Polynomial 3-SAT Encoding for *K*-Colorability of Graph

Prakash C. Sharma
Indian Institute of Technology
Survey No. 113/2-B, Opposite to Veterinary College, A.B.Road, Village Harnia Khedi, MHOW, Indore (M.P.), India, Pin 453441

Narendra S. Chaudhari
Indian Institute of Technology
Survey No. 113/2-B, Opposite to Veterinary College, A.B.Road, Village Harnia Khedi, MHOW, Indore (M.P.), India, Pin 453441,

ABSTRACT
Graph *k*-Colorability (for $k \geq 3$) Problem (GCP) is an well known NP-Complete problem; till now there are not any known deterministic methods that can solve a GCP in a polynomial time. To solve this efficiently, we go through Propositional Satisfiability, which is the first known NP-Complete problem [3]. However, to use the SAT solvers, there is a need to convert or encode an *k*-colorable graph to 3-SAT first. In this paper, we are presenting a polynomial 3-SAT encoding technique for *k*-colorability of graph. Alexander Tsiatas [1] gave a reduction approach from 3-Colorable graph to 3-SAT encoding. According to [1], total number of clauses generated in 3-CNF-SAT formula for 3-colorable graph $G = (V, E)$ is $((27*|V|) + (256*|E|))$. In our earlier formulation of reduction of *k*-colorable graph to 3-SAT [2], we generalized [1] for *k*-colorable graph and generated $((k^k*(k-2)*|V|) + (2^{2k+2} *|E|))$ clauses in 3-CNF. Here, we present our approach to encode a *k*-colorable graph to 3-CNF-Satisfiability (SAT) formula in polynomial time with mathematical proof. Our formulation generates total $(((k-2)*|V|) + (k*|E|))$ clauses in 3-CNF for *k*-colorable graph. Thus, our formulation is better than approach [1] and [2]. Also, we tested our encoding formulation approach on different graph coloring instances of DIMACS[8][9].

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

1. INTRODUCTION
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 properly color the vertices of *G* is called the chromatic number of *G*, denoted $\chi(G)$. A *k*-coloring of graph *G* is an assignment of colors $\{1,2,\ldots,k\}$ to the vertices of *G* in such a way that neighbors receive different colors. A graph is called *k*-colorable graph if and only if its 3-CNF-SAT formula is satisfiable.

Determining the *k*-colorability of any graph is an NP-Complete problem [3]. Graph Coloring Problem is very important because it has many applications; some of them are planning and scheduling problems such as timetabling, channel assignment in cellular network [2][4][5][6] and many others. Since, till now there are not any known deterministic methods that can solve a GCP in a polynomial time. There is an alternative approach to solve it efficiently by propositional Satisfiability which is first known NP-Complete problem. We can reduce any NP-Complete problem to/from SAT. As per the recent advancement in SAT and due to the availability of efficient SAT solver [7], we proceed for the reduction of the *k*-colorable graph to satisfiability in polynomial time. In our paper, we did reduction of *k*-colorable graph into a 3-CNF-Satisfiability (or SAT). Then, it is illustrated by an example of 3-colorable graph that has to be encoded into 3-CNF-SAT.

Previously, in [1], 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 [2], we mapped graph coloring concept in the field of cellular netwaork to solve channel assignment problem via propositional satisfiability and generalized [1]'s approach of 3-SAT encoding for *k*-colorable graph as $((k^k (k-2)*|V|) + (2^{2k+2} *|E|))$ clauses in 3-CNF-SAT expression.

However, [1] and [2] both gives exponential bounds in terms of number of clauses as 3-CNF-SAT for reducing *k*-colorable graph. Our paper explores polynomial 3-SAT encoding formulation of *k*-colorable graph and gives better results than [1] and [2]. In next section of this paper, we have explored basic detail of 3-SAT and *k*-colorable graph. Section 3 describes our formulation approach of polynomial 3-SAT encoding of *k*-colorable graph. In section 4, we tested our propositional encoding approach on different standard graph coloring instances of DIMACS [8][9].

2. BACKGROUND
2.1 3-Satisfiability (3-SAT)
Let *F* be a 3-CNF formula which is conjunctions of *k* clauses say $C_1 \wedge C_2 \wedge \ldots \wedge C_k$, where each clauses having at most *m* literals i.e. a clause of length-*m* (in case of 3-CNF, *m*=3) with disjunctive form say $l_1 \vee l_2 \vee \ldots \vee l_m$. Each literal may be variables or negation of these variables say $x_1, x_2, \ldots, x_n$ or $\neg x_1, \neg x_2 \ldots \neg x_n$ where $\neg$ indicates negation. Every literal can take a truth value (0 or false, 1 or true). In Satisfiability problem, a set of values for the literals should be found, in such a way that the evaluation of formula should be true; if it is true then formula is called satisfiable, and otherwise formula is unsatisfiable. Following expression *E* is an example of 3-SAT:

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

Here, E has 3 clauses (denoted by parentheses), four variables (x_1, x_2, x_3, x_4), and clause length m=3 (three literals per clause). Here, x_1 is a positive literal and $\neg x_2$ is a negative literal. 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. 3-SAT is an NP-complete problem and it is used as a starting point for proving that other problems are also NP-hard. This is done by polynomial-time reduction from 3-SAT to the other problem.

2.2 *k*-Colorable Graph

Coloring a Graph with k colors or k-Coloring Problem is as follows: Is it possible to assign one of k colors to each vertex of a graph $G = (V, E)$, such that no two adjacent nodes be assigned the same color? If the answer is positive (or YES), we say that the graph is k-colorable and k is the chromatic number of graph G. It is possible to transform k-Coloring problem into Satisfiability problem; that means, for a given graph $G = (V,E)$ and a number k, it is possible to derive a CNF formula F such that F is satisfiable only in the case if G is k-colorable. To convert from k-colorability to 3-SAT, the vertices and end points of edges of graph must be encoded in a Boolean formula as 3-CNF.

3. PROPOSITIONAL 3-CNF-SAT ENCODING OF *K*-COLORABLE GRAPH

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 same color. 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 is presented below:

3.1 Polynomial 3-SAT Encoding Formulation Approach

3.1.1 Vertex Constraint Approach

As per vertex constraint, color each vertex of a graph G as v_{ic} in such a way that vertex should have at least one color among available k-colors as follows:

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

Where v_{ic} is a vertex v_i (i = 1,2,…,n vertices) is colored by available colors c (c = 1,2,…,k colors). 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 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:

$$(x_1 \vee x_2 \vee \neg y_1) \wedge (x_3 \vee y_1 \vee \neg y_2) \wedge (x_4 \vee y_2 \vee \neg y_3) \wedge \ldots \wedge (x_{k-2} \vee y_{k-4} \vee \neg y_{k-3}) \wedge (x_{k-1} \vee x_k \vee y_{k-3}) \tag{2}$$

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.

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

$|F_v| = (k\text{-}2)*|V|$ clauses in 3-CNF-SAT (3)

Where F_v is the conjunction of all the n vertices of graph G which is 3-CNF-SAT encoded by applying vertex constraint approach as:

$$F_v = (v_{1c} \wedge v_{2c} \wedge \ldots \wedge v_{nc}) \tag{4}$$

3.1.2 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 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 be assigned different 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:

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

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 as per edge constraint approach i.e.

$|F_e| = k*|E|$ clauses in 3-CNF-SAT (6)

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

$$F_e = (e_1 \wedge e_2 \wedge \ldots \wedge e_m) \tag{7}$$

3.1.3 Bounds of Final 3-CNF-SAT Formula

To get final 3-CNF-SAT encoded formula F, we conjunct formula obtained by vertex constraint approach (4) and formula obtained by edge constraint approach (7) as:

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

Finally, to get total number of clauses in 3-CNF-SAT formula, we combine (3) and (6) as number of clauses obtained by vertex constraint approach and from 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 i.e.

$|F| = (k\text{-}2)*|V| + k*|E|$ (9)

3.2 Algorithm: *k*-Colorable Graph to *k*-CNF

1. Take Input expression in the form of adjacency matrix of a given graph, through a file. Store the incidence matrix in an two dimensional integer array (say, arr[][]). Since this matrix is symmetrical we will only consider the lower half of this matrix.
2. Read the lower half of this array one by one for each value.
   2.1 If its value is 0, don't do anything
   2.2 If its value is 1, apply the vertex constraint approach (1) and edge constraint approach (5) for those vertices of an edge and then generate encoded formula for same as below:

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

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

3. Take CNF expression obtained by above (2), Count the number of clauses in this expression and store this expression in a file (say, newstr.txt).
4. Convert this string into a two dimensional char array such that each row contains a single clause with its literals separated by "$", store this array in a file (say, arrange1.txt).
5. Store the final expression which is in *k*-CNF in the file (newstr.txt).

3.3 Algorithm: *k*-CNF to 3-CNF (for *k*≥3)

1. To convert *k*-CNF to 3- CNF, we take two files simultaneously newstr.txt and a new file (diff.txt). Read the file newstr.txt clause by clause in which *k*-CNF expressions are stored.
2. Take a clause, store this clause into a one dimensional array, and count the number of literals in this clause.
   2.1 If no. of literals ≤ 3
   Then, write this clause as it is in the new file (diff.txt)
   2.2 If no. of literals > 3
   Then store the literals of this clause in a new array and write the first two literal as it is in the file and then write the expression "zn)&(-zn+literal" till only two literals remain, where n is the count for number of z inserted. Append the last two literals in the file and Write ")" to the file; If it is not the last clause write "&" in the file.
3. Repeat the steps 2 till the end of the file (newstr.txt).
4. Final 3-CNF expression will be stored in the file (diff.txt).

3.4 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)

From 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

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

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

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

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

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

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

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

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

= (*m*-1)*|*V*| + (*m*+1)*|*E*|

From 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.

3.5 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*:

$F = (v_i \wedge e_j)$ It can also be written as:

Where $v_i = v_{1,}\ v_{2,\ldots,}v_n$ are *n* vertices and $e_j = e_{1,}\ e_{2,\ldots,}\ e_m$ are *m* edges of the graph *G* that has to be *k*-colorable by vertex constraint approach and edge constraint formulation respectively. Above expression can also be expanded as:

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

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.

$$F_v = (v_{11} \vee v_{12} \vee \ldots \vee v_{1k}) \wedge (v_{21} \vee v_{22} \vee \ldots \vee v_{2k}) \wedge \ldots \wedge (v_{n1} \vee v_{n2} \vee \ldots \vee v_{nk})$$

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.

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

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.

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

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.

3.6 Illustration by 3-Colorable Graph to 3-CNF-SAT

Here, we are taking an example of Petersen [10] 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:

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

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

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:

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

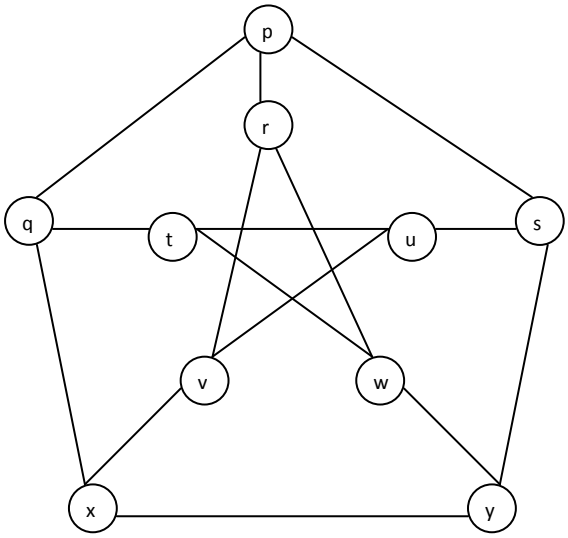

Fig 1: Petersen Graph

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.

4. EXPERIMENTAL RESULTS

We have implemented the above formulation of 3-CNF-SAT encoding corresponding to *k*-colorable graph. We found that there are total ((*k*-2)*|*V*| + *k**|*E*|) clauses of length-3 for any *k*-colorable graph which is polynomial 3-SAT encoding approach and better than earlier approaches [1][2]. Here, we analyzed the encoding formulation for 3-color and 4-color on various benchmark problems (graph coloring instances) of the DIMACS challenge [8][9]. Results are compiled at table 1.

Table 1: 3-CNF-SAT clause generation for color *k*=3 and 4

Graph Coloring Instances	No. of Vertices	No. of Edges	Total 3-CNF-SAT clauses generated (when *k*=3)	Total 3-CNF-SAT clauses generated (when *k*=4)
myciel3	11	20	71	102
myciel4	23	71	236	330
queen5_5	25	160	505	690
mugg100_1	100	166	598	864
myciel5	47	236	755	1038
queen6_6	36	290	906	1232
miles250	128	387	1289	1804
queen7_7	49	476	1477	2002
myciel6	95	755	2360	1700

5. CONCLUSION

In this paper, we have presented an approach using vertex constraint approach and edge constraint approach to encode the *k*-colorable graph in 3-CNF-SAT expressions. By our formulation, *k*-colorable graph is encoded by ((*k*-2)*|*V*| + *k**|*E*|) clauses in 3-CNF-SAT expression; which is polynomial reduction from *k*-colorable graph to 3-CNF-SAT. Whereas, in [1],Alexander Tsiatas has generated total number of clauses in 3-CNF-SAT formula for 3-colorable graph is ((27*|*V*|) + (256*|*E*|)) clauses and [2] generalized it for *k*-colorable with ((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 [1][2] which were exponential.

6. REFERENCES

[1] 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).

[2] 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

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

[4] Daniel Marx, "Graph Colouring Problems and their applications in Scheduling", Periodica Polytechnica Ser El. Eng Vol.48, No.1, pp. 11-16 (2004)

[5] Maaly A. Hassan and Andrew Chickadel, "A Review Of Interference Reduction In Wireless Networks Using Graph Coloring Methods" , International journal on applications of graph theory in wireless ad hoc networks and sensor networks (GRAPH-HOC) Vol.3, No.1, March 2011, pp 58-67.

[6] Mohammad Malkawi, Mohammad Al-Haj Hassan and Osama Al-Haj Hassan, "New Exam Scheduling Algorithm using Graph Coloring", The International Arab Journal of Information Technology, Vol. 5, No. 1, January 2008, pp 80-87

[7] 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.

[8] Graph Coloring Instances, http://mat.gsia.cmu.edu/ COLOR/instances.html.

[9] DIMACS Implementation Challenges, http://dimacs.rutgers .edu/Challenges/

[10] Petersen graph, http://en.wikipedia.org/wiki/ Petersen _gra ph

APPENDIX

For above Peterson's graph, 3-CNF-SAT expression *F* by 3-color can be obtained by conjuncting all the encoded expression of vertices and edges as:

$$F = F_v \wedge F_e$$

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

However, Peterson graph having 10 vertices and 15 edges. When we encoded it by our formulation, we found that the final expression having only 25 clause in 3-CNF for 3-colorable.