A Tree-Based Graph Coloring Algorithm Using Independent Set

Harish Patidar and Prasun Chakrabarti

Abstract This paper introduces a tree data structure-based graph coloring algorithm. Algorithm explores vertices in the tree form to finds maximal independent set, than these independent sets are colored with minimum colors. Proposed algorithm is tested on various DIMACS standard of graph instances. Algorithm is design to solve graph coloring problem for high degree graphs, i.e. the proposed algorithm is highly efficient for those graphs which has number of edges to number of vertices ratio is very high. Worst and best case time complexity of proposed algorithm is also discussed in this paper.

Keywords Maximal independent set · Complement edge table
Graph coloring · Chromatic number

1 Introduction

Graph coloring problem has three different areas of problem. Fist is vertex coloring; second edge coloring and third one is phase coloring. This paper is focused on vertex coloring problem. In the vertex coloring for any given graph G = (V, E), where V is set of vertices E is set of edges, and C = {1, 2, 3, …, K} is set of colors, each vertex must assign a color from set of color in such a way that colors of two connected vertices must be different, i.e., f: V_C such that for each [u, v], f(v) is not equal to find f(u).

H. Patidar (✉) · P. Chakrabarti
Department of Computer Science and Engineering,
Sir Padampat Singhania University, Udaipur, India
e-mail: harish.patidar@gmail.com

P. Chakrabarti
e-mail: prasun.chakrabarti@spsu.ac.in

C. R. Panigrahi et al. (eds.), *Progress in Advanced Computing and Intelligent Engineering*, Advances in Intelligent Systems and Computing 714,
https://doi.org/10.1007/978-981-13-0224-4_48

1.1 Applications of Graph Coloring Problem

Graph coloring problem has a wide area of applications like nearest neighbor search [1], register allocation in compiler [2], social networking, puzzles like Sudoku solving, frequency allocation in cellular network, cognitive radio dynamic frequency distribution and many more.

1.2 Algorithms of Graph Coloring Problem

Vertex coloring or more formally it can be called as graph coloring algorithms are designed and implemented to solve the graph coloring problems. There are certain objectives to design graph coloring algorithm. The primary objective of most graph coloring algorithm is to find minimum number of colors for coloring vertices of graphs. Many algorithms also try to achieve some other objectives like optimization of time and space complexity, improvement in execution success rate [3], finding efficient algorithm for large graphs where number of vertices in graph is high and many more.

On the basis of execution pattern, graph coloring algorithm can be sequential and parallel. Graph coloring algorithm is broadly divided into two categories one is exact approach and another next one is approximate [4]. Exact method's execution success rate is high but they are not efficient for the large graphs. Approximate algorithm give results on optimum time for large graphs but their execution success rate is low.

There are many algorithms already proposed by researchers like ant colony optimization algorithm [5, 6], Cuckoo optimization, Parallel genetic algorithm [7], Modified cuckoo optimization GCA [3], constructive hyper heuristic algorithm [8], and many more.

2 Proposed Algorithm

This paper proposed an algorithm to solve the vertex coloring problem for higher degree graph. Proposed algorithm is based on finding maximum independent set using tree data structure.

2.1 Maximum Independent Sets

The subsets of the graph containing those vertices that are not connected, i.e. no element in the set is connected to any other element of the same set, are known as

independent sets. And, as the name suggests "Maximal Independent Sets" are those independent sets that contain maximum number of vertices.

2.2 *Proposed Algorithm on Petersen Graph*

For the graph in Fig. 1 the independent sets are shown in Table 1.

In Table 1, there are two maximal independent sets starting from vertex 1, i.e., Set1 and Set5. Based on the rules of proposed algorithm, Set1 is selected for further exploration.

Now, as Vertex 2 is not included in Set1, Vertex 2 is to explore independent sets for with those vertices that are not included in Set1. Table 2 shows the independent set starts from vertex 2.

Again, based on the rules, according to algorithm Set11 is selected and so Vertex 5 is explored as shown in Table 3, being the first un-included vertex till now.

As, all the vertices of graph are now included, three maximal independent sets can be selected, Set1, Set11, and Set17. Now, as it is known that each element in a maximal independent set is disconnected with others. A single color can be assigned to all the elements of each maximal independent set. Thus, assigning every maximal independent set a different color, proper minimum coloring can be

Fig. 1 Petersen graph

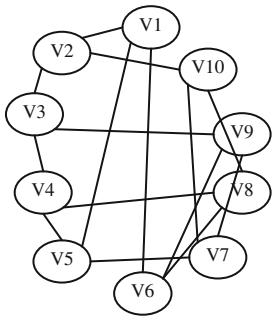


Table 1 Independent sets starting from vertex 1

S. No.	Set	Vertices
1	Set1	V1, V3, V7, V8
2	Set2	V1, V3, V8
3	Set3	V1, V3, V10
4	Set4	V1, V4, V7
5	Set5	V1, V4, V9, V10
6	Set6	V1, V4, V10
7	Set7	V1, V7, V8
8	Set8	V1, V8, V9
9	Set9	V1, V9, V10
10	Set10	V1, V10

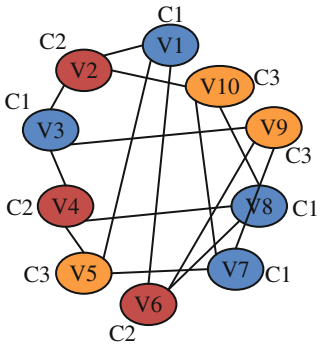
Table 2 Independent sets starting from vertex 2

S. No.	Set	Vertices
1	Set11	V2, V4, V6
2	Set12	V2, V4, V9
3	Set13	V2, V5, V6
4	Set14	V2, V5, V9
5	Set15	V2, V6
6	Set16	V2, V9

Table 3 Independent sets starting from vertex 3

S. No.	Set	Vertices
1	Set17	V5, V9, V10
2	Set18	V5, V10

Fig. 2 Colored Petersen graph using the proposed algorithm



assigned as shown in Fig. 2. C1 color is assigned to Set1, C2 color to Set11, and C3 color to Set17.

2.3 Algorithm

Here, this paper proposed a vertex coloring algorithm that calculates minimum colors for graphs. Entire algorithm is divided into three steps. The first step is the development of complement edge table. Second, is finding maximum independent sets. And the third step is coloring of the maximal independent sets.

Step 1: *Complement Edge Table*

Complement edge table is the opposite of edge table. In order to find maximal independent sets, it is required to put together those vertices that are not connected by each other. And, so if complement edge table defines which vertices are not connected, it would reduce the time complexity significantly.

By scanning the edge table, make a new edge table that comprises of only those edges which were not in the original edge table. Also, include only those edges which originate from a vertex of smaller numbering than its destination.

In this step, when making the complement edge table, the algorithm also calculates the number of occurrences of every vertex, i.e., the degree of each vertex.

Step 2: *Finding Maximal Independent Sets*

This is the core of proposed algorithm. In this step, algorithm proceeds by exploring a tree for each maximal independent set. This step itself is a multistep process, which are as follows:

i. *Initiation Step*: To find the sets, algorithm needs to commence from somewhere, and so, the initiation step is defined to select the first vertex that is not yet included in any maximal independent set. If, there is no maximal independent set yet, algorithm can be start from vertex 1. Make a new empty maximal independent set.

ii. *Tree exploration*: This section has two main activities.

a. Every vertex that is greater in numbering than the selected vertex and is a connection in the complement edge table is made a child of the selected vertex given it is not included in any maximal set till now.

b. For all children, explore one by one by making the vertices that are in connection with complement edge table, are not included in any maximal set, and are siblings of the vertex being explored. This step is repeated till there are no more vertices which can be explored.

iii. *Path selection*: Then select the path with maximum length. If more than one path has maximum length, then the sum of degree for each such path is calculated and selects the path with minimum sum of degrees. The sum of degree is calculated as

$$\text{Sum} = \text{sum of degrees of all vertices in the path} - \text{L} * (\text{L} - 1)/2 \qquad (1)$$

where L is the length of the path.
If, more than one longest path has minimum sum of degrees, first traversing left to right is selected.

iv. Path to the maximal independent set is added for each vertex in the paths, degree of all its connections is decremented.
v. Step i through iv is repeated until all the vertices are included in some maximal independent set.

