2013 Fifth International Conference on Computational Intelligence, Modelling and Simulation

A New Genetic Algorithm for Graph Coloring

Raja Marappan,
Department of Computer Applications,
SASTRA University,
Tamil Nadu - 613401, India
e-mail: raja_csmath@cse.sastra.edu

Gopalakrishnan Sethumadhavan,
Department of Computer Applications,
SASTRA University,
Tamil Nadu - 613401, India
e-mail: sgk@mca.sastra.edu

***Abstract* - Graph coloring problem is a classical example for NP-hard combinatorial optimization. Solution to this graph coloring problem often finds its applications to various engineering fields. This paper exhibits the robustness of genetic algorithm to solve a graph coloring. The proposed genetic algorithm employs an innovative single parent conflict gene crossover and a conflict gene mutation as its operators. The time taken to get a convergent solution of this proposed genetic method has been compared with the existing approaches and has been proved to be effective. The performance of this approximation method is evaluated using some benchmarking graphs, and are found to be competent.**

Keywords: Graph coloring, Genetic algorithm, NP-hard

I. INTRODUCTION

A simple graph G = (V, E) consists of vertex set V(G) = $\{v_1, v_2, \ldots, v_n\}$ and an edge set E(G) = $\{e_1, e_2, \ldots, e_m\}$ such that each edge e_k is mapped to an unique and unordered pair of vertices (v_i, v_j). The adjacency matrix A(G) of G is an n × n symmetric binary matrix where $A_{ij} = 1$ if there is an edge between the vertices v_i and v_j and also called adjacent vertices; and $A_{ij} = 0$, otherwise.

The minimum number of colors, $\chi(G)$ used to color the vertices of a graph G with no two adjacent vertices with same color is a Graph Coloring Problem (GCP) [2]. For a simple graph G with n vertices the optimal color $\chi(G)$ can be obtained and found resides in the solution space of size n!. Approximation method which minimizes the search space is needed.

The GCP is used to solve the real world problems such as scheduling, register allocation, channel assignment and noise reduction in VLSI circuits [3] etc. Since GCP is a NP-hard problem, until now there is no method devised to solve it in a polynomial time [1]. Because of its NP-completeness, a fast and effective algorithm to generate an optimal coloring is required. Hertz and De Werra [4] proposed a genetic graph coloring algorithm for random graphs using natural 1-exchange neighborhood. Fleurent and Ferland [5] proposed a hybrid genetic algorithm using specialized crossover and Tabu search algorithm. Mumford [6] has designed a genetic algorithm using heuristic combinations of order-based crossovers. Solutions of GCP are also found using methods such as branch-and-bound, branch-and-cut, and backtracking algorithms [7], [8], [9]. Lixia Han and Zhanli Han[10] designed a bi-objective genetic algorithm. Tamás Szép and Zoltán Ádám Mann [11] has designed typical heuristic genetic algorithm with random mutation and it is compared with branch-and-bound algorithm.

This paper presents a new genetic method to solve GCP, which applies fitness-proportionate selection method for selecting a better gene during each generation. Single Parent Conflict Gene crossover (SPCGX) and conflict-edge mutation operators has been devised to reduce the search space and also to minimize the generations. This procedure has been tested with some benchmarking graphs such as queen, Mycielski, huck.col, jean.col, games120.col, miles250.col, david.col, anna.col and besides some planar graphs and found to be effective.

II. THE NEW GENETIC ALGORITHM

The following notions used in this paper are presented below:

A. Notation 1: Genes Sequence

Let the gene sequence $(g_1, g_2, \ldots, g_n)$ be the color assignments of the vertex set V, where g_i $(1 \le i \le n)$ is a non zero positive integer.

B. Notation 2: Population Initialization

Let p_c and p_m are the probabilities of crossover and mutation respectively. Let g be the generation number and P_g be the random collection of gene sequences at generation g and also termed as population. Let $N = |P_g|$ be the population size and also the number of distinct genes sequences at generation g and P_0 be the population at g=0. Generate N random gene sequence for P_0 such that the value of each gene drawn from $[1, \chi(G)]$.

