A Novel Exact Heuristic Graph Coloring Algorithm based on Finding Independent Set

Sukrati Agrawal
M.Tech. Scholar CS Dept.
MIST, Indore (M.P.), India

Vishal Chhabra
Asst. Prof. CS Dept.
MIST, Indore (M.P.), India

ABSTRACT
Vertex coloring is a graph coloring technique which has a wide application area to provide solution for many real world problems. The high computational complexity of graph coloring algorithm led the development of exact heuristic algorithm which can be executed in optimal time. This paper explores some existing graph coloring algorithms to propose taxonomy of exact graph coloring algorithm which is capable to execute large graphs also. This paper presented experimental result on DIMACS graph instances.

Keywords
Graph Coloring, independent set, exact algorithm, approximate algorithm, sequential algorithm and parallel algorithms.

1. INTRODUCTION
A graph G (V, E) is a set of vertices V and a set of edges E. The edges are unordered pairs of the form (i, j) where i, j ∈ V. Two vertices i and j are said to be adjacent if and only if (i, j) ∈ E and non-adjacent otherwise.

The graph coloring problem (GCP) - Graph coloring (mainly vertex coloring) of a graph is an assignment of colors to the vertices such that no two adjacent vertices are assigned the same color. Alternatively, a coloring is a partition of the vertex set into a collection of vertex-disjoint independent sets. Each independent set is called a color class. The GCP is then to find a vertex coloring for a graph using the minimum number of possible colors.

There are numerous applications of graph coloring like time tabling and scheduling [1], register allocation [2], frequency assignments [3], crow management, air traffic management [4] etc. The graph coloring begins with coloring the map. In 1852 Francis Guthrie, while trying to color the map of countries of England, noticed that four colors are sufficed. Subsequently, he proposed that 4 colors are enough to color any map. Successive efforts made to prove Guthrie's 4-color inference led to the development of much of graph coloring. Later on this map coloring is also known as face coloring.

2. EXISTING ALGORITHMS
Graph coloring has wide scope of problem solving capabilities. So many researchers and mathematicians tried to discover different algorithms. On the basis of problem solving capabilities graph coloring algorithms are divided into two categories, one is exact and another one is approximate algorithm [5].

2.1 Exact and Approximate Algorithm
Finding optimum solution through exact algorithm is a NP-hard problem. Exact algorithms give very precious results. But there are certain issues related to exact algorithms are observed. The main problem with the exact algorithms is that most of the exact algorithms are able to execute small size graphs which have less than 100 vertices [6] [7].

Approximate algorithms [5] are algorithms used to find approximate solutions for optimization problems. Heuristic parallel algorithms are the approximate algorithms which color the vertices of the graph in parallel to reduce the coloring time and also reduce the number of colors used in coloring of graph. Solution given by the approximate algorithms is not surely precious. But it has been observed that approximate algorithms are able to provide solution for large graphs having more than 100 vertices.

2.2 Sequential and Parallel Algorithms
On the basis of execution behavior algorithms are also divided in to sequential and parallel [8]. Sequential algorithms use to color vertices with minimum colors. These algorithms are much efficient for small size graphs but take time to execute large graphs. There are certain algorithms which uses sequential method to solve the graph coloring such as sequential greedy algorithm [9], first fit [10], largest-degree-first-ordering [11], incidence-degree-ordering [12], and saturation-degree-ordering [13].

Parallel computing is an effective way to find a solution in optimal time complexity for any algorithm. Hence in the same way graph coloring researchers are also using parallel computing to solve GCP. There are some sequential algorithms like largest- degree-first algorithm [11] and the smallest-degree-last algorithm [14] which converted in to parallel algorithms to achieve better performance.

3. PROBLEM IDENTIFICATION
By reviewing of literatures related to the graph coloring, it has been found that most of the researcher's focus is to find better chromatic number to solve vertex coloring problem i.e. finding optimum chromatic number is always a primary objective for researchers. Exact algorithm is more reliable for solving GCP. Because result generated by exact algorithms are more accurate and optimum, but the major problem with the existing exact algorithms that they are not suitable to execute large graphs. So it is a real challenge to find such an algorithm, which can execute large graphs in finite time.

4. PROPOSED SOLUTION
This paper presents an exact graph coloring algorithm to solve GCP, which can find coloring sequence and feasible chromatic number for large graphs also. Proposed algorithm is based on theory of finding independent set, because all the vertices of any independent set can be colored with same color. Proposed algorithm uses iterative approach to find maximal independent sets to assign colors to the vertices.

4.1 Maximal Independent Set

In graph theory, an independent set is a set of vertices in a graph such that no two vertices are adjacent from given set. That is, it is a set V of vertices such that for every two vertices in V, there is no dedicated edge connecting the two vertices. This is also known as stable set. If this set is maximal i.e. contain maximal vertices as compared to other sets then it is known as maximal independent set.

