Perspectives in Science (2016) **8**, 384—387

Available online at www.sciencedirect.com

ScienceDirect

journal homepage: www.elsevier.com/pisc

New approximation algorithms for solving graph coloring problem — An experimental approach☆

Raja Marappan [a,*], **Gopalakrishnan Sethumadhavan** [a,*], **R.K. Srihari** [b]

[a] *Department of Computer Applications, School of Computing, SASTRA University, Thanjavur, Tamilnadu, India*
[b] *TATA Consultancy Services, Chennai, Tamilnadu, India*

Received 3 February 2016; accepted 11 April 2016
Available online 30 April 2016

KEYWORDS
Graph coloring;
Chromatic number;
Divide & conquer;
Approximation

Summary Some of the real world applications require the solution to graph coloring problem, an NP-hard combinatorial optimization problem. $\chi(G)$, the chromatic number of a graph G, the minimum number of colors required to color the vertex set $V(G)$ with adjacent vertices assigned with different color can also be obtained using evolutionary methods. This paper exhibits two new approximation methods of solving graph coloring based on $\mu_{1/2}(G)$, the median of the degrees $V(G)$. In the first method, a heuristic procedure is designed to color $V(G)$ which works in two stages. In the first stage, to minimize the conflicting edges, the vertices of $V(G)$ whose degrees are $\geq\mu_{1/2}(G)$ are colored. Then the remaining vertices are colored through a heuristic procedure. The second method is implemented using divide & conquer strategy. These new approximation algorithms are exhibited on some of the small, intermediate and large benchmark graphs and the results are compared. The proposed algorithms significantly reduce the computational complexity in obtaining the near optimal solution and also the results are found to be better than the existing approaches.

Introduction

The vertex set $V(G)$ and edge set $E(G)$ of a simple graph G with n vertices are $V(G)$: $\{v_1, v_2, \ldots, v_n\}$ and $E(G)$:$\{e_1, e_2, \ldots, e_m\}$ respectively such that each $e_k \in E(G)$ is uniquely mapped to disordered end vertices pair (v_j, v_i). The adjacency matrix of G is $A(G)$, an $n \times n$ symmetric binary matrix,

☆ This article belongs to the special issue on Engineering and Material Sciences.

* Corresponding authors.
E-mail addresses: raja_csmath@cse.sastra.edu (R. Marappan), sgk@mca.sastra.edu (G. Sethumadhavan).

http://dx.doi.org/10.1016/j.pisc.2016.04.083

Table 1 Computation of near optimal solution on small graphs ($n < 100$).

Graph no	Graph (G)			Minimum color obtained in		
	Graph type	Instances	$\chi(G)$	Existing method	Method 1	Method 2
1	queen5_5.col	$n = 25$; $m = 320$	5	5	5	5
2	queen6_6.col	$n = 36$; $m = 580$	7	7	7	7
3	queen7_7.col	$n = 49$; $m = 952$	7	7	7	7
4	queen8_8.col	$n = 64$; $m = 1456$	9	9	9	9
5	myciel5.col	$n = 47$; $m = 236$	6	6	6	6
6	myciel6.col	$n = 95$; $m = 755$	7	7	7	7
7	myciel4.col	$n = 23$; $m = 71$	5	5	5	5
8	myciel3.col	$n = 11$; $m = 20$	4	4	4	4
9	huck.col	$n = 74$; $m = 301$	11	11	11	11
10	jean.col	$n = 80$; $m = 254$	10	10	10	10
11	david.col	$n = 87$; $m = 406$	11	11	11	11
12	queen8_12.col	$n = 96$; $m = 2736$	12	12	15	12
13	queen9_9.col	$n = 81$; $m = 2112$	10	10	11	10

where $A_{ji} = 1$ if v_j and v_i are adjacent such that $(v_j, v_i) \in E(G)$; $A_{ji} = 0$ when (v_j, v_i) is not in $E(G)$. $\chi(G)$, minimum number of colors used to color $V(G)$ with adjacent vertices are colored with different color, is called a graph coloring problem (GCP) (Méndez-Diaz and Zabala, 2006). Different methods were proposed to solve GCP (Hertz and Werra, 1987; Galinier and Hao, 1999; Marappan and Sethumadhavan, 2013; Sethumadhavan and Marappan, 2013; Marappan and Sethumadhavan, 2015a,b,c). This paper exhibits two new approximation methods to find $\chi(G)$. These approximation schemes are devised by combining the greedy and divide & conquer algorithmic design approaches. Greedy design strategy is applied to minimize the number of conflicting vertices. The first method (method 1) initially computes the median degree, $\mu_{1/2}(G)$ and colors all the vertices with degree $\geq \mu_{1/2}(G)$ and then colors the remaining vertices. The divide & conquer design strategy is applied to split $V(G)$ into two smaller subsets viz. left and right. Initially the coloring procedure is called to the left subset while assigning colors to right the adjacency between the vertices present in the left to right is considered to avoid the conflicts (method 2). To minimize the number of conflicting edges and to minimize the computational time the subset which has more number of conflicting edges with the other subset is solved first independently. The conflicting edges of the other subset are solved concurrently based on the color assignment of other subset. Section 2 focuses on the new approximation methods; Experimental outcomes and evaluations with well known methods are presented in Section 3 and the conclusions are drawn in Section 4.

