**Flow Free Solver**

solve the flow free problem with genetic algorithm

Li-Hsuan Chang  
 Department of Computer Science National Yang Ming Chiao Tung University  
 HsinChu City 300, Taiwan  
 grace534266.cs13@nycu.edu.tw

Zi-Hui Li  
 Department of Computer Science National Yang Ming Chiao Tung University  
 HsinChu City 300, Taiwan  
zephyrlii.cs13@nycu.edu.tw

Ying-ping Chen  
 Department of Computer Science National Yang Ming Chiao Tung University  
 HsinChu City 300, Taiwan ypchen@nclab.tw

**ABSTRACT**

Genetic algorithms (GAs) are adaptive heuristic search techniques inspired by the process of natural evolution. They are widely used to solve optimization problems, particularly in scenarios with high complexity or multiple constraints, such as resource allocation, pathfinding, and scheduling. This study applies a GA to a grid-based pathfinding problem, Flow Free. Aiming to connect predefined color-coded start and end points on a grid. The problem presents unique challenges, including ensuring all paths are non-overlapping, maintaining connectivity for each color pair, and adhering to fixed start and end points. Additionally, optimizing the solution within a constrained search space while minimizing computational overhead adds further complexity. To address these challenges, the GA incorporates fixed-point enforcement, adaptive mutation, and a custom-designed fitness function to balance connectivity and resource efficiency.The main contributions of this study lie in the design of a specialized initialization strategy and a multi-step fitness function tailored for the grid-based pathfinding problem.

**Initialization Strategy**: The algorithm incorporates a targeted initialization process that ensures all predefined fixed points, including start and end locations, are preserved while optimizing the placement of intermediate path segments. This approach reduces the search space and improves the quality of the initial population.

**Fitness Function**: The fitness function evaluates solutions in three distinct steps,

* Cluster Analysis: Penalizes too many clusters
* Path Connectivity: Penalizes unconnected paths
* Overuse/underuse the Colors : Penalizes overuse colors

These contributions enable the genetic algorithm to effectively handle the constraints and complexities of the problem while converging toward optimal solutions. Project Page and code: [*https://github.com/Hsuan1079/generate\_map\_test.git*](https://github.com/Hsuan1079/generate_map_test.git)

