A comparative study of Genetic algorithm, Simulated Annealing and their hybrid for graph coloring problem

Anik Bhattacharya
Department of Information Technology
Heritage Institute of Technology
Kolkata, India
anikbhattacharya93@gmail.com

Pankhudi Jain
Department of Information Technology
Heritage Institute of Technology
Kolkata, India
pankhudi.0804@gmail.com

Anindya Jyoti Pal
Department of Information Technology
Heritage Institute Of Technology
Kolkata, India
aj.pal@heritageit.edu

Abstract—**In this paper, we have used two heuristic algorithms, namely Genetic Algorithm and Simulated Annealing and hybridized them with each other for solving the graph coloring problem. Although the existing pure algorithms provide us with near optimal solutions for different graphs, the new hybridized algorithm shows improved results over the existing ones. A comparative study of these heuristic algorithms with their hybridized version to solve a large number of benchmark graphs, namely the DIMACS, shows an improvement in terms of the near optimal solutions achieved when the hybridized algorithms are used.**

Keywords— Planar Graph Coloring, Genetic Algorithm, Simulated Annealing, hybrids, DIMACS.

I. INTRODUCTION

The graph coloring problem is a topic that has been the subject of intense research by computer scientists and mathematicians for decades. The graph coloring problem requires us to color the vertices or nodes of a graph with the minimum number of color possible, subject to the constraint that no two adjacent vertices be colored with the same color.

What makes the graph coloring problem so intriguing is the fact that it is an NP complete problem [1]. By definition, NP complete are the problems which fall in the categories of NP and NP hard problems. The problems that lie in the intersection of NP and NP hard are said to fall under the NP complete category. [2] The difficulty while trying to solve these type of problems is that it is not known whether a solution of the problem exists or not. Also the time required to solve such problems increases rapidly with the size of problem. Hence higher the number of vertices of the graph more will be the time it will take to reach an optimum solution.

The increasing time complexity with the increase in problem size, due to the NP completeness property of the problem makes heuristic algorithms a better solution for these kinds of problems than deterministic algorithms. Although they produce near optimal solution (and not the optimum solution) in most cases, the reduced time complexity makes them attractive for obtaining minimum color required for graphs with large set of vertices and edges.

Graph coloring problem has many practical applications including, but not limited to[6] timetable scheduling [3], assignment of mobile radio frequency [4], map coloring [4] and register allocation [5].Due to the challenging nature of the problem and the wide range of applications it has, we decided to hybridize two heuristic algorithms that not only finds a near optimal solution but also takes minimal amount of time to do that.

Evolutionary algorithms are population based heuristic algorithms which generate random probable solutions, and then modifies that to reach an optimum value. Usually, evolutionary algorithms use techniques which are inspired from the biological concepts of evolution, and uses techniques like population generation, recombination, mutation etc. to generate a fitter population which is analogous to the nearest optimal result possible.

Let us take an undirected graph G= (V, E). The vertices v are partitioned into k-subsets $C_{i,}$ i= {1…k) such that no two adjacent vertices have the same color, i.e., are in the same subset. The graph coloring problem is to find the smallest value of k possible. The smallest possible value of k is known as the chromatic number $\chi(g)$ of graph G. If two adjacent vertices a and bare colored with the same color, vertices a and b are said to be conflicting vertices, the edge {a, b} is called a bad edge, and x is called a conflicting color. The k-coloring of a graph is said to be legal if there are no bad edges. The graph coloring problem is to determine the smallest value of k for which we will find a legal k-coloring of G.

In this paper, we have solved the graph coloring problem using two such algorithms, namely Genetic Algorithm (GA) and Simulated Annealing (SA) and then compare the results obtained with the results of the newly devised hybrid algorithm. Strictly speaking, simulated annealing does not fall under the category of Evolutionary algorithms as it does not use mechanism of evolution but it still has striking similarities with the fundamental concept of these evolutionary algorithms.

II. Previous Works

As already mentioned, the graph coloring problem has been a topic of research for a long period of time and hence over the years, many research papers have been published. A brief review of the previous work done is given below.

