Using Genetic Algorithms To Solve The Graph Coloring Problem

Ahmed M. Zarie
The Computer Science and Engineering Department
The American University in Cairo
Cairo, Egypt
zare3@aucegypt.edu

1- Introduction

The graph coloring problem is the trial to color each node U in a graph G with a certain color Ki, such that no two adjacent nodes have the same color Ki. Also, the number of total number of colors, |K| should be minimal.

This problem is a very interesting one because it can model several different problems, and is regarded to be an NP-complete problem.

For my approach, I will be using Genetic Algorithms to find the solution for this problem. Also, I will be introducing a new approach (The Helper) to minimize population saturations.

2- Genetic Parameters

For this section of the paper, I will be talking about all the algorithms and parameters used for the training.

2- a- Chromosome

Each chromosome in my population represents the nodes in the graph. i.e. Each gene g = chromosome[i] is the value of the color assigned to node n = graph[i]

2-b- Population Initialization

Each chromosome in the population is assigned to have all of its genes of the same value (color)

2-c- Crossover

```
Crossover(parent p1, parent p2)
        pivot = RAND_IDX%p1.length()
        child1 = upperLeftParent1TillPivot
      +         bottomRightParent2TillPivot
        child2 = bottomLeftParent2TillPivot
        +       upperRightParent1TillPivot
return pair (child1,child2)
```

2-d-Mutation

```
Mutate(Chromosome ch)
  For each edge connecting gene[u]->gene[v]
      If (gene[u] == gene[v])
              C = RANDCOLOR
               such that C != gene[v]
              gene[u] = C
```

2-e-Genetic Parameters

Population size: 200
Maximum Fitness: -0.1
UpperMutation Rate: 100%
LowerMutationRate: 50%
Maximum Number of Iterations: 5000
Patience Time: 70
MaxUselessCount: 5

Upper Mutation Rate: for each two new children, what is the possibility of mutating at least one of them?

Lower Mutation Rate: for each two new children, given that Upper Mutation Rate has been met, each child gets a possibility of 50% of being the child to be mutate

Patience Time:

```
for each new generation
        if (currentMaxChromosome==PrevMaxChromosome)
                patienceCnt++
        if (patienceCnt>=Patience Time)
                Helper()
                UselessCount++
```

Helper and Useless Count will be explained later in the paper

```
if (UselessCount >= MaxUselessCount)
        EndTraining()
```

2-f- Helper

Before explaining the Helper, lets denote:

- The most fit chromosome found so far, as *maxCh.*
- The max chromosome in the current generation (the generation at which Helper was called), as *currentGenMaxCh.*
- The fitness of maxCh as f
- The fitness of *currentGenMaxCh* as f ’

Helper ()

For each gene g in currentGenMaxCh
For each gene g' in maxCh
change g to g'
if (f ' gets improved)
keep changes
else
revert changes

then:
*- repeat popSizeTimes*0.4-*
newChild = mutate(currentGenMaxCh)
add newChild to population

2-g- Fitness Function

Let's denote:
- Each edge that connects two nodes of the same color as a *bad edge*
- *N = Number of nodes in the graph*

*fitness =((100 * number of bad edges) + (50 * number of colors used – 1))/N*

3- Results

For this section, I will compare the results:
- With and without using the helper function
- With another paper with the same problem, *An Efficient Hierarchical Parallel Genetic Algorithm for Graph Coloring Problem* HPGAGCP

this data was collected based on the DIMACS dataset for graph clique numbers

	|V|	|E|	Act-ual Ans-wer	GA Answer W/o Helper	GA Ans.. With Help-er
david.col	87	812	11	56	11
fpsol2.i.1.col	496	11654	65	404	65
games120.col	120	1276	9	85	9
homer.col	561	3258	13	423	15
huck.col	74	602	11	46	11
jean.col	80	508	10	49	10
miles250.col	128	774	8	78	10
miles1000.col	128	6432	42	101	45
anna.col	138	986	11	85	11

	|V|	|E|	Act-ual Ans-wer	HPGAGCP	Helper GA
david.col	87	812	11	11	11
fpsol2.i.1.col	496	11654	65	65	65
games120.col	120	1276	9	9	9
homer.col	561	3258	13	13	15
huck.col	74	602	11	11	11
jean.col	80	508	10	10	10
miles250.col	128	774	8	8	10
miles1000.col	128	6432	42	42	45
anna.col	138	986	11	11	11

4- Further Work

- Improving the *UselessCount* to be based on an estimate for solution correctness based on the output of the helper function.
- Improving the helper function to give better results.
- Experimenting more with the genetic parameters.
- Parallelizing the algorithm to run on the GPU instead of the CPU
- Experiment with the algorithm on more constraint optimization problems.
- Experiment with more methods for crossover

5- References

Abbasian, Reza, and Malek Mouhoub. "A Hierarchical Parallel Genetic Approach For The Graph Coloring Problem". *Applied Intelligence* 39.3 (2013): 510-528. Web.