Step 3: *Coloring the Maximal Independent Sets*

This is the final step of the proposed minimum coloring algorithm. This step assigns a different color to each maximal independent set, i.e. all the vertices that belong to

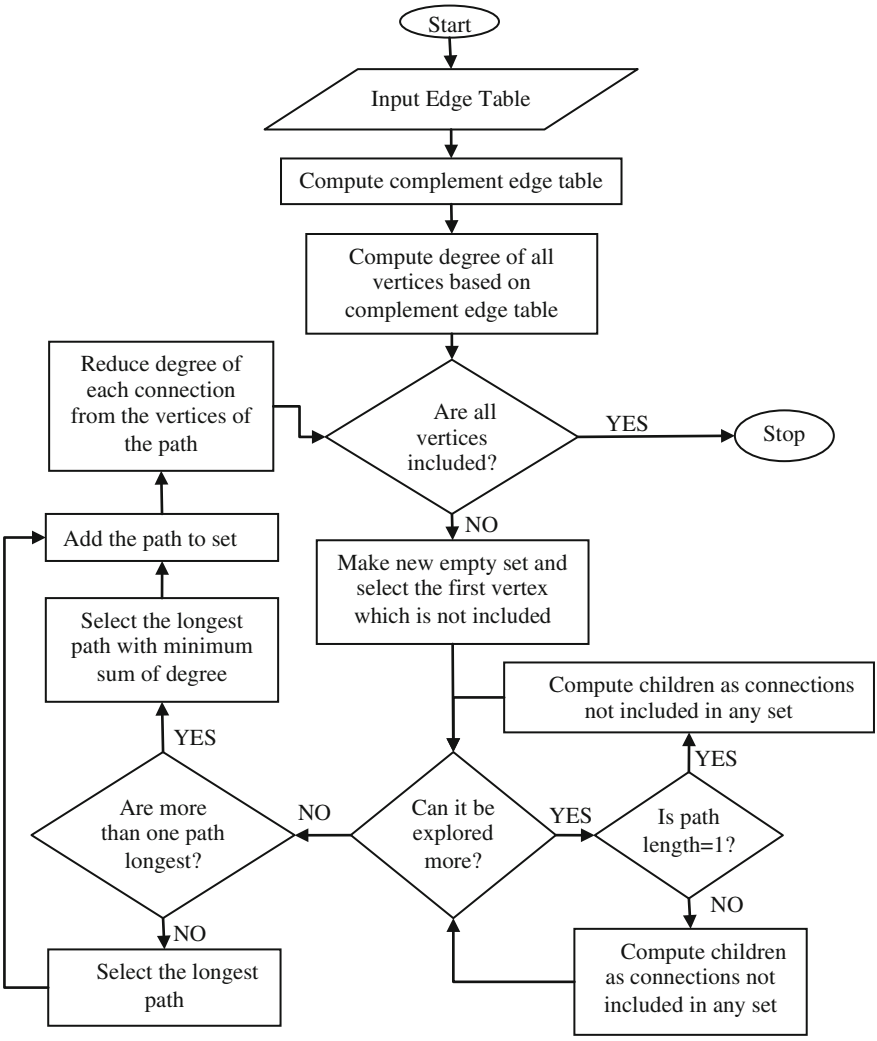


Fig. 3 Flowchart of the proposed algorithm

the same maximal independent set are assigned the same color and all the vertices that belong to different independent sets, now have different colors.