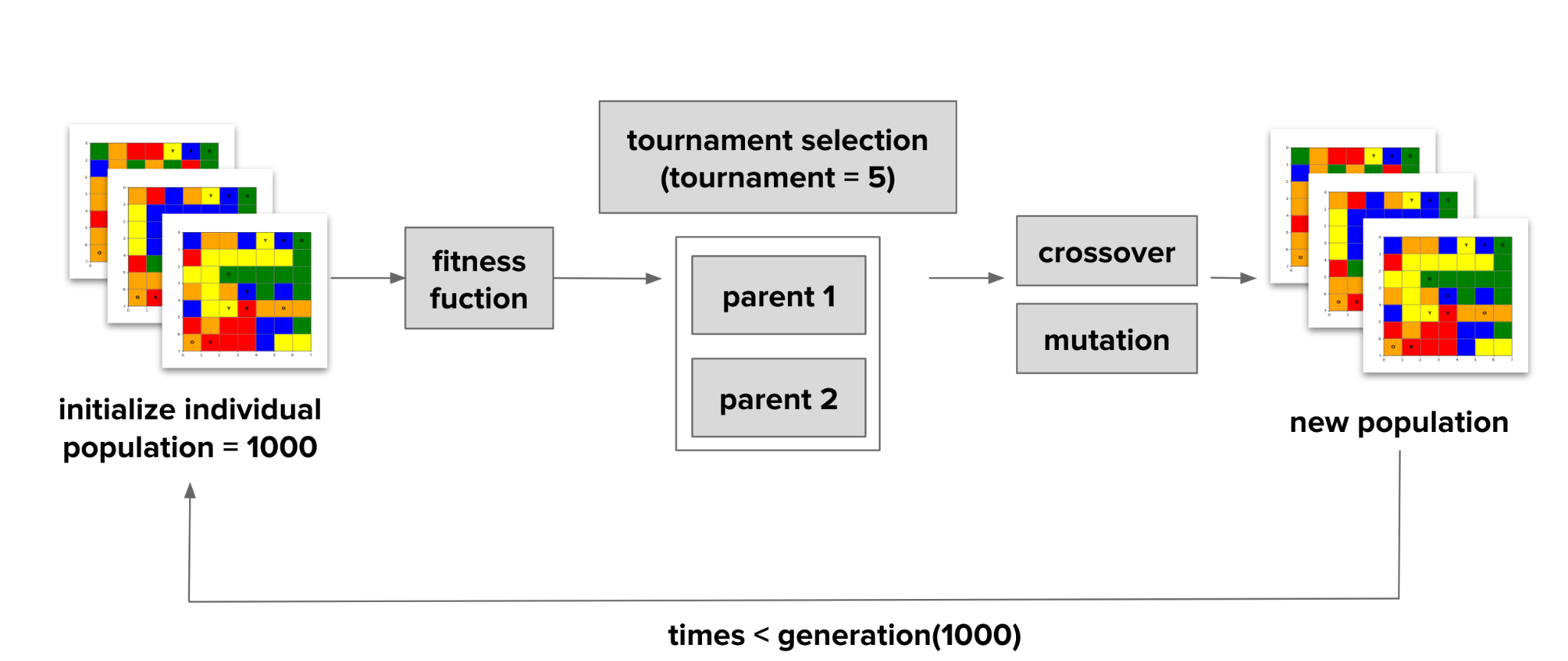


Figure 1: **Genetic algorithm framework flow**

**KEYWORDS**

Flow Free, Genetic Algorithm, GA,Graph, Puzzle, Path finding, Population, Crossover, Mutation, Fitness Function

**ACM Reference format:**

FirstName Surname, FirstName Surname and FirstName Surname. 2018. Insert Your Title Here: Insert Subtitle Here. In *Proceedings of ACM Woodstock conference (WOODSTOCK’18). ACM, New York, NY, USA, 2 pages.* https://doi.org/10.1145/1234567890

**1 INTRODUCTION**

Flow Free is an immensely popular logic puzzle game with over 100 million downloads on the Google Play Store. The game originates from the classical Numberlink puzzle, first introduced in 1897 by Sam Loyd.

In Flow Free, players are tasked with connecting pairs of colored cells on an 𝑛×𝑛grid using paths that must collectively cover all the cells. The puzzle-solving process is intricately tied to the disjoint covering paths problem, which generalizes the search for Hamiltonian paths with specified start and end points. Prior studies have shown that the problem can be modeled as finding vertex-disjoint paths within the graph representation of the grid. Unlike classical Numberlink puzzles, which allow uncovered vertices, Flow Free requires complete grid coverage, adding to the problem's complexity.

The computational challenges of solving Flow Free puzzles have been well-documented. For instance, it has been proven that solving both Flow Free and its variant, Zig-Zag Numberlink, is NP-complete[1].

In this work, we propose the use of a genetic algorithm (GA) to address the problem of solving Flow Free puzzles. As the grid size increases, finding the correct solution becomes significantly more difficult for human solvers. This characteristic makes Flow Free an ideal candidate for optimization techniques like GA, which excels at efficiently exploring large solution spaces. The GA's adaptive search capabilities make it particularly effective for identifying unique solutions to larger, more complex Flow Free grids.

The remainder of this paper is organized as follows: Section 2 outlines the implementation of the proposed GA-based approach for solving Flow Free puzzles. Section 3 presents the experimental results and a detailed discussion. Finally, Section 4 concludes the paper with insights and potential future directions.

**2 METHOD**

To solve Flow Free puzzles using a genetic algorithm (GA), we designed a framework that iteratively improves candidate solutions through a series of evolutionary operations. The process includes initializing individuals, evaluating their fitness, selecting the fittest candidates, and applying genetic operators such as crossover and mutation to generate a new population. This section details the implementation of each component.

**2.1 Initialization of Individuals**

To achieve a comprehensive initialization of the grid during the setup phase, the process begins with a thorough connectivity analysis. The algorithm scans the grid to identify target points with only one valid path available for extension. When such a path is detected, it is extended, and the newly occupied cell is marked as an edge. Subsequently, the algorithm reevaluates edge points to identify additional unique extension paths. This process of updating and extending paths continues iteratively until no further changes are possible, ensuring all fixed points are properly connected.

