The Graph Colouring Problem: Genetic Algorithm vs. Heuristic

COZLOSCHI LUCA
BUTUC FLAVIUS

January 2024

1 Abstract

This report presents a comprehensive study of solving the Graph Coloring Problem using Genetic Algorithms (GAs) and a heuristic approach, specifically Simulated Annealing (SA). The primary objective is to compare the efficiency and effectiveness of these methods in finding optimal or near-optimal solutions. The research involved implementing both algorithms, conducting numerous trials under various parameter settings, and analyzing the results statistically. The findings indicate that while both approaches have their merits, GAs demonstrated a superior capability in consistently finding better solutions under the constraints posed by this NP-hard problem. This study contributes to a deeper understanding of combinatorial optimization techniques and their practical applications in complex problem-solving scenarios.

2 Introduction

Graph Coloring is a well-known NP-hard problem that has significant implications in various fields such as scheduling, network coloring, and register allocation in compilers. The complexity of the problem lies in assigning colors to the vertices of a graph such that no two adjacent vertices share the same color, while minimizing the total number of colors used. This report delves into the application of Genetic Algorithms (GAs) and Simulated Annealing (SA) as methods to address this challenge.

2.1 Motivation

The motivation behind this study stems from the need to explore efficient computational strategies that can handle the complexity of NP-hard problems. GAs and SA are popular choices in combinatorial optimization; however, their comparative effectiveness in graph coloring is a subject that warrants deeper exploration.

2.2 Report Overview

This report begins with a discussion of the methodologies employed, detailing the implementation of both GAs and SA. This is followed by a comprehensive experimental setup, where various parameters are fine-tuned, and the algorithms are subjected to numerous trials. The results are then statistically analyzed to draw meaningful comparisons between the two methods. The conclusion synthesizes these findings, offering insights into the algorithms' performance and suggesting potential areas for future research.

3 Methods

3.1 Overview of Genetic Algorithm

Genetic Algorithms (GAs) are inspired by the process of natural selection. They are particularly effective for solving problems where traditional optimization methods are inadequate. In this project, a GA is employed to solve the Graph Coloring Problem, an NP-hard problem in combinatorial optimization.

3.2 Implementation Details

3.2.1 Chromosome Representation

Each solution to the Graph Coloring Problem is represented as a chromosome, with each gene corresponding to a node's color in the graph.

3.2.2 Initial Population

The initial population of chromosomes is generated with the same color.

3.2.3 Fitness Function

The fitness function evaluates each chromosome based on the formula:

$$\text{Fitness} = \frac{50 \times \text{number of unique colors} + 1000 \times \text{number of conflicting edges}}{\text{number of nodes}} \tag{1}$$

This function aims to minimize the number of colors used and the number of edge conflicts.

3.3 Genetic Operations

3.3.1 Selection Method

The selection process is tournament.It uses tournament selection for choosing parents for crossover. Specifically, it randomly selects four chromosomes from the current population and then picks the better one out of each pair to be the parents for crossover

3.3.2 Crossover (Marry Function)

The crossover function, termed ‘marry‘, combines genes from two parent chromosomes to produce offspring. A pivot point is selected randomly to mix the genes, creating diverse offspring.

3.3.3 Mutation (Mutation Function)

Mutation introduces random alterations in chromosome genes, vital for exploring new solutions and maintaining genetic diversity.

3.4 Intervention Mechanism

When the algorithm experiences stagnation, an intervention mechanism is triggered:

- The fittest chromosome is identified for potential modification.
- Each gene is examined, testing different values to explore nearby solution space.
- If a new gene value improves fitness, it is retained; otherwise, the original value is restored.
- The modified chromosome is then added back to the population to foster diversity.

3.5 Termination Condition

The algorithm terminates either after reaching a maximum number of generations or if no significant improvement is observed for a predefined number of generations, indicated by the patience count.

3.6 Computational Considerations

Given the potential computational intensity of the intervention mechanism, especially for larger chromosomes, parallel processing techniques and selective gene modification strategies are considered to enhance efficiency.

4 Heuristic Algorithm Description

4.1 Overview

The heuristic algorithm implemented for solving the Graph Coloring Problem is based on the Simulated Annealing (SA) technique. SA is a probabilistic method used for approximating the global optimum of a given function. It is particularly effective in graph coloring for minimizing the number of edges that connect nodes of the same color, known as bad edges.

4.2 Algorithm Components

- **Graph Representation:** The graph is represented as an adjacency matrix, where `graph[i][j]` is `1` if there is an edge between nodes `i` and `j`, and `0` otherwise.
- **Initial Solution Generation:** A random initial coloring is generated where each node is assigned a random color from the total number of nodes in the graph.
- **Cost Calculation:** The `calculateCost` function assesses the quality of a coloring solution. It calculates the number of bad edges and computes the solution cost based on the number of unique colors used and the bad edges.
- **Solution Modification:** The current solution is slightly altered by changing the color of a randomly selected node to explore different coloring configurations.
- **Simulated Annealing Process:** The SA algorithm starts with a high temperature that gradually cools down. It probabilistically accepts new solutions based on their quality and the current temperature, allowing the algorithm to escape local optima.

