GRAPH COLORING: AN EXACT ALGORITHM BASED ON PRELIMINARY VERTEX ORDERING

A. Zakrevskij and I. Vasilkova

Institute of Engineering Cybernetics of National Academy of Sciences of Belarus, Surganov str., 6, 220012, Minsk, BELARUS, zakr@newman.bas-net.by

ABSTRACT

The well-known graph coloring problem is regarded in this paper. A new algorithm is suggested and investigated, based on fixing vertex ordering, which allows to minimize the run-time of inner cycles. A series of computer experiments on a flow of pseudo-random graphs having up to thousand vertices was conducted to find dependencies of the run-time and the chromatic number on the number of vertices. The area of preferable application of the algorithm is estimated.

1. INTRODUCTION

The problem of graph coloring is a good example of a hard combinatorial problem that seems will never be solved completely [1, 2]. It has a lot of useful applications in different theoretical and engineering areas and that is why practically efficient algorithms for its solution are needed and developed - chiefly approximate [3, 4], because exact ones could be too expensive. Nevertheless, the latter are sometimes quite necessary, if only because for estimating the quality of solutions produced by heuristic algorithms.

That argument stimulated constructing an exact algorithm described in this paper. It is based on the well-known tree searching technique used already before for the same problem [5, 6]. Some new rules are added to it in order to speed up the search.

First, the vertices of any regarded graph are ordered preliminary, to avoid spending much time to look for the next vertex each time. In its turn, that decision implies the task of finding some optimal order (or rather sub-optimal, to be realistic) on the set of vertices.

Second, a convenient data representation by Boolean vectors and matrices is used. That enables to execute the most of fulfilled work by efficient component-wise operations over Boolean vectors.

Third, a special attention is paid to minimize the time spent in inner cycles. Some means of non-abundant logical inference are used for that.

The suggested algorithm has been programmed and subjected to computer experiments the results of which are given below.

2. GRAPH REPRESENTATION AND VERTEX ORDERING

Let us regard a graph G presented on fig. 1.

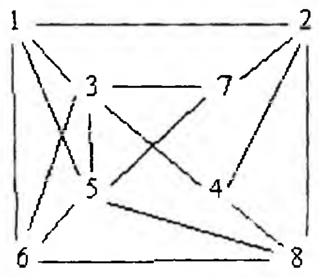


Figure 1. Graph G

Its adjacency Boolean square matrix G looks as follows:

$$\begin{array}{c|cccccccc}
 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\
2 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\
3 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\
4 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\
5 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\
6 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\
7 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\
8 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0
\end{array}$$

The following procedure is applied for ordering the vertices (similar to that used in the heuristic algorithm for graph coloring suggested in [7]). One by one, it finds in a regarded graph G vertices of minimum degree (corresponding to rows in matrix G with minimum number of 1s) and eliminates them together with incident edges. If several vertices have the minimum degree, the first of them is chosen. In such a way some linear order is constructed on the set of vertices. After that it is reversed.

In our example, vertices with numbers 4, 2, 7, 8, 1, 3, 5 and 6 are selected consecutively, so the reverse order (6, 5, 3, 1, 8, 7, 2, 4) is accepted. The vertices are renumbered for convenience (set (6, 5, 3, 1, 8, 7, 2, 4) is mapped onto (1, 2, 3, 4, 5, 6, 7, 8)) and the adjacency matrix G is correspondingly transformed:

$$\begin{array}{c|cccccccc}
 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
\hline
1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\
2 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\
3 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\
4 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\
5 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\
6 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\
7 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \\
8 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0
\end{array}$$

3. Tree searching technique

Consider a growing tree which levels correspond to the vertices of graph G, according to the given order: the first level (the nearest to the root and shown on the left in figures presented below) corresponds to the first vertex, the second to the second, etc. The process of looking for some optimal coloring of graph G may be regarded as moving along the tree and increasing it by adding new branches. The current vertices are colored and re-colored by that, according to the following rules.

Suppose we can use for coloring q different paints $c_1, c_2, \ldots, c_q$ and can vary q during the coloring process, for example beginning with $q=n$ (the number of vertices). Let us present the current result of coloring by Boolean matrix C : $c_{kj}=1$ iff vertex v_j was given paint c_k.

