
**CS152 Assignment 1**

**Micromouse Challenge**

**Steven Tey**

**Prof Shekhar**

**PEAS Description**

*Write down a well-justified PEAS description for the task environment of the micromouse challenge.*

- **Performance**
    - **Time taken** for the mouse to get from the selected starting square to one of the four center squares of the maze (max 10 mins)
    - If the mouse doesn't get to the center squares, its score will be determined by the maximum number of unique cells that the mouse traversed in a unique run.
    - If there's a tie in terms of the time taken between two mice, **the number of unique cells traversed** will be the tiebreaker.
- **Environment**
    - **Maze size** - the maze contains 16 x 16 multiples of an 18 cm x 18 cm unit square.
    - **Passageway width** - the passageways between the walls are 16.8 cm wide
    - **Colors of the walls and the floors** - The sides of the maze walls are white and the floor of the maze is finished with a nongloss black paint.
    - The **starting square orientation** (start state) - it is located at one of the four cornersand has walls on three sides.
    - The **center four squares** (goal state) - that's the only part of the maze where the squares do not have walls all four of its vertices.
- **Actuators**: Motors via interrupts programming, engine, wheels.
- **Sensors**: Obstacle sensors installed on the front, back, left and right of the mouse

**State Space**

*Provide a candidate description of the state space for the micromouse problem, being sure to describe the state dimension, the number of states and possible actions. There are multiple ways of defining the state space, but whichever one you choose needs to be justified.*

- To understand the state space of the micromouse problem to the fullest extent, we have to consider the number of unit squares as well as the number of states that the mouse can take on in each unit square.
- To make the problem more tractable, instead of viewing the transition between two unit squares as a continuous distribution of steps (with inifinitesimally small step sizes), we will use an abstraction and consider the transition between unit squares to be negligible.
- Given that there are 16 x 16 squares, and the robot can be facing one of 4 different directions at each square, there are a total of $16 \times 16 \times 4 = 1,024$ different configurations/states in this state space.
- At each state, the mouse has 4 different actions: move forward, turn 90 degrees clockwise, turn 180 degrees clockwise, or turn 90 degrees anti clockwise
- We can further abstract the state space by combining the action of making a turn and moving forward into one state space, which will reduce the number of possible states that the mouse has to commit to memory, hence reducing the space complexity. We can also use heuristics to eliminate unnecessary states such as by neglecting routes that are facing in the opposite direction of the goal state (risk: might result in untraversed nodes).

**Programming The Mouse**

*Given the performance measure and state space that you have specified, explain how the mouse will need to be programmed to behave rationally (i.e. what actions will in need to take in response to the current state and percepts).*

- The mouse needs to know the coordinates of the start state and the goal state.
- While the mouse is not at the goal state, it should try and minimize the Euclidean distance between its current state and the goal state.
- When it reaches a junction, it can randomly pick a direction and map the path cost that it needs to take to get to the destination.
- To make sure that it explores all the possible paths in the maze, even when the mouse reaches the goal state in the first round, it should keep traversing all the different paths in the maze until it has went over all 16 x 16 squares of the maze.
- The initial round can be done using an uninformed search algorithm such as a depth-first search to allow it to traverse all the possible paths of the maze. The reason why DFS was used is because the mouse needs to at least be able to reach the goal state once - the time taken to reach the goal state is considered first before the number of steps (which can be maximized with breadth-first search).
- The second round will be done using an informed search algorithm like A* search, which takes into account the path cost to a certain node as well as the cost to get from the node to the goal state (which can be derived from the traversal data in the first round).

## Extension

*How would you classify the micromouse in terms of its agent program (as discussed in section 2.4.1 of Russell & Norvig)? Justify your answer. Extra credit will be given for sketching out the pseudocode of the agent program, using your derived state description.*

In the case of the micromouse challenge, the micromouse is a goal-based first utility agent - it receives a goal information that describes situations that are desirable (the only region in the maze that does not have walls on all 4 vertices of the squares) and does everything in its power to search for the goal state. However, it also needs to take into account the amount of time that it needs to achieve that goal - the shortest possible route is the most desirable.

With all this information in mind, here's the psuedocode that the micromouse can use to solve the problem:

```
start_state = (0, 0)
end_state = (x, y)

# this is to store the coordinates & walls data of traversed path
path_data = []

while current_state != end_state:

      path_data.append([current_state, walls_data])

      if not intersection:
            current_state += 1 # move forward
      elif intersection:
            current_orientation = random.direction
      else: # if it's a dead end
            backtrack to previous intersection
```

**LOs Used**

#aiconcepts

- In this assignment, I explained the objective of the micromouse and put the context of the maze into perspective using the PEAS framework. I then explained how the micromouse will think and act rationally based on the environment factors and performance metrics and find the most optimum route to get from the start state to the goal state. I also accurately classified the agent type of the micromouse as a goal-based first utility agent.

#search

- In this assignment, I defined the state space of the micromouse challenge and explained the type of search algorithm that it should employ to find the most optimum way to the goal state. I also accurately distinguished between depth-first search and breadth-first search and explained their relevance to the context of this question. I then wrote up a psuedocode of the agent program that shows how the micromouse can find its way to the goal state.

#constraints

- There are various constraints that the mouse faces in the micromouse challenge - the time limit (10 mins), the walls (the mouse cannot jump over walls), as well as the number of runs (max 10 runs). As a result, the mouse has to apply constraint satisfaction in its programming to make sure that it is able to find the optimum solution in the least amount of time & traverse the maximum number of unique squares.