In Fig. 3 entire flow of the proposed algorithm has been shown.

3 Complexity Analysis

Worst-Case Complexity: The complexity of the proposed algorithm depends on the tree exploration. And, the maximum number of nodes would be explored when all the vertices are isolated, i.e., the edge table is empty, making ${}^{n}C_{2}/2$ edges in the complement edge table, where n is the number of vertices. So, complexity can be calculated for exploring the tree when all the vertices are isolated.

If n is the number of vertices, then V1 would have n − 1 connections. Similarly, V2 would have n − 2, V3 would have n − 3 connections, and so on, since the algorithm is accounting only the edges to vertices greater in numbering. Equation 2 is the recursion equation for the exploration.

$$T(n) = T(n-1) + T(n-2) + T(n-3) + \cdots + T(2) + T(1) \tag{2}$$

And the initial conditions would be
$T(1) = 0$ and $T(2) = 1$
From Eq. (2)

$$T(n-1) = T(n-2) + T(n-3) + \cdots + T(2) + T(1) \tag{3}$$

Subtracting (3) from (2)

$$\begin{gathered} T(n) - T(n-1) = T(n-1), \\ T(n) = 2 * T(n-1) \end{gathered} \tag{4}$$

By Eq. 4, the worst-case complexity of the algorithm is O(2n − 1).

4 Result Analysis

4.1 Test Data Sets

DIMACS graph instances are taken as data set for experiment results analysis of the proposed algorithm. DIMACS (Center for Discrete Mathematics and Theoretical Computer Science) defined a format to represent undirected graphs [9], which has been used by most of the researchers for analysis of their graph coloring algorithms. In this format graph data are stored in an input file, which contains all information about graph. In this input file, nodes are numbered from 1 to n. Edges are stored in the form of edge list like “e 1 2”.

4.2 Algorithm Implementation Platform

The proposed algorithm is implemented using Java programming language. Window 7 Ultimate 64-bit operating system platform with Intel(R) Core(TM) 2 Duo 2.10 GHz with 2 GB Installed memory (RAM) is used for algorithm execution.

4.3 Experimental Results

The proposed algorithm is tested on 26 DIMACS graph instances, includes queen graphs, DSJC series graphs, miles graphs, random series graphs, insertion and full insertion graphs. In Table 4 all experimental results are shown. Table contains the

Table 4 Experimental results of the proposed algorithm

Instance	V	E	Avg degree	K (Result)
myciel3	11	20	3.64	4
myciel4	23	71	6.17	5
queen5_5	25	320	25.60	5
1-FullIns_3	30	100	6.67	4
queen6_6	36	580	32.22	10
2-Insertions_3	37	72	3.89	4
myciel5	47	236	10.04	6
queen7_7	49	952	38.86	7
queen8_8	64	1456	45.50	10
queen9_9	81	2112	52.15	12
queen8_12	96	2736	57.00	15
queen10_10	100	2940	58.80	13
queen11_11	121	3960	65.45	15
DSJC125.9	125	6961	111.38	52
miles1500	128	10,396	162.44	74
miles1000	128	6432	100.50	45
DSJC250.9	250	27,897	223.18	82
DSJC250.5	250	15,668	125.34	35
DSJC500.9	500	224,874	899.50	150
DSJR500.1c	500	121,275	485.10	104
latin_square_10	900	307,350	683.00	119
DSJC1000.9	1000	449,449	898.90	259
R100_9 g	100	4438	88.76	42
R100_9 gb	100	4438	88.76	42
R50_9g	50	1092	43.68	25
R50_9gb	50	1092	43.68	24

Table 5 Experimental results of the proposed algorithm on geometric series graphs

