2020

Forward Planning Agent

AI NANODEGREE PROJECT 2 REPORT

GORKEM BUZCU

Table of Contents

Table of Figures

Introduction

In this project our task was to complete the code in planning agent and then run search algorithms with different problems in order to create a benchmark. I have run all of the 11 search algorithms on the first two problems but due to the complexity concerns I only run one uninformed search(algorithm 1), two heuristics with greedy best first search(algorithms 4, and 5), and two heuristics with A*(algorithms 8, and 9) on problems 3 and 4.

Problem List

1	Air Cargo Problem 1
2	Air Cargo Problem 2
3	Air Cargo Problem 3
4	Air Cargo Problem 4

Algorithm List

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

Results

Unit Test Results

The unit test results of my_planning_graph:

```
root@233d4d3aab1f:/home/workspace# python -m unittest -v
test_1a_inconsistent_effects_mutex
(tests.test_my_planning_graph.Test_1_InconsistentEffectsMutex) ... ok
test_1b_inconsistent_effects_mutex
(tests.test_my_planning_graph.Test_1_InconsistentEffectsMutex) ... ok
test_1c_inconsistent_effects_mutex
(tests.test_my_planning_graph.Test_1_InconsistentEffectsMutex) ... ok
test_1d_inconsistent_effects_mutex
(tests.test_my_planning_graph.Test_1_InconsistentEffectsMutex) ... ok
test_1e_inconsistent_effects_mutex
(tests.test_my_planning_graph.Test_1_InconsistentEffectsMutex) ... ok
test_2a_interference_mutex (tests.test_my_planning_graph.Test_2_InterferenceMutex) ... ok
test_2b_interference_mutex (tests.test_my_planning_graph.Test_2_InterferenceMutex) ... ok
test_2c_interference_mutex (tests.test_my_planning_graph.Test_2_InterferenceMutex) ... ok
test_2d_interference_mutex (tests.test_my_planning_graph.Test_2_InterferenceMutex) ... ok
test_2e_interference_mutex (tests.test_my_planning_graph.Test_2_InterferenceMutex) ... ok
test_3a_negation_mutex (tests.test_my_planning_graph.Test_3_NegationMutex) ... ok
test_3b_negation_mutex (tests.test_my_planning_graph.Test_3_NegationMutex) ... ok
test_3c_negation_mutex (tests.test_my_planning_graph.Test_3_NegationMutex) ... ok
test_4a_competing_needs_mutex
(tests.test_my_planning_graph.Test_4_CompetingNeedsMutex) ... ok
test_4b_competing_needs_mutex
(tests.test_my_planning_graph.Test_4_CompetingNeedsMutex) ... ok
test_4c_competing_needs_mutex
(tests.test_my_planning_graph.Test_4_CompetingNeedsMutex) ... ok
test_4d_competing_needs_mutex
(tests.test_my_planning_graph.Test_4_CompetingNeedsMutex) ... ok
test_4e_competing_needs_mutex
(tests.test_my_planning_graph.Test_4_CompetingNeedsMutex) ... ok
test_5a_inconsistent_support_mutex
(tests.test_my_planning_graph.Test_5_InconsistentSupportMutex) ... ok
test_5b_inconsistent_support_mutex
(tests.test_my_planning_graph.Test_5_InconsistentSupportMutex) ... ok
test_6a_maxlevel (tests.test_my_planning_graph.Test_6_MaxLevelHeuristic) ... ok
test_6b_maxlevel (tests.test_my_planning_graph.Test_6_MaxLevelHeuristic) ... ok
test_6c_maxlevel (tests.test_my_planning_graph.Test_6_MaxLevelHeuristic) ... ok
test_6d_maxlevel (tests.test_my_planning_graph.Test_6_MaxLevelHeuristic) ... ok
test_6e_maxlevel (tests.test_my_planning_graph.Test_6_MaxLevelHeuristic) ... ok
test_7a_levelsum (tests.test_my_planning_graph.Test_7_LevelSumHeuristic) ... ok
test_7b_levelsum (tests.test_my_planning_graph.Test_7_LevelSumHeuristic) ... ok
test_7c_levelsum (tests.test_my_planning_graph.Test_7_LevelSumHeuristic) ... ok
test_7d_levelsum (tests.test_my_planning_graph.Test_7_LevelSumHeuristic) ... ok
test_7e_levelsum (tests.test_my_planning_graph.Test_7_LevelSumHeuristic) ... ok
test_8a_setlevel (tests.test_my_planning_graph.Test_8_SetLevelHeuristic) ... ok
test_8b_setlevel (tests.test_my_planning_graph.Test_8_SetLevelHeuristic) ... ok
test_8c_setlevel (tests.test_my_planning_graph.Test_8_SetLevelHeuristic) ... ok
```

```
test_8d_setlevel (tests.test_my_planning_graph.Test_8_SetLevelHeuristic) ... ok
test_8e_setlevel (tests.test_my_planning_graph.Test_8_SetLevelHeuristic) ... ok

----------------------------------------------------------------------
Ran 35 tests in 4.218s

OK
```

Search Algorithm Run Results

Number of actions are shown below:

Problem	Number of Actions
Air Cargo Problem 1	20
Air Cargo Problem 2	72
Air Cargo Problem 3	88
Air Cargo Problem 4	104

Table 1 Number of Actions

The number of actions available for a problem roughly indicates the size of our problem.