Once the fixed points are established, the grid initialization proceeds by generating individuals through the filling process. For each unconnected color, the algorithm employs Breadth-First Search (BFS), a systematic method that explores all feasible routes between the color’s target points. During this traversal, BFS considers only paths that adhere to the puzzle's constraints, such as avoiding overlap with existing paths and excluding already occupied cells. If BFS successfully identifies a valid connection, the grid is updated immediately to incorporate the completed path, with the corresponding cells marked as occupied and unavailable for future paths.

After attempting to connect all identified unconnected points, the algorithm addresses any remaining empty spaces on the grid. These unoccupied cells are filled by randomly assigning them to available colors, ensuring that the grid is fully populated. This final step prevents the creation of invalid individuals by guaranteeing a complete initial configuration. By carefully connecting unlinked paths and filling unused cells, this process generates diverse and feasible initial solutions for the genetic algorithm, providing a strong foundation for optimization in subsequent iterations.

**2.2 Fitness Function Design**

The fitness function is a critical component in evaluating the quality of individuals within the genetic algorithm and guiding the optimization process. It is designed to measure how closely a candidate solution adheres to the rules of the Flow Free puzzle while penalizing deviations or incomplete configurations. To achieve this, the fitness function incorporates multiple evaluation criteria, each addressing a key aspect of solution validity and optimization.

The first criterion focuses on cluster number evaluation, a metric that computes the number of distinct clusters formed by unconnected or incomplete paths on the grid. Clusters represent sections of the grid where paths remain isolated, fragmented, or disconnected, highlighting a lack of coherence in the solution. A smaller number of clusters indicates a more connected and organized solution.(Figure 2.) To encourage such connectivity, the fitness function assigns higher scores to individuals with fewer unconnected groups, thereby penalizing fragmented solutions. By prioritizing solutions with minimal clustering, this criterion ensures the algorithm not only seeks to reduce disconnected paths but also promotes a greater degree of coherence, completeness, and overall connectivity across the grid.

If the number of clusters () is greater than the desired final number of clusters (), decrease the fitness:

If , the solution is marked as complete:

