A Critical Performance Failure Analysis of Genetic Algorithms for Graph Coloring

Jenik Gajera

December 2, 2025

Abstract

This report evaluates the performance of the genetic algorithm for solving the Graph Coloring Problem, a classic NP-hard task. The genetic algorithm is compared against highly optimized greedy algorithms such as Welsh–Powell (WP) and **DSATUR**. Across DIMACS and synthetic benchmark datasets, the genetic algorithm demonstrates severe scalability issues: a median slowdown of over **710**× relative to greedy methods, for a **zero-median improvement** in color count. Although the genetic algorithm can occasionally reduce color usage on dense random graphs, these gains are overwhelmed by crippling computational overhead. The findings confirm that the genetic algorithm is an unsuitable general-purpose method for graph coloring.

1 Fundamental Mechanics and Computational Overheads

Genetic algorithms are population-based, stochastic metaheuristics inspired by natural selection. While effective for certain optimization classes, they impose heavy computational demands in graph coloring because every generation requires full-graph fitness evaluation.

1.1 Chromosome Representation and Fitness Evaluation

- **Encoding:** Each chromosome is an array representing a vertex-to-color assignment:

$$\text{Chromosome}[i] = C(v_i)$$

- **Fitness Function:** Measures the number of conflicting edges:

$$F(C) = \sum_{(u,v)\in E} \mathbf{1}[C(u) = C(v)]$$

  A proper coloring satisfies $F(C) = 0$.

1.2 Expensive Operators and Parameter Sensitivity

The performance of the genetic algorithm is heavily influenced by parameters such as population size and mutation rate. These increase computational cost linearly or superlinearly.

Operator	**Role**	**Computational Cost and Sensitivity**
Selection	Chooses high-fitness parents.	Population size P directly multiplies the number of evaluations per generation. Larger P improves exploration but causes slowdowns.
Crossover	Combines parent chromosomes.	Crossover rate R_c governs exploration. High R_c increases recombinations, each requiring full fitness evaluation.
Mutation	Introduces diversity via random color changes.	Mutation rate R_m prevents stagnation but forces rechecking conflicts every generation.

Table 1: High-cost operators in the genetic algorithm and their impact on runtime.

---

2 Catastrophic Inefficiency: Data-Driven Analysis

A key structural disadvantage of the genetic algorithm is the **multiplicative generational cost**:

$$T_{\text{genetic}} \approx (\text{Generations}) \times P \times O(|E|)$$

In contrast, Welsh–Powell and DSATUR operate in approximately $O(|E|)$ once.

2.1 Runtime Inefficiency Metrics

Metric	**GA / DSATUR Runtime**	**DSATUR / GA Color Ratio**
Mean Inefficiency	**884.6**× slower	0.957
Median Inefficiency	**710.7**× slower	1.000
Worst Case Runtime	**2,321.5**× slower	—
Worst Case Quality	—	**0.222**

Table 2: Runtime and color usage comparison between the genetic algorithm and DSATUR.

The **median** result is the most telling: identical color counts for hundreds of times more computation.

2.2 Worst-Case: Zero Quality Gain, Maximum Penalty

Graph	**Vertices**	**GA (ms)**	**DSATUR (ms)**	**Ratio**
david	87	211.3	0.091	**2,321.5**×
planar_309_4	309	225.0	0.148	**1,519.9**×
planar_161_5	161	126.8	0.087	**1,457.2**×
myciel3	11	10.7	0.008	**1,334.2**×
homer	561	558.7	0.421	**1,327.0**×

Table 3: Extreme GA slowdowns relative to DSATUR, despite identical color counts.

---

3 Suitability and Practical Guidance

3.1 When the Genetic Algorithm Should Be Avoided

- **Trees, Bipartite Graphs, Grid Graphs ($\chi(G) \leq 4$):** Greedy methods solve these instantly. GA introduces $10^3\times$ slowdowns.
- **Complete Graphs:** Chromatic number is trivial; GA is pointless.

3.2 Rare Cases Where GA May Help

- **Dense Random Graphs (DSJC, Erdős–Rényi):** GA may reduce color count by 5–10%, but at massive computational cost.

---

4 Illustration: Mycielski Graph M_4

- **Vertices:** 23
- **Edges:** 71
- **Chromatic Number:** $\chi(G) = 5$
- **GA Runtime:** 18.604 ms
- **DSATUR Runtime: 0.019** ms
- **Slowdown: 979**× slower

Even small but chromatically difficult graphs show catastrophic slowdowns.

5 Conclusion

The genetic algorithm is fundamentally misaligned with efficient graph coloring. Its repeated full-graph fitness evaluations and parameter sensitivity cause it to run hundreds to thousands of times slower than greedy heuristics such as DSATUR, while producing almost identical results. Except for rare dense random graphs where runtime is irrelevant, the genetic algorithm should not be used for graph coloring.