Expansion Results

Table 2 shows number of expanded nodes and the number of actions combined. This table tells how the expansion is related to the actions.

	Actions	s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11
P1	20	43	21	60	7	6	6	6	50	28	43	33
P2	72	3343	624	5154	17	9	27	9	2467	357	2887	1037
P3	88	14663			25	14			7388	369		
P4	104	99736			29	17			34330	1208		

Table 2 Number of Expansions

In order to understand better a graph is shown below:

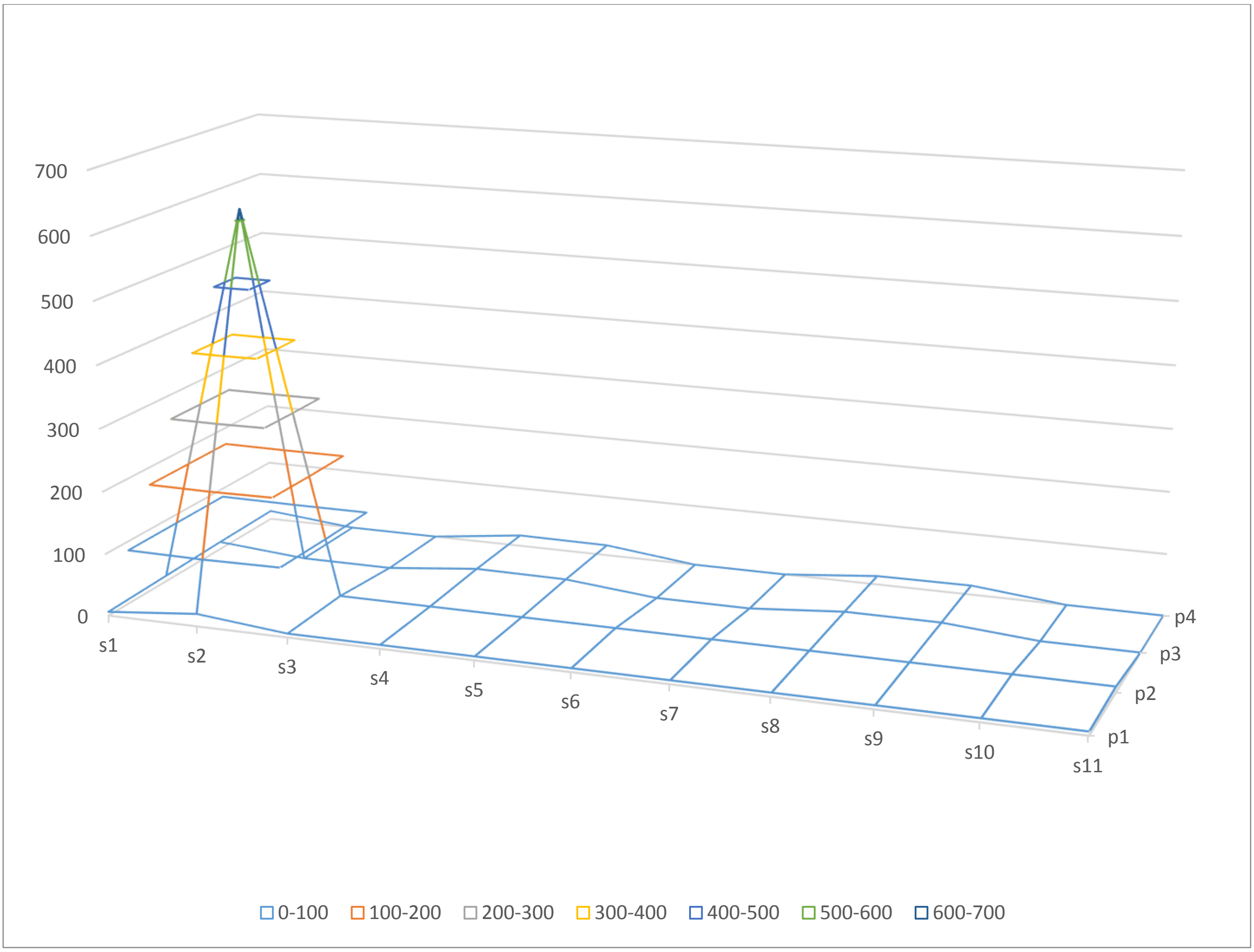

Figure 1 The Relation Between Expansions and Action Size

While Search algorithm 1 and 8 reacts very dramatically to action size, other algorithms are not as effected.

