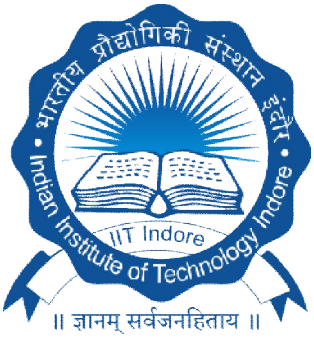
ANALYSIS OF GRAPH COLORING PROBLEM BASED ON SATISFIABILITY AND MAXIMAL INDEPENDENT SET

Ph.D. Thesis

By

PRAKASH CHANDRA SHARMA



DISCIPLINE OF COMPUTER SCIENCE & ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY INDORE

JULY 2018

ANALYSIS OF GRAPH COLORING PROBLEM BASED ON SATISFIABILITY AND MAXIMAL INDEPENDENT SET

A THESIS

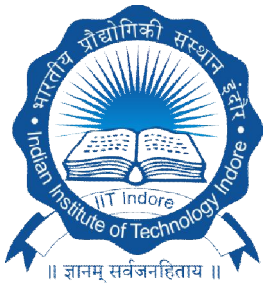
Submitted in partial fulfillment of the requirements for the award of the degree

of

DOCTOR OF PHILOSOPHY

by

PRAKASH CHANDRA SHARMA



DISCIPLINE OF COMPUTER SCIENCE & ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY INDORE

JULY 2018

INDIAN INSTITUTE OF TECHNOLOGY INDORE

CANDIDATE'S DECLARATION

I hereby certify that the work which is being presented in the thesis entitled**ANALYSIS OF GRAPH COLORING PROBLEM BASED ON SATISFIABILITY AND MAXIMAL INDEPENDENT SET**in the partial fulfillment of the requirements for the award of the degree of **DOCTOR OF PHILOSOPHY** and submitted in the **DISCIPLINE OF COMPUTER SCIENCE & ENGINEERING, Indian Institute of Technology Indore**, is an authentic record of my own work carried out during the time period from January 2011 to July 2018 under the supervision of Dr. Narendra S. Chaudhari, Professor, Discipline of Computer Science & Engineering, Indian Institute of Technology Indore.

The matter presented in this thesis has not been submitted by me for the award of any other degree of this or any other institute.

Dated: Signature of the student
(PRAKASH CHANDRA SHARMA)

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Dated: Signature of Thesis Supervisor
(NARENDRA S. CHAUDHARI)

PRAKASH CHANDRA SHARMAhas successfully given his Ph.D. Oral Examination held on…….

Signature of Chairperson (OEB) Signature of External Examiner Signature of Thesis Supervisor
Date: Date: Date:

Signature of PSPC Member #1 Signature of PSPC Member #2 Signature of Convener, DPGC
Date: Date: Date:

Signature of Head of Discipline
Date:

Dedicated to My Parents

Acknowledgements

I would like to take this opportunity to express my heartfelt gratitude to a number of persons who in one or the other way contributed making this time as learnable and bearable. At first, I would like to thank my supervisor Dr. Narendra S. Chaudhari, Professor, IIT Indore for his invaluable guidance, suggestions, encouragement and direction throughout this work. I am thankful to Dr Ram Bilas Pachori and Dr. Anand Parey, my research committee members for their interesting discussions and suggestions towards my research. I am grateful to Dr. Surya Prakash, Head of Department, for his constructive feedback. I would also like to thank Dr. Pradeep Mathur, Director, Indian Institute of Technology Indore, for his help, support, interest and valuable suggestions.

I extend my sincere thanks to government body Department of Science & Technology (DST), Government of India New Delhi and Indian Institute of Technology Indore to help me with financial support to attend international conferences which helped a lot to groom my research work.

I wish to thank all my colleagues and staff from the Discipline of Computer Science and Engineering for their help, motivations, suggestions and friendship. I express my special appreciation to my friends Rajkumar Jain, Varun Bajaj, Jaya Thomas, Neetesh Saxena, RudresDwedi, Navneet, Ashish Jain, Rajat Saxena, Mayank, Nikhil and Rajendrawho helped and supported me in many ways during my thesis work.

I also would like to thank the cooperative staff members Tapes Parihar, Shailendra Verma, Satish Bisen, Ambar Dixit, Prahlad Panwar, and Lalit Jainfor their extended help and support during my thesis.I wish to special thank to (Late) Dr ShreedharBajpei, Manoj Shukla, Dr K K Mehta, Ravindra Singh and Narendra Sharma for their motivation, support and help at all time during PhD thesis completion.

Whatever little I have achieved in life, the credit goes to my family members. I would like to thank my family, especially my mother, father, brother (Deepak Rashmi Sharma), sisters (Jyoti Sunil Shukla andKiran Vipin Sharma) and my family for always believing in me, for their continuous source of inspiration andtheir support in my decisions.

I take great pride in dedicating my thesis tomy parents(Maa-Babuji). They have kept trust and faith in whatever I did.Last but not the least; I would also like to thank GOD for showering upon me his blessings since my childhood.

I am delighted to thank all my friends and well wishers whose name I am missing to mention here and who directly or indirectly walked along with me throughout.

Thank you all.

Prakaash Sharma

Abstract

Nowadays, everywhere resource scheduling is an important task. In general, it is observed that resources are limited and users are quite more than resources; then question is how to maximize utilization of resources without conflict or with minimum conflict. The graph coloring problem is mainly used for resource scheduling. A *k*-colorability of graph *G* is an assignment of colors {1,2,…,*k*} to the vertices of *G* in such a way that neighbor vertices of graph should not receive the same colors. The minimum number of colors needed to properly color the vertices of *G* is called the chromatic number of *G*.Graph coloring problem has several important real-world applications including register allocation problem, channel assignment problem in cellular network, time tabling problem, aircraft scheduling problem, etc. Since graph coloring problem is an NP-Complete problem; therefore no exact solution could be found for large graph. There is so many heuristic algorithm used to find out approximate solution till date.

This thesis presents two variations of solution approach for graph coloring problem. First is optimization based solution for graph coloring problem and second one is decision based solution for graph coloring problem.

In the optimization variation of graph coloring problem, its goals to calculate the minimum possible colours*k*, so that a propercolouring of graph*G*could be possible. A *k*-colorable graph divides an array of vertices *V* into *k* dissimilarcolor classes, where each member of the class has the same color. In order to have the same color, the members of each class must be pairwise non-adjacent, which by definition makes them an independent set. In our thesis, we presented an algorithm of finding maximal independent sets from the initial graph and it gives solution for graph coloring problem.

Satisfiability (SAT) is recognized as the first NP-Complete problem and one of the classic problems in computational complexity. Since, the Satisfiability problem (SAT) is interesting because it can be used as a stepping stone for solving decision problems. In our thesis, we presented Satisfiability (SAT) based solution approach for decision based graph coloring problem. In the form of a decision problem, graph *k*-colorability problem can be stated as follows: Is it possible to assign one of the *k* colors to vertices of a graph $G = (V, E)$, such that no two adjacent nodes are 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*; otherwise it returns “unsatisfiable”.

Satisfiability (SAT) is used as a starting point for proving that other problems are also NP-hard. We can reduce any NP-Complete problem to/from SAT. Therefore, in our thesis we presented a generalized reduction approach for *k*-colorable graph to/from 3-CNF-SAT

In our thesis, we presented a polynomial 3-SAT encoding technique of *k*-colorable graph. This approach introduces two coloring constraint say vertex coloring constraint and edge coloring constraint for proper coloring of a graph. Since, there have been dramatic improvements in SAT solver technology over the past decade. This has lead 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.In thesis, we analyze an efficient SAT solver MiniSAT to investigate SAT based solution of graph *k*-colorability problem. Encoded 3-SAT expression will be input for SAT solver and then it gives decision based solution.

In this thesis, we have analyzed the behavior of two NP-Complete problem say 3-Satisfiability and Graph 3-Colorability during reduction from each other with the help of phase transition phenomenon.

Since, the channel assignment problem is very similar to the graph *k*-colorability problem. Reduction from graph *k*-colorability problem to satisfiability is an important concept to solve channel assignment in cellular network. In our thesis, we mapped a cellular network with frequency assignment and then introduced a 3-SAT encoding of channel assignment problem.

List of Publications

International Journals

1. Sharma P.C., Chaudhari N.S., A Tree based Novel Approach for Graph Coloring Problem using Maximal Independent Set, **Springer**Journal of Wireless Personal Communication. (Under Review: Submitted on August 2017)

2. Sharma P.C.**,** Chaudhari N.S. (2016), Investigation of Satisfiability Based Solution Approach for Graph Coloring Problem,International Journal of Engineering and Advance Technology (IJEAT), 6(1),106-112.

3. Sharma P.C., Chaudhari N.S. (2015),Maximal Independent Set Based Approach for Graph Coloring Problem, International Journal of Computer Engineering and Application (IJCEA), 9(2), 205-214.

4. Sharma P.C., Chaudhari N.S. (2012), A New Reduction from 3-SAT to Graph K-Colorability for Frequency Assignment Problem,International Journal on Computer Application, Special Issue on Optimization and On-chip Communication ooc(1), 23-27.

5. Sharma P.C., Chaudhari N.S. (2011), Polynomial 3-SAT Encoding for *k*-Colorability of Graph,International Journal on Computer Application, Special Issue on Evolution in Networks and Computer Communications (1), 19-24.

International Conferences

1. Sharma P.C., Chaudhari N.S. (2012), Phase Transition in Reduction between 3-SAT and Graph Colorability for Channel Assignment in Cellular Network, 4thIEEE International Conference on Computational Intelligent and Communication Networks (CICN) in Mathura, India, pp 164-168.

2. Sharma P.C., Chaudhari N.S. (2012), Channel Assignment Problem in Cellular Network and Its Reduction to Satisfiability using Graph *k*-Colorability,7th IEEE Conference on Industrial Electronics and Application (ICIEA)in**Singapore,** pp 1734-1737.

3. Sharma P.C., Chaudhari N.S. (2011), A graph coloring approach for channel assignment in cellular network via propositional satisfiability,International Conference on Emerging Trends in Networks and Computer Communications (ETNCC) in Udaipur, India, pp 23-26.

National Convention

1. Sharma P. C., Chaudhari N.S. (2011), Cellular Networks, *k*-Coloring and 3-SAT, 25th National Convention of Computer Engineers and National Seminar on Networked Home Systems and Services (NHSS) in Udaipur, India, pp 57-58.

Contents

List of Figures xvii

List of Tables xix

List of Abbreviations xxi

1. Introduction 1

1.1 Motivation..……3

1.2 Objectives……..4

1.3 Contribution…… 5

1.4 Organization of the Thesis.....6

2. Background Details 9

2.1 NP Complete Problem……9

2.2 Satisfiability Problem……...11

2.3 Graph Coloring Problem……..13

2.4 Channel Assignment Problem……..13

2.5 Independent Set……15

2.6 Analysis ofSome Heuristic Approaches for Graph Coloring Problem……15

2.6.1 Greedy Algorithm……..……...16

2.6.2 First Fit…….……17

2.6.3 Largest-Degree-First-Ordering.……17

2.6.4 Smallest-Degree-Last–Ordering……..17

2.6.5 Incidence-Degree-Ordering…….18

2.6.6 DSATUR (Degree of Saturation) Approach……18

2.6.7 Recursive Largest First (RLF) Approach.……19

2.6.8 Parallel Maximal Independent set……19

2.6.9 Jones-Plassmann Approach……..20

2.7 Related Work……21

2.7.1 Reduction of 3-colorable graph to 3-CNF-SAT……...21

2.7.2 Reduction of 3-CNF-SAT to 3-Colorable Graph.…….22

2.7.3 Approaches Based on Independent Set Extraction……..22

3. A Tree Based Novel Approach for Graph Coloring Problem using Maximal Independent Set 25

3.1 Introduction ……25

3.2 Review of previous work ……27

3.3 Our Proposed Algorithm: Tree Based Maximal Independent Set …..28

3.3.1 Complementary Edge Table ……28

3.3.2 Finding Maximal Independent Sets: Tree Exploration……29

3.3.3 Coloring the Maximal Independent Sets…….30

3.4 Illustration by an Example…..……...……...30

3.4.1 Creating Complementary Edge Table …….30

3.4.2 Finding Maximal Independent Set by Tree Exploration ……31

3.4.3 Coloring the Maximal Independent Set …….33

3.5 Algorithm ……... ……... 34

3.5.1 Notations used in our algorithm……...34

3.5.2 Algorithm for finding Maximal Independent Sets (MIS) ….. 34

3.6 Results and Discussion ……35

3.7 Summary …….. 37

4. Polynomial 3-SAT Encoding Technique for *k*-colorable Graph and Analysis of Graph Coloring Problem based on Satisfiability 39

4.1 Introduction…… 39

4.2 Polynomial 3-SAT Encoding Formulation Approach of *k*-Colorable Graph……41

4.2.1 Vertex Constraint Approach ……41

4.2.2 Edge Constraint Approach ……42

4.2.3 Bounds of Final 3-CNF-SAT Formula ……..43

4.2.4 Algorithm: k-Colorable Graph to 3-CNF …… 43

4.2.5 Bounds on number of clauses in 3-CNF expression …… 44

4.2.6 Justification of Propositional Encoding Formulation of *k*-Colorable Graph ……..46

4.3 Illustration of Encoding of 3-Colorable Graph to 3-CNF-SAT …….47

4.4 Solution Approach for Graph K-Colorability using SAT Solver …48

4.5 Results and Discussion ……49

4.6 Summary ……54

5. A New Reduction from 3-SAT to Graph K-Colorability for Frequency Assignment Problem 55

5.1 Introduction ……55

5.2 Polynomial reduction from 3-CNF-SAT to k-colorable graph ……..56

5.2.1 Reduction of 3-SAT to Graph 3-Colorability (3-SAT ≤p 3-Color) ……..57

5.2.2 Reduction of 3-SAT to Graph 4-Colorability (3-SAT ≤p 4-Color) ……..59

5.2.3 Reduction of 3-SAT to Graph 5-Colorability (3-SAT ≤p 5-Color) ……..61

5.2.4 Reduction of 3-SAT to Graph *k*-Colorability (3-SAT ≤p *k*-Color) ……..63

5.3 Graph k-colorability to frequency assignment problem ……64

5.4 Summary ……65

6. Phase Transition in Reduction between 3-SAT and Graph Colorability 67

6.1 Introduction ……..67

6.2 Phase Transition ……68

6.3.1 Phase Transition in 3-SAT 69

6.3.2 Phase Transition in 3-Colorability …… 70

6.3 Phase transition of reduced 3-colorable graph corresponding to 3-SAT instance ……. 70

6.4 Phase transition of reduced 3-CNF-SAT corresponding to 3-Colorable Graph ……..72

6.5 Results and Discussion ……74

6.6 Summary ……75

7. Channel Assignment Problem in Cellular Network and its Reduction to Satisfiability using Graph K-Colorability 77

7.1 Introduction ……77

7.2 Channel Assignment Problem ……78

7.3 Problem Formulation: Channel Assignment Problem as Graph *k*-colorability ……..79

7.4 Reduction Approach to Satisfiability using Graph k-Colorability …..81

7.4.1 Base Station Constraint Approach ……. 81

7.4.2 Interference Constraint Approach …… 82

7.4.3 Maximum bound of generated 3-CNF-SAT Formula ……83

7.5 Illustration by an Example…….. 84

7.6 Results and Discussion……85

7.7 Summary ……85

8. Conclusion and Scope for Future Work 87

8.1 Conclusion ……87

8.2 Scope for Future Work ……88

Appendix A 89

Bibliography 93

List of Figures

1.1 (a) 3-colorable graph ……..1

1.1 (b) 4-colorable graph …….1

2.1 Relationship between P and NP class problem……..9

3.1 Properly Colored 5 Vertex Star Graph……..25

3.2 Petersen Graph …… 30

3.3 (a)First Maximal Independent Set……..32

3.3 (b) Second Maximal Independent Set……. 32

3.3 (c)Third Maximal Independent Set…….33

3.4 Properly colored Petersen graph given in figure 3.2 using our proposed approach……...34

4.1 Petersen Graph…… 47

4.2 Analysis of 3-CNF-SAT clause generation for $k = 3$……...51

4.3 Analysis of 3-CNF-SAT clause generation for $k = 4$……...51

5.1 Reduced 3-Colorable Graph.……59

5.2 Reduced 4-Colorable Graph.……61

5.3 Reduced 5-Colorable Graph.……63

6.1 Generated 3-colorable graph from an instance of 3-CNF-SAT……...71

7.1 Mapping of graph k-colorability and channel assignment problem in cellular network.……80

7.2 Small Channel Assignment Problem Instance…….84

List of Tables

2.1 Comparison of sequential algorithm in terms of number of colors required for graphs…… 20

2.2 Comparison of parallel algorithm in terms of average number of colors required for graphs…… 21

2.3 Comparison of parallel algorithm in terms of time taken in seconds for coloring graphs…….21

3.1 Edge and Complementary Edge Table for Figure 3.2…….....31

3.2 Comparison of our result with E2COL approach [2], DSATUR approach [16] and Malguti's approach [17]……36

4.1 3-CNF-SAT clause generation for color k=3 and 4…….50

4.2 Results of Some DIMACS Graph Instances which are encoded as 3-CNF-SAT by our reduction approach and then solved by Minisat 2.2…….52

4.3 Results of Some UNSAT graph instance given by Minisat 2.2……...53

6.1 Comparisons of Phase Transition of reduced 3-SAT and 3-Colorable Graph with standard and previously generatedphase transition values……..74

List of Abbreviations

SAT: Satisfiability

CNF: Conjuctive Normal Form

DNF: Disjunctive Normal Form

GCP: Graph Coloring Problem

G*k*CP: Graph *k*-Colorability Problem

CAP: Channel Assignment Problem

MIS: Maximal Independent Set

NPC: Non Deterministic Polynomial-Complete Problem

ET: Edge Table

CET: Complementary Edge Table

Chapter 1

Introduction

A Graph *k*-colorability is an assignment of colors {1,2,…,*k*} to the vertices of a 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. A graph *G* is called *k*-colorable, if there exists a legal coloring with at most *k* colors and the minimum number of colors needed to color the vertices of the graph *G* is called the chromatic number of *G*, denoted as χ (*G*). Following figures 1(a) and 1(b) show a 3-colorable and 4-colorable graph respectively. Graph *k*-colorability problem (for $k \geq 3$) is among the 21 NP-complete problems [10][25] originally given by Richard Karp in the year 1972.

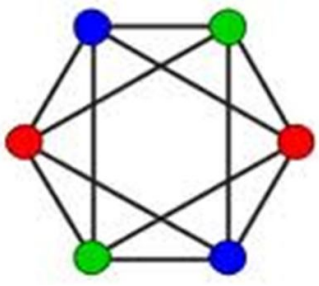


Figure 1(a): 3-colorable graph

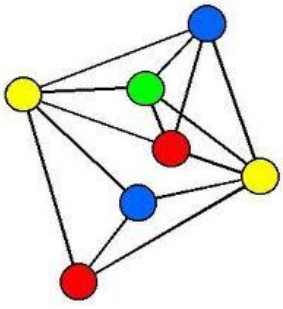


Figure 1(b): 4-colorable graph

The wide range of applications along with its combinatorial complexity elevates the graph coloring problem to one of the most famous and most researched problems in graph theory. Generally, the GCP is NP-complete [10] [25] and we currently know of no efficient algorithm for large graphs.

There are many approaches proposed to solve the graph coloring problem (GCP) till date. Reviews of major approaches on graph coloring problem have been given in [41]. An alternative approach to solve this problem is by satisfiability (SAT). Satisfiability (SAT) is the first known NP-complete problem [25]. The SAT problem is usually expressed in conjunctive normal form (CNF). A CNF formula on binary variables is the conjunction of clauses; each clause is a disjunction of one or more literals, where a literal is the occurrence of a variable or its complement. In general, the SAT problem is defined as follows: Given a Boolean formula in CNF, find an assignment of variables that satisfies the formula or prove that no such assignment exists.