Biman Ray, Anindya J Pal, Debnath Bhattacharyya and Tai-hoon Kim [6], have implemented genetic algorithm using a special kind of mutation known as multipoint guided mutation. The algorithm using this technique turns out to be more efficient than the simpler GA. Their work was published in 2010 and all the earlier work on genetic algorithm to solve graph coloring problem is discussed there. The results given in this paper clearly shows marked improvements in reaching a near optimal solution. They have tested their results on benchmark DIMACS graphs which confirm the efficiency. A new operator is introduced in this paper which is called double point Guided mutation and the algorithm that is proposed here changes the performance of the conventional GA very drastically. The basic job of the new mutation operator is to reduce colors from multiple locations. Two mutation operators are used here. One, the conventional mutation that does not guarantee an improved solution but helps the solution to get out of local minima and second, the multi point guided mutation that aims at reducing the number of color classes.

The main aim of their work was to get a solution to the complex graphs in reasonable time and they have been successful in doing so.

Musa M. Hindi and Roman V. Yampolskiy [7] in 2012 have incorporated the theory of Wisdom of artificial crowd when the result of genetic algorithm does not improve for a fixed number of iterations. They have used two mutations and crossover operators. They use the operators depending on the chromatic numbers of the chromosome.

Artificial crowd is being used here only in that situation when after a fixed number of iterations the algorithm doesn't reach a solution then in that case this technique is used. The technique is basically used to decide and remove the bad edges so that a legal solution is reached.
The algorithms in this paper were tested on DIMAC benchmark graphs and the paper was concluded by showing better results for a few more graphs than the earlier work. However, results of implementing the algorithms on bigger graphs were not discussed..

In 2012 Anindya Jyoti Paul, Biman Ray, Nordin Zakariaa and Samar Sen Sarma [8] worked on the Simulated Annealing algorithm and also introduced a modified version. The modified algorithm has a new operator known as the random change operator. This paper has described in details all the work done previously on solving Graph coloring problem using Simulated Annealing.
In the modified algorithm, instead of randomly generating the neighboring solutions, a new operator is used that generates the solution under some guidance and works at a particular point of the solution. This is one of the main reasons why the modified version of the algorithm gives a better solution as the solution generation is more guided than being random as found in conventional Simulated Annealing. A repair function is also used that checks whether the solution generated is valid or not and if it found to be invalid, then necessary changes are done to validate the solution .Although the Modified algorithm works well for most of the graphs but there are a few graphs for which the algorithm does not to give the optimal colors.

Szymon Łukasik, Zbigniew Kokosiński and Grzegorz Świętoń [9] strived to solve the graph coloring problem by implementing parallel simulated annealing. The master-slave model of the parallel algorithm was what they used. The paper mentions all the methods and parameters the authors have used while implementing the algorithm. The paper includes a comparison of the results obtained after implementing parallel genetic algorithm to solve the same problem of graph coloring. Multiple processors were used which worked concurrently in a particular sequence and accepted the solution obtained after a fixed amount of time. The algorithms were tested on a few DIMAC graphs. The authors showed that the efficiency of the algorithm depends on the initial solution chosen and the cooling schedule. The comparison result with parallel genetic algorithm showed that none is really a superior algorithm and that there was scope to combine the two algorithms to produce a better hybrid that would benefit from both Genetic Algorithm and Simulated Annealing.

That is exactly what we have tried to do in our work. Combine two equally efficient algorithms to produce a hybrid version.

III. Evolutionary Algorithm

Evolutionary algorithms have been used in artificial intelligence to solve complex problems. They are used to design computational techniques to mainly solve optimization problems; they do this mainly by imitating the biological process of Evolution. A brief description of the Algorithms used is given below.

A. Genetic Algorithm

It is one of the most well known evolutionary algorithms whose working principle is inspired from the fundamental concepts of genetics, i.e., selection, mutation, crossover etc. First of all in Genetic algorithm (GA), we need to encode the probable solution to a specific problem in a chromosome like structure. A pool of such chromosomes is created, and their respective fitness is calculated. The main aim after that is to improve the chromosome through crossover and mutation and take the fittest population to the next generation.

For Graph coloring we have implemented the algorithm by initializing chromosomes of the same length as the number of vertices in the graph. Values in these chromosomes will be randomly assigned colors to each vertex. For example if there is a graph with 6 vertices a sample chromosome will look like this:

1	5	3	2	4	6

Fig 1: Chromosome Example

After generating a specific number of such chromosomes, their validity is checked, i.e., it is made sure that no two adjacent vertices of the graph have the same color, and if any such conflict is found where the chromosomes are not valid, then the conflicting colors in one of the vertices is replaced by a suitable color.

Once the valid pool is generated, functions like mutation and crossover to generate a fitter population is applied to the chromosome. Fitness in this case is calculated by the number of distinct colors used in the chromosome. Like in the example mentioned above the fitness of that chromosome are 6.