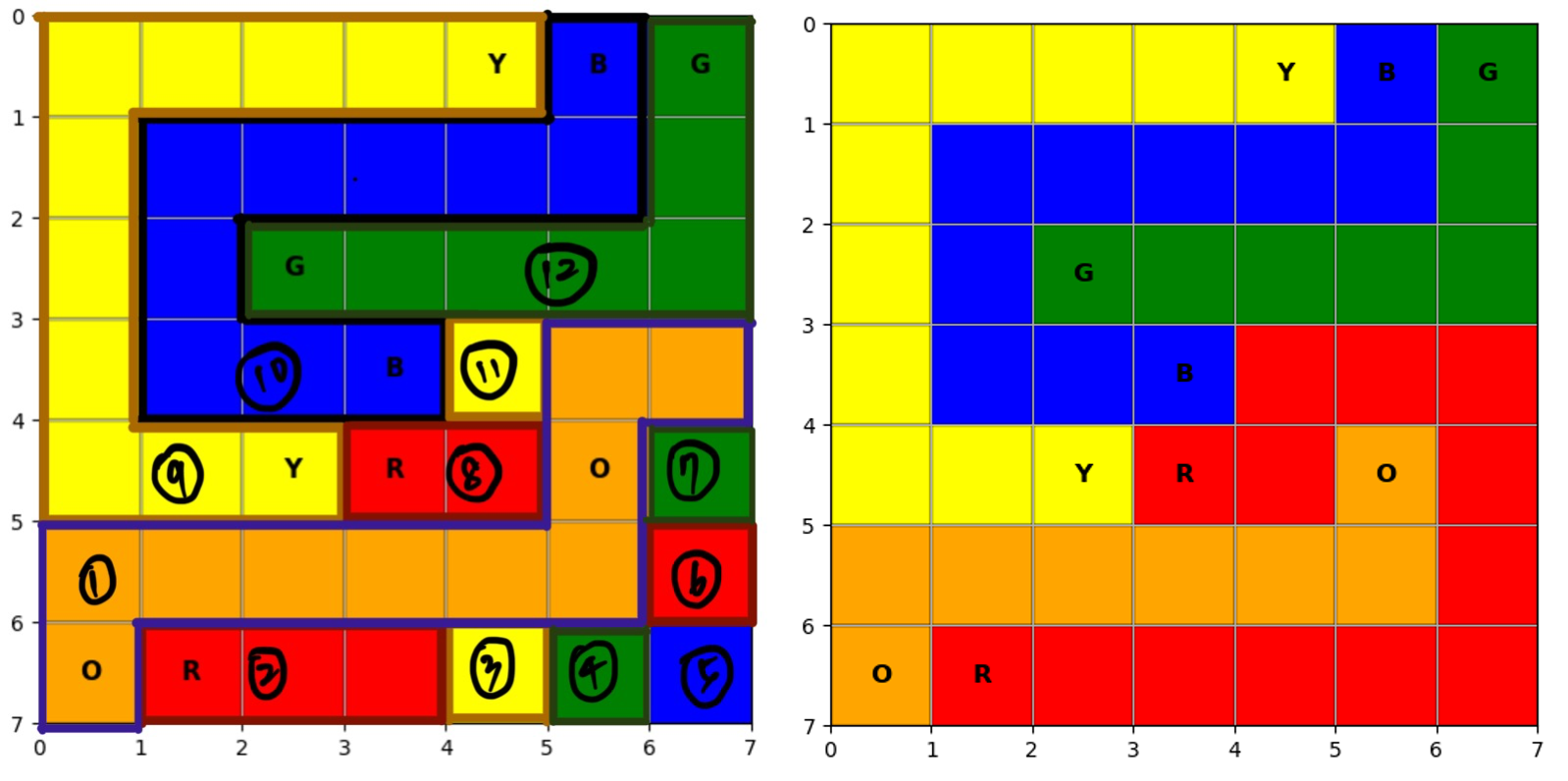


Figure 1: **Cluster numbers**, left side with 12 clusters; right side which is the final answer with 5 clusters

The second criterion involves an endpoint connectivity check. This step verifies whether the endpoints of all color pairs are successfully connected. Incomplete connections violate the fundamental rules of the puzzle and severely degrade the quality of the solution. To reflect this, the fitness function imposes significant penalties for any disconnected endpoints, encouraging the algorithm to resolve such gaps in subsequent generations. To implement the fitness function for evaluating the solution, the process begins with initializing key variables, including and , which track the quality of connections and the penalties for incomplete paths, respectively. Additionally, an empty list,, is used to keep track of the colors whose endpoints are successfully connected. For each color, the algorithm retrieves its respective start and end points and verifies their connectivity using a predefined function, such as is\_connected(start, end). If the two points are connected, the algorithm increments the by 10 and adds the color to the list. Conversely, if the connection is incomplete, a penalty of 5 is added to the .

The final criterion focuses on validating color usage and ensuring the population maintains the required color counts. While the initial population satisfies these requirements, crossover and mutation may disrupt them, leading to discrepancies. The fitness evaluation checks if each color’s path length matches the required count. Overuse or underuse of colors, such as assigning too many or too few cells to a color, violates puzzle constraints and reduces solution quality. Penalties are applied for excess length or incomplete paths, with higher penalties for underuse. Additionally, temporarily connected colors have their probability of occurrence reduced to prevent overrepresentation. This approach ensures balanced color usage and adherence to the puzzle’s specifications.

If the color already connected :

If not connected(check if it satisfy the require counts):

By combining these criteria, the fitness function provides a comprehensive assessment of each individual’s quality. It rewards configurations that closely match the puzzle's constraints while applying penalties to guide the population away from invalid or suboptimal solutions. This balanced approach ensures the genetic algorithm effectively converges toward valid and optimal solutions.

**2.3 Selection**

The selection process identifies parent individuals from the population to participate in crossover, using tournament selection as the method. In this approach, a subset of individuals is randomly selected from the population, and the one with the highest fitness score is chosen as a parent. For this implementation, a tournament size of 5 is used, meaning that each tournament randomly selects five individuals for comparison.

This method ensures that individuals with higher fitness scores have a better chance of being chosen while still maintaining diversity by giving lower-fitness individuals occasional opportunities to compete. By focusing on relative performance within the tournament group, this strategy effectively balances exploitation of high-quality solutions and exploration of the broader search space.

**2.4 Crossover**

In this step, crossover is performed to combine solutions from two parent individuals and generate new offspring. The process begins by randomly determining whether to execute a horizontal or vertical crossover(Figure 3). A splitting line is drawn accordingly, dividing the grid into two segments. These segments are then swapped between the two parents to create two new individuals. This method ensures that portions of the solution are inherited from both parents while introducing variation to explore different configurations in the search space.

