Maze Solving using Flood Fill Algorithm

Rayan Bouhal and Niko Paraskevopoulos

October 2023

Introduction to Micromouse

In the world of Robotics there is a competition called Micromouse. Micromouse uses Robotics, Computer Science, and Combinatorics to create an autonomous mouse that is no bigger than $16cm \times 16cm$ which can plan its own path to solve a $2.88m \times 2.88m$ maze. The mouse that finds the center of the maze in the shortest time wins. [7].

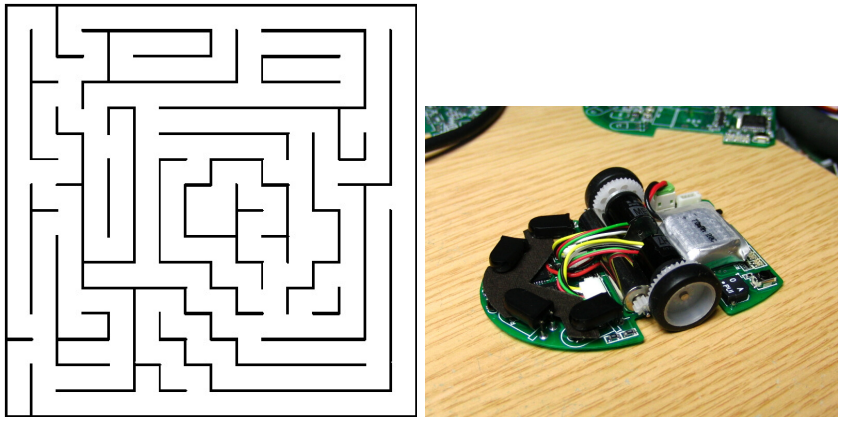


Figure 1: Example Micromouse maze and mouse [7][2]

Core Concepts

Robotics

The true feat of Micromouse competitions lies in the remarkable design achievement of constructing hardware that adheres to stringent size requirements while allowing the mouse to execute intricate maneuvers, including right, left, and diagonal turns. These miniature marvels employ infrared sensors which map the maze in a limited number of runs, and then deploy their path finding algorithms to race to the center of the maze. Typically, the fastest route is also the shortest

path; however, depending on a mouse's hardware configuration, exceptions may arise. In one memorable competition, the winning mouse traversed a path that deviated from its competitors. This path, technically longer, featured fewer 90° turns, allowing the mouse to achieve higher speeds despite its extended route.

While the hardware component of the competition is undeniably pivotal and presents its own set of challenges, this research paper will exclusively focus on the software and mathematical concepts of Micromouse. It is worth noting that without the hardware component, the Micromouse challenge would not be complete.

Computer Science

Another core concept at work in Micromouse is the software design of an intelligent decision making algorithm. There are many algorithms that have been used to solve a Micromouse maze, the most popular are Flood Fill, Dijkstra's, A*, and wall-following [4][3]. Currently, the best and most popular algorithm used in competitions is Flood Fill, with Dijkstra's and A* being slightly behind, and wall-following being considered the poorest algorithmic choice. The development of these algorithms has taken 30 years to perfect for this challenge [4].

In this research project we will present a visualization of the Flood Fill Algorithm that solves a randomly generated Micromouse maze, to gain a deeper understanding of the computing and mathematical concepts introduced by Micromouse. [1].

Combinatorics

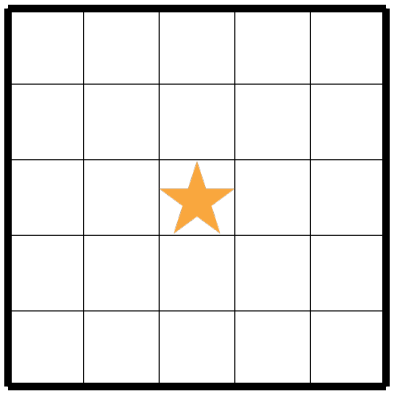
All of the algorithms needed to solve Micromouse are derived from combinatorial concepts with the main concepts being graph theory, shortest paths, and network algorithms. In the following section we will present an overview of the Flood Fill Algorithm by defining the necessary combinatorial contexts.

Combinatorial Definitions

Shortest Path Algorithms can be represented as networks. A network is a labeled graph, which assigns an integer value k to all edges. This value is used to weight the value or cost of an edge [6]. In the case of Flood Fill k represents the distance or proximity from the exit of the maze.

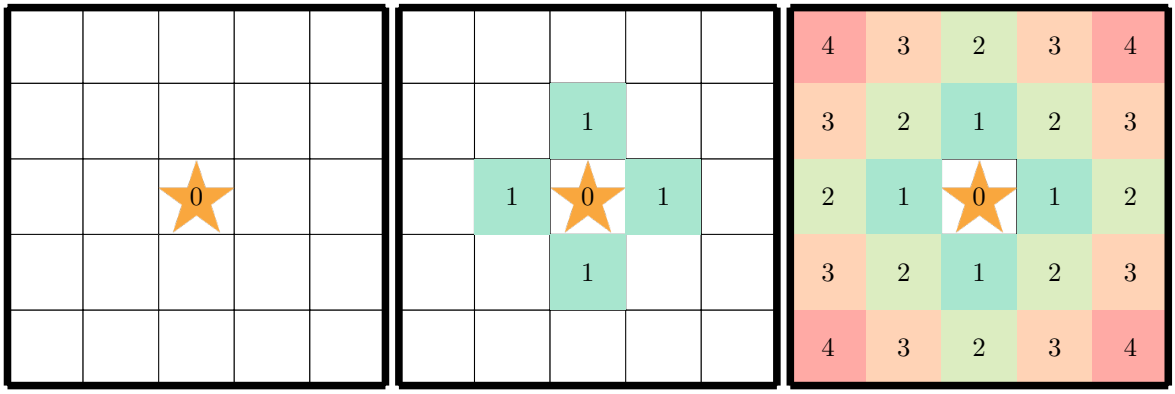
The maze is an undirected and connected graph. It is our goal to find a shortest path from vertex a to vertex b. We say a shortests path because there could be several shortests paths depending on a lot of different factors.

Flood Fill Algorithm



We can visualise our maze as an $m \times m$ grid. Generally, for Micromouse $m = 16$. However, for simplicity we will reduce our m to 5. With the end of the maze being the center block, represented with a yellow star.

Proximity K-Values



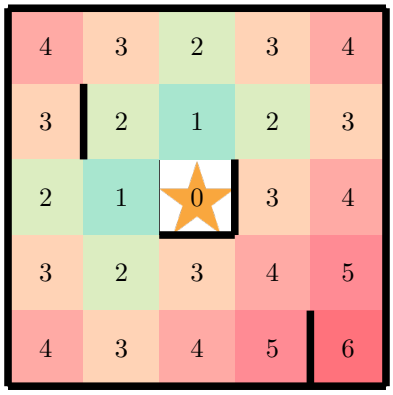
The first step of the flood fill algorithm is to assign numeric proximity values to all the squares in our grid. We previously defined this value as k.

1. The destination value (in our case the star in the center) is assigned a 0
2. All squares that are next to 0 are assigned a 1
3. All squares that are next to 1 are assigned a 2
4. Repeat this numbering process until all squares are enumerated

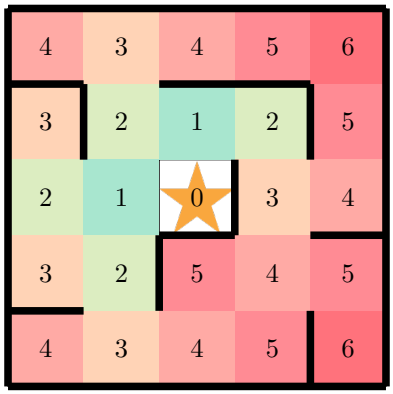
These numbers work such that if the mouse is in a square with a label 1 then it is 1 square away from the finish.

The initial mapping depicted above is how the mouse first views the maze. On the first iteration the mouse will act as if the maze has no walls. Then as

the mouse runs through the maze and senses walls it recomputes the proximity values.



Here is a simple illustration to show, that by adding just a few walls the Flood Fill Algorithm completely recalculates the proximity values.

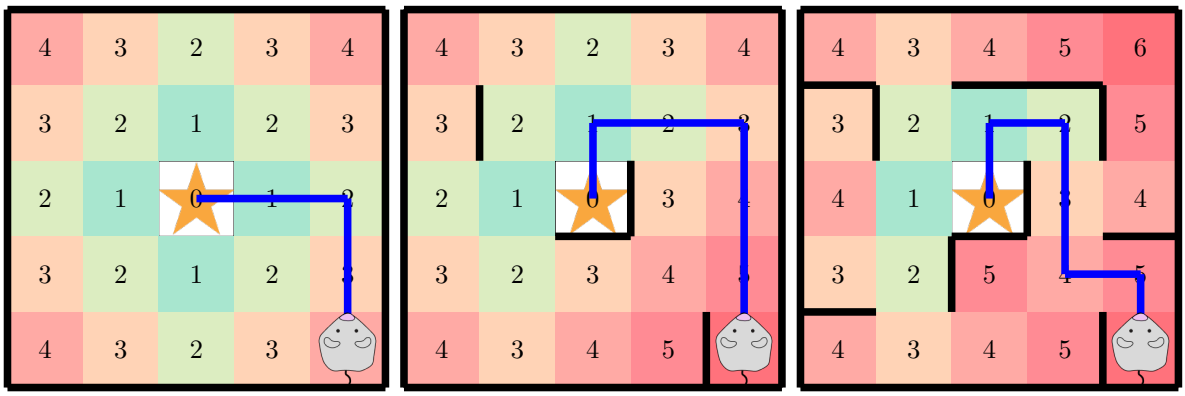


Here is a more complex maze that better imitates the Micromouse competition.

Shortest Path

Now that we have mapped the proximity k-values we are going to use those values to help us find the shortest path. Let's consider all three mazes we just viewed.

Let the starting position of the mouse be the bottom right corner facing the top of the maze. The Flood Fill Algorithm will find the shortest path by following descending proximity values. If it reaches a square where the values surrounding that square are equal, it will prioritize the straight path, since avoiding turns is generally faster for a vehicle.



Research Approach

We aim to develop a graphical simulation of a mouse navigating a maze using the Flood Fill Algorithm by integrating insights from various research papers. Komák and Pivarčiová employed CAD, a 3D modeling software, to visually represent the maze [1]. In contrast, we will utilize Python—a language we are proficient in—for both graphics and the algorithm's implementation.

Komák and Pivarčiová emphasized the need for the underlying code to detect maze walls and facilitate the mouse's movements—forward, left, and right [1]. In their model, users manually control the mouse's movement. The simulation concludes once the mouse either encounters an obstacle or reaches its destination [1].

Our goal is to enhance this model. We aspire to design an autonomous Micromouse maze simulation that leverages the Flood Fill Algorithm to determine the shortest path. Mirsha et al. and Tjiharjadi et al. have illustrated how to integrate the Flood Fill Algorithm with physical components to navigate a real Micromouse maze [3] [4] [5]. Adapting their work, we will focus solely on the algorithm, setting aside the hardware considerations.

Enhancements to Our Research

The Flood Fill Algorithm currently stands as the most prominent method for Micromouse competitions. Consequently, from an algorithmic standpoint, there's limited scope for refining our approach. However, a significant aspect we're choosing not to address in our research pertains to the diagonal movements of the mouse in real-life Micromouse contests. Our simulation restricts the mouse to move only forward, left, or right. A potential avenue for augmenting our research would be to modify both our simulation and algorithm to incorporate this diagonal movement capability. Nevertheless, for reasons of time constraints and simplicity, we have opted not to delve into this dimension for now.

References

[1] Martin Komák and Elena Pivarčiová. "Creating a Simulation Environment for the Micromouse". English. In: *TEM Journal* 11.1 (Feb. 2022). Copyright - © 2022. This work is published under https://creativecommons.org/licenses/by-nc-nd/4.0/ (the "License"). Notwithstanding the ProQuest Terms and Conditions, you may use this content in accordance with the terms of the License; Last updated - 2022-09-06, pp. 479–483. URL: http://login.ezproxy.lib.vt.edu/login?url=https://www.proquest.com/scholarly-journals/creating-simulation-environment-micromouse/docview/2702222085/se-2.

[2] *Micromouse robot runs maze in record-breaking five seconds (w/ Video)*. Image source. 20120. URL: https://phys.org/news/2010-11-micromouse-robot-maze-record-breaking-seconds.html.

[3] Swati Mishra and Pankaj Bande. "Advanced Algorithms for Micro Mouse Maze Solving". In: *Proceedings of the 2009 International Conference on Embedded Systems & Applications, ESA 2009, July 13-16, 2009, Las Vegas Nevada, USA*. Ed. by Hamid R. Arabnia and Ashu M. G. Solo. CSREA Press, 2009, pp. 78–84.

[4] Swati Mishra and Pankaj Bande. "Maze Solving Algorithms for Micro Mouse". In: *2008 IEEE International Conference on Signal Image Technology and Internet Based Systems*. 2008, pp. 86–93. DOI: 10.1109/SITIS.2008.104.

[5] Semuil Tjiharjadi, Marvin Wijaya, and Erwin Setiawan. "Optimization Maze Robot Using A* and Flood Fill Algorithm". In: *International Journal of Mechanical Engineering and Robotics Research* 6 (Sept. 2017), pp. 366–372. DOI: 10.18178/ijmerr.6.5.366-372.

[6] A. Tucker. *Applied Combinatorics, 6th Edition*. Online access: Center for Open Education Open Textbook Library. Wiley, 2012. ISBN: 9781118210116. URL: https://books.google.com/books?id=hdgbAAAAQBAJ.

[7] Bob White. *APEC MicroMouse Contest Rules*. 2004. URL: https://www.thierry-lequeu.fr/data/APEC/APEC_MicroMouse_Contest_Rules.html.