Search Time Results

	Actions	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11
P1	*20*	0.006	0.003	0.012	0.003	0.68	0.55	0.89	0.013	1.779	1.876	1.969
P2	*72*	3.012	4.23	4.79	0.02	12.11	24.29	16.62	2.4	305.34	2616.3	2169.1
P3	*88*	17.39			0.07	39.33			14.12	720.4		
P4	*104*	161.16			0.1	72.2			89.92	3892		

Table 3 Search Time Results

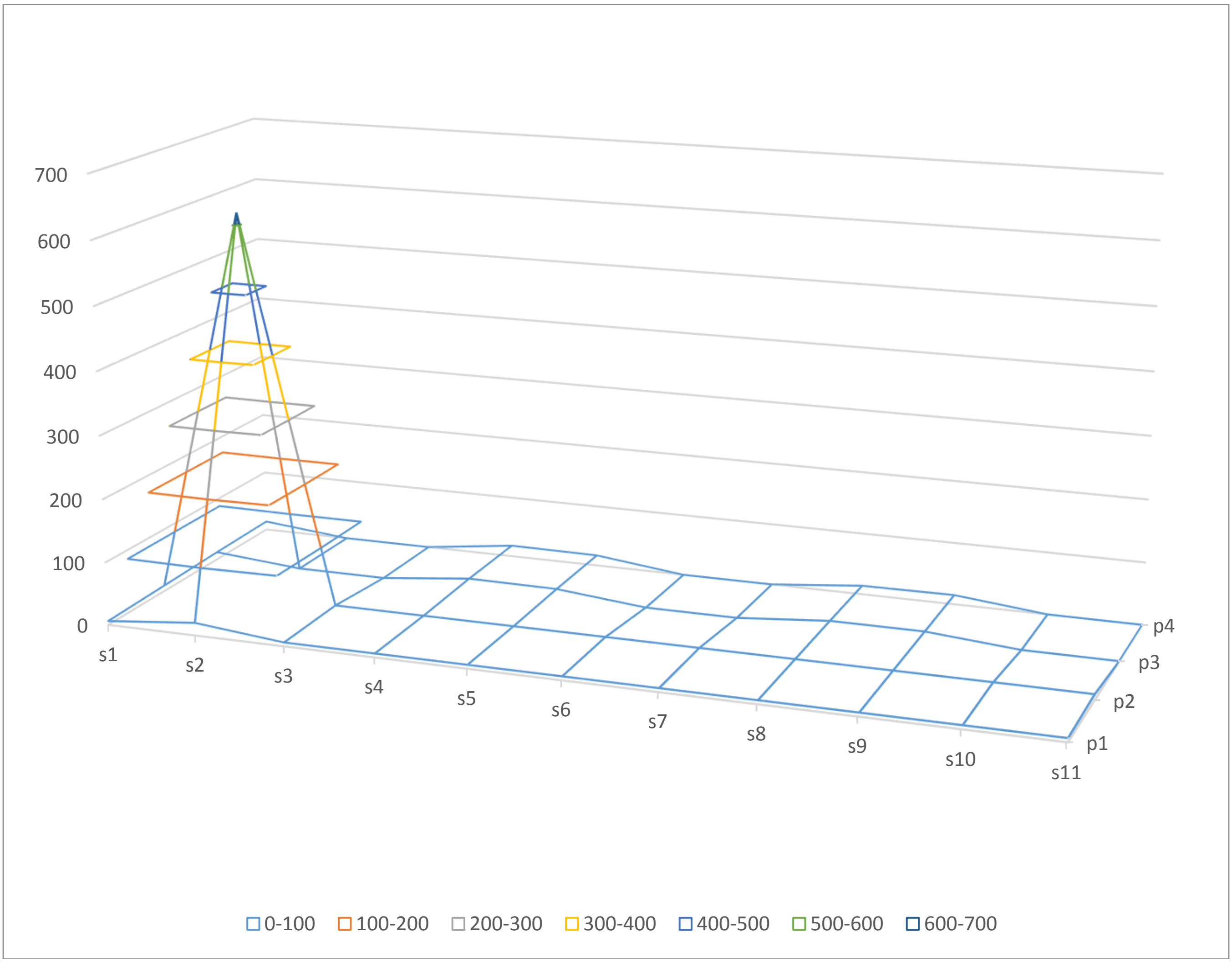

Figure 2 The Relation between Time and Action Size

Search algorithms 1, 8, and 9 grow exponentially. Search algorithms 4 and 5 remained fastest even in bigger size problems.