C. Notation 3: Objective Function of GCP

Let f(G)>0 be the general fitness function of the graph G and is also the number of distinct integers present in the gene sequence during each generation which is being determined using the survival of the fittest strategy. Now the objective function of GCP is to minimize f(G) such that $f(G) - \chi(G) = 0$.

2166-8531/13 $26.00 © 2013 IEEE
DOI 10.1109/CIMSim.2013.17

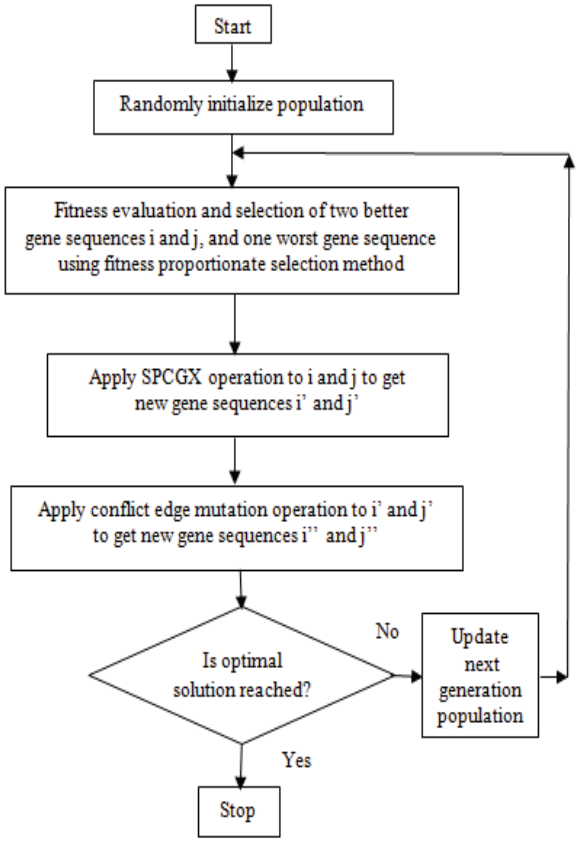

Figure 1: Flowchart of the new genetic algorithm to solve GCP

D. Notion 4: Conflicting Genes/edges and Gene Sequence

In a given gene sequence $s = (s_1, s_2, \ldots, s_i, \ldots, s_j, \ldots, s_n)$, the genes s_i and s_j are conflicting if and only if $si = s_j$ and $(i, j) \in E(G)$. And hence the gene sequence, s.

The flowchart of the new genetic algorithm to solve GCP is shown in Figure 1 and the novel genetic operators to solve GCP have been presented below:

E. Fitness Evaluation and Selection

Fitness proportionate selection is used to select some specific gene sequences in each generation g and is done as follows:

a. Compute the fitness evaluation function F(i) for each gene sequence i ($1 \le i \le N$) in P_g as the number of conflicting edges present in that sequence.
b. Compute the probability for each gene sequence i as $p(i) = F(i) / \sum_{i=1}^{N} F(i)$.
c. Compute the expected number of observations for each i as E(i) = N*p(i).
d. Choose two better gene sequences i and j such that E(i) and E(j) are minimum and next minimum among all the expected number of observations.
e. Choose the worst gene sequence k where E(k) is maximum among all expected number of observations.

F. Crossover Operation

Crossover operation has been performed on the selected gene sequences to generate two new gene sequences. The selected gene sequences i and j are used to generate the offsprings, i' and j', a set of gene sequences using the SPCGX *crossover* operator with a chosen crossover probability p_c. The new SPCGX crossover operator is defined as follows:

a. Let $i = (i_1, i_2, i_3, \ldots, i_q, \ldots, i_r, \ldots, i_n)$ and $j = (j_1, j_2, j_3, \ldots, j_s, \ldots, j_t, \ldots, j_n)$ be the two selected sequences.
b. Choose random probability p_{cr}.
c. If $p_{cr} > p_c$ go to Step-d of crossover operation, otherwise go to Step-G.
d. Identify all pairs of conflicting edges in i and j.
e. Let (q, r) and (s, t) be such two pairs of conflicting edges (i.e., $i_q = i_r$ and $j_s = j_t$) of gene sequences i and j, then apply the following conflict gene crossover operation on these gene sequences to generate two offsprings: $(i_1, i_2, i_3, \ldots, i_q, \ldots, i_r+1, \ldots, i_n)$ and $(j_1, j_2, j_3, \ldots, j_s, \ldots, j_t+1, \ldots, j_n)$.
f. Repeat Step-e of crossover operation to all the identified conflicting pairs in Step-d of crossover operation and generate two updated gene sequences $i' = (i_{1'}, i_{2'}, i_{3'}, \ldots, i_{f'}, \ldots, i_{g'}, \ldots, i_{n'})$ and $j' = (j_{1'}, j_{2'}, j_{3'}, \ldots, j_{k'}, \ldots, j_{l'}, \ldots, j_{n'})$.

G. Mutation Operation

With a chosen mutation probability p_m the gene sequences i' and j' are mutated using the following *conflict gene/edge mutation* operator to generate the offsprings, i'' and j'':

a. Choose random probability p_{mr}.
b. If $p_{mr} > p_m$ go to Step-c of mutation operation, otherwise go to Step-H.
c. Identify all pairs of conflicting edges in i' and j'.
d. Let (f', g') and (k', l') be such two pairs of conflicting edges (i.e., $i_{f'} = i_{g'}$ and $j_{k'} = j_{l'}$) of gene sequences i' and j', then apply the following conflict gene mutation operation on these gene sequences to generate two offsprings: $(i_{1'}, i_{2'}, i_{3'}, \ldots, i_{f'}-1, \ldots, i_{g'}, \ldots, i_{n'})$ and $(j_{1'}, j_{2'}, j_{3'}, \ldots, j_{k'}-1, \ldots, j_{l'}, \ldots, j_{n'})$.
e. Repeat Step-d of mutation operation to all the identified conflicting pairs in Step-c of mutation operation and generate two updated gene sequences $i'' = (i_{1''}, i_{2''}, i_{3''}, \ldots, i_{f''}, \ldots, i_{g''}, \ldots, i_{n''})$ and $j'' = (j_{1''}, j_{2''}, j_{3''}, \ldots, j_{k''}, \ldots, j_{l''}, \ldots, j_{n''})$.

H. Termination and Update population:

Find f(i'') and f(j'') for the two updated gene sequences i'' and j''. If any of f(i'') or f(j'') is equal to $\chi(G)$ then go to Step-E. Otherwise g = g + 1 and update the next generation population P_g as follows:

a. If F(i'') < F(worst) then replace the worst gene with i''.

b. If F(j'') < F(worst) then replace the worst gene with j'' and go to Step-E.

I. Optimal Solution:

Optimal solution obtained. Terminate the generation.

III. THEORETICAL RESULTS AND GLOBAL CONVERGENCE

A. Colorable Case:

The random generated probabilities p_{cr} and p_{mr} determines the conflict edge crossover and mutation operations. Let (q, r), and (s, t) be the conflicting edges of gene sequences i and j respectively. Let $f_g(i)$ and $f_g(j)$ be the fitness function values of gene sequences i and j at generation g. Initially $f_0(i) \leq f_0(j)$. The proposed crossover and mutation reduces the fitness value of gene sequences monotonically in the subsequent generations. However the following cases are noticed:

Case 1: In the subsequent generations the value of the fitness function is monotonically decreasing and converging to an optimal solution say at (q+1) generation. That is, either $f_0(i) \geq f_1(i) \geq f_2(i) \geq \ldots \geq f_q(i) \geq (f_{q+1}(i) = 0)$ or $f_0(j) \geq f_1(j) \geq f_2(j) \geq \ldots \geq f_q(j) \geq (f_{q+1}(j) = 0)$. In the case of Mycielski graph with 11 vertices and for $f_0(i) = 7$ and $f_0(j) = 10$ we found that the values of the fitness function i and j in the subsequent generations are obtained and are 3, 1, 0 and 3, 3, 1 respectively.