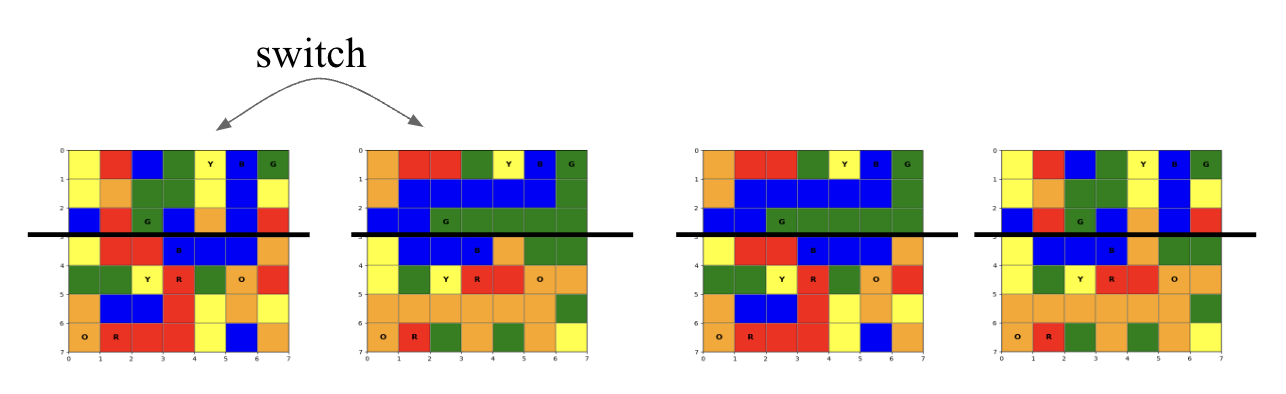


Figure 3 : **Crossover schematic diagram**

**2.5 Mutation**

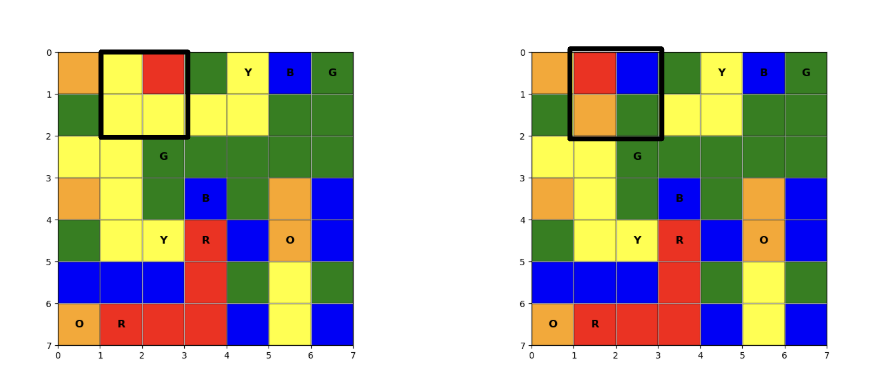
Mutation introduces randomness to maintain diversity in the population and prevent premature convergence. A random region (size = 1 to Grid\_size / 2)on the grid is selected (Figure 4), and the colors within this region are shuffled and reassigned to the same positions. However, to preserve solution integrity, fixed points—those that cannot be altered due to puzzle constraints—are left unchanged. This careful adjustment helps the algorithm refine solutions while maintaining adherence to the required rules and constraints.

Figure 4 : **Mutation schematic diagram**

**3 EXPERIMENT**

**3.1** **Test the population size**

To evaluate the impact of population size on the fitness progression, we conducted experiments with three different population sizes: 500, 1000, and 1500 (Figure 5). For each population size, the experiment was repeated five times to account for variability and randomness in the evolutionary process. The grid size was fixed at 7 across all tests. The objective was to analyze both the overall trends and the consistency of results across different trials.

Experiment Setup

* Grid Size: 7
* Population Sizes Tested: 500, 1000, 1500
* Repetitions: Each experiment was repeated 5 times.
* Generations: Evolutionary progress was tracked over several generations.
* Metrics Observed: Maximum fitness value over generations.