There are two different operations: coloring and re-coloring.

Going forward (to the right) we regard some current vertex v_i and color it selecting the minor possible paint from $c_1, c_2, \ldots, c_t$ already used for coloring left vertices, adding to them new paint c_{t+1} if $t<q$. Note that paint p_k is possible if the following condition is satisfied

$$g_i \wedge c_k \neq 0,$$

which is not difficult to check when using the suggested graph representation.

Having done that when $i=n$ (coloring the last vertex) we have a solution, the first one or better than any found before. We save it, diminish q by one in order to look later only for better solutions, and go one step back, to vertex v_{i-1}. We go back also when there is no possible paint for coloring the regarded vertex.

Going back (to the left) we try to re-color the current vertex with paints not used yet. If there are no possible alternative, we continue to retreat to the left until find a new possible paint for some vertex. In that case we re-color the vertex and switch to the going forward mode. If we return to the root of the tree, the algorithm terminates, and the last saved solution turns out to be optimal.

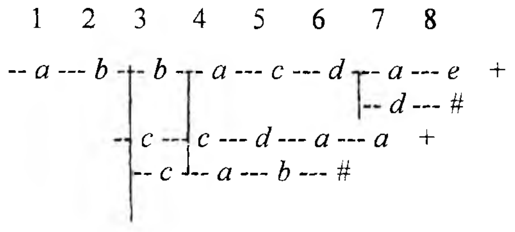
Let us demonstrate this algorithm on the given example, first with not ordered yet set of vertices. Matrix G could be simplified for convenience - by deleting all 1s in the top-right half.

$$\begin{array}{c|cccccccc}
 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
\hline
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
3 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
4 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
5 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
6 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\
7 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\
8 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0
\end{array}$$

Suppose we have an ordered set of paints $a, b, c, d, e, f, \ldots$. According to the formulated above rules, we find first coloring (a, b, b, a, c, d, a, e) for vertices (1, 2, 3, 4, 5, 6, 7, 8) and memorize it. As a result, matrix C takes the following value:

$$\begin{array}{c|cccccccc}
 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
\hline
a & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\
b & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
c & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
d & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
e & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{array}$$

The found solution has five paints, so we are looking for some other solution with not more than four paints - paint e is forbidden from now on. The search for that is illustrated by fig. 2.



1 2 3 4 5 6 7 8
-- a --- b -- b --- a --- c --- d --- a --- e +
-- d --- #
c --- c --- d --- a --- a +
-- c -- a --- b --- #

Figure 2. Search for a best coloring of graph G

Returning to the nearest vertex for which some other paint is possible - number 7 in our example, we use paint d for it. But in that case the next vertex 8 cannot be colored by any of the allowed four paints. Hence we have to go back, this time to vertex 4. Comparing the corresponding row

0 1 1 0 0 0 0 0

of reduced matrix G with rows of C, taken one by one beginning with c_a , we find that the first row which does not intersect with it is c_c . Hence we re-color vertex 4 using paint c.

The new branch leads to the four-paint solution (a, b, b, c, c, d, a, a), better than the previous one. So we memorize it and continue the search, looking now for solutions having three paints or less. Returning to vertex 3 we initiate the fourth branch of the search tree, but reaching vertex 6 see that finding better solution is impossible.

That search process is much simplified by preliminary ordering the vertices. In that case they get new numbers, and the simplified adjacency matrix looks as follows:

$$\begin{array}{c|cccccccc} & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 3 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 4 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 5 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 6 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 7 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 8 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \end{array}$$

The search tree becomes this time very simple (fig. 3), because the first branch presents an optimal four-paint solution.

```
1     2     3     4     5     6     7     8
-- a --- b --- c --- d ---c ---a --- b --- a --  +
```

Figure 3. Search for a best coloring of graph G after preliminary vertices ordering

4. Experiments

A series of experiments were conducted on a flow of pseudo-random graphs with controlled parameters n (the number of vertices) and p (the density of edges, i. e. the probability of existing an edge connecting two arbitrary vertices). Two dependencies on n and p were investigated: the chromatic number γ and the run-time t. The obtained results are shown below.