New approximation methods

Notions, variables, data structures and procedures

The following notions, variables, data structures and procedures are used in the proposed methods:

1. Median degree $\mu_{1/2}(G)$: The degree $d(v_i)$ of v_i is the number of e_j's incident onto v_i.

$$\mu_{1/2}(G) = \text{Median}\{d(V_i) | V_i \in V(G), \quad 1 \leq i \leq n\}$$

2. mincolor: A variable to store a possible minimum integer value.

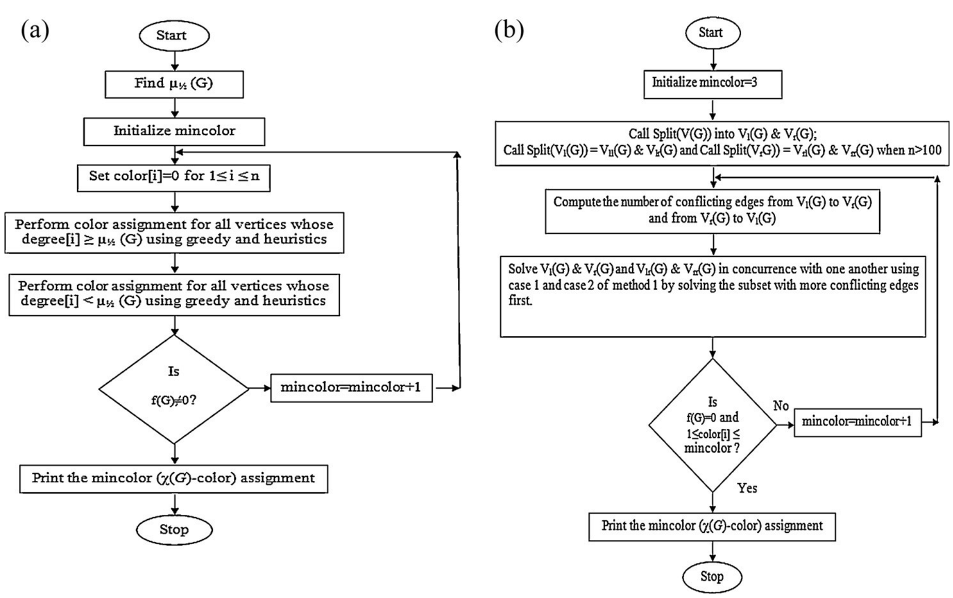

Fig. 1 (a) Flowchart for method 1 (b) Flowchart for method 2. (C) Two level divide & Conquer on Method 1 (Method 2).

Table 2 Computation of near optimal solution on intermediate graphs ($100 \leq n < 500$).

