Genetic algorithm Flow Free Solver

（evolutionary computation applications）

Team member

Student ID	Chinese name	Email address
313551079	張芷烜	grace534266.cs13@nycu.edu.tw
313551092	李姿慧	zephyrlii.cs13@nycu.edu.tw

Introduction

- **Flow Free game rules**

1. Connect matching colored dots by drawing continuous lines
2. Fill the entire grid with these lines without overlapping

- **Goal : Find the only solution**

We aim to apply a GA to tackle the Flow Free puzzle, a complex path-finding and constraint-satisfaction problem. As the grid size increases, finding the correct solution becomes significantly more challenging for human solvers, making it an ideal problem for GA's optimization capabilities. The GA's inherent ability to explore large solution spaces efficiently makes it well-suited for finding the unique solution in larger, more complex Flow Free grids.

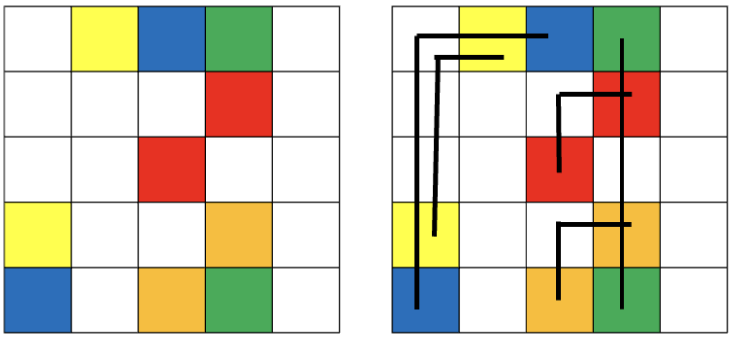
Implement Step

The most challenging aspect of implementing this project is finding an effective method to initialize the first generation of candidates. It is crucial to ensure that the initial candidates are of sufficient quality to enable the genetic algorithm to generate a strong and viable next generation. Additionally, determining an effective fitness function to select the best candidates for parenthood is critical, as it directly impacts the convergence of the solution.

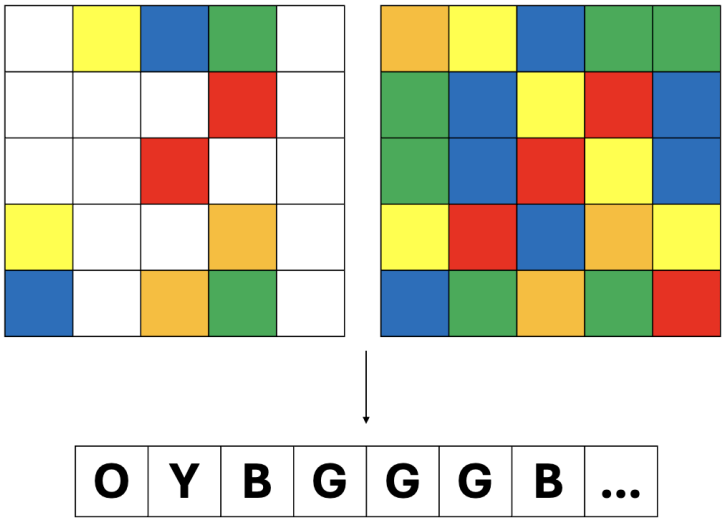
Below are our initial considerations for the fitness function design, aiming to achieve convergence towards the unique solution.

Initialize

We start by calculating the shortest path between each pair of points, and then use this information to randomly assign colors to the position on the grid. After that, any remaining positions that have not been assigned a color are filled randomly. For example, in a 5x5 grid with 5 pairs of points, if the shortest path for the red pair is 1, for green is 3, for blue is 5, for yellow is 3, and for orange is 1, we randomly allocate the required number of grid positions for each color based on the shortest path. The remaining 2 unoccupied positions are then filled with random colors. This process ensures that enough positions are allocated for each color, allowing for a valid path to be formed between the target endpoints.



- shortest path



- random example

phenotype : 2D grid random fill in different colors
genotype : transfer two dimensional grid into one dimensional array
[O,Y,B,G,G,G,B,Y,R,B,...]

Fitness funtion

1. Grid Coverage: The entire grid must be filled, ensuring that no empty spaces remain.
2. Connected Regions:
    a. A cluster is formed when color segments are adjacent (up, down, left, right). Fewer clusters are ideal. The closer the number of clusters is to the total number of color pairs (n), the better the fitness score.
    b. When the clustering counts toward the coverage of the answer, we need to check that each cluster is able to connect from the starting to the ending point, and that each point is visited exactly once. if the contraction cannot be qualified, lower down the fitness.
3. Correct Endpoints: Each path must start and end at its designated target points, ensuring that the correct color connects the corresponding start and end points.

parents selection method

We use tournament selection as the parent selection method.A few individuals (solutions) are randomly chosen from the population.The one with the highest fitness among the group is selected as a parent.This process is repeated to select two parents for crossover.

crossover

Using N-point crossover to swap segments, increasing diversity.
for example: single point crossover
[O,Y,B,G,G,G,B,Y,R,B,...] → [O,Y,B,G,G,G,B,Y,Y,R,...]
[R,R,B,G,O,B,O,Y,Y,R,...] → [R,R,B,G,O,B,O,Y,R,B,...]

mutaion

Using scramble mutation to choose a random section of grid and shuffle the colors within that section when the number of clusters is already optimal,but

a valid path has not yet been found. This method helps explore new configurations, especially when the solution is stuck in a local minimum, allowing the algorithm to escape and potentially discover better paths.

Contributions

Our approach contributes a novel perspective on solving the Flow Free problem using genetic algorithms, providing an alternative to traditional shortest path algorithms like Dijkstra's. Instead of directly calculating and optimizing the shortest path, we try to leverage genetic algorithm techniques to explore multiple potential solutions simultaneously, allowing for more diverse and flexible path-finding strategies. By using fitness functions that account for grid coverage, minimizing clusters, and ensuring correct endpoint connections, this method encourages paths that not only connect target points but also consider the overall structure and optimal filling of the grid. The genetic algorithm's capability to evolve solutions over iterations offers a more dynamic and adaptive approach compared to the rigidity of shortest path algorithms, potentially discovering solutions faster than traditional methods.

Future work

In routing problems, we often need to address issues like overlapping paths and coverage. Traditionally, solutions are approached in one of two ways: a top-down method, which looks at the entire system first, or a bottom-up method, which focuses on solving local parts before combining them. However, our proposed clustering-based approach for solving the Flow Free game could potentially be applied to routing problems as well. While we can't guarantee optimal results or convergence, exploring this new perspective is still valuable, as it may reveal innovative ways to tackle the problem.