Trajectory Methods and Genetic Algorithms in Graph Coloring Problem

Georgiana Diana Ojoc
Vlad Corjuc

January 9, 2020

Abstract

In this paper we review the application of Simulated Annealing and a genetic algorithm on the graph coloring problem, comparing the results with the standard DIMACS benchmark tests. The genetic algorithm described here uses more than one mutation and crossover methods, depending on the fitness of the last generations. The results provided by these two methods are similar in most cases, but only the genetic algorithm we presented here guarantees a valid coloring for each example of chosen graph.

1 Introduction and Motivation

The Graph Coloring Problem asks for a coloring of each vertex of a graph with as few colors as possible such that no adjacent vertices have the same color. We are interested in the Graph Coloring Problem as many practical problems which involve the partitioning of objects into disjoint classes can be solved using graph coloring.
These problems range from everyday life to specific technologies. One textbook example is timetable scheduling, where a number of lectures are to be assigned to different time slots. Some of the lectures might not be allowed in the same time slot because they are held by the same teacher or take place in the same room. In a graph where these conflicting lectures are connected, the colors of a coloring represent the time slots of a valid assignment.
Graph coloring is used in practice by processors for register allocation. Each register can hold one value at a time and a value has a timespan in which it is accessed. Processors construct an interference graph in which the loadable values are connected if their timespans overlap. A coloring of the interference graph yields an assignment of values to registers with each color representing one register.
Besides these practical applications, graph coloring also finds use in a number of mathematical problems.

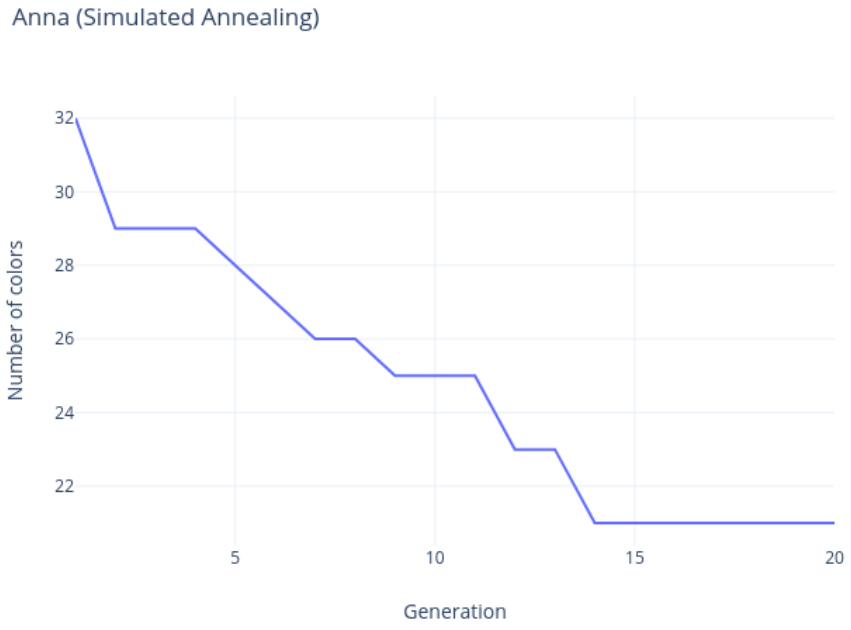
Generally, the Graph Coloring Problem is NP-complete and we currently do not know any efficient algorithm for large graphs. The wide range of applications along with its combinatorial complexity elevates the Graph Coloring Problem to one of the most famous and most researched problems in graph theory.
Probably the most well-known result related to graph coloring is the four color theorem. Initially proposed as a conjecture, it states that four colors suffice to color an arbitrary planar graph. The theorem gained much publicity not only because of many false alleged proofs, but also because it was the first to be proved extensively computer-aided. While we are able to find colorings using only four colors for any planar graph in quadratic time, it is already NP-hard to decide whether three colors suffice for the same graph.
Since the Graph Coloring Problem is NP-hard, until now there are not known any deterministic methods that can solved it in polynomial time. So non-deterministic algorithms have been built to solve it and two of them are Simulated Annealing and Genetic Algorithms, that have been used on the Graph Coloring Problem giving good results. However, these methods are not complete algorithms and they not always get the optimal solution.

