# Introduction to Artificial Intelligence: Assignment #1
# Fast Trajectory Replanning

Yuyang Chen. – yc791
Simiao fan – sf578
Zhaohan Yan – zy134

Rutgers University — February 25, 2019

## Introduction

**Info:** This document is Homework 1 for 198:440 Introduction to Artificial Intelligence at Rutgers University Spring 2019.

## 1 Part 1 - Understanding the methods

Read the chapter in your textbook on uninformed and informed (heuristic) search and then read the project description again. Make sure that you understand A* and the concepts of admissible and consistent h-values.

### 1.1

Question 1

Explain in your report why the first move of the agent for the example search problem from Figure 8 is to the east rather than the north given that the agent does not know initially which cells are blocked.

Answer:

When the agent begins in the maze program, it does not initially know which cells are blocked. It has to survey its surroundings as it takes each proceeding step. In Figure 8, the agent takes a step to the east rather than north because A* would calculate that it would take a shorter distance to the goal to go east, rather than first north and then east. Our agent can only see which cells are blocked at each subsequent step, so at its initial position, it cannot see that the path directly east is blocked both to the right and from above. Only once it moves east will it realize that it is not the correct path, and then move back to go north.

### 1.2

Question 2

This project argues that the agent is guaranteed to reach the target if it is not separated from it by blocked cells. Give a convincing argument that the agent in finite gridworlds indeed either reaches the target or discovers that this is impossible in finite time. Prove that the number of moves of the agent until it reaches the target or discovers that this is impossible is bounded from above by the number of unblocked cells squared.

Answer:

In this gridworld, the agent is guaranteed to reach the target of the maze (goal) or discover it is impossible to as A* search leads our agent through every possible cell in the grid to the goal. If the agent detect that the goal state is blocked on all sides by obstacles such as the grid edge or blocked cells, in the next A* search, the open list would be empty before the the algorithm pushing the goal stage node into the open list, and therefore we know we can't reach to the goal stage. Otherwise, the agent will keep moving along each cell until it reaches goal.

Use induction to prove: "The number of moves of the agent until it reaches the target or discovers that this is impossible is bounded from above by the number of unblocked cells squared. Let n be the total number of cells, let m be the number of unblocked cells, and let k be the total number of moves the agent took

Base case: n = 2, when there is only start state and goal state. If the goal state is blocked, then the total move would be k = 0, m = 1.

$$k < m^2 - - - check$$

If the goal state is unblocked, then m = 2, k = 1, because we directly move from start to goal with a single step.

$$k < m^2 - - - check$$

Inductive step: For n > 2, Suppose

$$k < m'^2$$

is true for all m' belong [1, m),need to prove

$$k < m^2$$

is also true. We know that for each unblocked cell, the agent would at most visit it twice (the first time follow the path to the goal, and later come back with the updated blocked information). Therefore, $k <= 2m$.

$$(\frac{2m}{m^2}) = (\frac{2}{m})$$

,

since $m > m' >= 1$, therefore $m >= 2$, and further

$$(\frac{2}{m}) <= 1$$

.

Thus k'/2m <= 1. Therefore $k < m^2$ hold true. Based on the base step and inductive step, we know $k < m^2$ for all m >= 1.

# 2 Part 2 - The Effect of Ties

Repeated Forward A* needs to break ties to decide which cell to expand next if several cells have the same smallest f-value. It can either break ties in favor of cells with smaller g-values or in favor of cells with larger g-values. Implement and compare both versions of Repeated Forward A* with respect to their runtime or, equivalently, number of expanded cells.

Question 3

Explain your observations in detail, that is, explain what you observed and give a reason for the observation.

Answer:

After implementing tie-breaking in regards to the g-value, we ran our project 50 times for Repeated Forward A* and compared the average run time. In our A* code, implementation of tie-breaking begins around line 34 in "classinfor". Here are the run-times:

Repeated Forward A*

**Choosing smaller g-value** Average run time: 5.512 s

**Choosing larger g-value** Average run time: 0.594 s

We observed that tie-breaking by the larger g-value proved to be almost ten times faster than the other method. From the A* algorithm, f(s) = g(s) + h(s). In terms of the tie-breaking, we can conclude that having a bigger g-value also means having a smaller h-value. In our case, the g-value represents the cost of the path the agent has finished while the h-value is a prediction of the best case in the future which can potentially increase a lot. In other words, having a larger h-value gives the agent more risks of increasing its overall cost in the future. With the same f-values, the one with the larger g-value will more likely to finish its path with less cost to be added to its original prediction. It is also why the method of choosing the larger g-value is faster than the method of choosing the smaller g-value.

