Genetic Algorithm for Domination Coloring

Shervin Emranin
Student ID: 9912223407

Introduction

This LaTeX document provides documentation for a Python script implementing a Genetic Algorithm to solve the Graph Coloring Problem. The Graph Coloring Problem involves assigning colors to the vertices of a graph in such a way that no two adjacent vertices have the same color, while using the minimum number of colors.

Input

The user is prompted to provide input for creating the graph. The input includes the size of the array (`n`) and space-separated values for each row of the adjacency matrix. The adjacency matrix is symmetric, and diagonal elements are set to 0. The input is used to represent the connectivity between vertices in the graph.
****Alert: The minimum number of vertices shoud be 4.****

Example Input

```
Enter the size of the array: 4
Enter space-separated values for row 1: 0 1 1 0
Enter space-separated values for row 2: 1 0 1 1
Enter space-separated values for row 3: 1 1 0 1
Enter space-separated values for row 4: 0 1 1 0
```

Graph Creation

The script creates an adjacency matrix based on the user input. It ensures symmetry and sets diagonal elements to 0. The resulting graph is then printed for verification.

Upper Bound for Coloring

The script calculates the upper bound for coloring based on the maximum degree of vertices in the graph.

Code Overview

The code uses a genetic algorithm to evolve a population of colorings for a given graph. The genetic algorithm involves creating individuals, performing crossovers, mutations, and selecting individuals based on their fitness. The fitness function checks if adjacent vertices have the same color and are connected in the graph.

Genetic Algorithm

The Genetic Algorithm is employed to find a valid coloring of the graph using the minimum number of colors. The algorithm includes the following steps:

1. **Initialization**: A population of individuals is created, where each individual represents a potential coloring solution.
2. **Fitness Function**: The fitness function evaluates the quality of an individual's coloring based on the number of conflicts (edges between vertices with the same color). In this modified fitness function, color-adjacency is a dictionary that maps each color to the set of vertices that are adjacent to vertices of that color. The non-dominated-count counts the number of vertices that are not dominated by any color class, which the fitness function aims to minimize.
3. **Crossover**: Parents are selected from the population and crossover is performed to create new individuals.
4. **Mutation**: Two mutation strategies (`mutation1` and `mutation2`) are applied to introduce diversity in the population.
5. **Roulette Wheel Selection**: Individuals are selected for the next generation based on their fitness using roulette wheel selection.
6. **Termination Criteria**: The algorithm terminates when either a satisfactory coloring is found or a maximum number of generations is reached.

Dominance Check

A dominance check is implemented to ensure that each color class is dominated by at least one vertex.

Algorithm Execution

The algorithm iteratively runs until a domination coloring is found or a maximum number of generations is reached. The best fitness and corresponding individual are printed at regular intervals.

Output

The script prints the best fitness, generation, and the corresponding individual at regular intervals. Once a valid coloring is found, the script displays the result with one less color than the actual coloring. If a valid coloring is not found, the script iteratively decreases the number of colors until a valid coloring is obtained or the number of colors becomes zero.

Result

The final result will indicate the minimum number of colors required to create a domination coloring for the given graph.

Dependencies

- `array` module: Imported for array operations.
- `random` module: Used for random number generation.

Conclusion

This genetic algorithm demonstrates how a domination coloring can be found for a graph. It uses a combination of crossovers, mutations, and dominance checks to evolve a population towards a proper domination coloring.