Case 2: Initially the fitness function is monotonically decreasing up to finite number of generations, say q. i.e., $f_0(i) \geq f_1(i) \geq f_2(i) \geq \ldots \geq f_q(i)$ and $f_0(j) \geq f_1(j) \geq f_2(j) \geq \ldots \geq f_q(j)$. Then based on the values of p_{cr} and p_{mr} the fitness function values again increases up to finite number of generations, say t. i.e., $f_q(i) < f_{q+1}(i) < \ldots < f_t(i)$ and $f_q(j) < f_{q+1}(j) < \ldots < f_t(j)$. Then after finite number of generation, say u, optimal solution has been reached. In the case of queen graph with 25 vertices and for $f_0(i) = 32$ and $f_0(j) = 32$ we found that the values of the fitness function i and j in the subsequent generations are obtained and are 32, 32, 24, 24, 24, 29, 29, 32, 32, 32, 17, 17, 24 and 32, 32, 24, 24, 24, 29, 31, 32, 32, 32, 17, 17, 0 respectively.

B. Non-colorable Case:

For the non-colorable case, after finite number of generation, say q, $f_g(i) \geq c$ and $f_g(j) \geq c$, for all g = q, q+1, q+2, etc.

C. Global Convergence:

Now we focus to analyze the global convergence of the described genetic approximation algorithm. To do this we first explain the general framework [10] that has been followed in this algorithm. During the iterations of this approximation the population is modified by a number of successive probabilistic transformations. Hence the new population generated during the current iteration in a probabilistic way is depends on the population present in the previous iterations. This property, known as the Markov property, reveals that Markov processes are appropriate models to analyze the probabilistic behavior of this genetic approximation algorithm. In order to prove the global convergence of the algorithm with probability one [12], [13], it is required to introduce the following concept.

D. Reachable from gene sequence $i = (i_1, i_2, i_3, \ldots, i_q, \ldots, i_r, \ldots, i_n)$ to gene sequence $i'' = (i_{1''}, i_{2''}, i_{3''}, \ldots, i_{r''}, \ldots, i_{g''}, \ldots, i_{n''})$.

Let $i' = (i_{1'}, i_{2'}, i_{3'}, \ldots i_{r'}, \ldots i_{g'}, \ldots, i_{n'})$ be a gene sequence obtained from gene i by applying SPCGX crossover operator. Let $i'' = (i_{1''}, i_{2''}, i_{3''}, \ldots i_{r''}, \ldots i_{g''}, \ldots, i_{n''})$ be a gene sequence obtained from gene sequence i' by applying conflict gene/edge mutation operator. Let p(E) defines the probability of the random event E. If $0 < p\{i'' = (\text{crossover(i) and mutation(i')})\} < 1$ then gene sequence i'' is said to be reachable from gene sequence i'.

E. Convergent Conditions to Global Optimum Solution

Let S be the finite search space of the genetic algorithm. The genetic algorithm converges to global optimum if the following conditions are satisfied [12], [13]:

a. For any two gene sequences i, j ϵ S, j is reachable from i.

b. The population P_g is monotone.

Theorem 1: The proposed genetic algorithm with SPCGX and conflict gene mutation always converges to the optimal solution.

Proof:

To prove (a):

Let p_c be the crossover probability of gene sequence i to apply for crossover operation. Let p_m be the mutation probability of gene sequence i' to apply for mutation operation. Clearly $0 < p_c < 1$ and $0 < p_m < 1$ for all gene sequences in the population P_g. Thus p(i'') satisfies,

$$p\{i'' = (\text{crossover(i) and mutation(i')})\} \geq p_c.\ p\{i' = \text{crossover(i)}\}.p_m.\ p\{i'' = \text{mutation(i')}\} \quad (1)$$

Probability of generating gene sequence i' from gene sequence i ($1 \leq i \leq N$) using SPCGX crossover operator is $1/2^k$ where $k > 0$ defines multiple crossovers for all conflicting genes. Therefore

$$p\{i' = \text{crossover(i)}\} = 1/2^k > 0. \quad (2)$$

Probability of generating gene sequence i'' from gene sequence i' ($1 \leq i' \leq N$) using conflict gene mutation is $1/f(G)^l$ where $l > 0$ defines multiple mutations for all conflicting gene. Hence

$$p\{i'' = \text{mutation(i')}\} = 1/f(G)^l > 0. \quad (3)$$

Hence (1) becomes,

$$p\{i'' = (\text{crossover(i) and mutation(i')})\} \geq p_c.1/2^k.p_m.1/f(G)^l\} > 0. \quad (4)$$

Hence i'' is reachable from i' by applying crossover and mutation operators.

