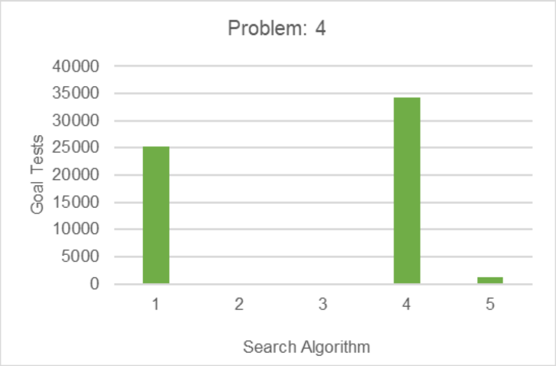


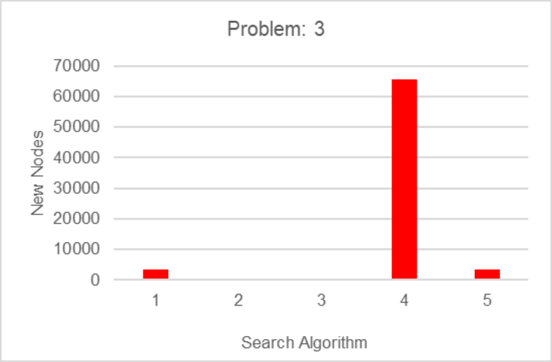


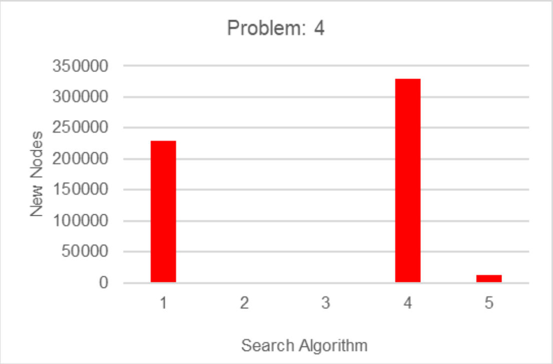

Problem: 3
Expansions
8000
7000
6000
5000
4000
3000
2000
1000
0
1
2
3
4
5
Search Algorithm

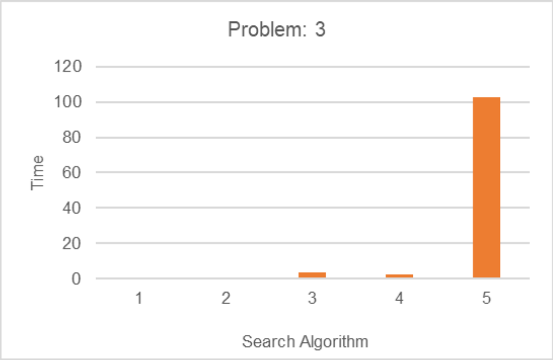

Problem: 4
Expansions
40000
35000
30000
25000
20000
15000
10000
5000
0
1
2
3
4
5
Search Algorithm

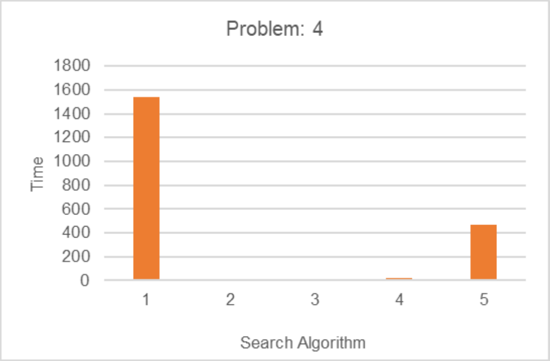

Problem: 3
Goal Tests
8000
7000
6000
5000
4000
3000
2000
1000
0
1
2
3
4
5
Search Algorithm


Problem: 4
Goal Tests
40000
35000
30000
25000
20000
15000
10000
5000
0
1
2
3
4
5
Search Algorithm


Problem: 3
New Nodes
70000
60000
50000
40000
30000
20000
10000
0
1
2
3
4
5
Search Algorithm


Problem: 4
New Nodes
350000
300000
250000
200000
150000
100000
50000
0
1
2
3
4
5
Search Algorithm


Problem: 3
Time
120
100
80
60
40
20
0
1
2
3
4
5
Search Algorithm


Problem: 4
Time
1800
1600
1400
1200
1000
800
600
400
200
0
1
2
3
4
5
Search Algorithm

(b) Report answers all required questions

I answer the following 3 questions which are written in the rubric.

- 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?

→ I think it's No.4 (greedy_best_first_graph_search, h_unmet_goals) because it's the fastest algorithm in a very restricted domain (project 1, 2).

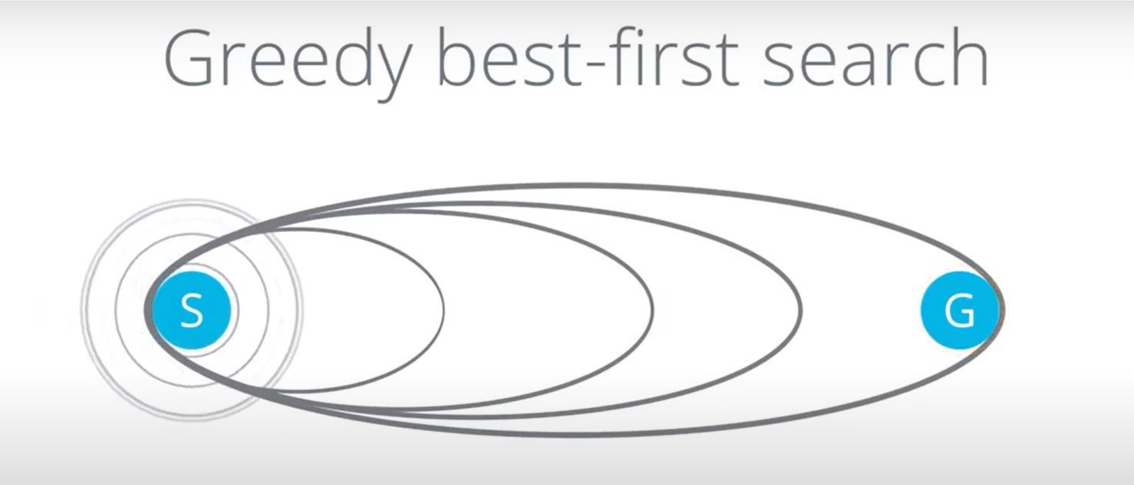
- Which algorithm or algorithms would be most appropriate for planning in very large domains (e.g., planning delivery routes for all UPS drivers in the U.S. on a given day)

→ I think it's No.4 (greedy_best_first_graph_search, h_unmet_goals) because it's the fastest algorithm in a very large domain (project 3, 4).

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

→ I think it's Greedy Best First Graph Search (No.4, 5) because it not only is fastest but also has smallest Expansions, Goal Tests, New Nodes compared with other algorithms.

It's intuitively understandable as I learned in the Lesson as below.


Greedy best-first search
S
G