A special type of mutation is used in this case. It is called multi-point special mutation [6]. Here we try to reduce the number of classes by replacing one particular color at two or more places by one of the other used colors. [6]. Then we check whether the new chromosome generated is valid or not. If it is then after calculating the fitness it is introduced in the pool or else the process is repeated again.

Crossover used here is a simple one; a two point cross over method is used. Two chromosomes are selected, and a random point is chosen. Information after that point is interchanged to generate two new chromosomes. Their fitness is calculated and they are included in the pool. The algorithm to explain how the graph coloring problem was solved is given below.

```
INPUT: A simple graph represented by an adjacency list or an adjacency matrix.
Step 1) Create the pool of initial chromosomes. A chromosome iscreated by randomly assigning colorsto the vertices of the graph.
Step 2) Predetermine crossover and mutation probability.
Step 3) Calculate the validity of such chromosomes to make sure the chromosomes have a valid coloring by eliminating bad edges.
Step 4) Calculate the fitness of each of the valid chromosome.
Step 5) Continue till the termination condition is met.
    Step 5.1) Perform Crossover with a probability.
    Step 5.2) Ensure validity of chromosome by removing bad edges.
    Step 5.3) Calculate the fitness of the newly generated chromosomes and add them in the pool.
    Step 5.4) Perform mutation with a probability.
        Step 5.4.1) Either perform an extensive mutation by randomly selecting a color class and replacing it with other colors for all vertices that it was previous colored with
        Step 5.4.2) Or perform a normal mutation by selecting a single vertex and recolor it with a different color to ensure that the solution is not stuck at a local minima.
    Step 5.5) Ensure validity of new chromosome by removing bad edges.
    Step 5.6) Calculate the fitness and introduce new chromosome in the pool.
    Step 5.7) Sort the chromosome pool according to the fitness and select the best few from the pool for the next generation.
    Step 5.8) Continue till the fitness value of the best possible solution remains unchanged for a significant number of iterations to find and optimal solution.
```

B. *Simulated Annealing*

The main philosophy behind this concept is derived from metallurgy, and is mainly used in finding the global optima of a problem.

In simulated annealing one candidate solution is randomly generated and then it is improved for a certain number of iterations. As already mentioned, it is a simpler version of GA where the size of the population is 1.

For graph coloring problem, we have generated a random solution by assigning random colors to the vertex. We ensure the validity of the solution here as well. Then we initialize a Temperature which is set at a very high value initially, and it keeps on decreasing with each iteration. After the initial solution is generated, the solution is improved by minimizing the number of colors used. The fitness of the new solution is calculated and compared with the older one. If the new solution is better it is accepted otherwise it is accepted with a probability. This probability is calculated based on the formula: $\exp[(-(e\text{`}-e))/T]$.

The algorithm is explained below.

```
INPUT: A simple graph represented by an adjacency list or an adjacency matrix.
Step 1) Generate an initial candidate solution. The solution will be random colors assigned to the vertices.
Step 2) Calculate the validity of such chromosomes to make sure the chromosomes have a valid coloring by eliminating bad edges.
Step 3) Initialize initial temperature, T with a high value.
Step 4) Continue till the termination condition is met.
    Step 4.1) Decrease T with a fixed value
    Step 4.2) Try and improve the candidate solution by replacing a color used with another used color and generate a new candidate solution.
    Step 4.3) Ensure validity of chromosome by eliminating bad edges.
    Step 4.4) Calculate the fitness.
    Step 4.5) Compare the fitness of the new solution with the previous solution.
    Step 4.6) If the new solution is better it is accepted.
            Else, the solution is accepted with a probability, depending on the value of T. Worse solutions are accepted initially with high tolerance, which later keeps decreasing, thereby rejecting bad solutions during the final stages.
    Step 4.7) Continue till the fitness value of the best possible solution remains unchanged for a significant number of iterations.
```

C. Hybrid Algorithm

In order to solve the Graph Coloring Problem with more efficiency, we have attempted to develop a hybridized algorithm using Simulated Annealing (SA) and GA.The hybrid algorithm attempts to combine the two algorithms by creating a random candidate solution using simulated annealing and introducing it in the pool of chromosomes during the implementation of genetic algorithm.