To prove (b):

Fitness proportionate selection selects two better gene sequences in each generation. The probability of selecting better gene sequence i using fitness proportionate selection is

$$p(i) = F(i) / \sum_{i=1}^{N} F(i) > 0. \quad (5)$$

The worst gene sequence is replaced with better gene sequence and population is updated. Hence the population P_g is monotone for g=0, 1, 2, 3, ...

Hence this approximation algorithm always converges to an optimal solution.

IV. EXPERIMENTAL RESULTS FOR SOME BENCHMARKING GRAPHS

The experiments are conducted for both colorable and non-colorable cases on some of the benchmarking graphs using Intel Core i5-2450M 2.5GHz with Turbo boost up to 3.1GHz system in Java JDK 1.7 environment. Algorithm has been implemented using JAVA language and the results are tabulated.

Lixia Han and Zhanli Han designed the genetic algorithm [10] in which the population size was set N=20, and the average number of generations performed for Myciel3.col graph was 57.8. But the proposed genetic algorithm runs successfully 100% for N=5 with the average number of generations performed is 4.

Tamás Szép and Zoltán Ádám Mann designed the genetic algorithm [11] in which the percentage of successful runs obtained for queen7_7.col and queen8_8.col graphs are 1% and 2% respectively. But the proposed algorithm runs successfully 31.6%, 26.9% for queen7_7.col and queen8_8.col graphs respectively. Also the number of backtracks performed for queen8_8.col graph is 642758 [11]. But the average number of crossovers and mutations performed in the proposed algorithm for queen8_8.col graph are 505 and 1668 respectively. Also the average number of generations performed for queen8_8.col graph in the proposed method is 13. Thus the proposed method reduces computational complexity compared to the backtracking method [11].

Percentage of successful runs on different benchmarks graphs for $\chi(G)$, $\chi(G)$+1, and $\chi(G)$+2 colors are tabulated in Table I. We have conducted 1000 runs each with 25000 generations for all these graphs. It has been found that the performance of this algorithm is note worthy even for the benchmarking graphs like queen 7 and queen 8 [11]. Table 1 also reveals the following:

1. The number of crossovers is less than the number of mutations for $\chi(G\text{-}2)$, $\chi(G\text{-}1)$, $\chi(G)$, $\chi(G)$+1, and $\chi(G)$+2 colors.
2. Optimal coloring is obtained in minimal number of generations. The maximum average number of generations is 2053 for anna.col graph. The maximum value of g for Mycielski graph is 260 (myciel7.col).
3. Distinct optimal gene sequences are also get generated. For example 632 different optimal gene sequences are obtained for huck.col graph.
4. It has also been noticed that when n increases, the value of g also increases for the graphs huck.col, jean.col, david.col, miles250.col, anna.col.
5. $\chi(G)$-1, $\chi(G)$-2 colors explores the search space more and also proves the non-colorability. The maximal complexity is at $\chi(G)$-1 or $\chi(G)$-2 for most of graphs. It has also been noticed that when n increases, the maximal complexity reaches at $\chi(G)$ for most of the benchmarking graphs.
6. The complexity is monotonically decreasing in j when $j \geq \chi(G)$ for most of the graphs.

The simulated results of various graphs towards achieving $\chi(G)$ with three different categories of p_c & p_m ($p_c < p_m$, $p_c > p_m$ and $p_c = p_m$) are presented in Table II. For $p_c < p_m$ and $p_c > p_m$ categories experiments have conducted 900 runs. For $p_c = p_m$ experiments have conducted 225 runs. Table II also reveals the following:

1. When $p_c \leq p_m$ and for lower values of n ($n < 50$) the optimal solution is reached in minimal number of generations except for some of the queen graphs. Also more number of crossovers is performed than mutation.
2. When $p_c \geq p_m$ and for higher values of n ($n > 70$) the optimal solution is reached in minimal number of generations (graphs: huck.col, jean.col, myciel7.col, games120.col, miles250.col, david.col, anna.col). Also more number of mutations is performed than crossovers.
3. The optimal solution is reached for queen graphs in minimal number of generations when $p_c \geq p_m$ except queen 5 graph.
4. It has also been noticed that when n increases, many distinct optimal gene sequences are also get generated for some of the bench marking graphs (huck.col, jean.col, myciel7.col, games120.col, miles250.col, david.col, anna.col). The maximum value of x is 935 for huck.col graph.
5. It has also been noticed that when n increases, the value of g also increases for some of the bench marking graphs (huck.col, jean.col, myciel7.col, games120.col, miles250.col, david.col, anna.col). The maximum average number of generations performed is 3921 for miles250.col graph when $p_c > p_m$.

The average number of crossovers & mutations performed around $\chi(G)$ for various graphs are shown in Figure 2. The various inputs and the results along with byte code implementation are available at http://www.4shared.com/folder/548wDW59/Raja.html.

TABLE I. PERCENTAGE OF SUCCESSFUL RUNS ON GRAPHS FOR $\chi(G)$, $\chi(G)+1$ & $\chi(G)+2$

Graph No	Graph (G)			Percentage of successful runs of			x *	g @
	Graph Type	Instances	$\chi(G)$	$\chi(G)$	$\chi(G)+1$	$\chi(G)+2$		
1	queen5_5.col	n=25; m=320	5	75.0%	89.8%	93.2%	6	130
2	queen6_6.col	n=36; m=580	7	44.9%	61.8%	81.8%	4	13
3	queen7_7.col	n=49; m=952	7	31.6%	35.2%	39.5%	2	16
4	queen8_8.col	n=64; m=1456	9	26.9%	39.4%	43.2%	1	13
5	myciel5.col	n=47; m=236	6	93.0%	98.2%	100.0%	23	254
6	myciel6.col	n=95; m=755	7	88.3%	100.0%	100.0%	13	7
7	myciel4.col	n=23; m=71	5	100.0%	100.0%	100.0%	3	5
8	myciel3.col	n=11; m=20	4	100.0%	100.0%	100.0%	3	4
9	Heawood graph	n=25; m=67	4	63.0%	95.8%	100.0%	3	2
10	Hexagonal grid graph	n=20; m=43	3	98.3%	100.0%	100.0%	1	4
11	Planar graph	n=6; m=10	3	100.0%	100.0%	100.0%	2	5
12	Planar graph	n=10; m=21	4	100.0%	100.0%	100.0%	8	6
13	Dodecahedron Schlegel	n=20; m=30	3	71.1%	100.0%	100.0%	3	6
14	5-connected planar	n=12; m=30	4	78.5%	100.0%	100.0%	5	7
15	5-connected planar	n=28; m=70	4	84.7%	100.0%	100.0%	3	6
16	huck.col	n=74; m=301	11	63.2%	85.9%	93.8%	632	547
17	jean.col	n=80; m=254	10	27.9%	75.2%	89.8%	279	667
18	myciel7.col	n=191; m=2360	8	22.1%	71.4%	97.5%	221	260
19	games120.col	n=120;m=638	9	18.3%	73.4%	93.5%	183	888
20	miles250.col	n=128;m=387	8	1.4%	30.8%	72.7%	14	1196
21	david.col	n=87;m=406	11	6.5%	39.6%	80.4%	65	1176
22	anna.col	n=138;m=493	11	6.1%	46.9%	82.7%	61	2053

* Number of distinct optimal gene sequences generated; @ Average number of generations performed for $\chi(G)$