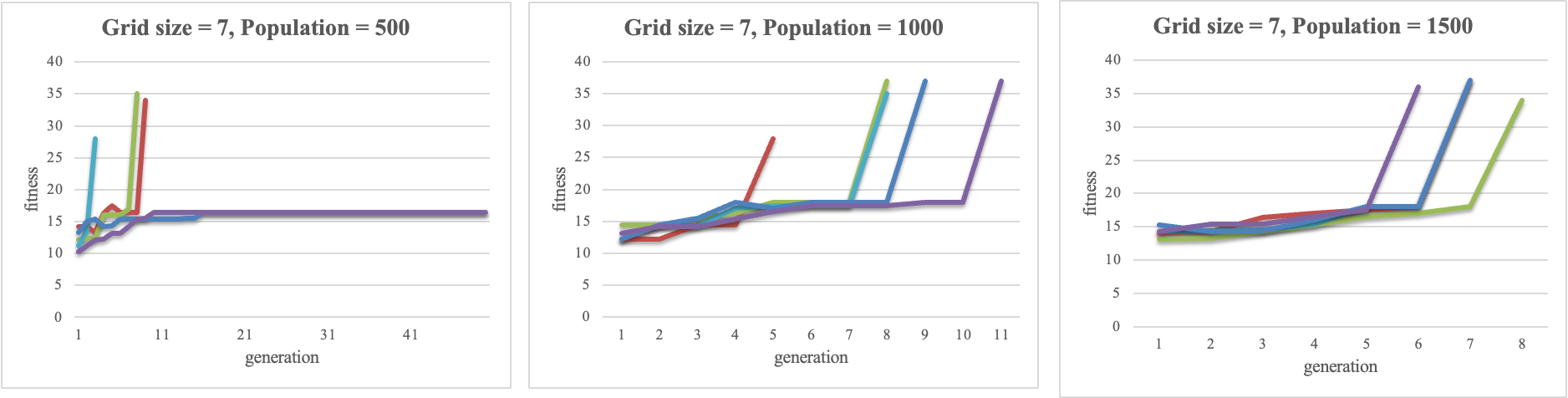


Figure 5 : **Test Result (500,1000,1500 from left to right)**

We could find that larger population sizes not only resulted in higher maximum fitness levels but also converged quickly.

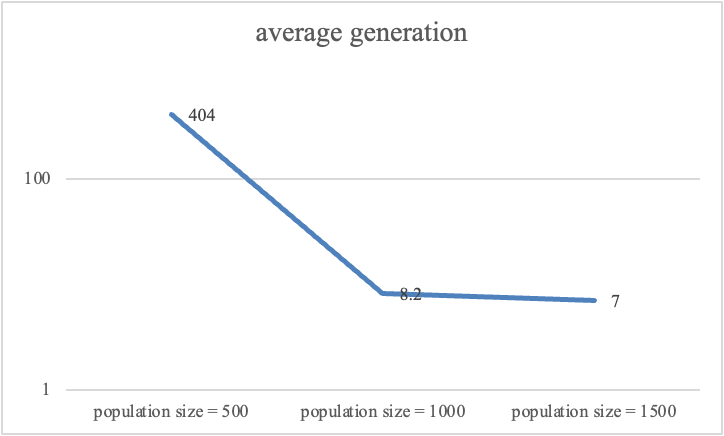
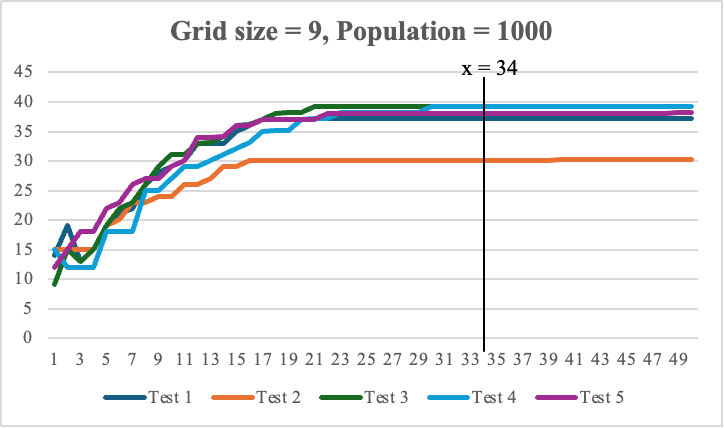


Figure 6: **compare the average generate times**

In this case, we could find that the convergence time didn’t change when increasing the population size from 1000 to 1500. The result related to population size is balanced.