hello.py

```
#! /usr/bin/python

import sys
sys.stdout.write("Hello␣World!\n")
```

Fusce eleifend porttitor arcu, id accumsan elit pharetra eget. Mauris luctus velit sit amet est sodales rhoncus. Donec cursus suscipit justo, sed tristique ipsum fermentum nec. Ut tortor ex, ullamcorper varius congue in, efficitur a tellus. Vivamus ut rutrum nisi. Phasellus sit amet enim efficitur, aliquam nulla id, lacinia mauris. Quisque viverra libero ac magna maximus efficitur. Interdum et malesuada fames ac ante ipsum primis in faucibus. Vestibulum mollis eros in tellus fermentum, vitae tristique justo finibus. Sed quis vehicula nibh. Etiam nulla justo, pellentesque id sapien at, semper aliquam arcu. Integer at commodo arcu. Quisque dapibus ut lacus eget vulputate.

Command Line

```
$ chmod +x hello.py
$ ./hello.py

Hello World!
```

Vestibulum sodales orci a nisi interdum tristique. In dictum vehicula dui, eget bibendum purus elementum eu. Pellentesque lobortis mattis mauris, non feugiat dolor vulputate a. Cras porttitor dapibus lacus at pulvinar. Praesent eu nunc et libero porttitor malesuada tempus quis massa. Aenean cursus ipsum a velit ultricies sagittis. Sed non leo ullamcorper, suscipit massa ut, pulvinar erat. Aliquam erat volutpat. Nulla non lacus vitae mi placerat tincidunt et ac diam. Aliquam tincidunt augue sem, ut vestibulum est volutpat eget. Suspendisse potenti. Integer condimentum, risus nec maximus elementum, lacus purus porta arcu, at ultrices diam nisl eget urna. Curabitur sollicitudin diam quis sollicitudin varius. Ut porta erat ornare laoreet euismod. In tincidunt purus dui, nec egestas dui convallis non. In vestibulum ipsum in dictum scelerisque.

# 3 Part 3 - Forward vs. Backward

Implement and compare Repeated Forward A* and Repeated Backward A* with respect to their runtime or, equivalently, number of expanded cells. Explain your observations in detail, that is, explain what you observed and give a reason for the observation. Both versions of Repeated A* should break ties among

cells with the same f-value in favor of cells with larger g-values and remaining ties in an identical way, for example randomly

Answer:

After implementing tie-breaking in regards to the g-value, we ran our project 50 times for Repeated Forward A* and compared the average run time. In our A* code, implementation of tie-breaking begins around line 34 in "classinfor". Here are the run-times:

# 4 Part 4 - Heuristics in the Adaptive A*

The project argues that “the Manhattan distances are consistent in gridworlds in which the agent can move only in the four main compass directions.” Prove that this is indeed the case.

Furthermore, it is argued that “The h-values hnew(s) ... are not only admissible but also consistent.” Prove that Adaptive A* leaves initially consistent h-values consistent even if action costs can increase.

Answer:

After implementing tie-breaking in regards to the g-value, we ran our project 50 times for Repeated Forward A* and compared the average run time. In our A* code, implementation of tie-breaking begins around line 34 in "classinfor". Here are the run-times:

# 5 Part 4 - Heuristics in the Adaptive A*

Implement and compare Repeated Forward A* and Adaptive A* with respect to their runtime. Explain your observations in detail, that is, explain what you observed and give a reason for the observation. Both search algorithms should break ties among cells with the same f-value in favor of cells with larger g-values and remaining ties in an identical way, for example randomly.

Answer:

After implementing tie-breaking in regards to the g-value, we ran our project 50 times for Repeated Forward A* and compared the average run time. In our A* code, implementation of tie-breaking begins around line 34 in "classinfor". Here are the run-times:

# 6 Part 6 - Memory Issues

You performed all experiments in gridworlds of size $101 \times 101$ but some real-time computer games use maps whose number of cells is up to two orders of magnitude larger than that. It is then especially important to limit the amount of information that is stored per cell. For example, the tree-pointers can be implemented with only two bits per cell. Suggest additional ways to reduce the memory consumption of your implementations further. Then, calculate the amount of memory that they need to operate on gridworlds of size $1001 \times 1001$ and the largest gridworld that they can operate on within a memory limit of 4 MBytes.

Answer:

After implementing tie-breaking in regards to the g-value, we ran our project 50 times for Repeated Forward A* and compared the average run time. In our A* code, implementation of tie-breaking begins around line 34 in "classinfor". Here are the run-times: