International Journal of Engineering and Advanced Technology (IJEAT)
ISSN: 2249-8958 (Online), Volume-6 Issue-1, October 2016

Investigation of Satisfiability Based Solution Approach for Graph Coloring Problem

Prakash C. Sharma, Narendra S. Chaudhari

***Abstract:** Graph k-colorability (for k ≥ 3) problem (GCP) is a well-known NP-Complete problem. There are many approaches proposed to solve graph coloring problem till date. There is an alternative approach to solve it efficiently by Satisfiability which is first known NP-Complete problem. We can reduce any NP-complete problem to/from SAT. Reduction from graph k-colorability problem to satisfiability is an important concept to solve it using efficient SAT solver. In this paper, we are presenting a polynomial 3-SAT encoding technique for k colorable graph. Our formulation generates total (((k-2)*|V|) + (k*|E|)) clauses in 3-CNF for k-colorable graph. We tested our encoding formulation approach on different graph coloring instances of DIMACS[8][9] and then investigated the solution of graph coloring problem as a decision problem based on SAT approach using powerful SAT solver Minisat 2.2.*

***Keywords:** 3-SAT, CNF, DNF, graph coloring, NP-Complete, k-colorable, chromatic number, DIMACS.*

I. INTRODUCTION

A Graph k-colorability is an assignment of colors $\{1,2,\ldots,k\}$ to the vertices of graph G in such a way that neighbor vertices of graph should receive different colors. That means, in a proper graph coloring, if two vertices u and v of a graph share an edge (u, v), then they must be colored with different colors. The minimum number of colors needed to color the vertices of graph G is called the chromatic number of G, denoted as $\chi(G)$. A graph that can be assigned a (proper) k-coloring is k-colorable, and it is k-chromatic if its chromatic number is exactly k. Graph coloring was among the 21 NP-complete problems [7] originally given by Richard Karp in the year 1972. Graph coloring is a fundamental and extensively studied problem, which besides its theoretical significance also enjoys a lot of practical applications. The graph k-colorability problem has several important real-world applications [15][16], including register allocation, frequency assignment problem in cellular network, time tabling problem, aircraft scheduling problem and many other problems. Satisfiability (SAT) was the first problem shown to be NP-Complete [8]. The SAT problem is usually expressed in conjunctive normal form (CNF). A CNF formula on binary variables is the conjunction (logical AND) of clauses, each of which is a disjunction (logical OR) of one or more literals, where a literal is the occurrence of a variable or its complement. A clause is said to be satisfied if at least one of its literals is true, unsatisfied if all of its literals are set to false and unresolved otherwise.

Manuscript published on 30 October 2016.
* Correspondence Author (s)
Prakash C. Sharma, Department of Computer Science & Engineering, Indian Institute of Technology, Indore (M.P.), India.
Narendra S. Chaudhari, Department of Computer Science & Engineering, Indian Institute of Technology, Indore (M.P.), India.

A formula is said to be satisfied if all its clauses are satisfied, and unsatisfied if at least one of its clauses is unsatisfied. In general, the SAT problem is defined as follows: Given a Boolean formula in conjunctive normal form (CNF), find an assignment of variables that satisfies the formula or prove that no such assignment exists. In the following example, the 3-CNF (clause length=3) formula E consists of 4 variables and 3 clauses; each clause having at most 3 literals (length of clause=3 i.e. 3-CNF):

$$E = (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_4)$$

One of the truth assignments for satisfiability of above expression is $x_1 = x_3 =$ true, & $x_2 =$ false or $x_1 = x_2 =$ true & $x_3 =$ false. Note that a problem with n variables will have 2^n possible assignments to test. The above example with 3 variables has 8 possible assignments. The Satisfiability problem is particularly interesting because it can be used as a stepping stone for solving decision problems. The graph colouring problem can also be solved as a decision based using method of Satisfiability (SAT). Problem instances from domains such as Graph Colouring can be encoded into SAT and then solved by the help of SAT algorithms. Previously, in [6], Alexander Tsiatas gave a reduction approach from 3-Colorable graph to 3-SAT expression. He encoded the vertices and edges of the graph by 3-color as boolean encoded expression in DNF then that has to be converted into k-CNF. He used two recursive and one non-recursive method to convert a k-CNF expression into 3-CNF expression. Results of all three methods were observed and found that non- recursive method gave a better result than remaining. Finally, using this, Alexander generates total ((27*|V|) + (256*|E|)) clauses as 3-CNF-SAT formula for 3-colorable graph. In our earlier formulation of reduction of k-colorable graph to 3-SAT [1][5], we generalized Alaxander's approach for k-colorable graph and generated ((k^k *(k-2)*|V|) + (2^{2k+2} *|E|)) clauses in 3-CNF, which is an exponential bound complexity. In section 2, we discussed our 3-SAT encoding approach for k-colorable graph [1][5]. Section 3 illustrated SAT encoding approach of a 3-colorable graph. In section 4, we discussed the SAT approach for solving graph coloring problem and then in section 5 experimental results are discussed.

II. POLYNOMIAL 3-SAT ENCODING FORMULATION APPROACH OF K-COLORABLE GRAPH

We have formulated reduction approach of graph k-color ability to 3-SAT in [1]. Here we are briefing description of 3-SAT encoding technique of k-colorable graph as follows: let there be a graph $G = (V, E)$, where V is the set of n vertices $\{v_1\, v_{2,\ldots,} v_n\}$ and E is the set of m edges$\{e_1\, e_{2,\ldots,} e_m\}$. The graph has to be colored by k-color $\{1,2,\ldots,k\}$ in such a way that no two adjacent vertices have the same color.

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)

Then to encode this colored graph into propositional formula, we use two approach say vertex constraint approach and edge constraint approach which will apply on vertices and edges of the graph respectively. The polynomial 3-SAT encoding formulation of k-colorable graph [1] is presented below:

2.1. Vertex Constraint Approach

As per vertex constraint approach, color each vertex of a graph G as v_{ic} in such a way that vertex v_i (i = 1,2,…,n vertices) should have at least one color c (c = 1,2,…,k) among available k-colors as follows:

$$v_{ic} = (v_{i1} \vee v_{i2} \vee \ldots \vee v_{ik}) \quad \ldots (1)$$

Equation (1) generates one clause of length-k in CNF corresponding to each vertex of graph. But, now we have to reduce it in 3-CNF. There are several different ways of doing this, one of the non-recursive methods is to convert a k-CNF to 3-CNF is as follows: Consider a clause $F = x_1 \vee x_2 \vee \ldots \vee x_k$ where k (k > 3) is the length of the clause, which can be converted in 3-CNF by introducing some new variables like $y_1, y_2, \ldots, y_{k-3}$ as:

$$(x_1 \vee x_2 \vee \neg y_1) \wedge (x_3 \vee y_1 \vee \neg y_2) \wedge (x_4 \vee y_2 \vee \neg y_3) \ldots (x_{k-1} \vee x_k \vee y_{k-3}) \quad \ldots (2)$$

Expression (2) transforms a clause of length k into (k−2) clauses of length 3, and doing this requires introducing (k−3) new variables. For example, applying (2) to a clause of length 6 yields (1 clause of length 6) = (4 clauses of length 3) and this required an additional 3 variables, since the clause was of length k = 6.

Let F_v is the conjunction of 3-CNF encoding expression of all the n vertices of graph G by applying vertex constraint approach as:

$$F_v = (v_{1c} \wedge v_{2c} \wedge \ldots \wedge v_{nc}) \quad \ldots (3)$$

Applying (2) to (1) and finally, we get total (k-2)*|V| clauses in 3-CNF-SAT from graph G corresponding to vertex constraint approach.

$|F_v| = (k-2)*|V|$ clauses in 3-CNF-SAT … (4)

2.2. Edge Constraint Approach

As per edge constraint approach, color two end points of each edge e_j (j=1,2,….m) of a given graph G in such a way that two vertices (u, v) connecting with an arc should not have same colors. That is, any edge of a k-colorable graph can be encoded by generating a clause in such a way that two end point of an edge say u,v should not be assigned same color k. The purpose of this approach is to ensure that two adjacent vertex should not be assigned same color.

$$e_j = \neg(u_1 \wedge v_1) \wedge \neg(u_2 \wedge v_2) \wedge \ldots \wedge \neg(u_k \wedge v_k)$$

Above equation can also be written as:

$$e_j = (\neg u_1 \vee \neg v_1) \wedge (\neg u_2 \vee \neg v_2) \wedge \ldots \wedge (\neg u_k \vee \neg v_k) \quad \ldots (5)$$

Let F_e is the conjunction of 3-CNF encoded expression of all the m edges of graph G by applying edge constraint approach as:

$$F_e = (e_1 \wedge e_2 \wedge \ldots \wedge e_m) \quad \ldots (6)$$

Since, expression (5) is in 3-CNF-SAT, so there is no need to apply (2) on it. Finally we get, total k*|E| clauses in 3-CNF-SAT from graph G as per edge constraint approach i.e.

$|F_e| = k*|E|$ clauses in 3-CNF-SAT … (7)

2.3. Bounds of Final 3-CNF-SAT Formula

To get final 3-CNF-SAT encoded formula F of graph G, we conjunct encoded formula obtained by vertex constraint approach (3) and formula obtained by edge constraint approach (6) as:

$$F = ((v_{1c} \wedge v_{2c} \wedge \ldots \wedge v_{nc}) \wedge (e_1 \wedge e_2 \wedge \ldots \wedge e_m)) \quad \ldots (8)$$

Finally, to get total number of clauses in 3-CNF-SAT formula of graph G, we combine (4) and (7) as number of clauses obtained by vertex constraint approach and by edge constraint approach. Thus, total number of clauses in 3-CNF-SAT formula of k-colorable graph = (k-2)*|V| + k*|E|, which is polynomial encoding of k-colorable graph to 3-SAT.

$|F| = (k-2)*|V| + k*|E|$ … (9)

2.4. Algorithm: k-Colorable Graph to 3-CNF

1. Read input .col file of graph in the form of adjacency list. Here we have taken DIMACS graph coloring instances as input through a file.
2. Read number of colors k from user.
3. Read the number of vertices n and number of edges m from input file".
4. Start the encoding process for all the vertices from vertex 1 to last vertex by applying vertex constraint approach as follows:
   4.1 if number of colors k ==3

```
for (int i=1;i<=vertices;i++)
{
        for(int j=1;j<=k;j++)
        {
        write(i+"0"+j+" ");
        }
        write("0\n");
}
```

   4.2 if number of colors k > 3 and no of literals > 3 then write the first two literal as it is in the output file separated by an space " " and then write the expression "zNv)" "(-zNv+literal" till only two literals remain, where Nv is the count for number of extra variable z inserted. Append the last two literals in the file and Write ")" to the file; If it is not the last clause give space " " in the file.

5 Generate clauses by applying edge constraint approach on all the edges (u,v) of graph as follows:

```
for(int j=1;j<=color;j++)
{
```


Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

```
bw.write(""+edge[1]+"0"+j+"        -"+edge[2]+"0"+j+"
0\n");
}
```

6. Merge the clauses obtained by step 4 and 5.
7. Display the total number of generated clauses, number of extra variable needed, total execution time.

2.5. Bounds on number of clauses in 3-CNF expression

Property 1: The total number of 3-CNF clauses generated for a *k*-colorable graph is ((*k*-2)* |*V*| + *k**|*E*|) for *V* vertices and *E* edges of graph *G*

Proof *(by Induction):*

(a) Base Case: k=3 (k=no. of colors)

By Formula: Total number of clauses in 3-CNF expression:
(*k*-2)*|*V*| + *k**|*E*| = (3-2)*|*V*| + 3*|*E*| = |*V*| + 3*|*E*|
For one vertex and one edge it will be 1+3 = 4 clauses in 3-CNF

By expression: Let $v \wedge e$ are conjunction of encoded expression for a vertex *v* and an edge (*u*, *v*) of 3-colorable graph *G* by vertex constraint and edge constraint approach

$$v \wedge e = (v_1 \vee v_2 \vee v_3) \wedge (\neg u_1 \vee \neg v_1) \wedge (\neg u_2 \vee \neg v_2) \wedge (\neg u_k \vee \neg v_k)$$

For one vertex and one edge, above expression is generating 1+3 = 4 clauses in 3-CNF. Hence Base case is true.

(b) For k=m

By formula: (*k*-2)*|*V*| + *k**|*E*| = (*m*-2)*|*V*| + *m**|*E*|

By expression: For *m* colors $v_m \wedge e_m$ can be expressed as:

$$v_m \wedge e_m = (v_1 \vee v_2 \vee ... \vee v_m) \wedge (\neg u_1 \vee \neg v_1) \wedge (\neg u_2 \vee \neg v_2) \wedge ... \wedge (\neg u_m \vee \neg v_m)$$

But, we know that when we convert a single *m*-CNF clause with *m* different literals (*m*>3) into 3-CNF, we get (*m*-2) clauses in our 3-CNF expression. Hence, Number of clauses in 3-CNF expression by vertex constraint approach = (*m*-2)*|*V*| and number of clauses in 3-CNF by edge constraint approach = *m*. Therefore total number of clauses in 3-CNF from $v_m \wedge e_m$ is (*m*-2)*|*V*| + *m**|*E*|. So it is also true for *k*=*m*.

(c) For k = m+1

By formula: (*k*-2)*|*V*| + *k**|*E*| = (*m*+1-2)*|*V*| + (*m*+1)*|*E* = (*m*-1)*|*V*| + (*m*+1)*|*E*|

By expression: For *m*+1 color, the expression $v_{m+1} \wedge e_{m+1}$ can be represented as:

Number of 3-CNF clauses from v_{m+1} = (Number of clauses for v_m + clauses for (*m*+1)th color)
= ((*m*-2) + 1) = (*m*-1)

Number of 3-CNF clauses from e_{m+1} = (Number of clauses for e_m + clause for (*m*+1)th color) = (*m*+1)

Total no. of clauses in 3-CNF from $v_{m+1} \wedge e_{m+1}$ = (*m*-1)*|*V*| + (*m*+1)*|*E*|

So it is also true for *k*=*m*+1. Hence proved**.**

2.6. Justification of Propositional Encoding Formulation of k-Colorable Graph

Lemma: If a 3-CNF-SAT formula is satisfiable then graph is *k*-colorable.

Proof: Let us assume that an undirected graph *G*(*V*, *E*) that is *k*-colorable and the following is a 3-CNF-SAT formula corresponding to graph *G*:

$$F = (v_i \wedge e_j)$$

where $v_i = v_1, v_2, ..., v_n$ are *n* vertices and $e_j = e_1, e_2, ..., e_m$ are *m* edges of the graph *G* that has to be *k*-colored by vertex constraint approach and edge constraint formulation respectively. Above expression can also be expanded as:

$$F = ((v_1 \wedge v_2 \wedge ... \wedge v_n) \wedge (e_1 \wedge e_2 \wedge ... \wedge e_m))$$

For satisfiable of *F*, each one of these expressions should be true. Let's take an encoded vertex expression F_v for *k*-color by vertex constraint approach; F_v will be true when all of its clauses is true.

$$F_v = (v_{11} \vee v_{12} \vee ... \vee v_{1k}) \wedge (v_{21} \vee v_{22} \vee ... \vee v_{2k}) \wedge ... \wedge (v_{n1} \vee v$$

By this, it is clear that every vertex will be assigned at least one color. Similarly, take encoded expression of edge F_e for *k*-color by edge constraint approach. The expression F_e will be true when all of its edge clauses is true.

$$F_e = (e_1 \wedge e_2 \wedge ... \wedge e_m)$$

Let's take an encoded edge clause e_1 (v_1,v_2) from F_e; e_1 will be true if all its clauses is true. It means ends points of an edge will not be assigned same color.

$$e_1 = (\neg v_{11} \vee \neg v_{21}) \wedge (\neg v_{12} \vee \neg v_{22}) \wedge ... \wedge (\neg v_{1k} \vee \neg v_{2k})$$

Similarly, if we take other clauses, we will get the same conclusion that end points of an edge are colored with different color and this is true for each edge. Hence our graph is *k*-colorable.

III. ILLUSTRATION OF ENCODING OF 3-COLORABLE GRAPH TO 3-CNF-SAT

Here, we are taking an example of Petersen [9] graph *G* as figure 1, having 10 vertices and 15 edges to encode it by 3-color say 1, 2, 3 into propositional 3-satisfiability. Graph having following set of vertices and edges:

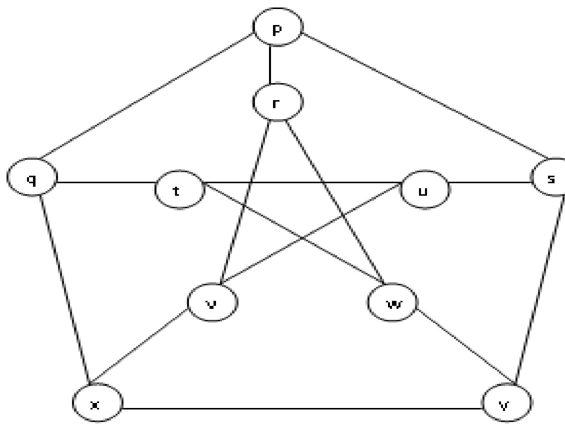

Fig 1: Petersen Graph [9]

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

$$V = \{p, q, r, s, t, u, v, w, x, y\}$$

$$E = \{(p,q),(p,r),(p,s),(q,t),(q,x),(r,v),(r,w),(s,u),(s,v),(t,u),(t,w),(u,v),(v,x),(w,y),(x,y)\}$$

Now we start polynomial 3-SAT encoding of following graph by 3-colors. As per the vertex constraint approach (1), we encode vertices of this graph and stored in F_v as:

$$F_v = (p_1 \vee p_2 \vee p_3) \wedge (q_1 \vee q_2 \vee q_3) \wedge (r_1 \vee r_2 \vee r_3) \wedge (s_1 \vee s_2 \vee s_3) \wedge (t_1 \vee t_2 \vee t_3) \wedge (u_1 \vee u_2 \vee u_3) \wedge (v_1 \vee v_2 \vee v_3) \wedge (w_1 \vee w_2 \vee w_3) \wedge (x_1 \vee x_2 \vee x_3) \wedge (y_1 \vee y_2 \vee y_3)$$

Similarly, we encode all the edges of graph as per the edge constraint approach (5), and stored 3-CNF-SAT expression in F_e. as:

$$F_e = (\neg p_1 \vee \neg q_1) \wedge (\neg p_2 \vee \neg q_2) \wedge (\neg p_3 \vee \neg q_3) \wedge (\neg p_1 \vee \neg r_1) \wedge (\neg p_2 \vee \neg r_2) \wedge (\neg p_3 \vee \neg r_3) \wedge (\neg p_1 \vee \neg s_1) \wedge (\neg p_2 \vee \neg s_2) \wedge (\neg p_3 \vee \neg s_3) \wedge (\neg q_1 \vee \neg t_1) \wedge (\neg q_2 \vee \neg t_2) \wedge (\neg q_3 \vee \neg t_3) \wedge (\neg q_1 \vee \neg x_1) \wedge (\neg q_2 \vee \neg x_2) \wedge (\neg q_3 \vee \neg x_3) \wedge (\neg r_1 \vee \neg v_1) \wedge (\neg r_2 \vee \neg v_2) \wedge (\neg r_3 \vee \neg v_3) \wedge (\neg r_1 \vee \neg w_1) \wedge (\neg r_2 \vee \neg w_2) \wedge (\neg r_3 \vee \neg w_3) \wedge (\neg s_1 \vee \neg u_1) \wedge (\neg s_2 \vee \neg u_2) \wedge (\neg s_3 \vee \neg u_3) \wedge (\neg s_1 \vee \neg y_1) \wedge (\neg s_2 \vee \neg y_2) \wedge (\neg s_3 \vee \neg y_3) \wedge (\neg t_1 \vee \neg w_1) \wedge (\neg t_2 \vee \neg w_2) \wedge (\neg t_3 \vee \neg w_3) \wedge (\neg t_1 \vee \neg u_1) \wedge (\neg t_2 \vee \neg u_2) \wedge (\neg t_3 \vee \neg u_3) \wedge (\neg u_1 \vee \neg v_1) \wedge (\neg u_2 \vee \neg v_2) \wedge (\neg u_3 \vee \neg v_3) \wedge (\neg v_1 \vee \neg x_1) \wedge (\neg v_2 \vee \neg x_2) \wedge (\neg v_3 \vee \neg x_3) \wedge (\neg w_1 \vee \neg y_1) \wedge (\neg w_2 \vee \neg y_2) \wedge (\neg w_3 \vee \neg y_3) \wedge (\neg x_1 \vee \neg y_1) \wedge (\neg x_2 \vee \neg y_2) \wedge (\neg x_3 \vee \neg y_3)$$

Finally, we conjunct F_v and F_e and obtained 3-CNF-SAT encoding expressions of 3-colorable Petersen graph.

$F = F_v \wedge F_e$

Total number of 3-CNF-SAT clause corresponding to above 3-colorable graph = (10 + 15) = 25, which is a polynomial reduction from graph to 3-SAT.

IV. SOLUTION APPROACH FOR GRAPH K-COLORABILITY USING SAT SOLVER

SAT-based approach is a general method to solve difficult combinatorial problems by encoding them into SAT (Satisfiability) problems and solving by using an efficient SAT solver. SAT solver is a program to find a solution of a SAT problem. Recent advances of SAT solver technology are remarkable. SAT solvers are used to solve hard problems by encoding them to SAT problems (SAT-based approach), such as scheduling, planning, Software and hardware verification. Since, there have been dramatic improvements in SAT solver technology over the past decade. This has led to the development of several powerful SAT algorithms that are capable of solving many hard problems consisting of thousands of variables and millions of constraints.

Reduction from graph *k*-colorability problem to SAT (satisfiability) is an important concept to solve it using efficient SAT solver. With the help of the polynomial encoding technique of graph *k*-colorability to SAT, we have reduced many graph coloring instances into 3-CNF-SAT expression. We have taken DIMACS benchmark instance [8][4] as input for the graph to encode into SAT. The DIMACS benchmark collects a large set of instances, which represent the standard set for experimenting algorithms for the Vertex Coloring Problem (see [8][14], all instances are available at ftp://dimacs.rutgers.edu/pub/challenge/graph/). The benchmark set includes: random graphs (DSJC), where for each pair of vertices $i, j \in V$, edge $(i, j) \in E$ is created with uniform probability; geometric random graphs (DSJR and *r*), where vertices are randomly distributed in a unit square, and an edge $(i, j) \in E$ is created if the distance between *i* and *j* is less than a given threshold.

After generating 3-CNF expression, now solved it by a powerful SAT solver. Here, we used a powerful SAT solver Minisat 2.2 [11] to solve 3-CNF-SAT expression. MiniSat [12][13] is a minimalistic, open-source Boolean satisfiability (SAT) solver, developed for both researchers and developers. MiniSat is a simple, well documented, implementation suitable for educational purposes and can solve a problem with 10^7 literals. Minisat gives output as truth assignment if formula is “SATISFIABLE”; otherwise it proves that expression is “UNSATISFIABLE”. Satisfiable expression also tells that graph is colored by exactly *k* colors.

V. RESULTS

We have implemented the formulation of polynomial 3-CNF-SAT encoding of *k*-colorable graph. Our formulation generates total $(((k\text{-}2)*|V|) + (k*|E|))$ clauses in 3-CNF for *k*-colorable graph, whereas previously, Alaxander generated $((k^k *|V|) + (2^{2k+2} *|E|))$ clauses in 3-CNF. Thus, our formulation is better than approach [6]. Here, we analyzed the encoding formulation for 3-color and 4-color on various benchmark problems (graph coloring instances) of the DIMACS challenge [8][14]. We implemented it in java (JDK 1.7). Results are compiled at table 1.

Graph Coloring Instances	No. of Vertices	No. of Edges	Alaxander's Approach (Total no of 3-CNF clause when *k*=3)	Our Approach Total 3-CNF-SAT clauses generated (when *k*=3)	Alaxander's Approach: Total no of 3-CNF clause when *k*=4)	Our Approach: Total 3-CNF-SAT clauses generated (when *k*=4)
myciel3	11	20	5417	**71**	8448	**102**
myciel4	23	71	18797	**236**	25024	**330**
queen5_5	25	160	41635	**505**	27200	**690**
mugg100_1	100	166	45196	**598**	108800	**864**
myciel5	47	236	61685	**755**	51136	**1038**
queen6_6	36	290	75212	**906**	39168	**1232**
miles250	128	387	102528	**1289**	139264	**1804**
queen7_7	49	476	123179	**1477**	53312	**2002**
myciel6	95	755	195845	**2360**	103360	**1700**

Table 1: 3-CNF-SAT Clause Generation for Color *k*=3 and 4

Now, in next step, we solve the obtained 3-CNF-SAT encoded expression using SAT solver. Here we used Minisat 2.2 which gives output as truth assignment if formula is "SATISFIABLE"; otherwise it proves that expression is "UNSATISFIABLE". Satisfiable expression also tells that graph is colored by exactly *k* colors. Here, we reported the computational results obtained by the Minisat 2.2 which takes input from 3-SAT expression obtained by the SAT encoding of *k*-colorable graph. All results of our algorithms were obtained on a Pentium IV 2.4 GHz with 2 GB RAM under Windows 7 as well as Linux (Ubuntu 12.4).

Table 2: Results of Some DIMACS Graph Instances which are Encoded as 3-CNF-SAT by our Reduction Approach and Then Solved by Minisat 2.2

Name of Graph Instance	No of vertices *n*	No of edges *m*	Satisfiable (SAT) on colors (chromatic number)
DSJC125.1.col	125	736	5
DSJC125.5.col	125	3891	5
DSJC125.9.col	125	6961	6
DSJC250.1.col	250	3218	5
DSJC250.5.col	250	15668	5
DSJR500.1.col	500	3555	5
le450_15a.col	450	8168	5
le450_15b.col	450	8169	5
le450_5a.col	450	5714	5
le450_5b.col	450	5734	5
le450_5d.col	450	9757	5
queen5_5.col	25	320	5
queen6_6.col	36	580	5
queen7_7.col	49	952	5
queen8_8.col	64	728	5
queen9_9.col	81	2112	5
myciel3.col	11	20	4
myciel4.col	23	71	4
myciel5.col	47	236	5
myciel6.col	95	755	5
myciel7.col	191	2360	5

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

1-Insertions_4.col	67	232	5
1-Insertions_5.col	202	1227	4
1-Insertions_6.col	607	6337	5
2-Insertions_4.col	149	541	4
2-Insertions_5.col	597	3936	5
3-Insertions_4.col	281	1046	4
4-Insertions_3.col	79	156	4
4-Insertions_4.col	475	1795	5
1-FullIns_3.col	30	100	4
1-FullIns_4.col	93	593	5
1-FullIns_5.col	282	3247	6
2-FullIns_3.col	52	201	5
2-FullIns_4.col	212	1621	6
2-FullIns_5.col	852	12201	5
3-FullIns_3.col	80	346	6
3-FullIns_4.col	405	3524	7
miles250.col	128	774	5
miles500.col	128	2340	5
miles700.col	128	4226	5
mulsol.i.1.col	197	3925	5
mulsol.i.2.col	188	3885	5
mulsol.i.3.col	184	3916	5

Computational result of graph coloring problem using Minisat 2.2 are stored in Table 2 and Table 3. Some of the DIMACS graph instance which are satisfiable on any color *k* are stored in Table 2. Some of the large graphs having 500 or more vertices are not satisfied and such list is contained by table 3.

Table 3: Results of Some UNSAT Graph Instance Given by Minisat 2.2

Name of Graph Instance	No of vertices *n*	No of edges *m*	Our Investigation: Satisfiable (SAT) on Colors
DSJC250.9.col	250	27897	UNSAT
DSJC500.1.col	500	12458	UNSAT
DSJC500.5.col	500	62624	UNSAT
DSJC500.9.col	500	1124367	UNSAT
DSJC1000.1.col	1000	49629	UNSAT
DSJC1000.5.col	1000	249826	UNSAT
DSJC1000.9.col	1000	449449	UNSAT
DSJR500.1.c.col	500	121275	UNSAT

VI. CONCLUSION

In this paper, we have formulated generalized polynomial encoding technique of graph *k*-colorability to/from 3-CNF-SAT. A polynomial reduction of a *k*-colorable graph to 3-CNF-SAT generates $((k\text{-}2)*|V| + k*|E|)$ clauses in 3-CNF-SAT expression. Whereas, in [6],Alexander Tsiatas had generated total number of clauses in 3-CNF-SAT formula for *k*-colorable graph having $((k^k *|V|) + (2^{2k+2} *|E|))$ clauses in 3-CNF. Thus, our encoding formulation of *k*-colorable graph to 3-CNF-SAT is polynomial and better than [6] which were exponential. Later on, we have investigated SAT based approach for solving graph coloring problem using a powerful SAT solver MiniSAT 2.2, the role played by SAT as an intermediate domain for solving problems. SAT technique only extracted that input number of color can be sufficient to color the graph or not. We tested many DIMACS graph instances and found some of the graph is *k* colorable i.e satisfiable. Whereas few of the graph having 500 or more vertices are not satisfiable. It means the solution of problem will be depend on the strength of SAT solver as well as encoding technique of *k*-colorable graph to 3-CNF-SAT expression.

REFERENCES

1. Prakash C Sharma and Narendra S Chaudhari, "Polynomial 3-SAT Encoding for K-Colorability of Graph", IJCA Special Issue on Evolution in Networks and Computer Communications (1): 2011, pp 19-24
2. Garey, M. R. and Johnson, D. S., Computers and Interactability: A Guide to the Theory of NP-Completeness, Freeman, San Francisco, 1979.
3. S. A. Cook, "The Complexity of Theorm Proving Procedures," in Proceeding of the ACM Symposium on the Theory of Computing, 2004, pp 151-158.
4. Koen Claessen, Niklas Een, Mary Sheeran and Niklas Sorensson, "SAT-solving in practice", Proceedings of the 9th International Workshop on Discrete Event Systems Goteborg, Sweden, pp 61-67,May 28-30, 2008.
5. Prakash C. Sharma and Narendra S Chaudhari, "A graph coloring approach for channel assignment in cellular network via propositional satisfiability", International Conference on Emerging Trends in Networks and Computer Communications (ETNCC) at Udaipur, 22-24 April 2011, pp 23-26

6. Alexander Tsiatas, “Phase Transitions in Boolean Satisfiability and Graph Coloring”, May 2008, Department of Computer Science, Cornell University,(www.cseweb.ucsd.edu/users/atsiatas/phase.pdf).
7. L. Adleman and K. Manders, “Reducibility, randomness and intractability (abstract)”, in STOC 77: Proceedings of the ninth annual ACM symposium on Theory of computing. New York NY, USA: ACM Press, 1977, pp. 151-163.
8. DIMACS Implementation Challenges, http://dimacs.rutgers .edu/Challenges/
9. Petersen graph, http://en.wikipedia.org/wiki/ Petersen _graph
10. N. Een and N. Sorensson. An extensible sat solver. In Proc. of the 6th Int. Conference onTheory and Applications of Satisfiability Testing, 2003.
11. N. Een and N. Sorensson. MiniSat v1.13 - A SAT Solver with Conict-Clause Minimization. System description for the SAT competition 2005.
12. The MiniSAT page by Niklas Een and N Sorensson. http://minisat.se/
13. MiniSAT User Guide: How to use the MiniSAT SAT Solver by David A. Wheeler. http://www.dwheeler.com/ essays/minisat-user-guide.html
14. Computational Series: Graph Coloring and Its Generalizations, http://mat.gsia.cmu.edu/COLOR04.
15. E. Malaguti, P. Toth, “A survey on vertex coloring problems”, International Transactions in Operational Research 17, 2010, pp 1–34.
16. W.K Hale, “Frequency Assignment: Theory and Applications”, in IEEE Proceeding, Vol.68, no.12, 1980, pp. 1497-1514.