Graph no	Graph (G)			Minimum color obtained in		
	Graph type	Instances	$\chi(G)$	Existing method	Method 1	Method 2
14	myciel7.col	$n=191$; $m=2360$	8	8	8	8
15	games120.col	$n=120$; $m=638$	9	9	9	9
16	miles250.col	$n=128$; $m=387$	8	8	8	8
17	anna.col	$n=138$; $m=493$	11	11	11	11
18	queen10_10.col	$n=100$; $m=2940$	11	11	14	11
19	queen12_12.col	$n=144$; $m=5192$	12	12	18	13
20	queen14_14.col	$n=196$; $m=8372$	14	14	20	16
21	queen15_15.col	$n=225$; $m=10360$	15	15	20	16
22	queen16_16.col	$n=256$; $m=12640$	16	16	22	17
23	queen11_11.col	$n=121$; $m=3960$	11	11	18	11
24	queen13_13.col	$n=169$; $m=6656$	13	13	19	14
25	miles500.col	$n=128$; $m=1170$	20	20	22	20
26	miles750.col	$n=128$; $m=4226$	31	31	32	31
27	miles1000.col	$n=128$; $m=6432$	42	42	48	43
28	miles1500.col	$n=128$; $m=10396$	73	73	78	76
29	zeroin.i.1.col	$n=211$; $m=4100$	49	49	51	49
30	zeroin.i.2.col	$n=211$; $m=3541$	30	30	30	30
31	zeroin.i.3.col	$n=206$; $m=3540$	30	30	30	30
32	mulsol.i.1.col	$n=197$; $m=3925$	49	49	50	49
33	mulsol.i.2.col	$n=188$; $m=3885$	31	31	32	31
34	mulsol.i.3.col	$n=184$; $m=3916$	31	31	33	31
35	mulsol.i.4.col	$n=185$; $m=3946$	31	31	32	31
36	mulsol.i.5.col	$n=186$; $m=3973$	31	31	32	31
37	le450_5a.col	$n=450$; $m=5714$	5	11	12	9
38	le450_5b.col	$n=450$; $m=5734$	5	11	12	9
39	le450_5c.col	$n=450$; $m=9803$	5	5	13	10
40	le450_5d.col	$n=450$; $m=9757$	5	5	13	11
41	le450_15b.col	$n=450$; $m=8169$	15	15	24	20
42	le450_15c.col	$n=450$; $m=16680$	15	15	25	22
43	le450_15d.col	$n=450$; $m=16750$	15	15	25	21
44	le450_25a.col	$n=450$; $m=8260$	25	25	38	30
45	le450_25b.col	$n=450$; $m=8263$	25	25	38	29
46	le450_25c.col	$n=450$; $m=17343$	25	34	40	32
47	le450_25d.col	$n=450$; $m=17425$	25	34	40	31
48	school1.col	$n=385$; $m=19095$	14	14	31	25
49	school1_nsh.col	$n=352$; $m=14612$	14	14	30	23
50	fpsol2.i.1.col	$n=496$; $m=11654$	65	65	70	65
51	fpsol2.i.2.col	$n=451$; $m=8691$	30	30	35	30
52	fpsol2.i.3.col	$n=425$; $m=8688$	30	30	34	31

Table 3 Computation of near optimal solution on large graphs ($n \geq 500$).

Graph no	Graph (G)			Minimum color obtained in		
	Graph type	Instances	$\chi(G)$	Existing method	Method 1	Method 2
53	homer.col	$n=561$; $m=1629$	13	13	13	13
54	inithx.i.1.col	$n=864$; $m=18707$	54	54	54	54
55	inithx.i.2.col	$n=645$; $m=13979$	31	31	31	31
56	inithx.i.3.col	$n=621$; $m=13969$	31	31	31	31

3. color[]: color[i] will hold the color value of the ith vertex.
4. SetGreedyColor(i): A procedure which sets a color to vertex i.
5. Conflicting vertices: Two adjacent vertices i and j are said to be conflicting iff color[i] = color[j].
6. Conflict function $f(G)$: This function assigns the total number of conflicting vertices in $V(G)$.
7. Split (S): Set $S=\{1, 2, 3, \ldots, n\}$ is split into two subsets S_l and S_r as follows:

$$S_l = \left\{1, 2, 3, \ldots, \frac{n}{2}\right\} \; \& \; S_r = \left\{\frac{n}{2}+1, \frac{n}{2}+2, \ldots, n\right\}$$

$$\text{with } \left|S_l\right| = \frac{n}{2} \; \& \; \left|S_r\right| = \frac{n}{2} \quad \text{when } n \text{ is odd.}$$

$$S_l = \left\{1, 2, 3, \ldots, n/2\right\} \; \& \; S_r = \left\{\frac{n}{2}+1, \frac{n}{2}+2, \ldots, n\right\}$$

$$\text{with } \left|S_l\right| = \left|S_r\right| = \frac{n}{2} \text{ when } n \text{ is even.}$$

Greedy and heuristics (Method 1)

This iterative method is evolved from the greedy design strategy with heuristics; by initializing mincolor as 3 and color[i] = 0 for $1 \leq i \leq n$.

Case 1: SetGreedyColor(i), a heuristic procedure is called on all the vertices whose degree $\geq \mu_{1/2}$ (G), a greedy approach as follows:i. Choose a vertex i such that color[i] = 0 and degree (i) $\geq \mu_{1/2}$ (G).ii. For $j=1$ to mincolor do

Boolean t = false;
For k = 1 to n do
If ($A_{ik} = 1$) then
If (color[k] == j) then
t = false; break;
Else t = true;
If (t = true) then color[i] = j; break;ii.
Repeat Steps i and ii for other vertices whose degree $\geq \mu_{1/2}(G)$.