According to Cook's theorem [10][25], we can reduce any NP-complete problem to/from SAT. Then obtained SAT expression can be solved separately by efficient SAT solver. Graph *k*-colorability problem as a decision problem targets the question if there exists a proper coloring for a given graph *G* and number of colors *k*. In this thesis, we have formulated a generalized reduction approach for the graph *k*-colorability to/from 3-CNF-SAT and then investigated SAT based approach for solving graph coloring problem.

In the optimization version, the graph coloring problem aims to find the lowest possible number of colors *k* under which a feasible coloring of *G* is possible. A *k*-colorable graph partitions a set of vertices *V* into *k* different color classes, where each member of the class has the same color. In order to have the same color, the members of each class must be pairwise non-adjacent, which by definition makes them an independent set. Therefore, finding maximal independent sets from a graph is an efficient approach for solving the graph coloring problem. Review of approaches for graph coloring using independent set is given in [2]. In our thesis, we have developed a tree based novel approach for graph coloring problem using maximal independent set.

1.1 Motivation

Graph coloring is a fundamental and extensively studied problem, which has its theoretical significance. The colouring problem of a graph is a famous problem category of NP-hard combinatorial optimization [10]. The graph *k*-colorability problem has several important real-world applications [46] including computer register allocation [11], timetabling and scheduling [12], frequency assignment problem [13, 42], and satellite range scheduling [14], time tabling problem [48] and aircraft scheduling problem [42] etc.

The Satisfiability problem is particularly interesting because it can be used as a stepping stone for solving decision problems. The graph coloring problem can also be solved as a decision based using the method of Satisfiability (SAT). Problem instances from domains such as Graph Coloring can be encoded into SAT and then solved by the help of SAT algorithms.

Since the channel assignment problem is very similar to the graph *k*-colorability problem [42]. But, till now there are not any known deterministic methods that can solve a graph *k*-colorability problem (GCP) (or any NP-complete problem) in a polynomial time. There is an alternative approach to solve it efficiently by propositional Satisfiability which is the first known NP-complete problem. The satisfiability problem (SAT) is one of the most prominent problem in theoretical computer science, which has become increasingly popular and important insights into our understanding of the fundamentals of computation. It is used as a starting point for proving that other problems are also NP-hard. We can reduce any NP-Complete problem to/from SAT. Since, there have been dramatic improvements in SAT solver technology over the past decade. This has lead 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 satisfiability is an important concept [62] to solve channel assignment problem in cellular network.

In the optimization form, the graph coloring problem, goals to calculate the lowest possible colours *k*, so that a realistic colouring of *G* could be possible. A *k*-colorable graph divides an array of vertices *V* into *k* dissimilar color classes,

where each member of the class has the same color. In order to have the same color, the members of each class must be pair wise non-adjacent, which by definition makes them an independent set. Therefore, finding maximal independent sets from the initial graph is an effective method for solving graph coloring problem.

Why coloring a graph? Graph colouring originates from the colouring of maps of countries and counties which can be represented as "planar graphs". All planar graphs (and maps) can be coloured using just four colours. However, other types of graphs require different numbers of colours. Many different algorithmic schemes have been developed for graph colouring. However, only a limited set of benchmark instances are typically considered in comparisons (e.g. DIMACS). Comparisons between algorithms are also difficult to draw because: (a) different experimental conditions are used (b) Often only the good results are reported (c) Researchers choose their own cut-off points and only report final (best) solutions.

1.2 Objectives

In our thesis, we analyzed the graph coloring problem based on maximal independent set and Boolean Satisfiability (SAT). Our research work mainly focuses on polynomial reduction approach of graph *k*-colorability to/from 3-CNF-SAT expression and then analyzed SAT based solution approach of graph coloring problem. However, the phase transition phenomenon is often associated with the hardness of complexity; therefore we investigated the phase transition of an encoded 3-colorable graph and generated graph from 3-SAT expression. Since all NP-complete problems can translate into one another, a study of phase transition gives a better understanding of NP-complete problems. Since, it is shown that the channel assignment problem in cellular network is similar to the graph *k*-colorability problem [42]; therefore, in our thesis, channel assignment problem (CAP) in a cellular network has also been reduced to satisfiability (SAT) using graph *k*-colorability. In order to realize these general aims, specific objectives of our research work are as below:

- To develop a tree based novel approach for graph coloring problem using maximal independent set.

- To formulate the generalized polynomial encoding technique for the reduction of graph *k*-colorability (for $k \geq 3$) to 3-CNF-SAT expression.

- To investigate the satisfiability (SAT) based approach for solving graph *k*-colorability problem.

- To formulate the generalized polynomial reduction of 3-CNF-SAT expression to *k*-colorable graph.

- To analyze and calculate the Phase Transition of a generated graph from 3-CNF-SAT expression and 3-CNF-SAT encoding of 3-colorable graph using our reduction method of 3-SAT to/from 3-colorable graph. By phase transition concept, we can discuss the hardness complexity of our above proposed reduction approach of 3-SAT to/from graph *k*-colorability.

- To formulate a reduction approach of channel assignment problem in cellular network into 3-CNF-SAT using graph *k*-colorability.

1.3 Contributions

In this research thesis, we addressed a number of issues associated with graph coloring problem, Satisfiability, and maximal independent set. We improved the state of knowledge in the following ways:

- A complementary edge table is introduced so that adjacency list of an input graph should be small; especially for the dense graph. A novel approach has been proposed for calculating independent set and then maximal independent set within a given graph and finally each maximal independent set has to assign a different color.
- Analyzed existing encoding technique of graph *k*-colorability to 3-CNF-SAT and developed two new constraints say vertex coloring constraint and edge coloring constraint to encode a graph as 3-CNF-

SAT expression. We encoded standard graph instances DIMACS using our approach.

- Generated 3-CNF-SAT expression from DIMACS graph has to be taken as input for SAT solver; so that decision based graph coloring problem can be solved using SAT algorithms. Here, we have taken an efficient SAT solver MiniSAT to analyze the encoded 3-CNF-SAT expression for finding solution of decision based graph coloring.
- Studied reduction of 3-CNF-SAT expression into 3-colorable graph and analyze it. We proposed reduction formula for 4, 5-colorable graph and in similar manner we generalized a polynomial reduction approach of 3-CNF-SAT to graph *k*-colorability.
- Discuss the concept of phase transition and standard phase transition of 3-SAT and 3-colorability problem. We proposed the calculation of the phase transition of systematically generated 3-colorability graph from 3-SAT and analysis of phase transition of encoded 3-SAT of 3-colorable graph.
- On basis of similarity in channel assignment problem with graph *k*-colorability problem, we mapped channel assignment problem with graph *k*-colorability. Then, we proposed an encoding technique of channel assignment problem into 3-CNF-SAT expression via graph *k*-colorablity concept.

1.4 Organization of the Thesis

In this thesis, we analyzed the well known graph coloring problem based on maximal independent set and satisfiability. The thesis is organized as follows: In the present chapter, the state-of-the –art topics covered in various chapters of the thesis have been enumerated.

Chapter 2 details the necessary background material required to understand the chapters have been presented. The chapter starts with the introduction of graph coloring problem where the characterization is governed by the problem features. Background study is presented for Graph Coloring Problem, Boolean Satisfiability, Satisfiability Solvers, Independent Set, and Channel Assignment Problem. The chapter also introduces and presents a brief

literature survey on graph coloring problem using maximal independent set and Satisfiability.

Chapter 3 illustrates the algorithm of finding maximal independent set from the given graph. In this chapter, a *k*-colorable graph divides an array of vertices *V* into *k* dissimilar color classes, where each member of the class has the same color. In order to have the same color, the members of each class must be pair wise non-adjacent, which by definition makes them an independent set. In this chapter, we have developed a tree based innovative approach for the graph coloring problem using maximal independent set.

Chapter 4 gives a generalized polynomial 3-CNF-SAT encoding technique of *k*-colorable graph. Vertex constraint and edge constraint approach for 3-SAT encoding of a graph has been discussed in this chapter. Also, justification of propositional encoding formulation of *k*-colorable graph is explored here. In this chapter, this approach is illustrated by a 3-ccolorable graph.

Chapter 5 explores polynomial reduction approach from 3-CNF-SAT to *k*-colorable graph. In this chapter, reduction approach of 3-CNF-SAT to 3-, 4- and 5-colorable graph has been discussed. At last, a generalized approach for reduction of 3-SAT expression to graph *k*-colorability is developed.

Chapter 6 calculates and analyze phase transitions of generated 3-CNF-SAT and 3-colorable graph using our reduction method of transforming 3-SAT to/from 3-Colorable graph. Then compare calculated phase transition with known phase transition. Since all NP-complete problems can translate into one another, study of phase transition gives a better understanding of NP-complete problems.

Chapter 7 explores about channel assignment problem in cellular network; also map this problem with graph coloring problem. This chapter illustrates the reduction approach of channel assignment problem to graph *k*-colorability instance using vertex and edge constraints. This chapter tells about the approach to solve channel assignment problem using SAT solver on basis of encoded 3-SAT expression.

Chapter 8 gives a summary of the work undertaken and provides a number of conclusions based on the results in the thesis. It also outlines recommendations for future research in this area.

Chapter 2

Background Details

2.1 NP-Complete Problem

P Class Problem: A problem which can be solved in polynomial time is known as P-class problem. For example: all sorting and searching algorithms.

NP Problem: A problem which cannot be solved in polynomial time but it is verified in polynomial time, is known as non deterministic polynomial or NP class problem. For example: Sudoku problem, Prime factor, Scheduling, Traveling Salesman problem etc. P class problem are tractable problems whereas NP class problem are intractable.

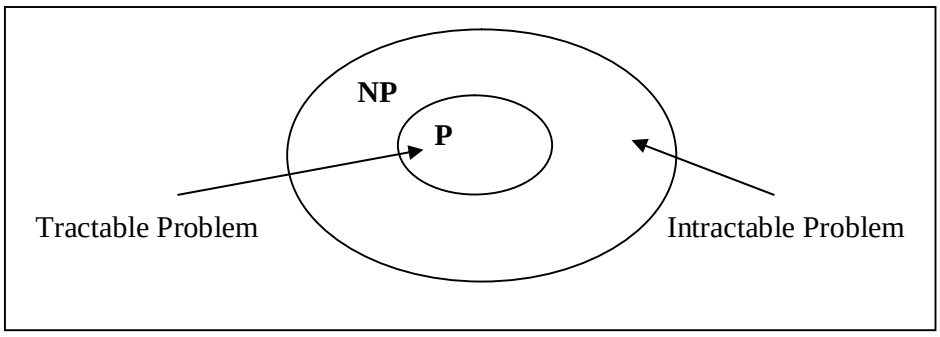


Figure 2.1 Relationship between P and NP class problem

Cook's Reducibility Concept: Let A and B are two problem then problem A reduces to problem B iff there is a way to solve A by deterministic algorithm that solve B in polynomial time. If A is reducible to B, then we denote it by $A \propto B$.

Reduction Properties:

(i) If problem A is reducible to problem B and B is solvable in polynomial time then A will also solvable in polynomial time.

(ii) If problem A is not solvable in polynomial time then it implies problem B will also not solvable in polynomial time.

NP Hard Problem: A problem is NP hard if every problem in NP can be polynomial reduced to it.

NP Complete Problem (NPC): A problem is NP complete if it is in NP and it is NP hard. All NPC problems are NP-hard but all NP-hard problems are not NPC. All the NPC problem is decision problem and the entire NP-hard problem are optimization problem.

In computational complexity theory, the complexity class NP-complete [10,52] (abbreviated NP-C or NPC) is a class of decision problems. A decision problem *L* is NP-complete if it is in the set of NP problems so that any given solution to the decision problem can be verified in polynomial time, and also in the set of NP-hard problems so that any NP problem can be converted into *L* by a transformation of the inputs in polynomial time.

NP-complete problem is a subset of NP, the set of all decision problems whose solutions can be verified in polynomial time. NP may be equivalently defined as the set of decision problems that can be solved in polynomial time on a nondeterministic Turing machine. A problem A in NP is also in NPC if and only if every other problem in NP can be transformed into problem B in polynomial time.

Formal definition of NP-completeness

A decision problem *C* is NP-complete if:

1. *C* is in NP, and

2. Every problem in NP is reducible to *C* in polynomial time.

C can be shown to be in NP by demonstrating that a candidate solution to *C* can be verified in polynomial time.

A problem *K* is reducible to *C* if there is a polynomial-time many-one reduction, a deterministic algorithm which transforms any instance $k \in K$ into an instance $c \in$ C, such that the answer to *c* is *yes* if and only if the answer to *k*

is *yes*. To prove that an NP problem *C* is in fact an NP-complete problem it is sufficient to show that an already known NP-complete problem reduces to *C*. Note that a problem satisfying condition 2 is said to be NP-hard, whether or not it satisfies condition 1.

2.2 Satisfiability (SAT) Problem

The Boolean Satisfiability Problem (SAT) is one of the most important and extensively studied problems in Computer Science and Engineering. In practice, SAT is a core problem in many applications such as Electronic Design Automation (EDA) and Artificial Intelligence (AI).

Given a Boolean formula, the problem of determining whether there exists a variable assignment that makes the formula evaluate to true is called the satisfiability problem. If the formula is limited to only contain logic operations *and*, *or* and *not*, then the formula is said to be a propositional Boolean formula. Determining the satisfiability of a propositional Boolean formula is called the Boolean Satisfiability Problem (SAT). Given a propositional Boolean formula, the SAT problem asks for an assignment of variables such that the formula evaluates to true, or a proof that no such assignment exists. SAT was the first problem shown to be NP-Complete [30].

The SAT problem is usually expressed in conjunctive normal form (CNF). A CNF formula on binary variables is the conjunction (AND) of clauses each of which is a disjunction (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, unit if all but a single literal are set to false, and unresolved otherwise. 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 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, 3 clauses; each clause having at most 3 literals (length of clause=3 i.e. 3-CNF) and 7 literals:

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

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.

Why almost all SAT solvers use CNF instead of DNF? Basically, a DNF formula is a disjunction of clauses $(c_1 \vee c_2 \vee \ldots \vee c_m)$, where each clause $c_i = (l_{i,1} \wedge \ldots \wedge c_{i,k})$ is a conjunction of literals. It seems that solving SAT is easier using DNF. But, let's call a clause c_i conflicting if and only if it contains both a literal l and its negation $\neg l$. A formula may have exponentially many solutions, so the corresponding DNF formula may have exponentially many clauses. After converting DNF to CNF we find that CNF is compact, while DNF is not; CNF is implicit, while DNF is explicit. An NP-Complete problem can be expressed in DNF as: Given a DNF instance is there an assignment of variables that falsifies all the clauses? In other words, if we have to get an optimal or accurate solution of NP-Complete problem then either SAT formula should be in CNF or in DNF with falsification; otherwise DNF cannot give an efficient solution.

The last few years have seen significant advances in Boolean satisfiability (SAT) solving. These advances have lead to the successful deployment of SAT solvers in a wide range of problems in Engineering and Computer Science. The first SAT solving algorithm is often attributed to Davis and Putnam, who proposed an algorithm that can solve general SAT problems in 1960 [8]. Since then, numerous algorithms and techniques have appeared in the literature to improve the efficiency of SAT solving. Because of its NP-Complete nature, it is unlikely that there exist algorithms that can solve SAT in polynomial time in the size of the instance description (unless P=NP). SAT instances with hundreds or even thousands of variables can often be solved by current state-of-the-art SAT solvers in seconds or minutes.

2.3 Graph Coloring Problem

Graph coloring was among the 21 NP-complete problems [3, 10, 62] originally given by Richard Karp in the year 1972. Graph coloring problem states that, given a graph G (V,E) where V is the set of vertices of the graph and E is the set of edges, how many colors are required to color the graph in such a way that no two adjacent vertices of the graph are colored with the same color. A coloring using at most k colors is called a (proper) k-coloring. The smallest number of colors needed to color a graph G is called its chromatic number, $\chi(G)$. 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 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, including register allocation, scheduling like frequency assignment, time tabling problem, aircraft scheduling and many other problems.

Unfortunately, determining the chromatic number of a graph is an NP-hard problem, hence we cannot expect to solve it efficiently for large graphs. So during modeling, it might happen that the graph of our application has some structure that makes coloring easier or if there is no hope for an efficient algorithm for coloring we give an approximation algorithm which does not give optimal solution but has some performance guarantee on quality of produced solution.

2.4 Channel Assignment Problem in Cellular Network

The assignment of channels to cells or mobiles is one of the fundamental resource management issues in a mobile communication system. A channel assignment problem [42][47][50] or the frequency assignment problem is nothing but the task of assigning frequency or channel from a frequency spectrum to a set of transmitters and receivers satisfying certain hard conditions. Channels are assigned to the cells or base stations such that

communication via these stations does not cause interference. Interference generally occurs when the same or close frequencies are assigned to stations that are situated near each other.

The channel assignment problem can be stated as follows: Given a set of n cells or base stations, a set of k channels and a set of interference constraints, assign each station a channel without violating any interference constraint using limited span of frequency spectrum. The channel assignment problem is more complicated than the graph coloring problem in the sense that an interference constraint does not just express that a pair of stations must be assigned different channels, but it also specifies a minimal required distance.

Channel assignment problem (CAP) is classified as an NP-complete problem [42][50], which means that as the size of the problem increases, the time required to solve the problem does not increase in a polynomial manner, but rather in an exponential one. These channels must be placed some distance apart in order to avoid interference. The assignment of channels to cells or mobile is one of the fundamental resource management issues in a mobile communication system.

It is shown that the channel assignment problem is similar to the graph k-colorability problem [50]. Determining the k-colorability of any graph is also an NP-Complete problem [42][50]. A k-colorability of graph G is an assignment of colors $\{1,2,\ldots,k\}$ to the vertices of 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 properly color the vertices of G is called the chromatic number of G, denoted $\chi(G)$.

Since, till now there are not any known deterministic methods that can solve a graph k-colorability problem (GCP) or any NP-complete problem in a polynomial time. There is an alternative approach to solve it efficiently by propositional Satisfiability which is first known NP-Complete problem. The satisfiability problem (SAT) is one of the most prominent problems in theoretical computer science, which has become increasingly popular and

important insights into our understanding of the fundamentals of computation. It is used as a starting point for proving that other problems are also NP-hard. We can reduce any NP-Complete problem to/from SAT. Since, there have been dramatic improvements in SAT solver technology over the past decade. This has lead 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 satisfiability is an important concept to solve channel assignment in cellular network.

2.5 Independent Set

An independent set (also known as a stable set) is a sub groups of vertices $S \subseteq V$ so that none of the vertices in *S* are neighbors. The subsets of the graph containing those vertices that are not attached, i.e. none of the element in the set are connected to any other element of the same set, are known as an independent set. The highest cardinality of a stable set of *G* is represented by $\alpha(G)$. A stable set is called maximal, if it is not a subset of any bigger independent set and it is assumed maximum if there is no bigger independent set within the graph.

A clique is a subset of *V* where all the vertices are pair wise adjacent. We can infer the complement graph $G' = (V, E')$ from a graph $G = (V, E)$, where $E' = \{(i, j) \mid i, j \in V, i \neq j \text{ and } (i, j) \notin E\}$. The Maximum independent set problem (MIS) is to determine an independent set in *G* of highest cardinality $\alpha(G)$. It is well understood that if *I* is an independent set of *G*, then *I* is a clique of *G'*.

2.6 Analysis of Some Heuristic Approaches for Graph Coloring Problem

The word heuristic is used for algorithms which find solutions among all possible solutions, but the solution found will be best is not sure, therefore they are considered as approximate algorithms. These algorithms, usually find a solution near to the best one and they find it fast and easily. Heuristic

algorithms are used when a feasible solution is required rapidly. Well known examples of heuristic algorithm are Traveling Salesman Problem and Knapsack Problem.

There are many approaches available to solve graph coloring problem. These approaches are used to reduce the time complexity of the algorithm and the number of colors used in graph. Here, we analysis the solution approaches of graph coloring problem in two ways; first in sequential way and second in parallel way. In sequential algorithm a node is selected according to some predefined criterion and then colored with legal color. The selection and coloring continues until all the nodes in the graph are colored. Here we study of few sequential algorithm of graph coloring problem such as: Greedy Algorithm, First Fit, Largest-Degree-First-Ordering, Incidence-Degree-Ordering, and Saturation- Degree-Ordering.

In parallel graph coloring, a number of the existing fast heuristics is based on the observation that an independent set of nodes can be colored in parallel. Depending on how the independent set is selected and colored, there are many parallel graph coloring techniques such as: Parallel Maximal Independent set (PMIS) also known as Luby's maximal independent set finding algorithm]. Other variants are the asynchronous parallel heuristic by Jones and Plassmann (JP). Some well- known sequential graph coloring algorithms like the Largest- Degree-First algorithm and the Smallest-Degree-Last algorithm has been parallelized.

2.6.1 Greedy Algorithm

Greedy algorithm is one of the simplest but most fundamental heuristic algorithms for graph colouring. The algorithm operates by taking vertices one by one according to some (possibly arbitrary) ordering and assigns each vertex its first available colour. Because this is a heuristic algorithm, the solutions it produces may very well be suboptimal; however, it can also be shown that GREEDY can produce an optimal solution for any graph. Sequential greedy algorithm [75] plays a significant role in the practical resolution of NP-hard problems. A greedy algorithm is a basic heuristic that finds a result by iteratively adding the locally best element into the solution as per pre defined

criteria. Greedy approach color the nodes of the graph by consider them in sequence and allocate each node the first available color. Greedy method is the simplest which takes an ordering of nodes of a graph and colors these with the smallest color fulfilling the constraints that no neighboring nodes are assigned similar colors. However, the Greedy method performs badly in practice for large graphs.

2.6.2 First Fit (FF)

The First Fit [76] coloring algorithm is supply the set of nodes in some random order. The algorithm sequentially assigns each node the lowest authorized color. First Fit has the advantage of being very simple and very fast. In other words, First Fit is an O (*n*) time algorithm.

2.6.3 Largest-Degree-First-Ordering (LDFO)

Ordering the nodes by decreasing degree, proposed by Avanthay et al. [77], was one of the earliest ordering strategies. This ordering works as follows. Assume the nodes v_1, v_2,…,v_{i-1} have been selected and colored. Node v_i is selected to be the node with the maximum degree among the set of uncolored nodes. Largest Degree First Ordering provides a better coloring because at each step it selects a node with the highest number of neighbors which produces the highest color. Note that this heuristic can be implemented to run in $O(n^2)$.

The Largest- Degree-First algorithm [77] can be parallelized by a very similar method to the Jones-Plassmann algorithm. The only difference is that instead of using arbitrary weights to construct the independent sets, the weight is selected to be the maximum degree of the node in the induced sub graph. Random numbers are only used to resolve conflicts between adjoining nodes having the same degree.

2.6.4 Smallest-Degree-Last–Ordering (SDLO)

The smallest-degree-last ordering heuristic [82] colors the nodes in the order induced by first removing all the lowest-degree nodes from the graph, then recursively coloring the resulting graph, and finally coloring the removed nodes.

The Smallest-Degree-Last algorithm [82] tries to get better upon the Largest-Degree-First Ordering algorithm by using a more complicated system of weights. To achieve this, algorithm works in two steps, a weighting step and a coloring step. The weighting step starts by searching all nodes with degree equal to the smallest degree *d* currently in the graph. These are given the current weight and detached from the graph, thus changing the degree of their adjacent. This continues until all nodes have been given a weight. The coloring step discovers the node which has highest weight; it colors itself using the lowest available color.

2.6.5 Incidence-Degree-Ordering (IDO)

Incident degree ordering was proposed by E.K. Burke et al.[79] and is defined as follows. At each step the node with the maximum incident degree is selected. The incidence degree of a node is defined as the number of its adjacent colored nodes. Note that it is the number of adjacent colored nodes and not the number of colors used by the nodes that is counted. For example, if a node *v* has degree 4 where one of its adjacent is uncolored, two of them are colored with color 1, while the last one is colored with color 3, then *v* has incident degree 3. Ties are resolved in favor of the node with the largest degree. Incident Degree Ordering is an O (*n*)-time algorithm.

2.6.6 DSATUR (Degree of Saturation) Approach

The DSATUR algorithm (abbreviated from “degree of saturation”) was originally proposed by Br´elaz [3]. In essence it is very similar in behavior to the GREEDY algorithm in that it takes each vertex in turn according to some ordering and then assigns it to the first suitable colour class, creating new colour classes when necessary. The difference between the two algorithms lies in the way that these vertex orderings are generated. With GREEDY the ordering is decided before any colouring takes place; on the other hand, for the DSATUR algorithm the choice of which vertex to colour next is decided heuristically based on the characteristics of the current partial colouring of the graph.

Saturated degree ordering (SDO) was given by E.Falkenauer [78] and is

defined as follows. At each iteration the node with the maximum saturation degree is chosen. The saturation degree of a node is defined as the number of its neighboring differently colored nodes. For example, if a node *v* has degree equal to four where one of its adjacent is uncolored, two of them are colored with color 1, while the last one is colored with color 3, then *v* has saturation degree of two. While selecting a node of maximum saturation degree, ties are resolved in support of the node with the leading degree. The heuristic can be implemented to run in O (n^2).

2.6.7 Recursive Largest First (RLF) Approach

While the DSATUR algorithm for graph colouring is similar in behavior and complexity to the classical GREEDY approach, the next constructive method we examine, the Recursive Largest First (RLF) algorithm follows a slightly different strategy. The RLF algorithm was originally designed by Leighton [4], in part for use in constructing solutions to large timetabling problems. The method works by colouring a graph one colour at a time, as opposed to one vertex at a time. In each step the algorithm uses heuristics to identify an independent set of vertices in the graph, which are then associated with the same colour. This independent set is then removed from the graph, and the process is repeated on the resultant, smaller subgraph. This process continues until the subgraph is empty, at which point all vertices have been coloured leaving us with a feasible solution. Leighton (1979) has proven the worst-case complexity of RLF to be $O(n^3)$, giving it a higher computational cost than the $O(n^2)$ GREEDY and DSATUR algorithms; however, this algorithm is still of course polynomially bounded.

2.6.8 Parallel Maximal Independent set (PMIS)

The Maximal Independent Set (MIS) algorithm proposed by Luby [80] colors the graph by continually getting the largest probable independent set of nodes (nodes which are not neibours) in the graph. All nodes in the first such set are assigned the same color and removed from the graph. The algorithm then finds a new MIS and assigned it a second color, and continues finding and coloring maximal independent sets until all nodes have been

colored. Luby mainly involves finding an independent set, removing these nodes and their adjacent nodes from the graph, and continuing this process, until all the nodes are detached.

2.6.9 Jones-Plassmann Approach (JP)

Jones-Plassmann given a parallel coloring algorithm that improves upon the parallel MIS algorithm [81]. The Jones-Plassmann algorithm behaves very much like the MIS algorithm, apart from that it does not find a maximal independent set at each step. It just finds an independent set in parallel using Luby's method of choosing nodes whose weights are local maxima. The nodes are colored separately using the smallest available color, i.e. the smallest color that has not previously been assigned to an adjacent node. This process is repetitive until the whole graph is successfully colored.

Result of some sequential algorithm given by Hussein [76] is shown in table 2.1. Results of some parallel algorithm given by Allwright [75] are shown in table 2.2 and table 2.3.

Table 2.1: Comparison of sequential algorithm in terms of number of colors required for graphs

No. of nodes	**Density**	**FF**	**LDFO**	**IDO**	**SDLO**
200	25%	20	18	18	17
200	50%	36	34	34	32
200	75%	58	55	56	53
1000	25%	64	62	63	58
1000	50%	127	123	126	116
1000	75%	217	212	214	204

Table 2.2: Comparison of parallel algorithm in terms of average number of colors required for graphs

Problem	PMIS	JP	LDFO	SDLO
LUNDA	29.4	28.9	25.0	23.7
LUNDB	30.2	29.7	25.0	24.1
GENT113	20.3	20.5	20.0	20.0
IBM32	9.0	9.3	8.0	8.0
CURTIS54	12.3	12.2	12.0	12.0

Table 2.3: Comparison of parallel algorithm in terms of time taken in seconds for coloring graphs

Problem	PMIS	JP	LDFO	SDLO
LUNDA	2.5	3.0	4.2	5.2
LUNDB	2.5	3.1	4.1	5.1
GENT113	1.2	1.3	1.1	2.7
IBM32	0.19	0.17	0.18	0.27
CURTIS54	0.38	0.30	0.32	0.81

2.7 Related Work

2.7.1 Reduction of 3-Colorable Graph to 3-CNF-SAT

In [27], 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 [10], we generalized Alaxander's approach [11] 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.

2.7.2 Reduction of 3-CNF-SAT to 3-Colorable Graph

Moret [44] gave an reduction approach from 3-SAT to 3-colorable graph. According to Moret, reduced 3-colorable graph having ($2n + 3m + 1$) vertices and ($3n + 6m$) edges, where n is the number of variables and m is number of clauses contained by 3-SAT formula. The brief description of Moret's Approach for reduction of 3-Colorable Graph to 3-CNF-SAT as follows: Given a 3CNF formula, we produce a graph as follows. The graph consists of a *triangle* for each variable and one *triangle* for each clause in the formula. All triangles for variables have a common vertex B (we can say base vertex) which preempts one color, so that the other two vertices of each such triangle corresponding to the variable and it's negation (or complement) must be assigned two different colors i.e. truth assignment either TRUE or FALSE. Then, we connect each vertex of a clause triangle to the corresponding literal vertex. Each such edge forces its two endpoints to use different colors.

2.7.3 Approaches Based on Independent Set Extraction

As observed in many studies, it is difficult, if not impossible, to find a proper k-coloring of a large graph G (e.g., with 1 000 vertices or more) with k close to $\chi(G)$ by applying directly a given coloring algorithm on G. A basic approach to deal with large graphs is to apply the general principle of "reduce-and-solve". This approach is composed of a preprocessing phase followed by a coloring phase. The preprocessing phase typically identifies and removes some (large) independent sets from the original graph to obtain a reduced subgraph (called "residual" graph). The subsequent coloring phase determines a proper coloring for the residual graph. Given the residual graph is of reduced size, it is expected to be easier to color than the initial graph. Now it suffices to consider each extracted independent set as a new color class (i.e., by assigning a new color to all the vertices of each of these sets). The coloring of the residual graph and all the extracted independent sets give a proper coloring of the initial graph. These approaches were explored with success in early studies like [69, 70, 71, 72]. Algorithms based on this approach can use different methods to find a large independent set in the graph. In [69], this was achieved with a simple greedy heuristic while in [69, 71], large independent sets were

identified by a dedicated tabu search algorithm. In [39], the authors introduced the XRLF heuristic which operates in two steps. First, a number of independent sets are collected using Leighton's Recursive Largest First (RLF) heuristic [29]. Then, an independent set is iteratively selected and extracted from the graph such that its removal minimizes the density of the reduced graph.

This process continues until the residual graph reaches a given threshold. For the subsequent residual graph coloring, various methods have been used including exhaustive search [72], tabu search [71], simulated annealing [69, 72] and hybrid genetic tabu search [70].

Chapter 3

A Tree Based Novel Approach for Graph Coloring Problem using Maximal Independent Set

3.1 Introduction

Suppose $G = (V, E)$ is a directionless graph where V is the set of vertices and E is the array of arcs. The colour problem of graph can be defined as a mapping of colours $C = \{1,2,\ldots,k\}$ with the vertices of G so that neighboring vertices of graph must not accept the same colors. It means, in an accurate graph colouring, if 2 nodes a and b of the graph share an arc (a, b), both node should paint by separate colours. The least number of colours, necessary for the colouring the nodes of G accurately are termed as the chromatic number of G, represented by $\chi(G)$. Graph colouring issue targets the smallest k for a specified graph G. The instance below illustrates the accurate minimum coloring of a 5 vertex Star Graph by minimum 3 colors.

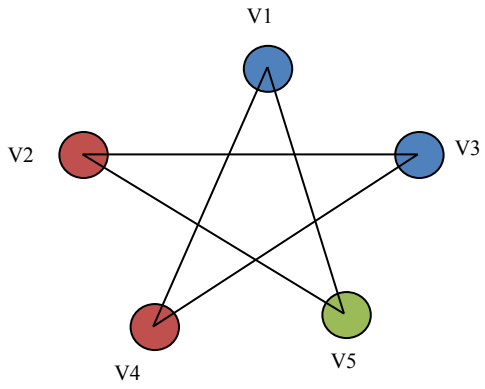


Figure 3.1: Properly Colored 5 Vertex Star Graph

In the optimization form, the graph coloring problem, goals to calculate the lowest possible colours k, so that a realistic colouring of G could be possible.

A k-colorable graph divides an array of vertices V into k dissimilar color classes, where each member of the class has the same color. In order to have the same color, the members of each class must be pair wise non-adjacent, which by definition makes them an independent set. Therefore, finding maximal independent sets from the initial graph is an effective method for solving graph coloring problem. In this manuscript, we have developed a tree based innovative approach for graph coloring problem using maximal independent set.

An independent set (also known as a stable set) is a sub group of vertices $S \subseteq V$ so that none of the vertices in S are neighbors. The subsets of the graph containing those vertices that are not attached to any other element of the same set are known as an independent set. The highest cardinality of a stable set of G is represented by $\alpha(G)$. A stable set is called maximal, if it is not a subset of any bigger independent set and it is assumed maximum if there is no bigger independent set within the graph.

A clique is a subset of V where all the vertices are pair wise adjacent. We can infer the complement graph $G' = (V, E')$ from a graph $G = (V, E)$, where $E' = \{(i, j) \mid i, j \in V, i \neq j$ and $(i, j) \notin E\}$. The Maximum independent set problem (MIS) is to determine an independent set in graph G of highest cardinality $\alpha(G)$. It is well understood that if I is an independent set of G, then I is a clique of G'.

In this chapter, we developed a new graph coloring method using maximal independent set by tree exploration. Ultimately, rather than removing independent sets in sequence, we have tried to recognize maximal sets of freelance vertices at each step. In this method, due to removing independent set in each iteration, then many vertices are eliminated from the original graph. Hence, in this way, coloring of the remaining graph is easier. We evaluated the performance of our graph coloring procedure on various large DIMACS standard graphs (with 100, 500 and 1000 vertices).

Further, this chapter is structured as follows. In segment two, we reviewed the heuristic based graph colouring strategies. In segment three, we provided a complete demonstration of the projected procedure. Segment four will describe our approach by an example. Segment five described our approach

through pseudo code. In segment six, we showed complexity analysis along with extensive experimental results and comparisons. The last segment summarized this chapter's research.

3.2 Review of previous works

The colouring problem of a graph is a problem category of NP-hard and graph k-colourability is a problem segment of NP-complete for every integer $k \geq 3$ (but 2-coloring is polynomial) [10, 22]. As observed in many pieces of literature, it is hard to acquire an accurate k-colourability of a massive graph G (for example: graph having 1000 nodes or more) with number of colors k nearby the chromatic number $\chi(G)$ to use a given colouring procedure directly onto graph G. There are many heuristic algorithms for graph colouring comprising the succeeding methods: greedy construction [3], Recursive largest first (RLF) heuristic [4], tabu search [5, 6, 7, 8], simulated annealing [15, 20], and evolutionary hybrid or population grounded search [17, 21, 23, 24]. A complete study of the most important heuristic methods can be found in Galinier and Hertz [26].

Another method for handling massive graphs for colouring is to use the general rule of "reduce-and-solve." This technique consists a "*preprocessing"* succeeded by a "*coloring phase"*. The first part usually recognizes and eliminates some independent sets from the initial graph to get a reduced sub graph (termed "residual" graph). The following part decides correct colouring for the "residual graph". Since the reduced graph is compact, it is easier to paint it than the original graph. Currently, it seems to think about every removed freelance set as a new color category (i.e., by assignment a new color to any or all the nodes of those sets). The coloring of the residual graph and the removed freelance sets offer acceptable coloring of the original graph. These techniques were explained successfully in earlier articles [7, 9, 15, 20].

Algorithms based on "reduce-and-solve" have the different way to find a large independent set in the graph. In [15], this was finished an easy greedy heuristic whereas in [7, 9], big independent sets has been recognized by a committed tabu search technique. In [20], the researchers presented the XRLF heuristic that works in 2 stages. Primarily, a variety of independent sets is generated through Leighton's Recursive Largest First (RLF) heuristic [4].

Secondly, a freelance set is sequentially picked and removed from the graph so that its elimination reduces the compactness of the reduced graph. This method progresses till the reduced graph touches a pre assumed threshold. For the successively reduced graph colouring numerous strategies are used including exhaustive search [20], tabu search [6][7], simulated annealing [15, 20] and hybrid genetic tabu search [9]. Most recently, this simple independent set removal method has been reviewed and improved outcomes have been found for many giant graphs [1, 2,].

3.3 Our Proposed Algorithm: Tree Based Maximal Independent Set

In this segment, we proposed an algorithm that calculates the minimum colouring for the graph using maximal independent set. It comprises of 3 steps. The first step is the creation of the complementary edge table. The main and second step is an iterative step to find maximal independent sets using tree exploration. Third and the final step is the coloring of the maximal independent sets. Now we elaborate these steps in detail in the following subsections:

3.3.1 Complementary Edge Table

The complement of graph $G = (V, E)$ is a graph $G' = (V, E')$, where $E' = \{(i, j) \mid i, j \in V, i \neq j \text{ and } (i, j) \notin E\}$. To find maximal independent sets, we need to put together those vertices that are not connected to each other and if we have a table that defines which vertices are not connected; it would reduce the time complexity significantly. So, we scan the edge table *ET* and make a new edge table that can be said complementary edge table *CET* that comprises a list of vertices that are not connected to each other. At the implementation level, we take the following step:

(i) Take input file of a graph (from DIMACS instances) as adjacency list of vertices of graph *G*; here we call it an edge table (ET).

(ii) Create complementary adjacency list of vertices of above input file and named it as complementary edge table (CET). CET having the list of vertices that are not connected to each other; it means CET comprises only those edges which were not in the original edge table (ET).

(iii) We include only those edges which originate from a vertex of smaller numbering than its destination as the graph is considered undirected.

3.3.2 Finding Maximal Independent Sets: Tree Exploration

This step itself is a multi-step process, which explores vertices of a graph to make a tree. Each sub tree will give a maximal independent set; this process will run until the entire vertex has been explored. The technical details of finding maximal independent sets are as follows:

(i) Select the first vertex which is not included yet in any maximal independent set *MIS* i.e. we start from vertex V_i (for i = 1,2,…,n). Now vertex V_i will be the root of general sub tree T_i (for i = 1,2,…,k).

(ii) Explore root vertex V_i of a sub tree T_i as follows: select all those vertices from complementary edge table which are listed against vertex V_i and make those vertices as children of root vertex V_i.

(iii)Repeat the following rule for further exploring every child node V_{ic} of tree T_i as follows:

a) Select only those vertices which are listed against being explored node V_{ic} in CET and it should be the sibling of explored node V_{ic} in sub tree T_i. Since we have to find independent set along a path of created sub tree; that's why we consider only sibling node of V_{ic}. This step avoids connectivity clashes among vertices of the path from the root node to leaf node. Also, check, it should not be included in any maximal independent set yet.

(iv) After completion of tree formation, select the path with the maximum length. If more than 1 path has maximum length then selects the first longest path while traversing left to right among them.

(v) Selected (maximum length) path of sub tree T_i (i=1,2,…,k) will be maximal independent set MIS_i (i =1,2,…,k) as every vertices in this path are not connected to each other in original graph.

(vi) Repeat step 1 to 5 until all the vertices of the graph will be included in any of maximal independent set MIS_i . It means, in each iteration, we get one MIS.

3.3.3 Coloring the Maximal Independent Sets

This is the final step of our minimum coloring algorithm. In this step we assign a different colour to each maximal independent set, i.e. all the vertices that belong to the same maximal independent set are allotted the same color and all the vertices that belong to different independent sets, now have assigned different colors. This is the core of our algorithm.

3.4 Illustration by an Example

Let us explore our approach by an example. We take following Peterson graph given in figure 1 and find the minimum coloring for this graph using maximal independent set.