Fig. 4 presents the dependence of chromatic number γ of random graphs (for three values of p: 0.25, 0.50 and 0.75) on the number of vertices n. It looks rather steady, which enables to predict the value of γ for arbitrary graphs. At the same time, it is interesting to note that in the region covered by experiments this dependence significantly differs from the asymptotic evaluation $\gamma(n)=n/2\log_2 n$ adduced in [8] for p=0.50: γ turns out to be at least two times larger.

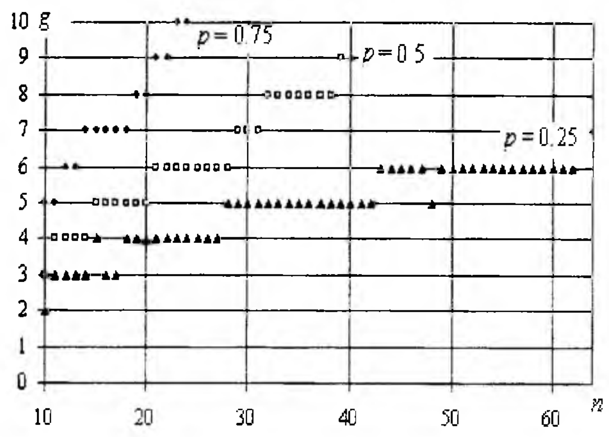


Figure 4. The dependence of the chromatic number (g) of pseudo-random graphs on the number of vertices n

As to the dependence of run-time t on n, it is far more complicated, being not so steady and is characterized by a big dispersion, as could be seen from tabl. 1.

Table 1. The dependence of the run-time t on n and p

n	t			n	t
	p=0.25	p=0.50	p=0.75		p=0.25
17	0	0	0	40	0
18	0	0	0	41	3
19	0	0	2	42	1
20	0	0	0	43	55
21	0	0	22	44	38
22	0	0	23	45	17
23	0	0	2970	46	67
24	0	1	2885	47	22
25	0	1		48	5
26	0	0		49	214
27	1	0		50	38
28	0	0		51	43
29	0	5		52	291
30	0	10		53	757
31	0	1		54	85
32	0	29		55	69
33	1	28		56	54
34	0	23		57	28
35	1	24		58	6
36	0	231		59	7
37	0	234		60	24
38	0	21		61	35
39	1	3458		62	1315

Time t is varying by rather an unpredictable way, jumping between ones, tens, hundreds and thousands. It depends greatly on the number of levels of the tree where the search takes place. As a rule, it grows with increasing of the chromatic number, but not obligatory.

Additional experiments were conducted on a series of pseudo-random graphs which chromatic number was specially decreased and made equal to parameter k, less than chromatic number γ typical for given n and p.

Such a graph could be obtained from the same pseudo-random graph with parameters n and p as before, splitting by random the set of its vertices into k non-empty parts and deleting inner edges in them. It should be noted, that in that case the value of p is decreased by $1/k$.

The next tabl. 2 presents the results of continuation of the previous experiment under p=0.25 for k=3, 4 and 5 instead of 6, beginning with n=62. It could be seen from them that the run-time is much less compared to the previous experiments characterized by the same n and p. The entry *** means that the tun-time exceeds 30 minutes. By reaching that the experiment was interrupted.

Table 2. The dependence of the run-time t on n under k=3, 4, 5 and p=0.25

n	t	n	t	
	k=3		k=4	k=5
62	0	62	1	548
300	1	63	1	35
700	2	64	6	48
800	4	65	18	3
900	5	66	20	4
1000	6	67	92	29
		68	74	33
		69	159	32
		70	19	9
		71	20	18
		72	19	28
		73	26	220
		74	45	108
		75	47	203
		76	6	545
		77	10	559
		78	25	122
		79	94	308
		80	552	2
		81	21	1450
		82	38	***
		83	38	
		84	200	
		85	532	
		86	735	
		87	730	
		88	52	
		89	1970	
		90	347	
		91	269	
		92	***	

Reaching *** does not mean that exact [illegible]ons could not be found for bigger values of n. That becomes clear after analyzing the results of similar experiments conducted under p=0.50 and p=0.75 and presented in tabl. 3, 4.

Each of these experiments can be regarded as a recognition process - finding the unique good solution defined by k and hidden in matrix G. The probability of existing a better solution (with $\gamma<k$) is rather low. Nevertheless, it happens sometimes - such events are marked in the tables with bold entries in t-columns.