Plan Length Results

	actions	s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11
p1	*20*	6	20	6	6	6	6	6	6	6	6	6
p2	*72*	9	619	9	9	9	9	9	9	9	9	9
p3	*88*	12			15	14			12	12		
p4	*104*	14			18	17			14	15		

Table 4 Plan Length Results

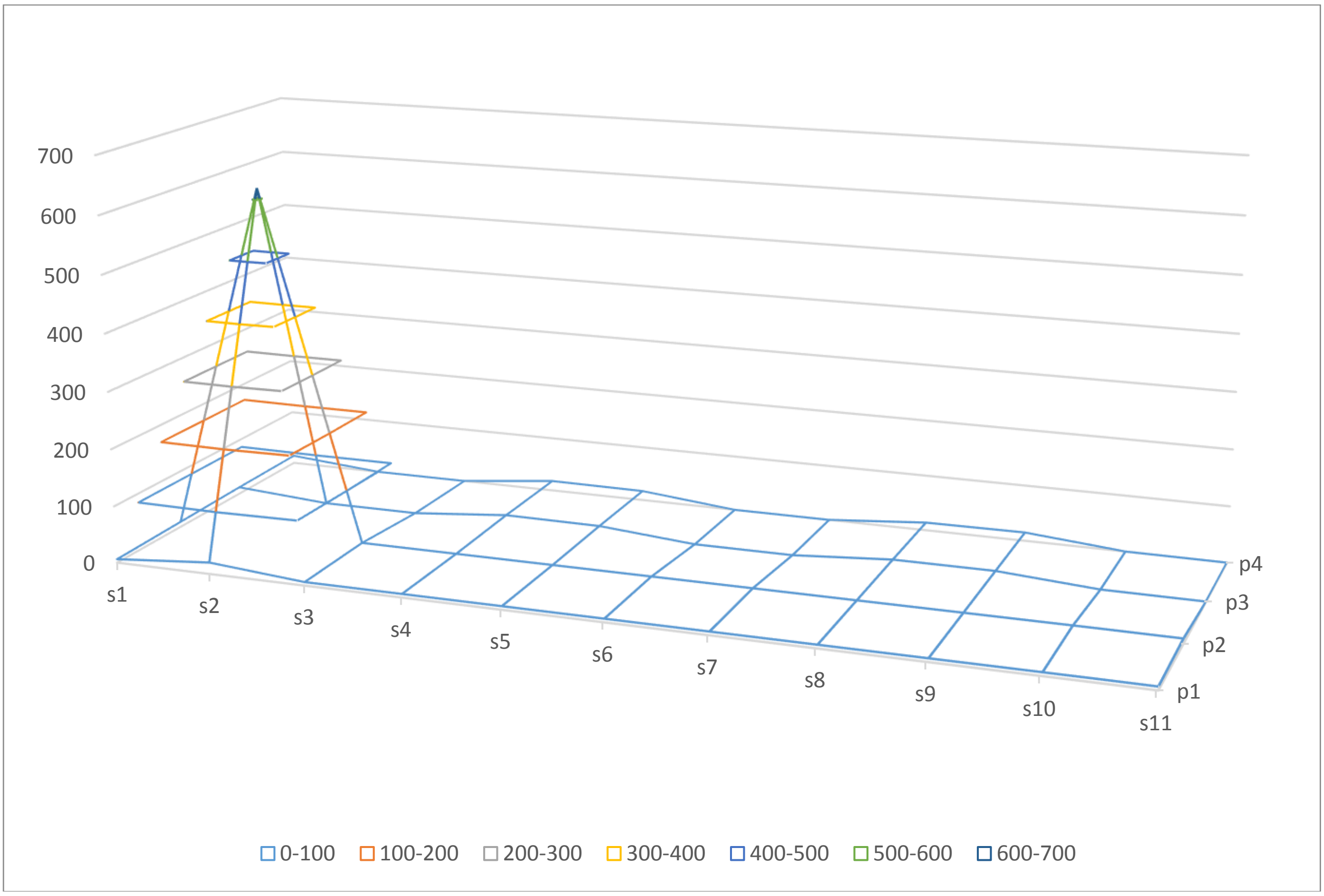

Figure 3 The Relation Between plan length and Action Size

Looking at Graph 3 clearly shows that search algorithm 2 (Depth First Graph) is the worst case. Other algorithms perform similarly to each other.

Questions and Answers

Q1

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?

Algorithm 4 would be the best choice since it is the fastest by far.

Q2

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)

For such a case both short route and speed are important. For speed of calculations algorithm 4 will be a good choice but for shorter route algorithm 8 will be the best choice.

Q3

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

For the optimum plan route, algorithms 1, and 8 are the best two choices.