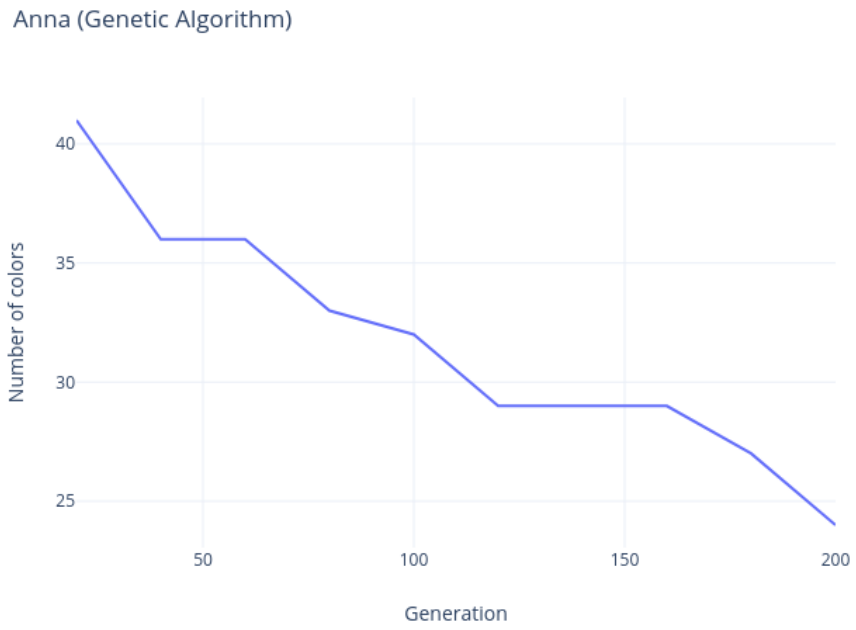
2 Method

A result is represented by the number of different numbers in the solution vector, which consists of nodes and their corresponding colors.
In order to evaluate the candidates in Simulated Annealing and to decide which individuals should be kept in the Genetic Algorithm, we used the fitness function described in the last reference: $fitness[i] = -(100 * conflictsNumber + 50 * (colorsNumber(i) - 1)/edgesNumber$, where $conflictsNumber$ represents the number of edges between nodes with the same color in the vector.
A new random color for a specific node was chosen from the set $\{1, ..., \Delta(G)+1\}$, excluding the colors of the nodes adjacent to it, where $\Delta(G)$ is the degree of the vertex with the greatest number of edges incident to it.
The Genetic Algorithm uses two Mutation methods, one in which only the node with the most conflicts has its color changed, and one which changes the color of every node that has conflicts. It also uses two Crossover techniques, Multi Point and Conflict Elimination.
In order to improve the Genetic Algorithm, we used the helper described in the last reference when for 50 generations the fitness of the current chromosome has not changed. This function tries to change 40% of the genes of the most fit chromosome, using a probability of 0.4.
The algorithms have been run 30 times for 13 examples of graphs. Simulated Annealing used 20 iterations, the temperature was set to 1000 and the cooling rate was equal to 0.1. The Genetic Algorithm used 200 generations, each chromosome started with 100 genes and the constants for Mutation, Crossover and Tournament Selection where 0.01, 0.3 and 5.

3 Results

Method / *File*	Information			Simulated Annealing		Genetic Algorithm	
	Number of nodes	**Number of edges**	**Expected number of colors**	**Minimum number of colors**	**Maximum number of colors**	**Minimum number of colors**	**Maximum number of colors**
Anna	138	986	11	21	24	24	36
David	87	406	11	19	23	29	30
Games 120	120	638	9	12	14	13	14
Homer	561	1629	13	80	83	71	80
LE 450 5	450	5714	5	36	39	33	39
LE 450 15	450	8168	15	86	88	82	88
LE 450 25	450	8260	25	86	88	84	86
Miles 500	128	1170	20	33	34	34	35
Miles 1500	128	5198	73	86	87	84	88
My Ciel 5	47	236	6	6	7	10	12
My Ciel 7	191	2360	8	46	48	39	45
Queen 8 12	96	1368	12	28	29	28	30
Queen 13 13	169	6656	13	44	46	41	46





4 Interpretation and Conclusion

There are examples of graphs on which the algorithms had good results, but in most cases we encountered big differences between the benchmark tests and our values, in our disadvantage. Even though Simulated Annealing had similar results to the Genetic Algorithm we chose, the colorings it returned had conflicts, while the second method always generated valid solutions. We also tried to prevent the incest between two individuals in the presented Genetic Algorithm, but this technique did not bring any improvements.
Therefore, the Genetic Algorithm we used is somehow better than the Simulated Annealing we implemented, but both of them can be improved by parallelizing the algorithm to run on the GPU instead of the CPU.

References

1. `https://mat.gsia.cmu.edu/COLOR/instances.html`
2. `http://ceur-ws.org/Vol-841/submission_10.pdf`
3. `http://algo2.iti.kit.edu/schulz/collection/thesises/williams.pdf`
4. Using-Genetic-Algorithm-To-Solve-The-Graph-Coloring-Problem-Ahmed-Zarie.pdf