4.2 Proposed Algorithm Flow Chart

This paper presents an efficient way of finding independent set from a given graph. The algorithm takes graph as an input and generates chromatic no. as an output. The algorithm works in iteration followed by multiple sub iterations. In external iteration the vertex with the maximum degree is selected to process first then this vertex further process by the sub iterations. The algorithm can be better understood by the flow chart given in figure 1.

4.2.1 Step1

Initially the data set has been initialized for which algorithm reads the graph data such as number of vertices, number of edges and edges information. This data set obtained from the DIMACS graph instance, which is available in .col format. Then algorithm creates an edge set through read edges information. One empty edge cover vertex set is also created to keep the record of vertex information.

4.2.2 Step 2

In this step algorithm checks the availability of vertex in vertex set. This condition is checked in all iteration whenever process of finding any independent set has been started. If all vertices are removed from vertex set then algorithm will terminate, otherwise enter into next step.

4.2.3 Step 3

In this step first of all copy of vertex set and edge set generated in step 1 is created in order to keep the record of original graph. Edge cover set is also need to clear for this step. After that degree of all vertices available in vertex set is calculated.

4.2.4 Step 4

In this step copy of edge set created in previous state is checked, that edge set is empty or not. Empty edge set shows that internal iterations of main iteration (used to find independent set) is over and control goes to the step 6. But if this edge set contain any edge shows internal iterations are not finished and controls transfer to step 5.

4.2.5 Step 5

In this step algorithm finds the vertex having maximum degree from vertex set (calculated in step 3). After finding maximum degree vertex algorithm add this vertex to edge cover set and delete it from vertex set copy (created in step 3). Algorithm also deletes all the edges connected this vertex from the copy of edge set (created in step 3). Finally maximum degree vertex is removed from vertex degree set. Then execution control go to step 4 and repeat step 4 and step 5 till edge set is not empty.

4.2.6 Step 6

This step assigns the same color to all the vertices present in a single independent set generated by algorithm. In each iteration algorithm take a new color to assign new independent set. After assigning the color to independent set controls goes to step 2 and whole iteration get repeated till all vertices is not removed from the edge set.

