CS152 Assignment 1

1. PEAS description for the task environment

Performance:

- If run is completed (micromouse succeeds to enter one of the center four destination squares out of 10 number runs in total of 10 minutes): fastest run time (time from the micromouse leave the starting square to enter one of the center four destination squares)

- If run is not completed: maximum number of unique cells that the mouse traversed in a unique run

- Ranking is based on fastest run time first, and then the most unique cells traversed in one run

Environment:

- maze made up of 16*16 number of 18cm*18cm sized unit squares (totaling of 16*16*18*18=82944 cm^2 total size) and enclosed entirely by outside walls

- floor made of wood and colored black

- walls 5cm high, 1.2cm thick, length being multiples of 18cm (atleast 18cm) since it takes one side of one unit square, colored white on the side and red on the top

- starting unit square located at one of the four corners with one open wall, so that the mouse starts at one specific orientation being either North, South, East, or West

- target four unit squares located at the center of the maze with open wall(s) to allow the micromouse arrive at the goal destination

- passageways between walls 16.8cm wide

- Some minor aspects of the environment are as the following quote from the given document on CAMM 2016 Competition Rules: “The dimensions of the maze shall be accurate to within 5% or 2 cm, whichever is less. Assembly joints on the maze floor shall not involve steps of greater than 0.5 mm. The change of slope at an assembly joint shall not be greater than 4 degrees. Gaps between the walls of adjacent squares shall not be greater than 1 mm. The illumination, temperature, and humidity of the room shall be those of an ambient environment. (40 to 120 degrees F, 0% to 95% humidity, noncondensing). Fading of colors may occur.”

Actuator:

- control over steering from either one of North, South, East, or West to another direction

- control over motor with accelerator and brake that help move the micromouse and decrease or increase its speed

Sensor:

- cameras or radars that detect location of walls
- speedometer that calculates the speed of the micromouse to know how much to accelerate and decelerate in the future
- accelerometer to know the current acceleration value
- GPS that tells the location of the micromouse

1. Description of state space

Assume that the micromouse only needs to turn at the intersection of two or more passageways, that the location of all walls and thus the shape of the maze is fully observable (using a camera or radar that is able to observe entire shape of the maze, above the walls), and that the robot itself can figure out how to turn based on its current facing direction and the relative direction from its current position to the next position it aims to reach.

Also just for simplicity, let's assume the maze is placed at the first quadrant, most south-west corner unit square of the maze is represented by the cardinal point (0,0), and each unit square is also represented by its south-west corner of the square. For example, the unit square right above the most south-west corner would be represented by (0,1).

Such assumptions allow us to abstract the originally continuous states space, since the micromouse originally could lie at any point in the square bounded by vertices (0,0), (16,0), (0,16) and (16,16), except the walls. Now we only need to consider possible states as the possible turning locations the micromouse may reach, since orientation or facing direction does not matter if we know the relative direction from its current position to its next position it aims to reach.

Thus, possible states are the intersection of two or more passageways in the maze and the size of state space or the number of states possible would be n, the number of these intersections. Possible actions would be to move from one intersection to another intersection, only vertically or horizontally undisturbed by any walls. We can call such intersections as being adjacent to each other in this report for simplicity.

According to the rules of the micromouse challenge, the initial state would be the starting unit square in either of the four corners, so one of (0,0), (15,0), (0,15), and (15,15). The goal state would be the target center four squares, so one of (7,7), (8,7), (7,8), and (8,8). There would be two action cost functions: one being the time taken from the action of moving from one intersection to its adjacent intersection or the number of unit squares passed in such action assuming that speed is constant throughout each movement action, another being the number of unique unit squares traversed in such action (in case the target squares are not reached)[1]

[1] Cs152-aiconcepts: summarizes the problem with PEAS description and abstracted state space by explaining appropriate terminologies such as performance measure, environment,

2. Classification of micromouse agent program

Micromouse is a utility-based and model-based reflex agent. It is model-based because it needs to keep track of its current location in the maze and tell how much it has come compared to the path calculated with its algorithm, rather than figure out its current location and which path to go next each time being in a new state like a simple reflex agent. The transition model would be that each action of moving from one intersection to an adjacent intersection would lead the micromouse to steer towards the new intersection and drive there. The sensor model may be that when the micromouse starts at the starting square, its camera sensor detects the whole shape of the maze, allowing for its algorithm to find the fastest path from the starting square to the target square.

It is also utility-based because the performance measure is not only about reaching the goal, but also taking the least time to reach the goal if the goal can be reached and the most unique unit squares traversed if the goal cannot be reached.

The utility function f may be represented as:

if t = 10 – (time for complete run) and n = (number of unique squares traversed in a run),

f = t + 16*16, when the goal is reached

f = n, when the goal is not reached

- t was set to be 10 – (time for complete run), because we want the utility to increase as the time for complete run decreases and the maximum time permitted for a unique run is 10 minutes.

- The total number of unit squares was added to t in the first equation, because the maximum number of unique number of unit squares abled to traverse is 16*16 and we want the utility function to be always larger when the goal is reached, compared to when the goal is not reached. This is simply because ranking is based on fastest run time first, and then the most unique cells traversed in one run, according to the the performance measure of the problem based on the given micromouse challenge rules.

We can also simplify the utility function as the following:

if t = 10 – (time for complete run), n = (number of unique squares traversed in a run), x=0 when the goal is reached, and x=1 when the goal is not reached,

f = (t + 16*16)*(1-x) + n*x [23]

---

actuator, sensor, initial state, goal state, and cost function.

[2] Cs152-aiprofessionalism: clearly classifies the program as model-based and utility-based program by defining the utility function with variables clearly explained and reason behind defining such function

[3] #utility: define the utility function to represent the performance measure and use it for

3. How the mouse will need to be programmed to behave rationally

The micromouse needs to use an algorithm combined of mainly two algorithms to find the optimal path. One being an informed search algorithm like the A* algorithm, to find path that goes through the least number of unit squares since that would mean it would be the path that takes the least time. (Since this is a maze, the algorithm will need to be modified a little by cutting off any redundant paths.) After figuring out the optimal path, the micromouse would also find places in the maze where it can move diagonally to fasten the time. The algorithm also needs to determine whether the goal is reachable and if the goal is not reachable whether because the algorithm fails to find it or the maze is designed to have no path in the first place, it needs to then use another second algorithm such as deep search that finds the path that would traverse through the most unique unit squares possible.

Such ultimate algorithm mainly combined of two algorithms would be operated based on the whole shape of the maze, perceived at the starting square with a camera looking over above the maze. And after the algorithm decides which path to take to maximize utility or the performance measure, it would also lastly decide what speed to take, when and how much to accelerate and decelerate etc. to move the micromouse in the intendeded path as fast as possible. Below is a more detailed explanation on the first two algorithms I mentioned:

The first A* algorithm would search through a tree with the root as the initial state or the starting square (0,0) and children as the possible successor state or in other words the parent node's adjacent intersections. It would expand on the nodes with the least f(n) = g(n) + h(n), when g is the actual path cost or the number of unit squares needed to reach from the starting square to the current state or unit square n, and h is the Manhattan distance from the current state to the closest goal state out of the four, because it is an admissible heuristic that will never overestimate the actual number of passed squares needed to reach target squares. After reaching the goal, the algorithm will also need to cutoff redundant paths such as A-B-C-B-A of returning back to A, which occur when the algorithm leads to a dead end and returns back. We should also let the algorithm to find paths that can be gone faster when driven diagonally to find the optimal path solution if one exists.[4]

We will know if there is no solution if A* algorithm goes into an infinite loop, so we can say that if the same sequence of about 16*16/2 unit squares (selected number as large as half of the maze to safely conclude it entered an infinite loop) were repeated during the search, A* is in an infinite loop and there is no detectable solution to the maze.[5]

The second algorithm is different from the first algorithm in that it is beneficial for the mouse

---

decision making

[4] #heuristics: use heuristic Manhattan distance in the A* algorithm and explain how it is admissible to help reach the optimal solution

[5] #plausibility: explains the plausibility of setting the standard of defining infinite loop as repeating path for 16*16/2 unit squares or half the total maze, explains for the reasons why a combination of A* and deep search algorithms is plausible for solving the problem

to go back and forth from dead ends and thus does not only need the intersections but also the dead ends as the state. In other words, state space changes differently from what I initially stated to location of both intersections and dead ends when we find out there is no solution. With such altered state space, we may do a deep search to go through the whole maze.[6][7]

[6] Cs152-search: applies two search algorithms which are A* and deep search algorithms and explains its details and any points to modify in the algorithms to fit the micromouse problem.

[7] #modeling: devises a model that would help solve the micromouse problem through defining state space and applying mainly two alogorithms