Udacity Artificial Intelligence Nanodegree Report:

Project 3 Building a Game Playing Agent

Author: JiaHsuan Lo

Date: 1/5/2018

# Introduction

The goal of this project is to build a game playing agent to play the isolation game. The project involves code implementation and analysis. The game playing agent was implemented using the Minimax and alpha-beta pruning algorithms. Several heuristic evaluation functions were also implemented and their performance were analyzed. This document describes the main ideas of the heuristic evaluation functions and reports the analysis results.

# Heuristic Evaluation Functions

### Difference of Liberties

### Close to Center

### Difference of Corner Areas

### Difference of Two-Layer Liberties

# Analysis:

### Number of Nodes Expanded Against Number of Actions in the Domain

The number of actions in the domain of the airport cargo problem 1 and 2 are 20 and 72, respectively. As can be seen in Figure 2, when the number of actions is small (20 actions), all eleven search methods are able to consistently find the solution with low numbers of nodes expanded. On the other hand, when the problem has a larger number of actions as in problem 2, the numbers of nodes required to solve the problem increase for all search algorithms.

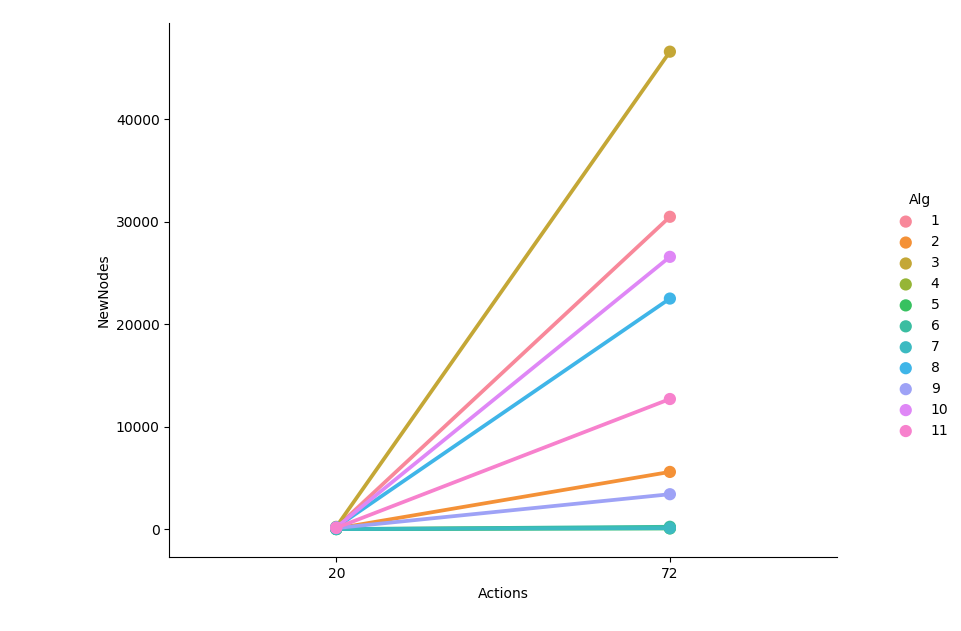


Figure : Number of nodes vs number of actions for each search algorithm

The increase in the number of nodes depends on the search algorithm. With a greater number of actions, there is a much wider variation in the number of nodes expanded, which can be shown in Figure 2 and Table 1. That indicates not all algorithms can efficiently solve the problem with a large design domain.

## 

Figure : Number of node expanded against number of actions

Table : Number of nodes expanded summary

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Number of New Nodes Expanded | | | |
| Number of Actions | min | max | mean | stdev |
| 20 | 24 | 240 | 114.3 | 80.3 |
| 72 | 86 | 46618 | 13509.0 | 15861.0 |

### Search Time against the Number of Actions in the Domain

As shown in Figure 3 and Table 2, the search time increases with increasing number of actions in the domain, regardless of search algorithm used. Moreover, there is a greater variation in the search time among search algorithms, when the number of actions is greater. These results indicate not all algorithms can efficiently (in terms of solving speed) solve the problem with a large design domain.

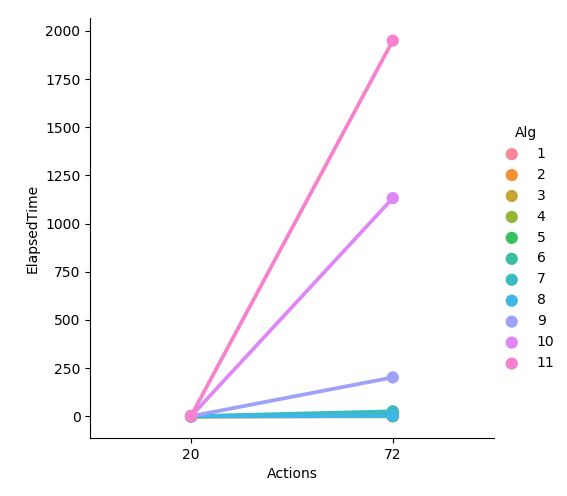


Figure : Elapsed time vs number of actions for each search algorithm

Table Elapsed time summary

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | ElapsedTime | | | |
| Actions | min | max | mean | std |
| 20 | 0.001254 | 2.310244 | 0.502025 | 0.703577 |
| 72 | 0.01659 | 1949.26 | 303.6412 | 641.0154 |

### Length of the Plans Returned by Each Algorithm on all Search Problems

As can be seen in Figure 4, almost all search strategies resulted in the same length of the plans, except for the depth-first (strategy 2) and the greedy\_best\_first-h\_pg\_setlevel (strategy 7) strategies. It is expected that depth-first search can result in non-optimal plan, because it can be stuck in the local minimum. The greedy\_best\_first-h\_pg\_setlevel strategy also put more emphasis on quickly finding the goal, so it may sometimes lead to non-optimal solution.

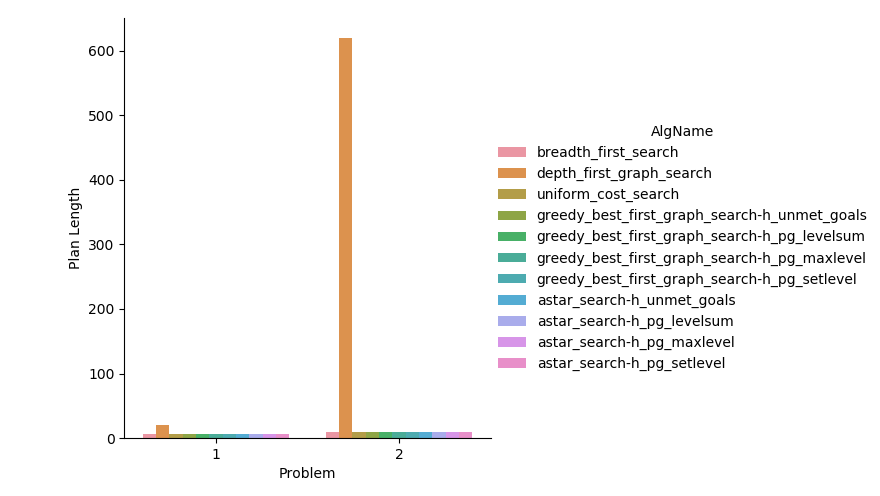


Figure : Length of plans by each algorithm

# Questions and Answers

### Summary of Problem 1 and 2

To answer the following questions, the search results of problem 1 and 2 are summarized in Table 3 for reference.

Table : Search results summary

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Problem | Algorithm | Actions | Expansions | GoalTests | NewNodes | PlanLength | ElapsedTime |
| Air Cargo Problem 1 | 1 breadth\_first\_search | 20 | 43 | 56 | 178 | 6 | 0.004 |
| 2 depth\_first\_graph\_search | 20 | 21 | 22 | 84 | 20 | 0.004 |
| 3 uniform\_cost\_search | 20 | 60 | 62 | 240 | 6 | 0.007 |
| 4 greedy\_best\_first\_graph\_search-h\_unmet\_goals | 20 | 7 | 9 | 29 | 6 | 0.001 |
| 5 greedy\_best\_first\_graph\_search-h\_pg\_levelsum | 20 | 6 | 8 | 28 | 6 | 0.361 |
| 6 greedy\_best\_first\_graph\_search-h\_pg\_maxlevel | 20 | 6 | 8 | 24 | 6 | 0.270 |
| 7 greedy\_best\_first\_graph\_search-h\_pg\_setlevel | 20 | 6 | 8 | 28 | 6 | 0.884 |
| 8 astar\_search-h\_unmet\_goals | 20 | 50 | 52 | 206 | 6 | 0.007 |
| 9 astar\_search-h\_pg\_levelsum | 20 | 28 | 30 | 122 | 6 | 0.833 |
| 10 astar\_search-h\_pg\_maxlevel | 20 | 43 | 45 | 180 | 6 | 0.841 |
| 11 astar\_search-h\_pg\_setlevel | 20 | 33 | 35 | 138 | 6 | 2.310 |
| Air Cargo Problem 2 | 1 breadth\_first\_search | 72 | 3343 | 4609 | 30503 | 9 | 1.510 |
| 2 depth\_first\_graph\_search | 72 | 624 | 625 | 5602 | 619 | 2.067 |
| 3 uniform\_cost\_search | 72 | 5154 | 5156 | 46618 | 9 | 2.434 |
| 4 greedy\_best\_first\_graph\_search-h\_unmet\_goals | 72 | 17 | 19 | 170 | 9 | 0.017 |
| 5 greedy\_best\_first\_graph\_search-h\_pg\_levelsum | 72 | 9 | 11 | 86 | 9 | 7.508 |
| 6 greedy\_best\_first\_graph\_search-h\_pg\_maxlevel | 72 | 27 | 29 | 249 | 9 | 15.084 |
| 7 greedy\_best\_first\_graph\_search-h\_pg\_setlevel | 72 | 12 | 14 | 111 | 10 | 26.123 |
| 8 astar\_search-h\_unmet\_goals | 72 | 2467 | 2469 | 22522 | 9 | 1.793 |
| 9 astar\_search-h\_pg\_levelsum | 72 | 357 | 359 | 3426 | 9 | 202.120 |
| 10 astar\_search-h\_pg\_maxlevel | 72 | 2887 | 2889 | 26594 | 9 | 1132.138 |
| 11 astar\_search-h\_pg\_setlevel | 72 | 1372 | 1374 | 12718 | 9 | 1949.260 |

### 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?

If the problem domain only has a few actions and it needs to operate in real time, the algorithm 4, **greedy\_best\_first\_graph\_search-h\_unmet\_goals** would be the most appropriate choice. It is because the algorithm 4 can solve the planning problem with the least node expansions and thus least solving time. Since the problem is assumed to be small, many other algorithms with more expansions works fairly well in terms of solving time. They include, 1 breadth\_first\_search, 3 uniform\_cost\_search, and 8 astar\_search-h\_unmet\_goals.

### 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 an algorithm to be able to solve a problem with a large design domain, it is important that the algorithm does not significantly increase the nodes expanded with increasing domain size. As shown in Figure 1. For some algorithms, such as 1 breadth\_first\_search and 3 uniform\_cost\_search, the nodes expanded increases drastically with increasing problem domain size.

On the other hand, the algorithm **5 greedy\_best\_first\_graph\_search-h\_pg\_levelsum** can solve the larger problem 2 with the least nodes expanded. So it would be the most appropriate algorithm for planning in very large domains.

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

If it is important to find only optimal plans, the depth first graph search and all greedy-based algorithm need to be excluded since they cannot guarantee to find the optimal solutions. Without considering the speed and memory usage, all possible algorithms suitable for finding optimal plans include:

1 breadth\_first\_search

3 uniform\_cost\_search

8 astar\_search-h\_unmet\_goals

9 astar\_search-h\_pg\_levelsum

10 astar\_search-h\_pg\_maxlevel

11 astar\_search-h\_pg\_setlevel