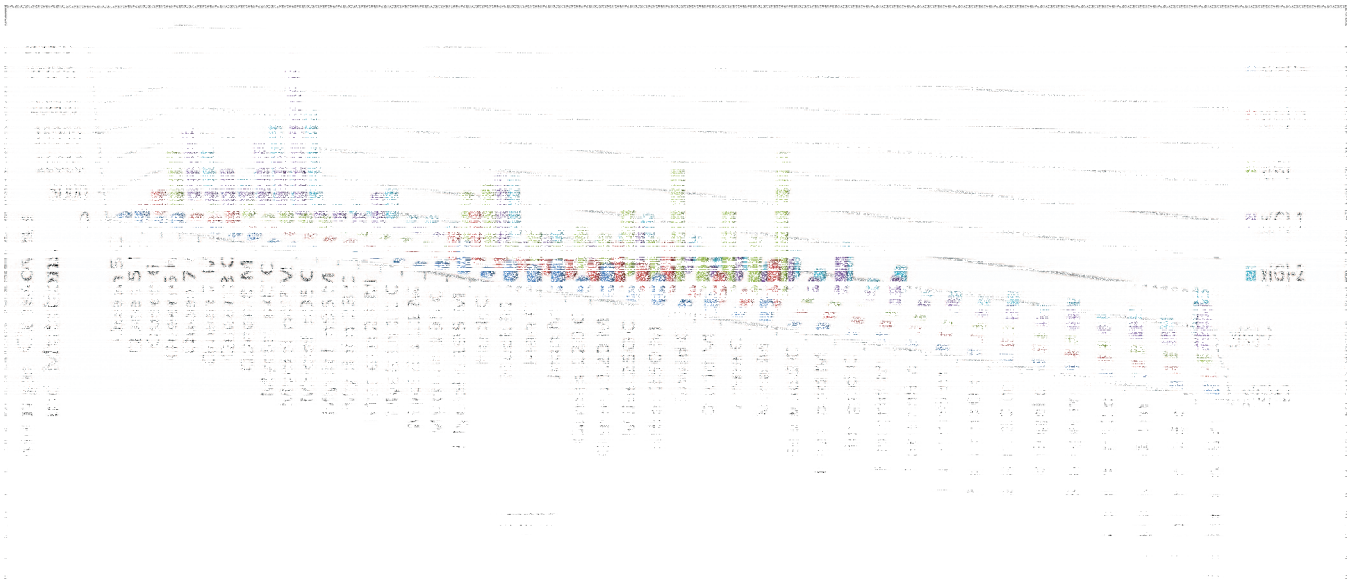

Figure 2: Average number of crossovers & mutations performed around χ(G), for various graphs

TABLE II. SIMULATED RESULTS ON GRAPHS TOWARDS ACHIEVING $\chi(G)$ WITH DIFFERENT CATEGORIES OF p_c & p_m

Graph No	$p_c < p_m$				$p_c > p_m$				$p_c = p_m$				x^*
	R	g	C	D	R	g	C	D	R	g	C	D	
1	99.5%	1	43	32	84.8%	3	43	145	99.1%	3	42	78	5
2	1.4%	15	652	1034	31.2%	15	508	1199	5.7%	20	1188	1295	4
3	0.7%	44	7918	5042	27.4%	20	1809	3230	8.8%	34	5779	4319	2
4	17.3%	22	3171	1849	33.4%	15	1101	1746	40.4%	21	2318	2009	1
5	99.6%	1	36	6	89.1%	4	52	67	99.1%	3	41	21	14
6	99.3%	1	13	7	88.5%	5	31	123	96.0%	2	16	21	10
7	100.0%	1	16	2	100.0%	4	22	35	100.0%	3	19	7	2
8	100.0%	1	16	5	99.8%	4	16	65	100.0%	3	16	22	3
9	99.2%	1	19	7	76.7%	2	20	31	97.7%	2	19	24	3
10	99.8%	1	23	12	97.8%	4	24	135	97.7%	2	23	30	1
11	100.0%	1	5	2	100.0%	4	6	28	100.0%	3	6	6	2
12	100.0%	2	19	7	99.8%	5	20	85	99.5%	5	19	21	10
13	6.8%	2	25	48	45.1%	6	25	180	14.2%	4	25	62	1
14	99.1%	1	9	2	82.2%	3	12	32	95.1%	3	11	8	2
15	98.5%	1	37	14	85.4%	5	48	334	96.0%	3	38	35	3
16	33.5%	1430	7398	2910	58.7%	530	2753	4695	46.2%	682	3938	2610	935
17	12.4%	817	5627	2096	20.5%	1316	4928	8634	21.3%	97	498	391	345
18	7.1%	779	23292	7563	16.8%	423	6873	13437	8.9%	523	13008	8656	235
19	8.1%	2048	18878	6398	15.8%	820	5226	8542	12.0%	2562	28136	18125	242
20	0.6%	368	6850	1329	0.8%	3921	25554	49871	0.5%	43	1019	1174	13
21	1.7%	1516	12558	5219	5.2%	833	3797	6758	3.6%	63	585	420	70
22	0.3%	6963	26638	6892	2.9%	2414	11110	19790	0.9%	37	747	393	31

R- Percentage of successful runs; g- Average number of generations; C Average number of crossovers; D Average number of mutations

V. CONCLUSION

In this paper, we presented a new genetic algorithm for graph coloring. We conducted several experiments with benchmark graphs to assess the computational complexity of graph coloring. The experimental results of proposed algorithm are compared with the existing approaches and it is found to be competent. According to this new genetic algorithm, $\chi(G)$ coloring is obtained in minimal number of generations. The proposed approximation genetic algorithm reduces the computational complexity and always produces an optimal solution. The new SPCGX crossover and conflict-gene mutation operators minimize the number of generations of the genetic algorithm and also reduce the search space significantly towards reaching an optimal solution.

ACKNOWLEDGEMENT

The authors would like to acknowledge the support rendered by the SASTRA Management by the way of providing necessary infrastructure and financial support for this research.

REFERENCES

[1] Garey, M. R. and Johnson, D. S., Computers and Intractability: Guide to the Theory of NP-Completeness, Freeman, San Francisco, 1979.

[2] Tommy R. Jensen, Bjarne Toft, Graph Coloring Problems, Wiley-Inter science, 1995.

[3] Noise Reduction in VLSI Circuits using Modified GA Based Graph Coloring - International Journal of Control and Automation, Vol. 3, No. 2, June, 2010.

[4] A. Hertz, and D. E Werra. "Using tabu search techniques for graph coloring", Computing Vol. 39, No. 4, pp. 345-351, 1987.

[5] C. Fleurent and J. A. Ferland. "Genetic and hybrid algorithms for graph coloring", Annals of Operations Research, 63, pp. 437-463, 1995.

[6] C. L.Mumford. "New order-based crossover for the graph coloring problem", T. P. Runarsson et al. (Eds.): PPSN LX, vol. 4193, pp. 80-88, 2006.

[7] Anuj Mehrotra and Michael A. Trick. A column generation approach for graph coloring. INFORMS Journal on Computing, 8(4):344–354, 1996.

[8] Isabel Méndez-Diaz and Paula Zabala. A branch-and-cut algorithm for graph coloring. Discrete Applied Mathematics, 154(5):826–847, 2006.

[9] R. Monasson. On the analysis of backtrack procedures for the coloring of random graphs. In E. Ben-Naim, H. Frauenfelder, and Z. Toroczkai, editors, Complex Networks, pages 235–254. Springer, 2004.

[10] Lixia Han and Zhanli Han, A Novel Bi-objective Genetic Algorithm for the Graph Coloring Problem, 2010 Second International Conference on Computer Modeling and Simulation.

[11] Tamás Szép and Zoltán Ádám Mann, Graph coloring: the more colors, the better?, CINTI 2010, 11th IEEE International Symposium on Computational Intelligence and Informatics, 18–20 November, 2010, Budapest, Hungary.

[12] G. Rudolph. "Finite Markov chain results in evolutionary computation: A tour Horizon", Fundamenta Informaticae, vol. 35, no.2, pp. 67-89.

[13] Back T. "Evolutionary algorithms in theory and practice", New York Oxford University Press, pp. 21-28, 1996.