We perform simulated annealing on the problem for a reduced number of iterations. It is performed instead of the mutation operation of the genetic algorithm. Apart from that one change, we allow the genetic algorithm to run uninterrupted, and every time mutation is needed to be performed, we use SA instead and then introduce that solution inside the pool.

```
INPUT: A simple graph represented by an adjacency list or an adjacency matrix.

Step 1) Create the initial pool of chromosomes. A chromosome will be a random assignment of colors to the vertices of the graph.
Step 2) Calculate the validity of such chromosomes to ensure that the chromosomes have a valid coloring by eliminating bad edges
Step 5) Calculate the fitness of each of the valid chromosome.
Step 6) Continue till the termination condition is met.
    Step 6.1) Perform Crossover with a probability.
    Step 6.2) Ensure validity of chromosome by eliminating bad edges.
    Step 6.3) Calculate the fitness of the newly generated chromosomes and add them in the pool.
    Step 6.4) Perform SA instead of Mutation (This is performed only whenmutation was supposed to be performed with a probability)
        Step 6.4.1) The solution generated after a certain reduced number of SA iteration is introduced in the pool after calculating the fitness.
    Step 6.3) Sort the chromosomes according to the fitness and choose the first few for the next generation.
    Step 6.4) Continue till the fitness value of the best possible solution remains unchanged for a significant number of iterations.
```

IV. Results and Discussions

We have developed the algorithms and tested them on DIMACS graphs [10]. They are benchmark graphs on which testing of graph algorithms can be done.

DIMACS sponsor challenges to determine the performances of algorithms on several problems like graph coloring, travelling salesman, shortest path problem and the like.

We coded our algorithms in ANSI C and the OS we used were FEDORA 21 and Mac Os X Yosemite. The processor that was used was 1.4 GHz Intel core i5. As for the results we obtained, we can see that in majority of the graphs the hybrid algorithm has found the near optimal solution. We have calculated the time needed by the respective algorithms and the chromatic number for these graphs that are generated by these algorithms and we have made the following observations

1) The hybrid algorithm, when applied on the graphs mentioned in the table below, performs better than Simulated annealing roughly 45.8% of the time, and better than Genetic algorithm roughly 41.6% of the time.
1) Even though the time taken by the hybrid algorithm may be greater than the generalized versions of the algorithms, they have a greater efficiency when it comes to reaching the near optimal solution.
2) Depending on the nature of the graph, the efficiency of the algorithms varies. For those belonging to the category of myciel or book graphs, the minimal solution is reached but for those belonging to the category of queen and mul_sol, all the algorithms have reached a near optimal solution and not the minimal one.

The results table of the chromatic number of the graphs and the results obtained is given below.

TABLE I: Results Observed

Name of the graph	Actual Chromatic Number	Chromatic Numbers (time in seconds)		
		SA	*GA*	*Hybrid*
DSJC125.1	5	9(0.18)	8(0.50)	8(6.55)
LE450_25B	25	28(15.89)	28(12.50)	27(27.30)
FPSOL2.I.1	65	65(45.31)	65(17.52)	65(17.46)
FPSOL2.I.3	30	33(29.74)	32(27.77)	30(32.24)
MULSOL.I.1	49	49(0.83)	55(1.51)	49(15.08)
MULSOL.I.5	31	36(1.36)	36(1.60)	33(8.49)
ZEROIN.I.1	49	49(1.88)	50(2.54)	49(17.45)
ZEROIN.I.2	30	30(2.18)	32(4.41)	30(19.20)
ZEROIN.I.3	30	30(0.93)	30(1.46)	30(12.42)
ANNA	11	11(0.80)	11(0.89)	11(7.26)
DAVID	11	11(0.86)	11(0.22)	11(6.32)
INITHX.I.1	54	57(121.94)	55(588.48)	55(571.33)
JEAN	10	10(0.07)	10(0.17)	10(9.94)
GAMES120	9	10(0.18)	9(0.45)	9(5.18)
MILES1500	73	77(0.63)	73(1.45)	73(8.70)
MILES250	8	10(0.19)	10(0.51)	8(9.59)
MILES500	20	22(0.27)	22(0.95)	20(7.11)
QUEEN5	5	8(0.01)	8(0.02)	6(1.10)
QUEEN6	7	9(0.01)	9(0.04)	8(2.72)
MYCIEL3	4	4(0.01)	4(0.01)	4(0.17)
MYCIEL4	5	5(0.05)	5(0.07)	5(0.92)
MYCIEL5	6	6(0.36)	6(0.45)	6(2.12)
MYCIEL6	7	7(1.69)	7(1.83)	7(11.69)
MYCIEL7	8	8(0.56)	8(1.27)	8(10.84)

Graph to show the results from the three algorithms is given below.