Table 3. The dependence of the run-time t on n under k=4, 5, 6, 7, 8 and p=0.50

n	t		n	T	n	t	
	k=4	k=5		k= 6		k= 7	k=8
39	0	0	40	1	39	25	5
400	1	1	50	3	40	26	5
500	2	2	60	0	41	38	5
600	3	3	70	2	42	12	7
700	5	6	80	1	43	3	1
800	6	7	90	41	44	4	1927
900	9	1	100	0	45	5	541
1000	12	14	110	3	46	20	949
			111	1	47	14	101
			112	0	48	8	9
			113	***	49	66	6
					50	65	7
					51	144	7
					52	140	58
					53	99	1912
					54	140	2054
					55	46	***
					56	157	
					57	766	
					58	570	
					59	572	
					60	267	
					61	144	
					62	311	
					63	401	
					64	222	
					65	56	
					66	517	
					67	***	

These results show that when there exists a good exact solution with low γ, it could be discovered by the suggested algorithm even for big n.

The most interesting in that sense is the case with p=0.75 and k=9. With increasing of n the run-time is changed at first chaotically, but then becomes smaller and smaller going down to only seconds when n is measured by hundreds.

That could be explained as follows. When n increases, there rises also the mathematical expectation of the number of such first vertices in the ordered sequence which constitute a complete sub-graph of graph G. That leads to a corresponding reduction of the number of tree levels involved in the search process, which, in its turn, reduces the run-time.

5. Conclusion

An exact algorithm for graph coloring is suggested, based on preliminary ordering of vertices. The computer experiments show that it could be applied in case of existence of high-quality solutions when its efficiency grows with increasing of the number of vertices.

Table 4. The dependence of the run-time t on n under k=5, 6, 7, 8, 9 and p=0.75

n	t				n	t
	k=5	k=6	k=7	k=8		k=9
24	0	0	0	0	24	0
400	1	2	2	2	29	9
500	2	3	3	3	30	10
600	5	5	5	8	31	4
700	7	7	7	8	32	203
800	11	10	11	12	33	382
900	14	15	16	19	34	380
1000	20	20	21	22	35	7
					36	80
					37	80
					38	12
					39	13
					40	13
					41	5
					42	6
					43	62
					44	60
					45	168
					46	64
					47	64
					48	71
					49	323
					50	467
					51	121
					52	455
					53	52
					54	72
					55	19
					56	32
					57	39
					58	73
					59	6
					60	52
					100	5
					200	6
					300	6
					400	7
					500	8
					600	10
					700	14
					800	18
					900	40
					1000	28

References

[1] Cho A., Bernson D. Linear and non linear circuits. 7th *Int.Workshop on Post-Binary Ultra-Large-Scale Integration Systems.* Fukuoka, Japan, 1998, p.137-145.

[2] Harary F. Graph theory. - Addison -Wesley Publishing Company, 1969.

[3] Zykov A. A. Theory of finite graphs. - Novosibirsk: Nauka, 1969 (in Russian).

[4] Wan W., Perkowski M. A. A new approach to the decomposition of incompletely specified multi-output functions based on graph coloring and local transformations and its application to FPGA mapping. - *European Design Automation Conference (Euro DAC)* Sept. 1992, pp. 230-235.

[5] Shneider A. A. Classification and analysis of heuristic algorithms for graph vertices coloring. - Kibernetika, 1984, **4**, pp. 15-22 (in Russian).

[6] Zakrevskij A. D. Algorithms for discrete automata synthesis. - Moscow: Nauka, 1971 (in Russian).

[7] Bykova S. V., Ivolga V. P. Efficiency evaluation for some algorithms of minimum breaking up. - *Vychislitelnaya technika v mashinostroenii*, **6** 1974. - Minsk, Institute of Engineering Cybernetics, pp. 79-86 (in Russian).

[8] Matula D. W., Marble G., and Isaacson J. D. Graph coloring algorithms. - Graph Theory and Computing (R. C. Read, ed.), *Academic Press*, New York, 1972, pp. 109-122.

[9] Korshunov A. D. Basic properties of random graphs with great number of vertices and edges. - UMN, 1985, v. 40, is.1, pp. 107-173 (in Russian).