4.3 Significance of Simulated Annealing

Simulated Annealing is apt for the Graph Coloring Problem due to its capability to escape local optima, a common issue in combinatorial optimization problems. It explores a broader solution space by probabilistically accepting worse solutions, especially in the initial stages. s.

5 Experimental Setup and Results

5.1 Experimental Setup

The experiments were conducted on various benchmark problems for the Graph Coloring Problem. The main parameters for the genetic algorithm were as follows:

- Population Size: `200`
- Mutation Rate: `100%`
- Maximum Number of Iterations: `2000`
- Patience Time: `70`
- Maximum Useless Shuffle: `2`

The main parameters for the heuristic algorithm were as follows:

- Maximum Number of Iterations:`30 000`
- Cooling rate: `0.96`
- Temperature: `1000`

Both heuristic and genetic algorithms were run on each problem instance to compare their effectiveness.

5.2 Results

The results of the experiments are tabulated below. The table compares the actual best-known solutions with the solutions found by the heuristic and genetic algorithms in terms of the number of colors used and the number of bad edges.

Nume problema	Nr. Noduri	Nr. Edges	Actual Answer	Heuristic Colors	Heuristic Bad Edges
Jean	80	508	10	9	11
Homer	561	3258	13	269	47
Huck	74	602	11	11	14
David	87	812	11	11	12
Fpsol2.i.l	496	11654	65	218	43
Games120	120	1276	9	25	20
Miles1000	128	6432	42	33	25
Miles250	128	774	8	28	21
Anna	138	986	11	23	23

Nume problema	Nr. Noduri	Nr. Edges	Actual Answer	Genetic Colors	Genetic Bad Edges
Jean	80	508	10	10	0
Homer	561	3258	13	15	0
Huck	74	602	11	11	0
David	87	812	11	11	0
Fpsol2.i.l	496	11654	65	65	0
Games120	120	1276	9	9	0
Miles1000	128	6432	42	44	0
Miles250	128	774	8	9	0
Anna	138	986	11	11	0

6 Comparison Discussion

The comparison between the heuristic and genetic algorithms reveals significant differences in their performance.

The heuristic algorithm tends to have a higher number of bad edges, indicating a suboptimal solution in terms of graph coloring. This suggests that while the heuristic approach is innefficient in using fewer colors, it compromises the primary objective of minimizing edge conflicts.

In contrast, the genetic algorithm consistently achieves zero bad edges in all problem instances, indicating a perfect graph coloring according to the problem constraints. Although it sometimes uses more colors than the heuristic method, it adheres strictly to the rule of no two adjacent nodes sharing the same color. This adherence to the primary objective of the problem shows the effectiveness of the genetic algorithm in finding optimal or near-optimal solutions.

The results demonstrate the genetic algorithm's superiority in solving the Graph Coloring Problem accurately. The ability of the genetic algorithm to explore a vast solution space and effectively escape local optima contributes to its success in finding solutions that satisfy all constraints.

7 Conclusions

7.1 Implications of the Results

The comparative analysis of the genetic and heuristic algorithms in solving the Graph Coloring Problem has revealed significant insights. Primarily, the genetic algorithm consistently achieved optimal or near-optimal solutions with zero bad edges across all tested cases. This demonstrates its robustness and effectiveness in adhering to the primary objective of minimizing edge conflicts in graph coloring. The ability of the genetic algorithm to explore a wide solution space and efficiently escape local optima has been clearly evidenced.

On the other hand, the heuristic approach often compromised the main objective by resulting in a higher number of bad edges, trying to get close color results. This indicates a trade-off between the number of colors used and the adherence to the graph coloring constraints. The heuristic method's reliance on probabilistic acceptance of solutions, particularly in simulated annealing, offers a rapid but less precise approach.

Overall, the results underscore the importance of selecting an appropriate algorithm based on the specific requirements of the problem at hand. While the genetic algorithm is more reliable for strict adherence to constraints, the heuristic method can offer quicker solutions with acceptable compromises in certain cases.

7.2 Future Research Directions

1. **Hybrid Approaches:** Future research could explore hybrid models that combine the robust constraint satisfaction of genetic algorithms with the speed and flexibility of heuristic methods. Such a hybrid approach could potentially offer a balance between precision and computational efficiency.

2. **Scalability and Real-World Applications:** Another direction for future work is to test the scalability of these algorithms on larger and more complex real-world graphs. Investigating their performance on real-world datasets could provide deeper insights into their applicability and efficiency in practical scenarios.

In conclusion, this study has demonstrated the strengths and limitations of both genetic and heuristic algorithms in graph coloring. The findings pave the way for further research in developing more sophisticated and efficient algorithms for combinatorial optimization problems.

References

[1] Test-Instances https://mat.gsia.cmu.edu/COLOR/instances.html

[2] Graph-Coloring-Problem https://www.geeksforgeeks.org/graph-coloring-applications/

[3] Genetic-Algorithm https://www.geeksforgeeks.org/genetic-algorithms/

[4] Simulated-Anealing https://www.mathworks.com/help/gads/what-is-simulated-annealing.html