3.4.1 Creating Complementary Edge Table

The edge table ET and the complementary edge table CET for the graph of figure 3.2 are listed below in table 3.1.

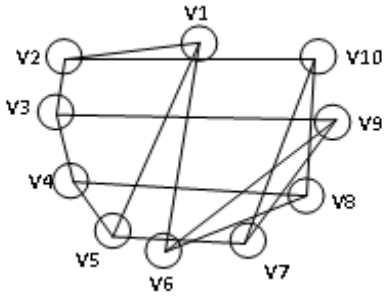


Figure 3.2: Petersen Graph [35]

Table 3.1: Edge and Complementary Edge Table for Figure 3.2

Edge Table (ET)		Complementary Edge Table (CET)	
1	2	1	3
1	5	1	4
1	6	1	7
2	3	1	8
2	10	1	9
3	4	1	10
3	9	2	4
4	5	2	5
4	8	2	6
5	7	2	7
6	8	2	8
6	9	2	9
7	9	3	5
7	10	3	6
8	10	3	7
		3	8
		3	10
		4	6
		4	7
		4	9
		4	10
		5	6
		5	8
		5	9
		5	10
		6	7
		6	10
		7	8
		8	9
		9	10

3.4.2 Finding Maximal Independent Set by Tree Exploration

Now, we start tree exploration according to above rule 3.3.2. Take first vertex V1 as the root of tree and then go to CET against V1, we find V3, V4, V7, V8, V9, V10; make these vertices children of V1. Now take V3 to be explored in next step. Again we go to CET against V3, we find V5,V6, V7, V8, V10; but we have to select only those vertices which are the siblings of V3 i.e. V7, V8, V10, so that connectivity clashes among vertices of a path of sub tree could be

avoided so that we could go ahead toward finding independent set. Similarly we explored all the vertices.

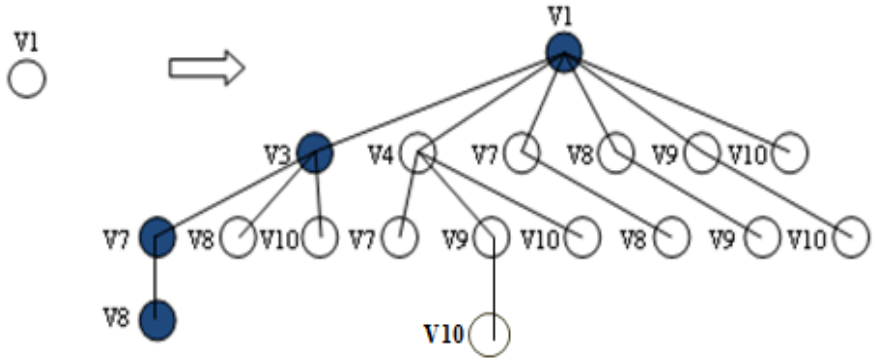


Figure 3.3(a): First Maximal Independent Set

In above sub tree which is drawn in figure 3.3 (a), there are two longest path {V1, V3, V7, V8} and {V1, V4, V7, V10}. Both are equal and as per our rule, we choose first longest path while traversing left to right. Since each path in this sub tree form an independent set and hence we make the longest path of sub tree as maximal independent set (MIS). We store all the vertices of selected longest path in MIS_i (for i=1,2,..,k) as well as in V_{MIS} which keeps record of vertices those are included in any MIS_i

MIS_1= {V1, V3, V7, V8}

V_{MIS}= {MIS_1} = {V1, V3, V7, V8}

Now we start exploration from next vertex which is not included in obtained MIS i.e. from V2; this is shown in figure 3.3 (b).

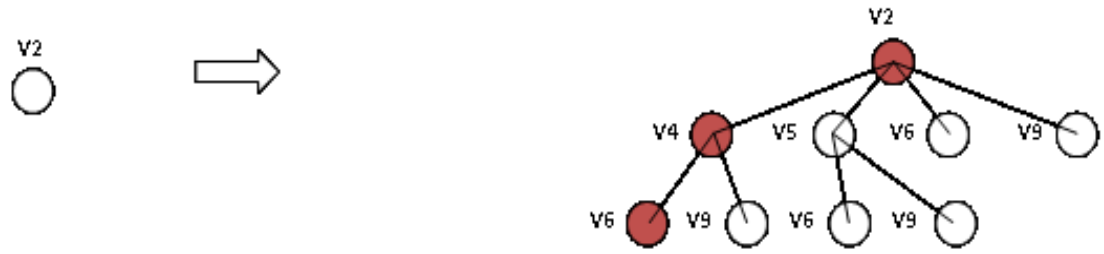


Figure 3.3 (b): Second Maximal Independent Set

In second case, longest paths of sub tree are {V2,V4,V6} {V2,V4,V9}, {V2,V5,V6} and {V2,V5,V9}. We select first longest path as maximal independent set according to our approach.

$MIS_2 = \{V2, V4, V6\}$

$V_{MIS} = \{MIS_1 \vee MIS_2\} = \{V1, V2, V3, V4, V6, V7, V8\}$

Again we start exploration from next node which is not included in any obtained MIS i.e. from V5. In following sub tree, there is only one path which is longest {V5, V9, V10}.this step is shown in figure 3.3 (c). Hence we select the longest path of sub tree as maximal independent set (MIS).

Figure 3.3 (c): Third Maximal Independent Set

$MIS_3 = \{V5, V9, V10\}$

$V_{MIS} = \{MIS_1 \vee MIS_2 \vee MIS_3\} = \{V1, V2, V3, V4, V5, V6, V7, V8, V9, V10\}$

Since $V = V_{MIS}$; now, stop the process of tree exploration as all the vertices of graph are included in any of maximal independent set (MIS).

3.4.3 Coloring the Maximal Independent Set

Since, by tree exploration of this graph we get 3 MIS and as we know that each independent set is a collection of all the disconnected vertices of graph; hence we can color each MIS by a unique color. Here, in graph given in figure 2, can be colored by 3 colors as it is obtained 3 MIS in this graph by our approach. First MIS_1 has to be assigned color blue; second MIS_2 has to be given color red and third MIS_3 has to be assigned color green. When we apply our algorithm on graph of figure 2, we get the following result:

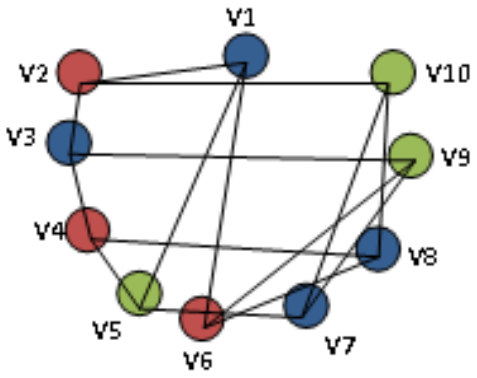


Figure 3.4: Properly colored Petersen graph given in figure 3.2 using our proposed approach

3.5 Algorithm for Graph Coloring Problem using Maximal Independent Set

3.5.1 Notations used in our algorithm:

- V_{MIS}: Set of vertices included in any of independent sets
- V: Set of all vertices
- X: Current node
- S_X: Set of siblings of current node X
- Xc : children of X
- N_X: Set of neighbors (adjacent nodes) of current node X
- i: temporary number
- T_k: k^{th} Sub Tree
- MIS_k: k^{th} MIS
- n: number of paths with maximum length
- P_L: Longest path while traversing left to right

3.5.2 Algorithm for finding Maximal Independent Sets (MIS)

1. Input: A directionless graph $G = (V, E)$, an integer k
2. Output: A proper k-coloring of graph G or display statement of failure

3. Begin
4. While ($V_{MIS} \neq V$)
5. $i = 1$ and $X = V_i$
6. If (($N_X \cap S_X$) = =$\emptyset$ and $X \notin V_{MIS}$)
7. Initialize a Sub Tree T_k and add vertex X as root to T_k
8. Repeat step 9 to10 for all neighbors of X
9. For all N_X : make child X_C of X as $X_C = N_X$
10. $i = i + 1$
11. Else
12. Repeat step 11 to 13 for all (V-1)
13. $X = X_C$ ($\forall X_C \in V$)
14. If ($\forall X \in V$ ($N_X \cap S_X$) $\neq \emptyset$ and $X \notin V_{MIS}$)
15. $\forall X \in V$ (X_C = ($N_X \cap S_X$))
16. End if
17. Find number of longest path n
18. If (n>1)
19. Select any one longest path P_L; make it MIS_k and set $V_{MIS} = MIS_k$
20. Set V= (V - V_{MIS})
21. End while
22. End

3.6 Results and Discussion

The property of obtained tree by our approach is equivalent to binomial trees. A binomial tree is an ordered set of element defined recursively. Let depth of the tree is d; the total number of node at order d in binomial tree is 2^d. We observed that height (or depth) of tree d = ($log\ n$). The complexity of our algorithm based on the creation of complementary edge table, tree exploration and selects the longest path of tree as maximal independent set. Creation of complementary edge table takes O (n^2) time where n is the number of vertices in graph G.

We have tested this algorithm on various DIMACS instances [18][19]. After testing, we found some interesting results. The algorithm gave colors precisely equal to the chromatic number of the graphs, no matter the sequencing of the

vertices. All results of our procedures were obtained on a Pentium IV 2.4 GHz with 2 GB RAM under Windows 7. We used Java (JDK 1.7) to implement our algorithm. Some of the instances that we have been tested and quantity of colors calculated by the algorithm are as shown in the table II below.

Table 3.2: Comparison of our result with E2COL approach [2], DSATUR approach [16] and Malguti's approach [17]

Graph Instance	$\|V\|$	$\|E\|$	k^*	k_{ours}	k_{E2COL} [2]	k_{DSATUR} [16]	$k_{malaguti}$ [17]
myciel6.col	95	755	7	7	-	7	-
myciel7.col	191	2360	8	8	-	8	-
queen6_6.col	36	580	7	8	-	-	-
queen7_7.col	49	952	8	8	-	-	-
queen8_8.col	64	728	9	9	-	9	-
queen9_9.col	81	2112	10	10	-	10	-
mulsol.i.1.col	197	3925	49	49	-	49	-
mulsol.i.2.col	188	3885	31	31	-	31	-
DSJC125.1.col	125	736	5	5	5	5	5
DSJC125.5.col	125	3891	17	17	17	19	17
DSJC125.9.col	125	6961	44	44	44	45	44
DSJC250.1.col	250	3218	8	8	8	9	8
DSJC250.5.col	250	15668	28	28	28	35	28
DSJC250.9.col	250	27897	72	72	72	87	72
DSJC500.1.col	500	12458	12	12	12	15	12
DSJC500.5.col	500	62624	48	48	48	63	49
DSJC500.9.col	500	1124367	126	126	126	160	127

In Table 3.2, $|V|$ denotes set of vertices, $|E|$ is the array of arcs, k^* is the chromatic number or the recognized limit of the chromatic number, k_{ours} is the number of color calculated for every graph using our algorithm, k_{E2COL} is the chromatic number obtained by extraction and expansion approach of coloring

[2], $k_{DSATUR}$is the chromatic number obtained by DSATUR approach [16], and $k_{malaguti}$ is the coloring number obtained by Malguti [17].

3.7 Summary

A novel heuristic approach has been proposed for the solution of graph coloring problem using the maximal independent set which is based on tree exploration. The first step converts a big graph into a series of gradually smaller graphs by eliminating maximal independent sets from the graph, while in the later step, to color the removed maximal independent sets. We have been observed that even with a basic tabu search coloring procedure, the planned method gets very reasonable outcomes on a set of DIMACS test standard graphs. By our approach, we are getting an optimized solution to the problem of coloring the graph. Computational outcomes are presented to prove the theoretical analysis.

Chapter 4

Polynomial 3-SAT Encoding Technique for *k*-colorable Graph and Analysis of Graph Coloring Problem based on Satisfiability

4.1 Introduction

A Graph *k*-colorability is an assignment of colors {1,2,…,*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 [28] 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 [41][42], 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. 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).

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

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 coloring problem can also be solved as a decision based using method of Satisfiability (SAT). Problem instances from domains such as Graph Coloring can be encoded into SAT and then solved by the help of SAT algorithms.

Previously, in [27], Alexander Tsiatas gave a reduction approach from 3-Colorable graph to 3-SAT expression. He encoded the vertices and an edge 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. We generalized Alexander'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 4.2, we discussed our 3-SAT encoding approach for *k*-colorable graph. Section 4.3 illustrated SAT encoding approach of a 3-colorable graph by an example. In section 4.4, we discussed the SAT based approach for

solving graph coloring problem and then in section 4.5 experimental results are discussed.

4.2 Polynomial 3-SAT Encoding Formulation Approach 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 should have the same color. Then to encode this k-colorable graph into 3-CNF-SAT propositional formula, we use two approaches 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 as below:

4.2.1 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:

$$v_{ic} = (v_{i1} \vee v_{i2} \vee \ldots \vee v_{ik}) \tag{4.1}$$

Equation (4.1) generates one clause of length-k in conjunctive normal form (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:

$$(x_1 \vee x_2 \vee \neg y_1) \wedge (x_3 \vee y_1 \vee \neg y_2) \wedge (x_4 \vee y_2 \vee \neg y_3) \wedge \ldots \wedge (x_{k-2} \vee y_{k-4} \vee \neg y_{k-3}) \wedge (x_{k-1} \vee x_k \vee y_{k-3} \tag{4.2}$$

Expression (4.2) 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 (4.2) to a clause of length 6 yields (1 clause of length 6) = (4 clauses of length 3) and this required an additional 3 variables.

Let F_v is the encoded formula (or expression) obtained by vertex constraint approach which is the conjunction of 3-CNF encoded expression of all the n vertices of graph G as:

$$F_v = (v_{1c} \wedge v_{2c} \wedge \ldots \wedge v_{ic}) \quad (4.3)$$

Applying (4.2) to (4.1) and finally, we get total (k-2)*|V| clauses in 3-CNF-SAT expression from graph G as per vertex constraint approach.

$|F_v| = (k\text{-}2)*|V|$ clauses in 3-CNF-SAT (4.4)

where $|F_v|$ shows total number of clauses in 3-CNF-SAT expression as per vertex constraint approach.

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. It means, 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:

$$e_j = (\neg u_1 \vee \neg v_1) \wedge (\neg u_2 \vee \neg v_2) \wedge \ldots \wedge (\neg u_k \vee \neg v_k) \quad (4.5)$$

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:

$$F_e = (e_1 \wedge e_2 \wedge \ldots \wedge e_m) \quad (4.6)$$

Since, expression (4.5) is in 3-CNF-SAT, so there is no need to apply (4.2) 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 (4.7)

4.2.3 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 (4.3) and formula obtained by edge constraint approach (4.6) as below:

$$F = (F_v \wedge F_e)$$

$$F = ((v_{1c} \wedge v_{2c} \wedge \ldots \wedge v_{nc}) \wedge (e_1 \wedge e_2 \wedge \ldots \wedge e_m)) \quad (4.8)$$

We combine (4.4) and (4.7) as number of clauses obtained by vertex constraint approach and by edge constraint approach. Finally, we get total number of clauses in 3-CNF-SAT formula |*F*| by polynomial 3-CNF-SAT encoding technique of *k*-colorable graph as:

$$|\boldsymbol{F}| = (\boldsymbol{k}\text{-}2)^*|\boldsymbol{V}| + \boldsymbol{k}^*|\boldsymbol{E}| \quad (4.9)$$

4.2.4 Algorithm: Encoding of *k*-Colorable Graph to 3-CNF-SAT Expression

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++)
{
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.

4.2.5 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

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

Proof by Induction:

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

Proof 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

$$v \wedge e = (v_1 \vee v_2 \vee v_3) \wedge (\neg u_1 \vee \neg v_1) \wedge (\neg u_2 \vee \neg v_2) \wedge (\neg u_3 \vee \neg v_3)$$

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

(b) For $k = m$

Proof by induction: $(k\text{-}2)*|V| + k*|E|$

$= (m\text{-}2)*|V| + m*|E|$

Proof by expression: For m colors $(v_m \wedge e_m)$ can be expressed as:

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

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 $= (m\text{-}2)*|V|$ 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 $(m\text{-}2)*|V| + m*|E|$. So it is also true for k=m.

(c) For $k = m + 1$

Proof by Induction: $(k\text{-}2)*|V| + k*|E|$

$= (m +1\text{-}2)*|V| + (m+1)*|E$

$= (m\text{-}1)*|V| + (m+1)*|E|$

Proof 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) = $((m\text{-}2) + 1) = (m\text{-}1)$

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 $v_{m+1} \wedge e_{m+1} = (m\text{-}1)*|V| + (m+1)*|E|$

So it is also true for $k = m+1$.

4.2.6 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 $v_i = v_1, v_2, \ldots, v_n$ 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*-colored 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 are true.

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

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

4.3 Illustration of Encoding of 3-Colorable Graph to 3-CNF-SAT

Here, we are taking an example of Petersen [9] graph *G* as figure 4.1, having 10 vertices and 15 edges to encode it by 3-color say 1, 2, 3 into propositional 3-satisfiability.

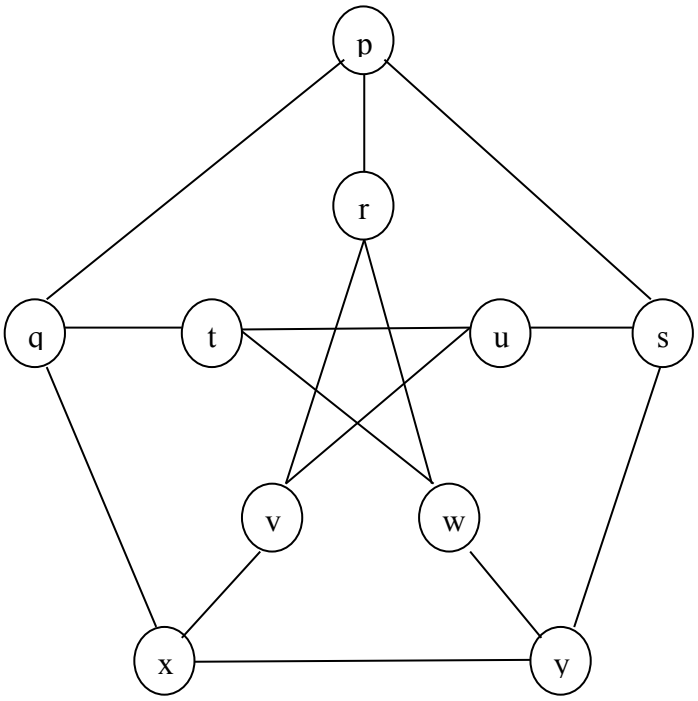


Figure 4.1: Petersen Graph [35]

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 above graph by 3-colors. As per the vertex constraint approach (4.1), 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 (4.5), and stored 3-CNF-SAT expression in F_e as below:

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

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

$F = F_v \wedge F_e$

Hence, total number of 3-CNF-SAT clauses corresponding to above 3-colorable graph Peterson graph = ((number of clauses as per vertex constraint approach) + (number of clauses as per edge constraint approach)) = (10 + 15) = 25, which is a polynomial reduction from 3-colorable graph to 3-CNF-SAT.

4.4 Solution Approach for Graph *k*-Colorability using SAT Solver

SAT-based approach is a decision based 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, and software & 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 [34][40] 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 ([34][40], 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 [36, 37, 38, 39] to solve 3-CNF-SAT expression. MiniSat [38][39] 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.

4.5 Results and Discussion

We have implemented a 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 which is a polynomial reduction, whereas previously, Alaxander [27] generated $((k^k * |V|) + (2^{2k+2} * |E|))$ clauses in 3-CNF, which is a exponential reduction. Here, we analyzed the encoding formulation for 3-color and 4-color on various benchmark problems (graph

coloring instances) of the DIMACS challenge [34][40]. We implemented it in java (JDK 1.7). Results are compiled at table 4.1.

Further, we solved 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 4.1: 3-CNF-SAT clause generation for color k=3 and 4

Graph Coloring Instances	No. of Vertices	No. of Edges	Alexander's Approach[27] (Total no of 3-CNF clause when k=3)	Our Approach Total 3-CNF-SAT clauses (when k=3)	Alexander's Approach[27] Total no of 3-CNF clause when k=4)	Our Approach: Total 3-CNF-SAT clauses (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**

Computational results of graph coloring problem using Minisat 2.2 are stored in Table 4.2 and Table 4.3. Some of the DIMACS graph instance which is satisfiable on any color *k* is stored in Table 4.2. Some of the large graphs

having 500 or more vertices are not satisfied and such list is contained by table 4.3.

We compared our approach of 3-CNF-SAT encoding with Alaxander's approach [27] by drawing a pie graph. Figure 4.2 shows the comparison between both the approaches with respect to 3-CNF clause generation for k = 3. Similarly, figure 4.3 shows analysis of 3-CNF-SAT clause generation for k = 4 corresponding to our and Alexander's encoding method.

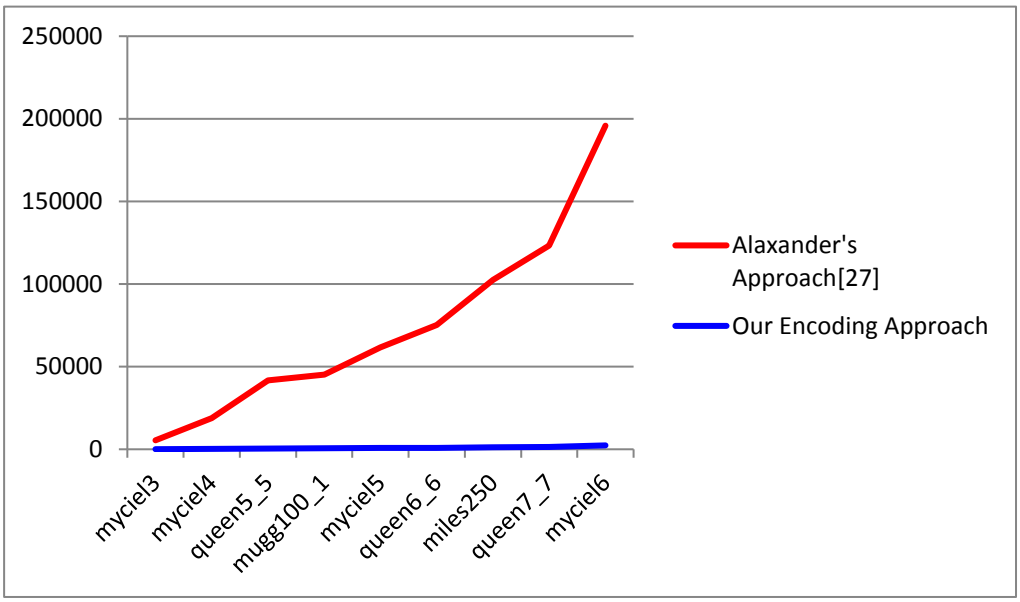


Figure 4.2: Analysis of 3-CNF-SAT clause generation for k = 3

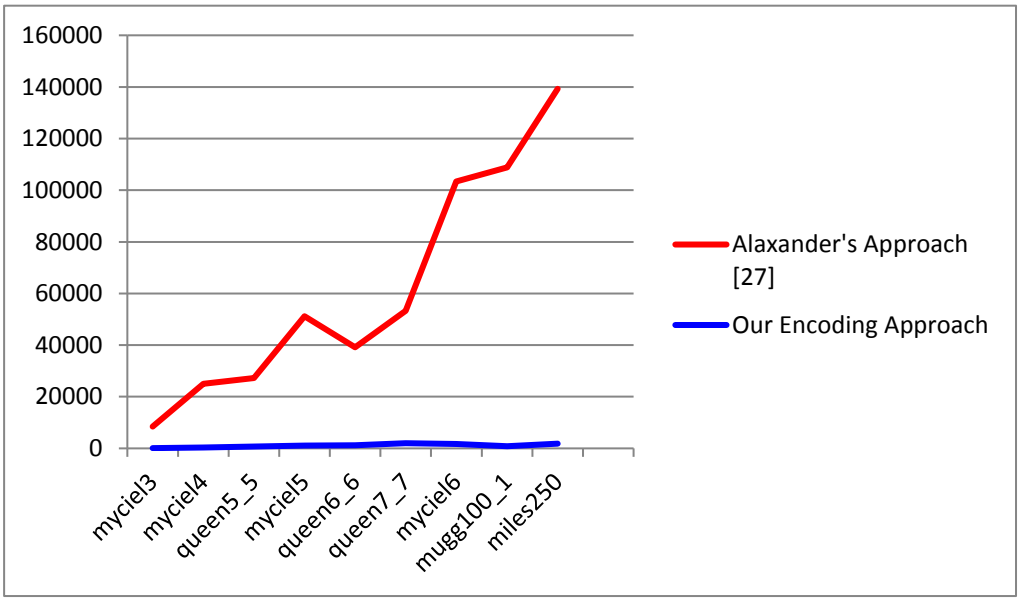


Figure 4.3: Analysis of 3-CNF-SAT clause generation for k = 4

Table 4.2: 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
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

Table 4.3: 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

4.6 Summary

In this chapter, we have presented a generalized polynomial 3-CNF-SAT encoding technique of graph *k*-colorability. Our encoding formulation of *k*-colorable graph to 3-CNF-SAT is polynomial and better than Alaxander's approach [27] which was 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 solver as an intermediate domain for solving problems in the form of decision based. SAT technique only extracted that *k* 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 number of vertices is equal to 500 or more are not satisfiable. It means the solution of problem will be depend on the strength of SAT solver along with encoding technique of *k*-colorable graph to 3-CNF-SAT expression.

Chapter 5

A New Reduction from 3-SAT to Graph *k*-Colorability for Channel Assignment Problem

5.1 Introduction

The satisfiability problem (SAT) is one of the most prominent problems in theoretical computer science, which has become increasingly popular and important insights into our understanding of the fundamentals of computation. SAT is the first known NP-Complete problem. 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. We can reduce any NP-Complete problem to/from 3SAT. Reduction from satisfiability problem to graph *k*-colorability problem or vice versa is an important concept to solve one of the hard scheduling problem, as frequency assignment in cellular network. The frequency assignment problem is very similar to the graph *k*-colorability problem [50].

The frequency band has become an important resource for communication service. There has been large increase in demand for using the frequency bands caused by the fast growth in mobile communication, satellite communication and mass communication service areas. To maximize utilization of frequency band, the limited band of available frequency is divided into a number of channels. A channel can be reused many times for different transmitters if the transmitters are far enough from one another so that the co-channel interference between them is low enough. If there are two close transmitters using the same channel simultaneously, they will suffer from severe co-channel interference and the quality of communication service will be unsatisfactory.

Since the available frequency band is limited, we are interested in using as small band of frequency as possible while satisfying all the frequency demand and the co-channel constraints. This can be efficiently done by the proper frequency assignment. It is shown that the frequency assignment problem is equivalent to an extended version of graph coloring problem [50]. A variation of the graph coloring problem is the graph *k*-colorability problem [49].

In this chapter, we introduce a new framework to represent SAT problems, for this, we proceed for the reduction of the instance of 3-CNF-SAT formula to *k*-colorable graph in polynomial time. Our reduction formula generate a *k*-colorable graph with $|V| = (2n + 3m + (k-2))$ vertices and $|E| = (3n + 6m)$ edges for $k = 3$ and $|E| = (|E|$ of $(k-1)$-colorable graph + $(|V|-1))$ edges for $k > 3$ corresponding to any instance of 3-CNF-SAT. Previously, in standard reduction approach from 3-SAT to 3-Colorable graph [27], the generated graph having $(2n+5m+3)$ vertices and $(3n+10m+3)$ edges. Further, Moret [44] gave an improved reduction approach from 3-SAT to 3-colorable graph. According to Moret, reduced 3-colorable graph will have $(2n + 3m + 1)$ vertices and $(3n + 6m)$ edges. Here, we generalized the reduction approach to reduce any instance of 3-CNF-SAT formula to a *k*-colorable graph in polynomial time with mathematical proof.

In next section of this chapter, we have explored basic detail of 3-SAT, *k*-colorable graph and frequency assignment problem. Section 5.3 describes our polynomial reduction approach from 3-SAT to *k*-colorable graph. Section 5.4 explored the formulation of graph *k*-colorability to frequency assignment problem.

5.2 Polynomial reduction from 3-CNF-SAT to *k*-colorable graph

The method of showing that a problem is NP-Complete by polynomial reduction is one of the most elegant and productive in computational

complexity [51]. To prove that problem *A* is NP-hard, reduce a known NP-hard problem to *A*. Cook [52] defines the following:

***Definition 1*:** Suppose that L_i is a language over $\sum i$, i =1,2.

Then $L_1 \leq_p L_2$ (L_1 is polynomially reducible to L_2) iff there is a polynomial-time computable function f: $\sum_1 \rightarrow \sum_2$ such that $x \in L_1 \leftrightarrow f(x) \in L_2$, for all $x \in \sum_1$.

***Definition 2*:** A language L is NP-Complete iff L is in NP, and $L' \leq_p L$ for every language L' in NP

***Proposition 1*:** Given any two languages, L_1 and L_2:

1) If $L_1 \leq_p L_2$ and $L_2 \in P$ then $L_1 \in P$.

2) If L_1 is NP-Complete, $L_2 \in NP$ *and* $L_1 \leq_p L_2$ then L_2 is NP-Complete.

5.2.1 Reduction of 3-CNF-SAT to Graph 3-Colorability (3-SAT $\leq_p$ 3-Color)

***Theorem 1*:** Graph 3-Colorability is NP-Complete [44].

Proof: First of all we have to proof that it is in NP then try to proof it is in NP-Hard. If it is both then it will be NP-Complete.

1. First we show that 3-Color $\in$ NP. Given a graph *G*, and a coloring assignment of the vertices, simply walk the graph and make sure that all adjacent vertices have a different color, and make certain that only 3 colors are used. This is clearly by $O(|V| + |E|)$, where $|V|$ is the number of vertices and $|E|$ is the number of edges of graph *G*.
2. Now show that 3-Color $\in$ NP-Hard. To do this, we reduce 3-CNF-SAT expression to 3-colorable graph, or show that 3-SAT $\leq_p$ 3-Color.

Graph Construction for 3-color: Start with an instance of 3-SAT formula *F* with *n* variables $x_1, x_2, \ldots, x_n$ and *m* clauses $c_1, c_2, \ldots, c_m$. Create a graph *G* such that *G* is 3-colorable iff *F* is satisfiable. Reduced graph *G* has vertices corresponds to variables and coloring to vertices is similar to truth assignment to variables from instance of 3-SAT formula. Given a 3-CNF formula, we produce a graph as follows. The graph consists of a triangle for each variable

and one triangle for each clause in the formula. All triangles for variables have a common vertex B (we can say base vertex) which preempts one color, so that the other two vertices of each such triangle corresponding to the variable and its negation (or complement) must be assigned two different colors i.e. truth assignment either TRUE or FALSE. Then, we connect each vertex of a clause triangle to the corresponding literal vertex. Each such edge forces its two endpoints to use different colors.

Correctness: A clause triangle can be proper colored if and only if all three of its corresponding literal vertices have not been given the same color, that is, a clause triangle will be proper 3-colored if and only if all three literals in the clause have not been assigned the same truth value. Thus, the transformed instance admits a solution if and only if the original 3-CNF-SAT instance does. Reduced graph from 3-SAT holds following two conditions for its correct solution as below:

- If the 3-SAT formula has a satisfying assignment then the graph has 3-coloring.
- If the graph has a 3-coloring, then the SAT formula has a satisfying assignment.

Bound: The transformation takes an instance of 3-SAT with n variables and m clauses and reduced a 3-colorable graph that will have the number of vertices and edges as follows:

$|V| = (2n + 3m + 1)$ vertices and

$|E| = (3n + 6m)$ edges

It is easily done in polynomial time.

Example 1: Transform following 3-CNF-SAT formula into 3-colorable graph:

$$(x \vee y \vee z') \wedge (x \vee y' \vee z') \qquad (5.1)$$

Here, number of variable $n = 3$ and number of clauses $m = 2$; corresponding to this instance of 3-CNF, following figure 5.1 shows reduced 3-colorable graph as per 3-colorable graph construction process. Total number of vertices $|V|$ and number of edges $|E|$ in reduced graph is calculated as:

$|V| = (2n + 3m + 1) = ((2*3) + (3*2) + 1) = 13$

$|E| = (3n + 6m) = ((3*3) + (6*2)) = 21$

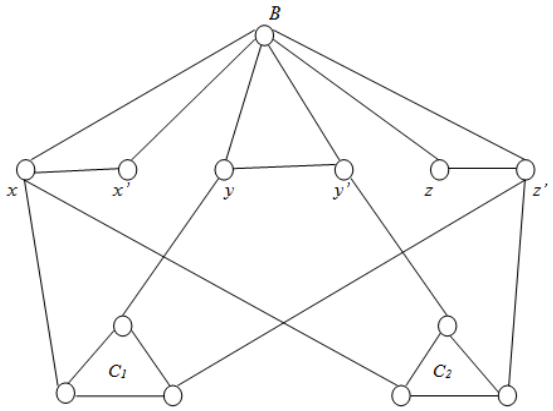


Figure 5.1: 3-Colorable Graph

5.2.2 Reduction of 3-SAT to Graph 4-Colorability (3-SAT $\leq_p$ 4- Color)

***Theorem 2*:** Graph 4-Colorability is NP-Complete

Proof: First of all we have to proof that it is in NP then proof that it is in NP-Hard. If it is both then it will be NP-Complete.

1. First we show that 4-Color $\in$ NP. Given a graph *G*, and a coloring assignment of the vertices, simply walk the graph and make certain that all adjacent vertices have a different color, and make certain that only 4 colors are used. This is clearly by $O(|V| + |E|)$.
2. Now show that 4-Color $\in$ NP-Hard. To do this, we reduce from 3-Color to 4-Color, or show that 3-Color $\leq_p$ 4-Color.

Graph Construction for 4-color: Let G^3 be an instance of 3-Colorable graph. Construct a new graph G^4 as follows: Add a single extra vertex B_1 and connect

it to every other vertex in the graph. This is clearly polynomial in the size of the graph.

Correctness: Now we must show that G^4 is a *yes*-instance of 4-Color if and only if G^3 is a *yes*-instance of 3-Color. Consider the following proof.

- Assume G^3 is 3-colorable. Therefore, G^4 is 4-colorable because the added vertex B_1, which is connected to all the other vertices in the graph, can be colored with a 4th color, and it will always be connected to vertices that are 1 of 3 other colors.
- Assume G^4 is 4-colorable. Because B_1 is connected to every vertex in the graph, B_1 must be the only vertex in G^4 that has a certain color. Therefore, all other vertices in the graph are colored 1 of 3 colors. Therefore, G^3 is 3-colorable.

Since we have shown that 4-Color $\in$ NP and 3-Color $\leq_p$ 4-Color, hence, it is proofed that 4-Color $\in$ NP-Hard. Therefore, 4-Color $\in$ NP-Complete.

Bound: The transformation takes an instance of 3-SAT with n variables and m clauses and generated a 4-colorable graph that will have the number of vertices and edges as follows:

$|V|$ = $|V|$ of 3-colorable graph + 1 = $(2n+3m+1) +1 = (2n + 3m + 2)$

$|E|$ = (($|E|$ of 3-colorable graph) + ($|V|$ of 3-colorable graph))

$= (3n + 6m) + (2n + 3m + 1) = (5n + 9m + 1)$ edges

It is easily done in polynomial time.

Example 2: Transform (5.1) into 4-colorable graph:

$(x \vee y \vee z') \wedge (x \vee y' \vee z')$

Here, number of variable $n = 3$ and number of clauses $m = 2$; corresponding to this instance of 3-CNF, following figure 5.2 shows reduced 4-colorable graph as per 4-colorable graph construction process. Total number of vertices $|V|$ and number of edges $|E|$ in reduced graph is calculated as:

$|V|$ = ($|V|$ of 3-colorable graph + 1) = $(2n + 3m + 1) +1 = (13 +1) = 14$

$|E|$ = $|E|$ of 3-colorable graph + $|V|$ of 3-colorable graph

= ((3*n* + 6*m*) + (2*n* + 3*m* + 1)) = 21 + 13 = 34.

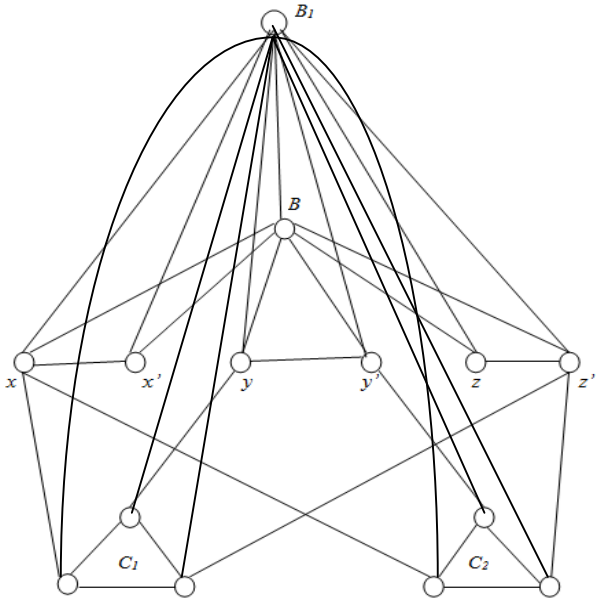


Figure 5.2: 4-Colorable Graph

5.2.3 Reduction of 3-SAT to Graph 5-Colorability (3-SAT $\leq_p$ 5-Color)

***Theorem 3*:** Graph 5-Colorability is NP-Complete

Proof: First of all we have to proof it as NP then NP-Hard. If it is both then it will be NP-Complete.

1. First we show that 5-Color $\in$ NP. Given a graph *G*, and a coloring assignment of the vertices, simply walk the graph and make certain that all adjacent vertices have a different color, and make certain that only 4 colors are used. This is clearly by *O* (|*V*| + |*E*|).
2. Now show that 5-Color $\in$ NP-Hard. To do this, we reduce from 4-Color to 5-Color, or show that 4-Color $\leq_p$ 5-Color.

Graph Construction for 5-color: Let G^4 be an instance of 4-Color. Construct a new graph G^5 as follows: Add a single extra vertex B_2 and connect it to every other vertex in the graph. This is clearly polynomial in the size of the graph.

Correctness: Now we must show that G^5 is a *yes*-instance of 5-Color if and only if G^4 is a *yes*-instance of 4-Color. Consider the following proof.

- Assume G^4 is 4-colorable. Therefore, G^5 is 5-colorable because the added vertex B_2, which is connected to all the other vertices in the graph, can be colored with a 5th color, and it will always be connected to vertices that are 1 of 4 other colors.
- Assume G^5 is 5-colorable. Because B_2 is connected to every vertex in the graph, B_2 must be the only vertex in G^5 that has a certain color. Therefore, all other vertices in the graph are colored 1 of 4 colors. Therefore, G^4 is 4-colorable.

Since, we have shown that 5-Color $\in$ NP and 4-Color $\leq_p$ 5-Color, hence, 5-Color $\in$ NP-Hard. Therefore, 5-Color $\in$ NP-Complete.

Bound: The transformation takes an instance of 3-SAT with n variables and m clauses and generated a 5-colorable graph that will have the number of vertices and edges as follows:

$|V|$ = $|V|$ of 4-colorable graph + 1 = $(2n + 3m + 2) + 1 = (2n + 3m + 3)$

$|E|$ = $|E|$ of 4-colorable graph + $|V|$ of 4-colorable graph

$= ((3n + 6m) + (2n + 3m + 1)) + (2n + 3m + 2) = (7n + 12m + 3)$

It is easily done in polynomial time.

Example 3: Transform (5.1) into 5-colorable graph:

$(x \vee y \vee z') \wedge (x \vee y' \vee z')$

Here, number of variable $n = 3$ and number of clauses $m = 2$; corresponding to this instance of 3-CNF, following figure 5.3 shows reduced 5-colorable graph as per 5-colorable graph construction process. Total number of vertices $|V|$ and number of edges $|E|$ in reduced graph is calculated as:

$|V|$ = $|V|$ of 4-colorable graph + 1 = $(2n + 3m + 2) + 1 = (14 + 1) = 15$

$|E|$ = $|E|$ of 4-colorable graph + $|V|$ of 4-colorable graph

$= ((3n + 6m) + (2n + 3m + 1)) + (2n + 3m + 2) = 34 + 14 = 48.$

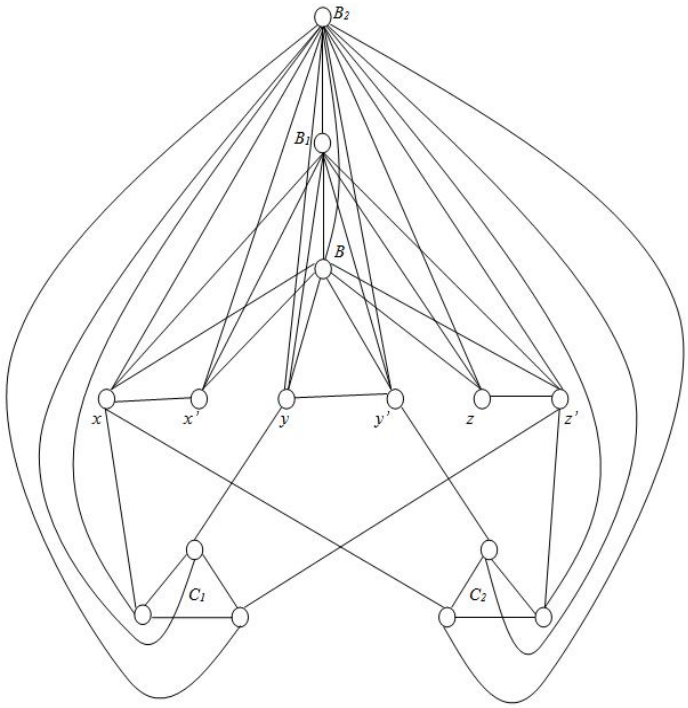


Figure 5.3: 5-Colorable Graph

5.2.4 Reduction of 3-SAT to Graph *k*-Colorability (3-SAT $\leq_p$ *k*-Color)

***Theorem 4*:** Graph *k*-Colorability is NP-Complete

Proof: First of all we have to proof it as NP then NP-Hard. If it is both then it will be NP-Complete.

1. First we show that *k*-Color $\in$ NP. Given a graph *G*, and a coloring assignment of the vertices, simply walk the graph and make certain that all adjacent vertices have a different color, and make certain that only *k* colors are used. This is clearly by O ($|V| + |E|$).

2. Now show that *k*-Color $\in$ NP-Hard. To do this, we reduce from (*k*-1)-Color to *k*-Color, or show that (*k*-1)-Color $\leq_p$ *k*-Color.

Graph construction for *k*-colorable graph: Let G^{k-1} be an instance of (*k*-1)-Color. Construct a new graph G^k as follows: Add a single extra vertex B_{k-3} and

connect it to every other vertex in the graph. This is clearly polynomial in the size of the graph.

Correctness: Now we must show that G^k is a *yes*-instance of *k*-Color if and only if G^{k-1} is a *yes*-instance of (*k*-1)-Color. Consider the following proof.

- Assume G^{k-1} is (*k*-1)-colorable. Therefore, G^k is *k*-colorable because the added vertex B_{k-3}, which is connected to all the other vertices in the graph, can be colored with a *k*th color, and it will always be connected to vertices that are 1 of (*k*-1) other colors.
- Assume G^k is *k*-colorable. Because B_{k-3} is connected to every vertex in the graph, B_{k-3} must be the only vertex in G^k that has a certain color. Therefore, all other vertices in the graph are colored 1 of (*k*-1) colors. Therefore, G^{k-1} is (*k*-1)-colorable.

Since we have shown that *k*-Color $\in$ NP and (*k*-1)-Color $\leq_p$ *k*-Color, we have shown that *k*-Color $\in$ NP-Hard. Therefore, *k*-Color $\in$ NP-Complete.

Bound: The transformation takes an instance of 3-SAT with *n* variables and *m* clauses and generated a *k*-colorable graph that will have the number of vertices and edges as follows:

$|V| = (2n + 3m + (k\text{-}2))$ vertices and

$|E| = (3n + 6m)$ edges for k=3

= (($|E|$ of (*k*-1)-colorable graph) + ($|V|$ of (*k*-1)-colorable graph)) edges for k >3

So, it is easily done in polynomial time.

5.3 Graph *k*-colorability to channel assignment problem

Formulate the channel assignment problem as a graph *k*-colorability problem. Let the vertices correspond to transmitters and edges correspond to interference between transmitters. Every vertex is labeled with a frequency range *Fi*. The question is whether one can allocate to each vertex a frequency

from its frequency range so that no vertices are connected with an edge having the same frequency.

For doing this, first of all we have to show that the frequency assignment problem is in NP. Guess (non-deterministic) a frequency assignment; go through each vertex and verify that its frequency is in the frequency set. Also go through each edge and verify that the endpoints of the frequencies are different. This takes linear time in the size of the graph.

In the second step we have to show that the channel assignment problem is NP-hard. For this, reduce graph *k*-colorability problem to frequency assignment:

Graph *k*-coloring(*G*, *k*) =

for each vertex v_i in the graph *G*

$Fi \leftarrow \{1,\ldots,k\}$

return Frequency Assignment (G,{F_i})

Finally, check correctness of above as there is a *k*-coloring of graph *G* iff there is a correct assignment of frequencies to *G*, where every vertex has frequency set {1,…,*k*}. Suppose we have a *k*-coloring of *G*. Number the colors from 1 to *k*. If a vertex has color *i*, we assign to the corresponding vertex (transmitter) in the frequency allocation problem the frequency *i*. This is a correct frequency assignment because we have been based on a correct *k*-coloring. In the other direction: assume that we have a correct frequency assignment. We get a *k*- coloring by allowing a vertex to have color *i* if the corresponding transmitter has been assigned frequency *i*.

5.4 Summary

The primary focus of this chapter is to introduce a generalized reduction approach from 3-SAT to *k*-colorable graph. Our polynomial reduction approach generate a *k*-colorable graph with $|V| = (2n + 3m + (k-2))$ vertices and $|E| = (3n + 6m)$ edges for $k = 3$ and $|E| = ((|E|$ of $(k$-1)-colorable graph + ($|V|$ of (k-1)-colorable graph)) edges for $k > 3$ corresponding to any instance of

3-CNF-SAT. Then, we give the formulation of graph *k*-colorability to frequency assignment problem in cellular network.

Chapter 6

Phase Transition in Reduction between 3-SAT and Graph Colorability

6.1 Introduction

In order to increase better understanding of NP-completeness, theoretical computer scientists have studied many different aspects of different problems. Even though solving an NP-complete problem is supposed to always take exponential time in the worst case, there are many special cases of NP-complete problems that can be solved relatively efficiently. Researchers were wondering why some problems were so much easier to solve than others, and they discovered that there is often a parameter characterizing a problem that affects the difficulty of solving it while exploring properties of NP-complete problems. The NP-complete Satisfiability problem by Cook and graph coloring problem by Karp [60] show this phenomenon known as a phase transition. Each of these problems has a parameter describing it, and when the parameter is increased to a critical value, the problems' solutions change dramatically. In the case of Boolean satisfiability, the problems are easy to solve if the formulas do not have too many clauses, compared with the number of variables. Graph coloring problems are easy to solve if the graphs do not have too many edges, compared with the number of vertices.

In this chapter, we calculated and analyzed phase transitions of generated 3-CNF-SAT and 3-colorable graph by our reduction method of transforming 3-SAT to/from 3-Colorable graph. Then we compare calculated phase transition with known phase transition. Since all NP-complete problems can translate into one another, study of phase transition gives a better understanding of NP-complete problems.

The chapter is organized as follows, section 6.2 discuss the basic concept about Boolean satisfiability, graph *k*-colorability. Section 6.3 discuss the concept of phase transition and standard phase transition of 3-SAT and 3-colorability problem. In section 6.4, we calculated the phase transition of systematically generated 3-colorability graph from 3-SAT, further section 6.5 gave the analysis of phase transition of reduction of 3-SAT which is reduced from 3-colorable graph. Finally results are compiled at section 6.6.

6.2 Phase Transition

A phase transition in a combinatorial structure occurs when a small change in the parameters of the structure results in a drastic change in the structure itself. This change occurs when the parameter reaches a certain critical value known as a threshold. This is analogous to physical phenomena such as ice melting when the temperature reaches 32^0 F. One another daily-life example of phase transitions is water changing from ice (solid phase) to water (liquid phase) to steam (gas phase) when temperature increases.

Existence of phase transition and location of threshold are known thoroughly only for relatively "easy" problems like 2-coloring and 2-SAT. In all these problems, the colorability or satisfiability depends on the presence of a cycle in the constraints which is relatively easy to distinguish. For "hard" problems like 3-SAT and 3-Coloring, existence of phase transition is not known strictly. Approximate locations of the thresholds are computed experimentally by Martin, Monasson and Zecchina [60] using non-rigorous methods of physics, showed 3-SAT has a phase transition [65] at $\alpha^* \approx 4.25$. It has also been shown that graph colouring problems for 3 color exhibit a phase transition, where problems change from being easy to colour, to being hard to colour, and on to problems that obviously cannot be coloured. The phase transition occurs for 3-colorable graph [61] [64] at a critical value of connectivity $G_c = 4.67$. The value of the parameter $\alpha^*$and G_c^* at which the transition occurs is known as the threshold value for 3-SAT formula and 3-colorable graph respectively.

For random instances of NP-complete problems, phase transitions provide some insight into both the structure of satisfiable instances and "hardness" of these instances. By hardness, we mean the time-complexity of a complete algorithm to determine whether an instance is satisfiable or not. Problem instances to be found below the threshold, i.e $\alpha < \alpha^*$, are *under-constrained* and are satisfiable with high probability, the instances to be found above the threshold, i.e. $\alpha > \alpha^*$, are *over-constrained* and are unsatisfiable with high probability and instances located near the threshold, i.e. $\alpha \approx \alpha^*$, are *critically-constrained*, and so the algorithm does a lot of back-tracking before finding either a solution or a contradiction.

One of the first discovered phase transitions in a mathematical structure is in the Erdos-Renyi random graph model: $G(n, p)$ [55]. $G(n, p)$ is a family of graphs that contain exactly n vertices, and between each pair of vertices, there is an edge with probability p. Erdos and Renyi discovered a phase transition in their $G(n, p)$ model: if $p < ((\ln n) / n)$, then almost surely, the graph is not fully connected, and if $p > ((\ln n) / n)$, then almost surely, the graph is fully connected. Thus, the point $((\ln n) / n)$ is known as a sharp threshold.

6.2.1 Phase Transition in 3-SAT

It is well-known that the Boolean satisfiability problems show a phase transition at threshold α. The parameter α can be defined as:

$$\alpha = \frac{Number\_of\_Clauses}{Number\_of\_Variables} \tag{6.1}$$

Now, one of the big question about the criteria for satisfiability of a 3-SAT formula which is randomly generated? Let a randomly generated 3-SAT instances having n variables and a standard critical value or a standard sharp threshold α^* for 3-SAT where it shows phase transition. Suppose, by any reduction method we generate 3-SAT instances systematically which shows its phase transition at threshold α then how we decide this generated 3-SAT formula will be satisfiable. In [63], there is a condition to check satisfiability of 3-SAT instances as: if $\alpha < \alpha^*$, the formula is almost surely satisfiable, and if $\alpha > \alpha^*$, the formula is almost surely not. This phase transition has been widely

studied, and for 3-SAT, the value of α^* has been empirically determined [65] [66] to be around $\alpha^* \approx 4.25$.

6.2.2 Phase Transition in 3-Colorability

The graph coloring problem on a graph $G = (V, E)$ with $|V|$ vertices and $|E|$ edges, also shows a phase transition [64]. In this case, the parameter involved is the graph connectivity, defined as

$$G_c = |V|p \tag{6.2}$$

where $|V|$ is the number of vertices in the graph, and p is the edge probability. Of course, since G is already defined, p is derived as

$$p = \frac{Number\_Of\_Edges}{Number\_Of\_PossibleEdges} = \frac{|E|}{\frac{|V|(|V|-1)}{2}} \tag{6.3}$$

Thus, combining (6.2) and (6.3), we get an easy-to-use formula for G_c is

$$G_c = \frac{2|E|}{|V|-1} \tag{6.4}$$

For 3-colorability, there is a known phase transition at a connectivity value $G_c^* \approx 4.67$. If $G_c < G_c^*$, then almost surely, the graph is colorable using three colors. If $G_c > G_c^*$, then almost surely, it is impossible to color the graph with only three colors.

6.3 Phase transition of reduced 3-colorable graph from 3-SAT instance

According to [54], polynomial reduction process from 3-SAT to 3-Colorability Graph is illustrated as follows: Create a separate *triangle* for each variable and each clause corresponding to the given 3-CNF-SAT formula. All triangles for variables have a common vertex B (we can say base vertex) which preempts one color, so that the other two vertices of each such triangle corresponding to the variable and its negation (complement) must be assigned two different colors i.e. truth assignment either TRUE or FALSE. Then, we connect each

vertex of a clause triangle to the corresponding literal vertex. Each such edge forces its two endpoints to use different colors.

Bound: Suppose the transformation takes an instance of 3-SAT formula with n variables and m clauses and generated a 3-colorable graph. Calculating the vertices in reduced 3-colorable graph as follows:

- 2 vertices per variable (variable and its negation): $2n$ vertices
- 3 vertices per clause: $3m$ vertices
- 1 vertex is common for all variable triangles: 1 vertex

Similarly, calculating the edges in reduced 3-colorable graph as follows:

- 3 edges per variable: $3n$ edges
- 3 edges per clause: $3m$ edges
- 3 edges per clause for connection with variables: $3m$ edges

Thus, the generated 3-colorable graph will have $(2n + 3m + 1)$ vertices and $(3n + 6m)$ edges.

Example 1: Transform following 3-CNF-SAT formula into 3-colorable graph:

$(x \vee y \vee z') \wedge (x \vee y' \vee z')$

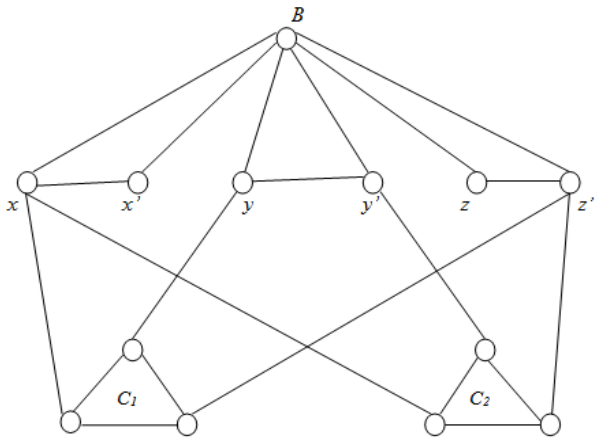


Figure 6.1: Generated 3-colorable graph from an instance of 3-CNF-SAT

The graph connectivity G_c can be computed from (6.4) with $|V| = (2n + 3m + 1)$ and $|E| = (3n + 6m)$:

$$G_c = \frac{2|E|}{|V|-1} = \frac{2(3n+6m)}{2n+3m}$$

From (6.1), $m = \alpha n$, substitute it on above, so

$$G_c = \frac{(6n+12\alpha n)}{2n+3\alpha n}$$

Then in the limit as $n \to \infty$,

$$G_c = \frac{(6+12\alpha)}{2+3\alpha} \tag{6.5}$$

Substituting the known phase transition for 3-SAT, $\alpha^* \approx 4.25$, into (6.5), which yields a phase transition in the generated 3-colorability problem at a connectivity value $G_c \approx 3.86$. The 3-colorability problem has a known phase transition at connectivity value $G_c^* \approx 4.67$. Since $G_c < G_c^*$, therefore we can say that our systematically generation approach of 3-colorable graph from 3-CNF-SAT is easy to generate for large formula also.

6.4 Phase transition of reduced 3-CNF-SAT expression from 3-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 3-color {1, 2, 3} in such a way that no two adjacent vertices should have same color. In [53], we have presented 3-CNF-SAT encoding procedure of k-colorable graph. This technique has two basic approaches; one is vertex constraint and second is edge constraint approach which will apply on vertices and edges of the graph respectively.

As per vertex constraint, color each vertex of a graph G in such a way that vertex v_{ic} should have at least one color among available 3-colors as follows:

$$v_{ic} = (v_{i1} \vee v_{i2} \vee_{i3}) \tag{6.6}$$

where v_{ic} is a vertex v_i (i = 1,2,…,n vertices) is colored by available colors c (c = 1,2,3 colors). Equation (6.1) generates one clause of length-3 in CNF corresponding to each vertex of graph. Finally, we get total $|V|$ clauses in 3-

CNF-SAT formula as per vertex constraint approach, where $|V|$ is the number of vertices in the graph.

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 3-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. The purpose of this approach is to ensure that two adjacent vertex should be assigned different color.

$$e_j = \neg(u_1 \wedge v_1) \wedge \neg(u_2 \wedge v_2) \wedge \neg(u_3 \wedge v_3)$$

Above equation can also be written in conjunctive normal form as:

$$e_j = (\neg u_1 \vee \neg v_1) \wedge (\neg u_2 \vee \neg v_2) \wedge (\neg u_3 \vee \neg v_3) \tag{6.7}$$

Equation (6.7) generates 3 clause of length-3 in CNF corresponding to each edge of graph. Finally, we get total 3*$|E|$ clauses in 3-CNF-SAT formula as per edge constraint approach, where $|E|$ is the number of edges in the graph.

Bound: Thus, total number of clauses in 3-CNF-SAT formula corresponding to 3-colorable graph =$|V|$ + 3*$|E|$, which is polynomial reduction of 3-colorable graph to 3-SAT, where $|V|$ and $|E|$ are number of vertices and edges of graph G.

Phase Transition: Phase transition in Boolean satisfiability is measured by a ratio α, it is presented as:

$$\alpha = \frac{Number\_of\_Clauses}{Number\_of\_Variables} = \frac{|V|+3|E|}{3|V|+3|E|} \tag{6.8}$$

Equation (6.4) can be rearranged so that α can be expressed in terms of edge connectivity G_c as:

$$|E| = \frac{(|V|-1)G_c}{2}$$

Applying this on (6.8) and substituting the known threshold for 3-colorable graph $G_c \approx 4.67$,

$$\alpha = \frac{8|V|-7}{10|V|-7}$$

Assuming $|V| \to \infty$ gives the ratio $\alpha = 0.8$, which is relatively very less to the known phase transition of 3-SAT, $\alpha^* \approx 4.25$ for random 3-SAT instances. Since $\alpha < \alpha^*$, therefore we can say that the 3-CNF-SAT generation technique from 3-colorable graph is quite easy.

6.5 Results and Discussion

Final result of the calculated phase transition for generated 3-SAT and 3-Colorable Graph by our reduction approach is compared with known phase transition and previously generated phase transition [27] in table 6.1 as follows:

Table 6.1: Comparisons of Phase Transition of reduced 3-SAT and 3-Colorable Graph with standard and previously generated phase transition values.

Reduced NP-Complete Problem	Known Phase Transition	Alaxander's Phase Transition [27]	Our Calculated Phase Transition
Generated 3-SAT expression corresponding to 3-colorable graph	$\alpha^* = 4.25$	$\alpha = 1.38$	$\alpha = 0.8$
Generated 3-Colorable Graph corresponding to 3-SAT expression	$G_c^* = 4.67$	$G_c = 3.91$	$G_c = 3.86$

6.6 Summary

In this chapter, we have just analyzed the behavior of two NP-Complete problem say 3-Satisfiability and Graph 3-Colorability during reduction from each other. Our reduction approach of 3-CNF-SAT to/from 3-colorable graph generates lower phase transition than known phase transition. The generated 3-colorable graph from an instance of 3-CNF-SAT gave phase transition at graph connectivity $G_c = 3.86$ whereas known phase transition of 3-colorable graph at $G_c = 4.67$. It means the reduction approach to generate 3-colorable graph is better than earlier one. Similarly, the generated 3-CNF-SAT from 3-colorable graph gave phase transition $\alpha = 0.8$ which is very lower than known phase transition of 3-SAT, $\alpha = 4.25$. It means the reduction method is more efficient and easy to generate 3-CNF-SAT from 3-colorable graph. The differences in phase transitions suggest that different reductions have different efficiencies that means the different methods of reducing 3-colorability to/from 3-SAT yielded different phase transitions.

Chapter 7

Channel Assignment Problem in Cellular Network and its Reduction to Satisfiability using Graph *k*-Colorability

7.1 Introduction

In cellular networks, a geographical area is divided into smaller service areas called cells and each of this cell's has a base station. All the terminals or the users in those cells communicate with their corresponding cell area's base stations. For these, communication links to be established, the available frequency spectrum should be used and reused very efficiently. The efficient reuse in the spectrum helps to reduce the cost of service by reducing the number of base stations and also accommodating more number of users per base stations. To maximize utilization of frequency band, the limited band of available frequency is divided into a number of channels. A channel can be reused many times for different transmitters if the transmitters are far enough from one another so that the co-channel interference between them should be low enough. If there are two close transmitters using the same channel simultaneously, they will suffer from severe co-channel interference and the quality of communication service will be unsatisfactory. This can be efficiently done by the proper channel assignment.

Since, the channel assignment problem is very similar to the graph *k*-colorability problem [50]. But, till now there are not any known deterministic methods that can solve a graph *k*-colorability problem (GCP) or any NP-complete problem in a polynomial time. There is an alternative approach to solve it efficiently by propositional Satisfiability which is first known NP-

Complete problem. The satisfiability problem (SAT) is one of the most prominent problems in theoretical computer science, which has become increasingly popular and important insights into our understanding of the fundamentals of computation. It is used as a starting point for proving that other problems are also NP-hard. We can reduce any NP-Complete problem to/from SAT. Since, there have been dramatic improvements in SAT solver technology over the past decade. This has lead 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 satisfiability is an important concept to solve channel assignment in cellular network. In this chapter, we study the channel assignment problem in cellular network and then reduce it to 3-CNF-SAT expression using polynomial reduction of the graph *k*-colorability to satisfiability [53].

The chapter is organized as follows, section 7.2 discuss the basic concept of graph *k*-colorability and channel assignment problem. In section 7.3, we formulate channel assignment problem using graph *k*-colorability, further section 7.4 gave reduction approach of channel assignment problem to satisfiability expression using graph *k*-colorability and then finally section 7.5 illustrate it by an example.

7.2 Channel Assignment Problem

The assignment of channels to cells or mobiles is one of the fundamental resource management issues in a mobile communication system. A channel assignment problem [67][68] or the frequency assignment problem is nothing but the task of assigning frequency or channel from a frequency spectrum to a set of transmitters and receivers satisfying certain hard conditions. There are basically two prime constraints that affect the channel assignment and its reusability. First constraint is co-channel interference and this interference is due to the allocation of the same channel to a certain pair of cells close enough to cause interference, i.e. a channel assigned to one cell cannot be reused in its nearby cells that are within its co-channel interference range. Second constraint is adjacent channel interference and this interference is due to the allocation of

adjacent channels (e.g., c_1 and c_2) to a certain pair of cells (normally adjacent pair) simultaneously, i.e. channel assigned to adjacent cells must maintain a minimum separation of a channel.

The above two constraint are also called hard constraint [47] for channel assignment problem and these are tackled by taking following approach. The co-channel interference is overcome by separating the channels by a distance "*d*" called "co-channel reuse distance" i.e., the minimum distance required between the centers of two cells using the same channel to maintain the desired signal quality is known as the co-channel reuse distance *d*. The cells with center-to center distance less than *d* belong to the same region. No channels are reused within that region. Second constraint the adjacent channel interference is approached by imposing channel separation. Let us consider two vertices *u* and *v* of graph *G*, let c_u and c_v be the channels assigned to them then the minimum channel separation should satisfy the following condition $|c_u - c_v| \leq d_{uv}$. This d_{uv} also represent interference strength (or interference weight on edge) between two endpoints *u,v* of an edge. Channel assignment should be in such a way that satisfies both co-channel re-use and adjacent channel interference conditions.

7.3 Problem Formulation: Channel Assignment Problem as Graph *k*-colorability

A graph *k*-colorability approach can be reduced to an instance of the channel assignment problem by considering an undirected graph $G = (V, E)$ where *V* represent the set of vertices in graph which are the base station or cells in cellular network and *E* represent the set of edges in graph which correspond to pairs of base stations in cellular network whose transmission regions intersect. Figure 7.1 represent a mapping between graph *k*-colorability with channel assignment problem in cellular network. The corresponding graph of cellular network is called an interference graph. In this graph, channels are assigned to stations (or vertices of graph) by colors. We assume that channels (colors) are non negative ordered number 1,2,,…,*k*. Suppose there is an cellular network

having n hexagonal cells say $x_1,x_2,\ldots,x_n$ each having one base station at the center transmitting with any of the k available channels c_m ($m=1,2,\ldots,k$). The minimal number of channels required to construct an interference-free channel assignment is equal to the minimal number of colours required to assign the color to the vertices of G. The channel assignment problem is more complicated than the graph coloring problem in the sense that an interference constraint does not just express that a pair of stations must be assigned different channels, but it also specifies a minimal required distance.

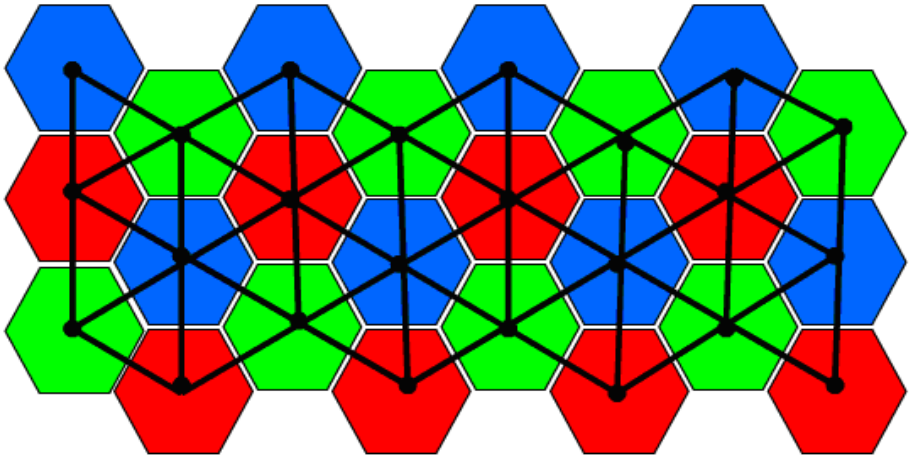


Figure 7.1: Mapping of graph k-colorability and channel assignment problem in cellular network.

The channel assignment problem can also be stated as follows: Given a set of n cells or base stations, a set of k channels and a set of interference constraints, assign each station a channel without violating any interference constraint using limited span of frequency spectrum. Channels need to be assigned to the cells or base stations such that communication via these stations does not interfere. Interference generally occurs when the same or close frequencies are assigned to stations that are situated near each other.

7.4 Reduction Approach of Channel Assignment Problem into Satisfiability using Graph *k*-Colorability

Let there is an interference graph $G = (V, E)$ corresponding to a cellular network, where V is the set of n base station $\{v_1, v_2, \ldots, v_n\}$ and E is the set of m edges $\{e_1, e_2, \ldots, e_m\}$ corresponding to interference between stations in network. The cells have to be assigned k-channel (color) $c_m = \{c_1, c_2, \ldots, c_k\}$ in such a way that it satisfies interference constraint $|c_u - c_v| \leq d_{uv}$, where any two channel c_u and c_v are assigned to two adjacent node u and v respectively. To encode this channel assignment in cellular network into propositional formula, we use two approach say base constraint approach and interference constraint approach which will apply on vertices (base stations) and edges (interference between two adjacent vertex) of the graph respectively. The 3-CNF-SAT encoding formulation of channel assignment problem using k-colorable graph is presented below:

7.4.1 Base Station Constraint Approach

Assign channels (colors) c_m (for $m = 1,2,\ldots,k$ channels) to each base station (or vertex) v_i (where $i = 1,2,\ldots,n$ stations) of a cellular network (or graph G) as v_{ic_m} in such a way that each station should be assigned at least one channel c_m (where $m = 1,2,\ldots,k$) among available k-channels as follows:

$$v_{ic_m} = (v_{ic_1} \vee v_{ic_2} \vee \ldots \vee v_{ic_k}) \tag{7.1}$$

Expression (7.1) generates one clause of length-k in CNF corresponding to each station in network. 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:

$$(x_1 \vee x_2 \vee \neg y_1) \wedge (x_3 \vee y_1 \vee \neg y_2) \wedge (x_4 \vee y_2 \vee \neg y_3) \wedge \ldots \wedge (x_{k-2} \vee y_{k-4} \vee \neg y_{k-3}) \wedge (x_{k-1} \vee x_k \vee y_{k-3}) \tag{7.2}$$

Expression (7.2) 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 (7.2) to (7.1) and we get total (k-2)*|V| clauses in 3-CNF-SAT corresponding to base station constraint approach.

Now, conjunct the 3-CNF-SAT encoding expressions of all n stations of graph G and store it in F_{bs}. Total number of clauses in formula F_{bs} which is obtained as per base constraint approach, is represented by $|F_{bs}|$ as:

$$F_{bs} = (v_{1c_m} \vee v_{2c_m} \vee \ldots \vee v_{nc_m}) \tag{7.3}$$

$|F_{bs}|$ = (k-2)*|V| clauses in 3-CNF-SAT (7.4)

7.4.2 Interference Constraint Approach

We assume that to avoid interference, two channels c_u and c_v used by two different cells u and v must differ at least by distance d_{uv} between these cells (the weight of the connecting edge that is also called interference strength) and follow the following hard constraint:

$|c_u - c_v| \leq d_{uv}$ for all channels c_m, where m =1,2,…,k (7.5)

An interference constraint is a triplet (u,v,d_{uv}), where $d_{uv} \geq 0$ is the frequency reused distance d_{uv} required between the channels assigned to cells u and v. As per interference constraint approach, assign channels to two end points of each edge e_j (j=1,2,…,.m) of a given graph in such a way that two station (u, v) connecting with an arc should satisfy the above interference constraint (7.5).

The interference constraints on an edge e = (u,v) can be modeled into propositional expression as:

$$e_j = \neg(u_{c_1} \wedge v_{c_1}) \wedge \neg(u_{c_1} \wedge v_{c_2}) \wedge \ldots \wedge \neg(u_{c_1} \wedge v_{c_k}) \wedge \neg(u_{c_2} \wedge v_{c_1}) \wedge \neg(u_{c_2} \wedge v_{c_2}) \wedge \ldots \wedge \neg(u_{c_2} \wedge v_{c_k}) \wedge \ldots \wedge \neg(u_{c_k} \wedge v_{c_1}) \wedge \neg(u_{c_k} \wedge v_{c_2}) \wedge \ldots \wedge \neg(u_{c_k} \wedge v_{c_k}) \tag{7.6}$$

$\forall$ Channel c_m (m=1,2,…,k) such that $|c_i - c_j| \leq d_{ij}$

That means above expression gives all the possible clauses between two end point of an edge (*u*, *v*) which satisfy above hard constraint (7.5) to avoid interference between channels. Since expression (7.6) is not in the 3-CNF; so it can be written in 3-CNF as:

$$e_j = (\neg u_{c_1} \vee \neg v_{c_1}) \wedge (\neg u_{c_1} \vee \neg v_{c_2}) \wedge \ldots \wedge (\neg u_{c_1} \vee \neg v_{c_k}) \wedge (\neg u_{c_2} \vee \neg v_{c_1}) \wedge (\neg u_{c_2} \vee \neg v_{c_2}) \wedge \ldots \wedge (\neg u_{c_2} \vee \neg v_{c_k}) \wedge (\neg u_{c_k} \vee \neg v_{c_1}) \wedge (\neg u_{c_k} \vee \neg v_{c_2}) \wedge \ldots \wedge (\neg u_{c_k} \vee \neg v_{c_k}) \quad (7.7)$$

$\forall$ Channel c_m (m=1,2,…,k) such that $|c_i - c_j| \leq d_{ij}$

It means (7.7) generates all clauses of channel assignment if and only if it satisfy hard constraint of (7.5). Since (7.7) is in 3-CNF-SAT, so there is no need to apply (7.2) on it. Finally we get, max k^2*$|E|$ clauses in 3-CNF-SAT as per interference constraint approach.

Now, conjunct the 3-CNF-SAT encoding expressions of all the *m* edges of graph *G* and store it in F_I. Total number of clauses in this formula is represented by $|F_I|$ as:

$$F_1 = (e_1 \wedge e_2 \wedge \ldots \wedge e_m) \quad (7.8)$$

$|F_I|$ = max(k^2*$|E|$) clauses in 3-CNF-SAT (7.9)

if and only if formula holds $|c_u - c_v| \leq d_{uv}$ for all channels c_m, where m =1,2,…,k

To get final 3-CNF-SAT encoded formula *F*, we conjunct formula obtained by base station constraint approach (3) and formula obtained by interference constraint approach (7.8) as:

$$F = ((v_{1c_m} \wedge v_{2c_m} \wedge \ldots \wedge v_{nc_m}) \wedge (e_1 \wedge e_2 \wedge \ldots \wedge e_m)) \quad (7.10)$$

where (7.10) satisfy hard constraint of channel assignment problem given in (7.5).

7.4.3 Maximum bound of generated 3-CNF-SAT Formula

Total no. of clauses in 3-CNF-SAT formula contains the clauses generated by (7.4) and (7.7). Finally we get 3-CNF-SAT formula *F* as:

$|F|$ = (((*k*-2)*$|V|$) + *max* (k^2*$|E|$)) clauses

if and only if it satisfy hard interference constraint of channel assignment (7.5) as: $\forall$ Channel c_m (m=1,2,…,k) such that $|c_i - c_j| \leq d_{ij}$

7.5 Illustration by an Example

Let there is an small instance of cellular network in figure 7.2, which has 4 base stations *u,v,w,x*; three available channels 2,4,7 and interference weight or channel reuse distance *d* are associated with each edge. Now we have to reduce this instance of channel assignment to 3-CNF-SAT formula.

First of all, we assign at least one channel to each station and encode it by base station constraint approach (6.4) as:

$$F_{bs} = (u_2 \vee u_4 \vee u_7) \wedge (v_2 \vee v_4 \vee v_7) \wedge (w_2 \vee w_4 \vee w_7) \wedge (x_2 \vee x_4 \vee x_7)$$

Now we encode edges in such a way that interference constraint (7.5) could be satisfied. There are 4 edges in network as $e_1(u,w)$, $e_2(u,v)$, $e_3(u,x)$ and e_4 (v,w).

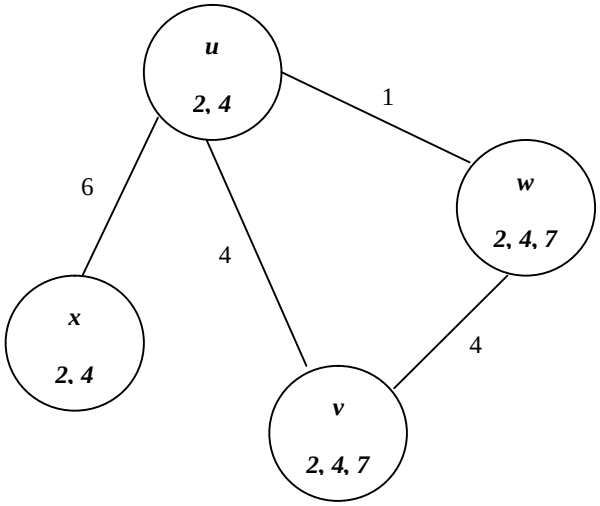


Figure 7.2: Small Channel Assignment Problem Instance

An edge connecting nodes *u* and *v* indicates that the channels assigned to stations *u* and *v* must be four or more than four apart (d_{uv}=4). Similarly d_{uw}=1, d_{ux}=6 and d_{vw}=4. This will generate following 3-CNF clauses:

$$e_1 = (\neg u_2 \vee \neg w_2) \wedge (\neg u_4 \vee \neg w_4)$$

$e_2 = (\neg u_2 \vee \neg v_2) \wedge (\neg u_2 \vee \neg v_4) \wedge (\neg u_4 \vee \neg v_2) \wedge (\neg u_4 \vee \neg v_4) \wedge (\neg u_4 \vee \neg v_7)$

$e_3 = (\neg u_2 \vee \neg x_2) \wedge (\neg u_2 \vee \neg x_4) \wedge (\neg u_4 \vee \neg x_2) \wedge (\neg u_4 \vee \neg x_4)$

$e_4 = (\neg v_2 \vee \neg w_2) \wedge (\neg v_2 \vee \neg w_4) \wedge (\neg v_4 \vee \neg w_2) \wedge (\neg v_4 \vee \neg w_4) \wedge (\neg v_4 \vee \neg w_7) \wedge (\neg v_7 \vee \neg w_4) \wedge (\neg v_7 \vee \neg w_7)$

To get final 3-CNF-SAT formula F, conjunct F_{bs} with e_1, e_2, e_3, and e_4. To model all the interference constraints in 3-CNF-SAT, 22 clauses are generated, that involving 10 variables.

$F = (F_{bs} \wedge e_1 \wedge e_2 \wedge e_3 \wedge e_4)$

$F = (u_2 \vee u_4 \vee u_7) \wedge (v_2 \vee v_4 \vee v_7) \wedge (w_2 \vee w_4 \vee w_7) \wedge (x_2 \vee x_4 \vee x_7) \wedge (\neg u_2 \vee \neg w_2) \wedge (\neg u_4 \vee \neg w_4) \wedge (\neg u_2 \vee \neg v_2) \wedge (\neg u_2 \vee \neg v_4) \wedge (\neg u_4 \vee \neg v_2) \wedge (\neg u_4 \vee \neg v_4) \wedge (\neg u_4 \vee \neg v_7) \wedge (\neg u_2 \vee \neg x_2) \wedge (\neg u_2 \vee \neg x_4) \wedge (\neg u_4 \vee \neg x_2) \wedge (\neg u_4 \vee \neg x_4) \wedge (\neg v_2 \vee \neg w_2) \wedge (\neg v_2 \vee \neg w_4) \wedge (\neg v_4 \vee \neg w_2) \wedge (\neg v_4 \vee \neg w_4) \wedge (\neg v_4 \vee \neg w_7) \wedge (\neg v_7 \vee \neg w_4) \wedge (\neg v_7 \vee \neg w_7)$

7.6 Results and Discussion

In this chapter, we have presented a simple approach to reduce the channel assignment problem in cellular network to 3-CNF-SAT using graph *k*-colorability. Our reduction formulation of channel assignment to 3-CNF-SAT generates total ((*k*-1)*|*V*| + max(k^2 *|*E*|)) clauses for all *k* channels c_m (*m*=1,2,…,*k*) such that $|c_i - c_j| \leq d_{ij}$;

7.7 Summary

Since, it is already proved that channel assignment problem in cellular network is equivalent to graph *k*-colorability; also both problems are NP-complete. To keep this in mind, in this chapter, we have presented a reduction approach for channel assignment problem into 3-CNF-SAT expression using graph *k*-colorability. Further, encoded 3-CNF expression can be solved using efficient SAT solver.

Chapter 8

Conclusion and Scope for Future Work

8.1 Conclusion

This thesis covered some research investigations of graph coloring problem and its applications based on Satisfiability and maximal independent set. We have designed a novel tree based approach for finding maximal independent set. Also we have attempted to formulate the encoding/reductions 3-CNF-SAT to/from graph *k*-colorability and drawn the important conclusions. We summarize the same as follows: In this thesis, we have developed polynomial 3-CNF-SAT encodings for the famous graph coloring problem. For any input graphs from DIMACS, the reduction has been performed on graphs to encode into 3-CNF-SAT formula. It has been observed that our formulations using adjacency list for the graphs generate generalized SAT formula. 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. To fetch the satisfiable values of the SAT formulas, the generated SAT clauses are passed to SAT solver. The number of satisfiable values so obtained reflects the unique color values for the input graphs. Thus, it is concluded that the using Satisfiability the number of unique colored vertices in the input graphs can be recognized.