**3.2** **Test with larger grid size**

To further investigate the impact of grid size on evolutionary progression and convergence, we expanded the grid size from 7x7 to 9x9. The objective was to determine how a larger search space (grid size) influences the convergence process and fitness progression.

****

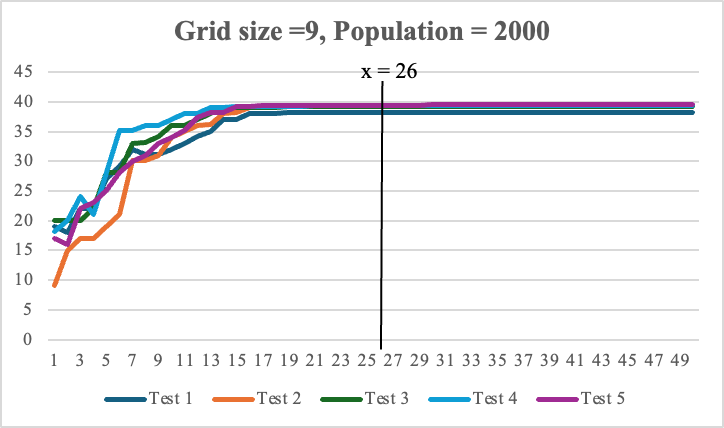
****

Figure 7 : **Gird size = 9 test with time reach local maximum**

Although the 9x9 grid size could not converge to the final solution within the given generations, the experiment demonstrates that a larger grid size requires significantly more generations and a larger population size to approach convergence compared to the 7x7 grid. As observed in **Figure 6**, the 7x7 grid achieved convergence around generation 8, while the 9x9 grid exhibited slower progression and did not fully stabilize even after 50 generations (reaching a local maximum around generation 34).While increasing the population size accelerated convergence to the local maximum (**Figure 7**), it was still insufficient to correctly converge to the final solution. This highlights the significant influence of population size on convergence results: larger grids demand proportionally larger population sizes to explore the expanded search space effectively. These findings underscore the need for carefully tuned parameters, including population size, when addressing the challenges posed by larger grid sizes in evolutionary algorithms.

**4. CONCLUSION**

Initially, we assumed that paths connecting colors would naturally emerge through the fitness function, without requiring specific initialization. However, experiments revealed that random grids often struggled to form viable paths, leading to poor performance and stagnation in the evolutionary process. In many cases, the genetic algorithm alone was unable to find the final solution within a reasonable number of generations. This highlights the importance of good initialization in improving the efficiency and effectiveness of the algorithm. By incorporating some pre-processing steps to generate better initial populations and applying post-processing techniques to refine solutions, we were able to achieve the desired final solutions. These findings underscore that well-designed initialization strategies are critical for navigating complex search spaces and overcoming local maxima in evolutionary algorithms.

**4.1** **Future Work**

One of the major challenges observed in this study is that larger grid sizes tend to stagnate at local maxima, making it difficult for the algorithm to progress toward the optimal solution. However, human judgment can often identify and construct optimal connections intuitively, even when the algorithm cannot.Future work could explore interactive approaches where human inputs are used to guide the generation process and correct the direction of convergence. For instance, incorporating user-driven feedback loops or semi-supervised methods into the genetic algorithm may help overcome local maxima and accelerate convergence. Such hybrid strategies could significantly improve performance in tackling complex and large-scale grid problems.

**ACKNOWLEDGMENTS**

We would like to express my sincere gratitude to the Evolutionary Computation course and our instructor for their invaluable guidance and support throughout this project. Additionally, we deeply appreciate the inspiration and constructive discussions with our classmates, which greatly enriched my understanding and contributed to the development of this work.

**REFERENCES**

[1] Hart, E., & McGinnis, J.A. (2022). Physical Zero-knowledge Proofs for Flow Free, Hamiltonian Cycles, and Many-to-many k-disjoint Covering Paths. ArXiv, abs/2202.04113.

[2] Flow free - apps on Google Play. url: [https://play.google.com/store/apps/details?id=com.bigduckgames.flow&amp;hl=en\_US&amp;gl=US.](https://play.google.com/store/apps/details?id=com.bigduckgames.flow&amp;hl=en_US&amp;gl=US)

[3] Flow free solution url: <https://flowfreesolutions.com/#google_vignette>