Instance	V	E	Avg degree	K (Result)
GEOM20	20	40	4.00	5
GEOM20a	20	57	5.70	5
GEOM20b	20	52	5.20	4
GEOM30	30	80	5.33	6
GEOM30a	30	111	7.40	6
GEOM30b	30	111	7.40	5
GEOM40a	40	186	9.30	7
GEOM40b	40	197	9.85	7
GEOM50a	50	288	11.52	10
GEOM50b	50	299	11.96	10
GEOM60a	60	399	13.30	11
GEOM60b	60	426	14.20	11

Table 6 Comparison of the proposed algorithm with HPGAGCP

Instance	K (Proposed)	K (HPGAGCA)
myciel3	4	4
myciel4	5	5
queen5_5	5	5
myciel5	6	6
queen7_7	7	8
queen8_8	10	10

instance name, number of vertices in graph (V), number of edges in graph (E), average degree of vertices in graph (AvgDegree) and number of colors required to color the graph, which are generated by the proposed algorithm (K).

From the experimental results, it has been found that proposed algorithm gives optimum results for high degree of graphs.

The proposed algorithm implementation is also tested on 12 geometric series graphs, these are weighted with bandwidth. But proposed algorithm implementation is not for weighted graphs so in algorithm weight is ignored from the data records. Also there is no use of bandwidth in the proposed algorithm, so that bandwidth is also ignored. In Table 5 experimental results of geometric series graphs (GEOM) can be seen.

In Table 6 few experimental results are also compared with a well-known hybrid genetic algorithm for graph coloring problem (HPGAGCP) [9]. Some interesting results are found in Table 6.

5 Conclusion

Generally complexity of any graph coloring algorithm is high for high degree graphs. But the proposed algorithm is based on complementary edge table. If the degree of graph is higher then size of complement edge table is small. Exploring tree through this complement edge table does not extend up to high level, i.e., algorithm is executed in optimum time and space complexity.

Tree-based graph coloring algorithm using independent set (proposed algorithm) is an efficient algorithm to calculate the number of colors required and assign to vertices. Time complexity of this algorithm is optimum for high degree graphs. The algorithm gives number of colors precisely equal to the chromatic number of graphs.

References

1. Berchtold, S., Böhm, C., Braunmüller, B., Keim, D. A., and Kriegel, H.-P.: Fast Parallel Similarity Search in Multimedia Databases, In ACM SIGMOD Int. Conf. on Management of Data, (1997)
2. Chaitin, G. J., Auslander, M. A., Chandra, A. K., Cocke, J., Hopkins, M. E., and Markstein, P. W.: Register Allocation via Coloring. Computer Languages, Vol. 6, Issue 1, (1981) 47–57
3. Mahmoudi, S., Lotfi, S.: Modified cuckoo optimization algorithm (MCOA) to solve graph coloring problem. ELSVIER, Applied Soft Computing, (2015) 48–64
4. Gupta A., Patidar H.: A Survey on Heuristic Graph Coloring Algorithm. International Journal for Scientific Research & Development, Vol. 4, Issue 04, (2016) 297–301
5. Salari, E., and Eshghi, K.: An ACO Algorithm for the Graph Coloring Problem. Interracial Journal Contemp. Math Sciences, Vol. 3, no. 6 (2008) 293–304
6. Thang, N. Bui, Nguyen, T. H., Patel, C. M., and Kim-Anh Phan, T.: An Ant-Based Algorithm for Coloring Graphs. Discrete Applied Mathematics 156 (2008) 190–200
7. Chen, B., Chen, Bo., Liu, H., Zhang X.: A Fast Parallel Genetic Algorithm for Graph Coloring Problem Based on CUDA. IEEE International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery, (2015) 145–148
8. Sabar, N. R., Ayob M., Qu, R., Kendall, G.: A Graph Coloring Constructive Hyper-Heuristic for Examination Time Tabling Problems. Online publication, Springer Science Business Media, (2011)
9. Hindi, M., and Yampolskiy, R. V.: Genetic Algorithm Applied to the Graph Coloring Problem, Proc. 23rd Midwest Artificial Intelligence and Cognitive Science Conf, (2012) 61–66