In this thesis, graph *k*-colorability can be reduced to channel assignment problem for the assignment of *k*- channel if and only if graph is *k*-colorable. Polynomial 3-CNF-SAT encoding of Graph *k*-colorability is the basis for reduction of channel assignment problem to 3-CNF-SAT. Reduction of the channel assignment problem in cellular network to 3-CNF-SAT using graph *k*-colorability generates total $((k\text{-}1)*|V| + \max(k^2 *|E|))$ clauses for all *k* channels c_m (m=1,2,…,k) such that $|c_i - c_j| \leq d_{ij}$.

In this thesis, we have formulated a generalized reduction approach from 3-CNF-SAT expression to *k*-colorable graph. Important formulations

have been developed to obtain *k*-colorable graph from 3-CNF-SAT clauses. A New reduction approach from 3-CNF-SAT formula to graph *k*-colorability generate a *k*-colorable graph with

$|V| = (2n + 3m + (k\text{-}2))$ vertices and

$|E| = (3n + 6m)$ edges for $k = 3$ or

$|E| = (|E|$ of $(k\text{-}1)$-colorable graph + $(|V|\text{-}1))$ edges for $k > 3$

8.2 Scope for Future Work

We believe that Satisfiability and maximal independent set based approach proposed in this thesis would provide promising outcomes for analysis of graph coloring problem. To facilitate further development of these approaches, we have highlighted few issues which are addressed below.

Since, channel assignment problem in cellular network is generalization of graph coloring problem i.e. multicolring graph problem or bandwidth graph coloring problem. In our thesis, we encoded a channel assignment problem into 3-CNF-SAT using graph *k*-colorability. So, there is a scope to solve this problem using efficient SAT solver. Also, one can try to solve the channel assignment problem using 3-SAT as a bandwidth graph coloring (multicoloring) problem.

In our thesis, we formulate a generalized encoding technique for *k*-colorable graph. To solve encoded SAT expression more efficiently, there is always a scope to develop a novel efficient SAT solver.

Appendix A

Dataset: DIMACS Graph Instances

Introduction

DIMACS (Center for Discrete Mathematics and Theoretical Computer Science) [34] defined a format for undirected graph, which has been used as a standard format for problems in undirected graphs. This format was also chosen for several DIMACS Computational Challenges. One purpose of the DIMACS Challenge is to ease the effort required to test and compare algorithms and heuristics by providing a common test bed of instances and analysis tools. To facilitate this effort, a standard format must be chosen for the problems addressed. This document outlines a format for graphs that is suitable for those looking at graph coloring. This format extends to a flexible format suitable for many types of graph and network problems.

Input Files

An input file contains all the information about an undirected graph. In this format, nodes are numbered from 1 up to *n* vertices in the graph. Files are assumed to be well-formed and internally consistent: node identifier values are valid, nodes are defined uniquely, exactly *m* edges are defined, and so forth. Input files having mainly following information:

- **Comments:** Comment lines give human-readable information about the file and are ignored by programs. Comment lines can appear anywhere in the file. Each comment line begins with a lower-case character c.

  c This is an example of a comment line.

- **Problem line:** There is one problem line per input file. The problem line must appear before any node or arc descriptor lines. The problem line has the following format.

  p FORMAT NODES EDGES

  The lower-case character p signifies that this is the problem line. The FORMAT field is for consistency with the previous Challenge, and should contain the word ``edge". The NODES field contains an integer value specifying *n*, the number of nodes in the graph. The EDGES field contains an integer value specifying *m*, the number of edges in the graph.

- **Edge Descriptors:** There is one edge descriptor line for each edge the graph, each with the following format. Each edge (*u*, *v*) appears exactly once in the input file and is not repeated as (*u*,*v*).

  e *u v*

  The lower-case character e signifies that this is an edge descriptor line. For an edge (*u*,*v*) the fields *u* and *v* specify its endpoints.

Graph Descriptions

All listed graphs are from the DIMACS benchmark [73], and, more exactly, they belong to the following families:

- **dsjcX.Y:** Random graphs generated by Johnson et. al [72] and used extensively afterwards by most graph coloring algorithms (like dsjc1000.5). The number of vertices is denoted by the first number while the second digit references the probability that any two vertices establish an edge (the density).

- **flatX_K**: Flat graphs due to J. Culberson. They are generated by partitioning the vertex set into K almost equal sized classes and then by selecting edges only between vertices of different classes. Finding the

best legal K-coloring is equivalent to restoring this initial partitioning. The second number K is hence the chromatic number (X is the vertex set size).

- **le450_K:** Leighton graphs, with the 450 vertices and with known chromatic number K (denoted by the second number) [4]; all graphs have a clique of size K.
- **dsjrX.Y and rX.Y:** Random geometric graphs:
    - **dsjrX.Y**: graphs presented by Johnson et. al in [72] along with the above dsjc random graphs. They are generated by picking points uniformly at random in a square and by setting an edge between all pairs of vertices situated within a certain distance. DSJRx graph instances are geometric random graphs with X nodes randomly distributed in the unit square.
    - **rX.Y**: graphs generated using the same idea by M. Trick using a program of C. Morgenstern; suffix "c" denotes the complement of a graph. Descriptions can be found in [74]. For R1000.5, only the clique number is available, but it is equal to the chromatic number and to the upper bound 234.
- **C2000.5 and C4000.5**: Huge random graphs with up to 4 million edges.
- **latin_square_10 and school*:** A latin square graph (and class scheduling graphs respectivelly) generated by Gary Lewandowski in the second Dimacs challenge.
- **myC:** Myciel graphs are based on the Mycielski transformation and These graphs are difficult to solve because they are triangle free but the coloring number increases in problem size.
- ***k*-Insertion and Full Insertion graph:** k-insertion graphs and full insertion graphs are a generalisation of myciel graphs with inserted nodes to increase graph size but not density. These instances are created by M. Caramia and P. Dell'Olmo.

- **queenX_Y:** A queen graph is a graph on n^2 nodes, each corresponding to a square of the board. Two nodes are connected by an edge if the corresponding squares are in the same row, column, or diagonal.

- **milesC:** In miles graphs nodes are placed in space with two nodes connected if they are close enough. The nodes represent a set of United States cities.

- **Leighton Graphs:** Leighton graphs are generated by Leighton's graph covering theorem (Two finite graphs which have a common covering have a common finite covering).

Bibliography

[1] Wu Q. and Hao J.K. (2012), Coloring large graphs based on independent set extraction, Computers & Operations Research, 39(2), pp 283–290.

[2] Hao J. K. and Wu Q. (2012), Improving the extraction and expansion method for large graph coloring, Discrete Applied Mathematics, 160 (16-17), pp 2397–2407.

[3] Brelez (1979), New methods to color the vertices of a graph, Communications of the ACM, 22(4), pp 251–256.

[4] Leighton (1979), A graph coloring algorithm for large scheduling problems, Journal of Research of the National Bureau of Standards, 84(6), pp 489–506.

[5] Blochligerand I., Zufferey N. (2008), A graph coloring heuristic using partial solutions and a reactive tabu scheme, Computers and Operations Research, 35(3), pp 960–975.

[6] Dorne R. and Hao J. K. (1998), Tabu search for graph coloring, T-colorings and set T-colorings, In Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization, pp. 77–92.

[7] Hertz A. and de Werra D (1987), Using tabu search techniques for graph coloring, Computing, 39, pp. 345–351.

[8] Porumbel D. C., Hao J. K. and Kuntz P. (2010), An evolutionary approach with diversity guarantee and well-informed grouping recombination for graph coloring, Computers and Operations Research, 37(10), pp 1822–1832.

[9] Fleurent C, and Ferland J A (1996), Genetic and hybrid algorithms for graph coloring, Annals of Operations Research, 63, pp 437–461.

[10] Garey M R, and Johnson D S (1979), Computers and intractability: a guide to the theory of NP-completeness, San Francisco: W.H. Freeman and Company.

[11] de Werra D, Eisenbeis C, Lelait S, and Marmol B (1999), On a graph-theoretical model for cyclic register allocation", Discrete Applied Mathematics, 93(2–3), pp 191–203.

[12] Burke E K, McCollum B, Meisels A, Petrovic S, and Qu R (2007), A graph-based hyper heuristic for timetabling problems, European Journal of Operational Research, pp 176-177.

[13] Smith D H, Hurley S, and Thiel S U (1998), Improving heuristics for the frequency assignment problem, European Journal of Operational Research, 107 (1), pp 76–86.

[14] Zufferey N, Amstutz P, and Giaccari P (2008), Graph colouring approaches for a satellite range scheduling problem, Journal of Scheduling, 11(4) pp 263–77.

[15] Chams M, Hertz A and de Werra D, Some experiments with simulated annealing for coloring graphs, European Journal of Operational Research, 32, pp 260–266.

[16] Mendez-Diaz I and Zabala P (2006), A branch and cut algorithm for graph coloring, Discrete Applied Mathematics, 154 (5), pp 826-847.

[17] Malaguti E, Monaci M, and Toth A (2008), A metaheuristic approach for the vertex coloring problem, INFORMS Journal Computation 20(2), pp 302–316.

[18] Malaguti E, Monaci M, Toth P (2011), An Exact Approach for the Vertex Coloring Problem, Discrete Optimization, vol. 8, no. 2, pp. 174-190.

[19] Graph coloring instances. http://mat.gsia.cmu.edu/COLOR/instances .html.

[20] Johnson D S, Aragon C R, McGeoch L A and Schevon C (1991), Optimization by simulated annealing: An experimental evaluation; Part II, graph coloring and number partitioning, Operations Research, 39(3), pp 378–406.

[21] Dorne R and Hao J K (1998), A new genetic local search algorithm for graph coloring, Lecture Notes in Computer Science, 1498, pp 745–754.

[22] Karp (1972), Reducibility among combinatorial problems, In R. E. Miller, J. W. Thatcher (eds.), Complexity of Computer Computations, Plenum Press, New York, USA, pp. 85–103.

[23] Galinier P, Hertz A and Zufferey N (2008), An adaptive memory algorithm for the *K*-colouring problem, Discrete Applied Mathematics, 156(2), pp 267–279.

[24] Lu Z and Hao J K (2010), A memetic algorithm for graph coloring, European Journal of Operational Research, 200(1), pp 235–244.

[25] Cook S A (2004), The Complexity of Theorem Proving Procedures, In: Proceeding of the ACM Symposium on the Theory of Computing, pp 151-158.

[26] Galinier P and Hertz A (2006), A survey of local search methods for graph coloring, Computers and Operations Research, 33(9), pp 2547–2562.

[27] Alexander T (2008), Phase Transitions in Boolean Satisfiability and Graph Coloring, Department of Computer Science, Cornell University, (www.cseweb.ucsd.edu/users/atsiatas/phase.pdf).

[28] Adleman L and Manders K (1977), Reducibility, randomness and intractability, in STOC 77: Proceedings of the ninth annual ACM symposium on Theory of computing. New York, USA,ACM Press, pp. 151-163.

[29] Chiarandini M, Stützle T (2010), An Analysis of Heuristics for Vertex Colouring, In: Proceeding Festa (ed.), Experimental Algorithms, Proceedings of the 9th International Symposium, (SEA 2010), vol. 6049 of Lecture Notes in Computer Science, pp. 326-337.

[30] Claessen K, Een N, Sheeran M and Sorensson N (2008), SAT-solving in practice, In: Proceedings of the 9th International Workshop on Discrete Event Systems Goteborg, Sweden, pp 61-67.

[31] Fleurent C, Ferland J. (1996), Object-oriented implementation of heuristics search methods for Graph Coloring, Maximum Clique, and Satisfiability, vol. 26 of DIMACS Series in Discrete Mathematics and Theoretical Computer Science, pp. 619-652.

[32] Gelder A. V. (2008), Another look at graph coloring via propositional satisfiability, Discrete Applied Mathematics, vol. 156, no. 2, pp. 230-243.

[33] Stockmeyer L (1973), Planar 3-Colorability is NP-Complete, SIGACT News, vol. 5, no. 3, pp. 19-25.

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

[35] Petersen graph, http://en.wikipedia.org/wiki/ Petersen _graph.

[36] Een N and Sorensson N (2003), An extensible sat solver, In: Springer Proceeding of the 6th International Conference on Theory and Applications of Satisfiability Testing, pp 502-518.

[37] Een N. and Sorensson N. (2005), MiniSat v1.13 - A SAT Solver with Conflict-Clause Minimization, System description for the SAT competition, 503-515.

[38] The MiniSAT page by Niklas Een and N Sorensson. http://minisat.se/

[39] MiniSAT User Guide: How to use the MiniSAT SAT Solver by David A. Wheeler. http://www.dwheeler.com/ essays/minisat-user-guide.html.

[40] Computational Series: Graph Coloring and Its Generalizations, http://mat.gsia.cmu.edu/ COLOR04.

[41] Malaguti E., and Toth P. (2010), A survey on vertex coloring problems, International Transactions in Operational Research 17, pp 1–34.

[42] Hale W K (1980), Frequency Assignment: Theory and Applications, in IEEE Proceeding, Vol.68, no.12, pp. 1497-1514.

[43] Paschos V. T. (2003), Polynomial approximation and graph-coloring, Computing, vol. 70, no. 1, pp. 41-86.

[44] Moret B M (1998), The Theory of Computation, Pearson Education, 1998, chapter 7, Proving Problem Hard, pp 226-252.

[45] Wood D C (1969), A Technique for Coloring a Graph Applicable to Large-Scale Timetabling Problems, Computer Journal, vol. 12, pp. 317-322.

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

[47] Hassan M A and Chickadel A (2011), 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, pp 58-67.

[48] Malkawi M, Al-Haj Hassan M and Al-Haj Hassan O (2008), New Exam Scheduling Algorithm using Graph Coloring, The International Arab Journal of Information Technology, Vol. 5, No. 1, pp 80-87.

[49] Taehoon P. and Lee, C.Y., (1994), On the *k*-coloring problem, Journal of Korean OR/MS Society, 19, pp. 219-233.

[50] De Werra D. and Gay Y. (1994), Chromatic scheduling and frequency assignment, Discrete Applied Mathematics 49, pp. 165-174.

[51] Adleman L and Manders K (1977), Reducibility, randomness and intractability, in STOC 77: Proceedings of the ninth annual ACM symposium on Theory of computing. New York, USA: ACM Press, pp 151-163.

[52] Cook S A (2000), The P versus NP problem, Computer Science Department, University of Toronto. http://www.claymath.org/ millennium/P_vs_NP/ Official_Problem_Description.pdf.

[53] Sharma P C and Chaudhari N S (2011), Polynomial 3-SAT Encoding for K-Colorability of Graph, Special Issue of International Journal of Computer Application on Evolution in Networks and Computer Communications (1), pp 19-24.

[54] Sharma P C and Chaudhari N S (2012), A New Reduction from 3-SAT to Graph *K*-Colorability for Frequency Assignment Problem, Special Issue of International Journal of Computer Application Issue on Optimization and On-chip Communication (5), pp 23-27.

[55] Erdos P, Renyi A (1960), On the evolution of random graphs, The mathematical institute of the Hungarian Academy of Science, pp 17-61.

[56] Johnson D S, Aragon C R, McGeoch L A, Schevon C. (1991), Optimization by Simulated Annealing: An Experimental Evaluation; Part II, Graph Coloring and Number Partitioning, Operations Research, vol. 39, no. 3, pp. 378-406

[57] Pardalos P, Mavridou T, Xue J (1998), The graph coloring problem: A bibliographic survey, Kluwer Academic Publishers, Boston, vol. 2, pp. 331-395.

[58] Johnson D S, Mehrotra A, Trick M A (2008), Special issue on computational methods for graph coloring and its generalizations, Discrete Applied Mathematics, vol. 156, no. 2, pp. 145-146.

[59] Held S, Cook W, Sewell E (2011), Safe lower bounds for graph coloring, Integer Programming and Combinatoral Optimization, pp. 261-273

[60] Martin O, Monasson R and Zecchina R (2001), Statistical mechanics methods and phase transitions in optimization problems, TCS 265, 3-67.

[61] Prosser P (1996), An Empirical Study of Phase Transition in Binary Constraint Satisfaction Problems. Artificial Intelligence, 82, pp 81-109.

[62] Karp R M (1972), Reducibility among Combinatorial Problems, Complexity of Computer Computations. New York: Plenum, pp. 85–103. http://www.cs.berkeley.edu/~luca/cs172/karp.pdf

[63] Selman B, Mitchell D G, and Levesque H J (1996), Generating hard satisfiability problems, Artificial Intelligence, 81(1–2), pp 17–29.

[64] Mulet R, Pagnani A, Weigt M, and Zecchina R (2002), Coloring Random Graphs, Research Article ,Phys. Rev. Lett. **89**, pp 268-301.

[65] Dubois O, Boufkhad Y, and Mandler J (2003), Typical random 3-SAT formulae and the satisfiability threshold. Technical Report TR03-007, Electronic Colloquium on Computational Complexity.

[66] Connamacher H and Molloy M (2004), The exact satisfiability threshold for a potentially intractable random constraint satisfaction problem. In Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer, Science, pp 590–599.

[67] Sung C W and Wong W S (1995), A Graph Theoretic Approach to the Channel Assignment Problem in Cellular System, IEEE 45th Vehicular Technology Conference, pp. 604–608.

[68] Park T, and Lee C Y (1996), Application of the Graph Coloring Algorithm to the Frequency Assignment Problem, Journal of the Operations Research Society of Japan, Vol. 39, No. 2, pp.258–265.

[69] Chams, M., Hertz, A., de Werra, D (1987), Some experiments with simulated annealing for coloring graphs. Eur. Journal. of Operation. Research. 32(2), pp 260–266.

[70] Fleurent, C., Ferland, J. (1996), Genetic and hybrid algorithms for graph coloring. Ann. of Operation Research, 63(3), pp 437–461.

[71] Hertz, A., de Werra, D. (1987), Using tabu search techniques for graph coloring, Computing. 39(4), pp 345–351.

[72] Johnson, D., Aragon, C., McGeoch, L., Schevon, C. (1991), Optimization by simulated annealing: An experimental evaluation; Part II, Graph coloring and number partitioning. Operation. Research, 39(3), pp 378–406.

[73] Johnson, D., Trick M. (1996), Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge. American Mathematical Society, 26, pp 1-10.

[74] Sewell E.C. (1996), An improved algorithm for exact graph coloring, Second DIMACS Implementation Challenge, volume 26 of DIMACS series in Discrete Mathematics and Theoretical Computer Science, pp 359--376. American Mathematical Society.

[75] Allwright J.R., Bordawekar R., Coddington P.D., Dincer K., Martin C.L. (1995), A comparison of parallel graph coloring algorithms, Technical Report SCCS-666, Northeast Parallel Architecture Center, Syracuse University, pp 1-19.

[76] Hussein A.H., Khair E.S. (2006), New Graph Coloring Algorithms, American Journal of Mathematics and Statistics 2(4), pp 739-741.

[77] Avanthay C., Hertz A, Zufferey P. (2003), A variable neighborhood search for graph coloring, European Journal of Operational Research 151 (2), pp 379–388.

[78] Falkenauer E. (1996), A hybrid grouping genetic algorithm for bin packing, Journal of Heuristics 2(1), pp 5–30.

[79] Burke E.K., McCollum B., Meisels A., Petrovic S., Qu R. (2007), A graph-based hyper heuristic for timetabling problems, European Journal of Operational Research 176, pp 177–192.

[80] Luby M. (1986), A simple parallel algorithm for the maximal independent set problem, Society for Industrial and Applied Mathematics Journal on Computing, 15, pp 1036-1053.

[81] Jones M.T., Plassmann P.E. (1993), A Parallel Graph Coloring Heuristic, Society for Industrial and Applied Mathematics Journal of Scientic Computing 14(3), pp 654-669.

[82] Matula D.W., Beck L.L.(1983), Smallest-last ordering and clustering and graph coloring algorithms, .Journal of the ACM, 30(3), pp 417-427.