Case 2: For all the vertices whose degree $< \mu_{1/2}(G)$, call SetGreedyColor (i). Compute $f(G)$. If $f(G) \neq 0$ set mincolor = mincolor + 1; repeat case 1 and case 2 until $f(G) = 0$. This greedy procedure assigns color[i] in [1,mincolor] $\forall\, 1 \leq i \leq n$. The flowchart is shown in Fig. 1(a).

This iterative method is evolved by applying the divide & conquer strategy in method 1; by initializing mincolor as 3 and color[i] = 0 for $1 \leq i \leq n$. This procedure is as follows:

i. Call Split($V(G)$) which partitions $V(G)$ into two subsets $V_l(G)$, $V_r(G)$.
ii. Call Split($V_l(G)$) and Split($V_r(G)$) when $n > 100$ that partitions $V_l(G)$ and $V_r(G)$ into pairs of two subsets $V_{ll}(G)$ & $V_{lr}(G)$ and $V_{rl}(G)$ & $V_{rr}(G)$, respectively.
iii. Compute the number of conflicting edges from $V_l(G)$ to $V_r(G)$ and from $V_r(G)$ to $V_l(G)$.
iv. Solve ($V_l(G)$, $V_r(G)$) and ($V_{lr}(G)$, $V_{rr}(G)$) in concurrence with one another using case 1 and case 2 of method 1 by solving the subset with more conflicting edges first.

Now compute $f(G)$. If $f(G) \neq 0$ or color[i] > mincolor ($1 \leq i \leq n$) set mincolor = mincolor + 1; repeat steps iii and iv until $f(G) = 0$. The flowchart for method 2 is shown in Fig. 1(b).

Experimental results and comparisons

The experiments are conducted on benchmark graphs using Intel Xeon Workstation with 256 GB DDR3 in Windows 8 Professional OS under JDK 1.8.0 environment. The devised methods are compared with some of the existing methods (Méndez-Diaz and Zabala, 2006; Hertz and Werra, 1987; Galinier and Hao, 1999) and the minimum color obtained is tabulated in Tables 1—3 respectively; and the following inferences are obtained.

1. The approximation methods find solution for most of the benchmark graphs in reduced computational complexities.
2. Method 2 significantly reduces the complexity and obtained the near optimal solution for most graphs compared to method 1.

Conclusion

New approximation methods to find $\chi(G)$ using greedy and divide & conquer strategies are presented in this paper. The methods are based on the median degree $\mu_{1/2}(G)$. The devised methods are evaluated using benchmark graphs and near optimal solution are obtained; and optimal solution is obtained for most of the graphs. The divide & conquer strategy is performed well for most graphs compared to method 1.

Acknowledgement

The authors would like to acknowledge the support rendered by the management of SASTRA University by the way of providing necessary infrastructure and financial support for this research. The authors would like to thank Gary Lewandowski, and Michael Trick for their support rendered in http://mat.gsia.cmu.edu/COLOR/instances.html to accessing the graph coloring instances available in DIMACS which are the main breeding source of this research.

References

Galinier, P., Hao, J.-K., 1999. Hybrid evolutionary algorithms for graph coloring. J. Comb. Optim. 3 3, 379—397.

Hertz, A., Werra, D.E., 1987. Using tabu search techniques for graph coloring. Computing 39 (4), 345—351.

Marappan, R., Sethumadhavan, G., 2013. A new genetic algorithm for graph coloring. In: CIMSim2013, 5th International Conference on Computational Intelligence, Modelling and Simulation, Seoul, South Korea, pp. 49—54.

Marappan, R., Sethumadhavan, G., 2015a. Solution to graph coloring problem using heuristics and recursive backtracking. Int. J. Appl. Eng. Res. 10 (10), 25939—25944, ISSN 0973-4562.

Marappan, R., Sethumadhavan, G., 2015b. Solution to graph coloring problem using evolutionary optimization through symmetry — breaking approach. Int. J. Appl. Eng. Res. 10 (10), 26573—26580, ISSN 0973-4562.

Marappan, R., Sethumadhavan, G., 2015c. Solving graph coloring problem for large graphs. Glob. J. Pure Appl. Math. 11 (4), 2487—2494.

Méndez-Diaz, I., Zabala, P., 2006. A branch-and-cut algorithm for graph coloring. Discrete Appl. Math. 154 (5), 826—847.

Sethumadhavan, G., Marappan, R., 2013. A Genetic algorithm for graph coloring using single parent conflict gene crossover and mutation with conflict gene removal procedure. In: 2013 IEEE International Conference on Computational Intelligence and Computing Research, 26—28 December, India, pp. 350—355.