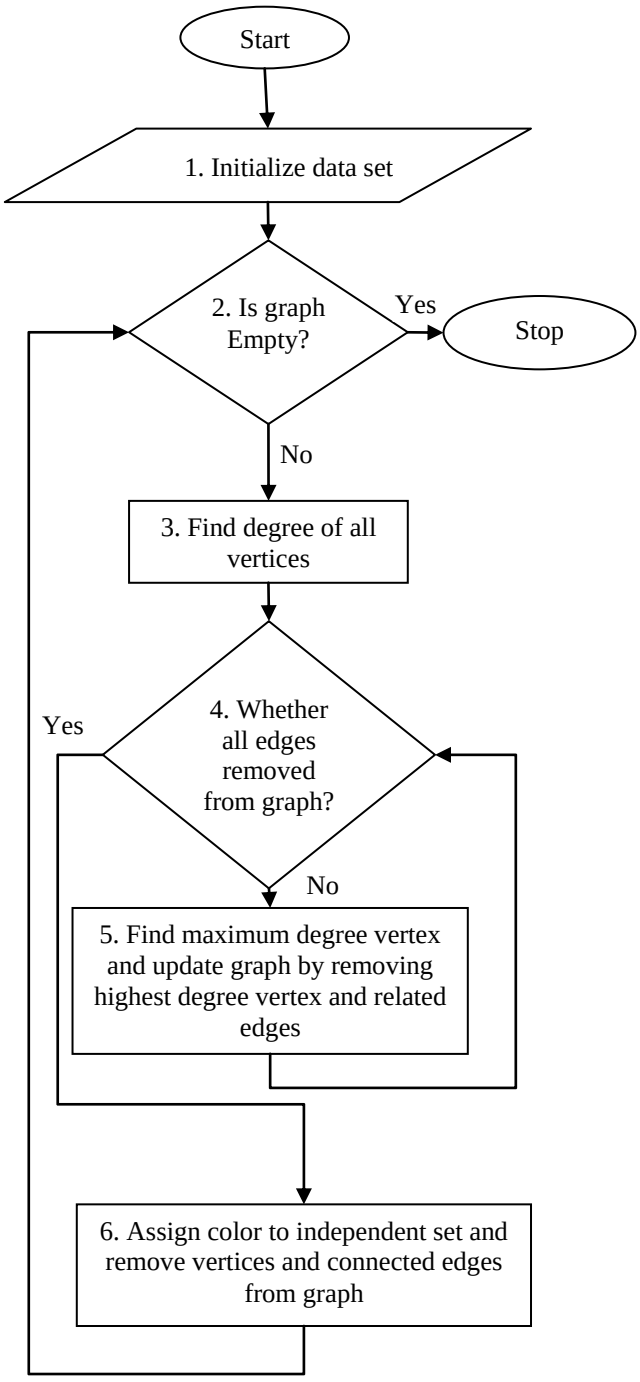

Fig 1: Algorithm Flow Chart

5. EXPERIMENTAL RESULTS AND RESULT ANALYSIS

5.1 Development Environment

Proposed algorithm is developed in Java Programming Language for experiments. Some important concepts like file handling and collection framework is used to implement the algorithm. Due to the platform independence of java algorithm is can be execute on different operating systems like Windows and Linux.

5.2 Data Set

The Center for Discrete Mathematics and Theoretical Computer Science (DIMACS) is collaboration between Rutgers University, Princeton University, and the research firms AT&T, Bell Labs, Applied Communication Sciences, and NEC. It is founded to support research activity and provide different open source data sets for research.

DIMACS also provides the different applications and randomly generated graph data sets for graph coloring. Most of the researchers use these data sources for their research. So to experiment the proposed algorithm graph instances provided by DIMACS is used [15].

5.3 Experimental Results

To execute the proposed algorithm Intel Pentium Dual CPU G640 @ 2.80 GHz processor and 2.00 GB RAM computer system is used. Table 1 shows the experimental results of proposed algorithm. Table 1 contain following information from first to sixth column serial number (Sr. No.), name of graph Instance (Instance), number of vertices in graph (V), number of edges in graph (E), chromatic number generated by proposed algorithm (K) and execution time in seconds (Time (s)).

Table 1: Experimental results of the proposed algorithm

Sr. No.	Instance	V	E	K	Time (s)
1	myciel3	11	20	4	0.031
2	GEOM20	20	40	5	0.031
3	myciel4	23	71	5	0.047
4	queen5_5	25	320	7	0.094
5	1-FullIns_3	30	100	4	0.047
6	GEOM30	30	80	6	0.044
7	GEOM30a	30	111	7	0.046
8	GEOM30b	30	111	6	0.062
9	queen6_6	36	580	10	0.171
10	2-Insertions_3	37	72	4	0.042
11	GEOM40	40	118	6	0.078
12	GEOM40b	40	197	7	0.094
13	myciel5	47	236	6	0.094
14	queen7_7	49	952	12	0.25
15	GEOM50	50	177	6	0.081
16	GEOM50b	50	299	10	0.142
17	R50_1g	50	108	5	0.065
18	R50_1gb	50	108	5	0.075
19	R50_5g	50	612	15	0.218
20	R50_5gb	50	612	15	0.218
21	R50_9g	50	1092	25	0.391
22	R50_9gb	50	1092	25	0.391
23	2-FullIns_3	52	201	5	0.078
24	3-Insertions_3	56	110	4	0.062
25	GEOM60	60	245	7	0.125
26	GEOM60b	60	426	12	0.203
27	queen8_8	64	1456	14	0.329
28	1-Insertions_4	67	232	5	0.094
29	GEOM70	70	337	9	0.174
30	GEOM70a	70	529	12	0.25
31	GEOM70b	70	558	12	0.172
32	R75_1g	70	251	6	0.125
33	R75_1gb	70	251	6	0.125
34	Huck	74	602	11	0.218
35	R75_5g	75	1407	16	0.391
36	R75_5gb	75	1407	16	0.359
37	R75_9g	75	2513	39	0.703
38	R75_9gb	75	2513	39	0.734
39	4-Insertions_3	79	156	4	0.094
40	3-FullIns_3	80	346	6	0.125
41	GEOM80	80	429	8	0.206
42	Jean	80	508	10	0.196
43	queen9_9	81	2112	15	0.375
44	David	87	812	12	0.235
45	mug88_1	88	146	5	0.109
46	mug88_25	88	146	4	0.109
47	GEOM90	90	531	10	0.246
48	GEOM90a	90	879	16	0.36
49	GEOM90b	90	950	18	0.36
50	1-FullIns_4	93	593	5	0.172
51	myciel6	95	755	7	0.219
52	queen8_12	96	2736	15	0.516
53	GEOM100	100	647	10	0.282
54	GEOM100a	100	1092	16	0.407
55	mug100_1	100	166	4	0.125
56	mug100_25	100	166	4	0.125
57	queen10_10	100	2940	17	0.5
58	R100_1g	100	509	8	0.235
59	R100_5g	100	2456	22	0.578
60	R100_9g	100	4438	44	1.08
61	R100_9gb	100	4438	44	1.063
62	GEOM110	110	748	11	0.313
63	4-FullIns_3	114	541	8	0.187
64	games120	120	1276	9	0.344
65	GEOM120	120	893	11	0.375
66	queen11_11	121	3960	18	0.703
67	DSJC125.1	125	736	8	0.284
68	DSJC125.5	125	3891	25	0.797
69	DSJC125.9	125	6961	56	1.739