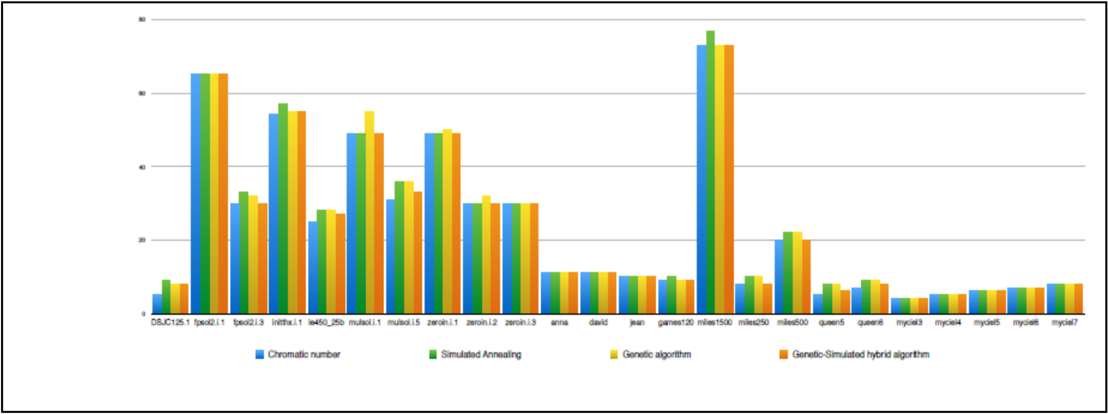

Fig 2: Graphical representation of chromatic numbers of the three algorithms.

V. Conclusion and Future Scope

In this paper, we considered hybridizing different heuristic algorithms with each other, a method that has not been previously tested for solving graph coloring problem and implemented them on a set of benchmark graphs, more specifically DIMAC graphs. Our experimental results show that hybridization of genetic algorithm with simulated annealing improves the value of k (minimum number of color required) but increases the time complexity. This particular hybridized algorithm can be implemented to achieve near optimal solutions for large problem set, provided one can tolerate the high time complexity required.

The work we have done needs further development and there is a lot of scope for improvement. We can test the hybridized algorithms on other classes of graphs**, t**ry and optimize the various operators in each algorithm more so as to get the best possible results for all family of graphs. We will also have to improve the time complexity of the algorithms. Finally, we would like to test the algorithms in practical situations by implementing them to solve industrial problems.

Acknowledgment

The authors gratefully acknowledge the works of all the other researchers whose works have been mentioned earlier. They have provided us with many useful information and tips to complete our work.

References

1. Gilles Brassard, Paul Bratley, Algorithmics: Theory and Practice. Englewood Cliffs, New Jersey : Prentice hall, 1989.
2. https://en.wikipedia.org/wiki/NP-complete
3. D. C. Wood, “A technique for coloring a graph applicable to large scale time-tabling problems”, Computer Journal, vol. 12,1969, pp. 317-319.
4. http://www.geeksforgeeks.org/graph-coloring-applications/
5. F. C. Chow and J. L. Hennessy, “Register allocation by priority based coloring”, Proceedings of the ACM Sigplan 84 symposium on compiler construction New York,1984, pp. 222-232.
6. Biman Ray, Anindya J Pal, Debnath Bhattacharyya , and Tai-Hoon Kim, “An efficient ga with multipoint guided mutation for graph coloring problems”, International Journal of Signal Processing, Image Processing and Pattern Recognition 3, no. 2 ,2010,: pp. 51-58.
7. Musa M. Hindi, and Roman V. Yampolskiy , “Genetic Algorithm Applied to the Graph Coloring Problem”, Proc. 23rd Midwest Artificial Intelligence and Cognitive Science Conf, 2012, pp. 61-66.
8. Anindya Jyoti Pal, Biman Ray, Nordin Zakariaa, and Samar Sen Sarma, "Comparative performance of modified simulated annealing with simple simulated annealing for graph coloring problem." Procedia Computer Science 9 ,2012,pp: 321-327.
9. Łukasik, Szymon, Zbigniew Kokosiński, and Grzegorz Świętoń, "Parallel simulated annealing algorithm for graph coloring problem." , *Parallel Processing and Applied Mathematics*,Springer Berlin Heidelberg,2008 , pp. 229-238.
10. http://mat.gsia.cmu.edu/COLOR/instances.html#XXSGB