70	miles1000	128	6432	51	1.406
71	miles1500	128	10396	81	2.588
72	miles250	128	774	10	0.297
73	miles500	128	2340	26	0.531
74	miles750	128	4226	39	0.953
75	Anna	138	986	12	0.281
76	queen12_12	144	5192	19	0.859
77	2-Insertions_4	149	541	5	0.203
78	5-FullIns_3	154	792	8	0.25
79	queen13_13	169	6656	20	1.046
80	mulsol.i.3	184	3916	31	0.719
81	mulsol.i.4	185	3946	31	0.715
82	mulsol.i.5	186	3973	31	0.672
83	mulsol.i.2	188	3885	31	0.672
84	myciel7	191	2360	10	0.422
85	queen14_14	196	8372	21	1.375
86	mulsol.i.1	197	3925	49	1.047
87	1-Insertions_5	202	1227	6	0.312
88	zeroin.i.3	206	3540	32	0.657
89	zeroin.i.1	211	4100	51	1.04
90	zeroin.i.2	211	3541	32	0.641
91	2-FullIns_4	212	1621	6	0.344
92	queen15_15	225	10360	25	1.86
93	DSJC250.1	250	3218	12	0.735
94	DSJC250.5	250	15668	42	3.578
95	DSJC250.9	250	27897	94	13.932
96	r250.5	250	14849	101	7.327
97	queen16_16	256	12640	27	2.221
98	3-Insertions_4	281	1046	5	0.312
99	1-FullIns_5	282	3247	6	0.469
100	flat300_28_0	300	21695	45	5.954

6. CONCLUSION

This paper presents algorithm is to solve vertex coloring problems which can be executed on different types of application data i.e. algorithm supports various application problems. The algorithm is successfully executed up to 300 vertices and gives results in finite time. Execution success rate is also high for proposed algorithm. In some cases it is found that proposed algorithm is not giving optimum chromatic number, so in future algorithm can be update to find optimum chromatic number for all types of graphs.

7. REFERENCES

[1] Hussin B., Basari A. S. H., Shibghatullah A. S. and Asmai S. A., Exam Timetabling Using Graph Colouring Approach, In Proc. IEEE Conference on Open Systems, Langkawi, 25-28 September (2011), p.139-144.

[2] Chaitin G. J., Register Allocation & Spilling via Graph Coloring, In Proc. SIGPLAN '82 Proceedings of the 1982 SIGPLAN symposium on Compiler construction, June (1982), p.98-105.

[3] S. Ahmed, "Applications of Graph Coloring in Modern Computer Science", International Journal of Computer and Information Technology, 2012, Vol. 3, Issue 2, pp. 1-7.

[4] Barnier N. and Brisset P., "Graph Coloring for Air Traffic Flow Management", Annals of Operations Research, 130 (1-4), 163–178, August (2004).

[5] A. Gupta and H. Patidar, "A Survey on Heuristic Graph Coloring Algorithm", International Journal for Scientific Research & Development Vol. 4, Issue 04, 2016, pp. 297-301.

[6] Mahmoudia S., Lotfi S., "Modified Cuckoo Optimization Algorithm (MCOA) to Solve Graph Coloring Problem", Applied Soft Computing, 33, 48-64 (2015).

[7] Torkestani J. A., Meybodi M.R., "A cellular learning automata-based algorithm for solving the vertex coloring problem", Expert Systems with Applications, 38, 9237–9247, (2011).

[8] Patidar H., Chakrabarti P., " Sequential and Parallel Approaches in Context to Graph Coloring Problems - A Survey", International Journal of Computer Systems, Volume 03– Issue 05, May, 201, pp. 403-406.

[9] Allwright JR, Bordawekar R, Codington PD, Dincer K, Martin CL,"A comparison of parallel graph coloring algorithms Technical" Report SCCS-666, Northeast Parallel Architecture Center, Syracuse University, 1995.

[10] Dr. Hussein Al-Omari and KhairEddinSabri: "New Graph Coloring Algorithms", American Journal of Mathematics and Statistics 2 (4): 739-741, 2006ISSN 1549-3636, March 2006.

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

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

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

[14] D. W. Matula and L. L. Beck. "Smallest-last ordering and clustering and graph coloring algorithms", JACM, 1983.

[15] DIMACS Graph Instances, available at "http://mat.gsia.cmu.edu/COLOR/instances.html"