15. Hopper, M.J. Harwell subroutine library, A catalogue of subroutines, Suppl. 2, 1973; AERE-R-7477, Suppl. 2, August 1977.
16. IMSL Library 1, Ed. 6. Houston, Texas.
17. Kowalik, J., and Kumar, S. Fast Givens transformations applied to the homogeneous optimization method. *Appl. Math. and Comp. 4* (1978), pp. 239–252.
18. Lawson, C. On the discovery and description of mathematical programming algorithms. Proc. Dundee Biennial Conf. Numerical Analysis, July 1–4, 1975; Lecture Notes in Mathematics 506, Springer–Verlag, pp. 157–165.
19. Lawson, C., and Hanson, R. *Solving Least Squares Problems.* Prentice-Hall, Englewood Cliffs, New Jersey, 1974.
20. Lawson, C., Hanson, R., Krogh, F., and Kinkaid, D. Basic linear algebra subprograms for Fortran usage. Sandia Lab. Tech. Rep. SAND77–0898, Albuquerque, New Mexico, 1978 (submitted to *Trans. Math. Software*, June 1977).
21. Orchard-Hays, W. *Advanced Linear-Programming Computing Techniques.* McGraw-Hill, New York, 1968.
22. Rosen, J.B., and Suzuki, S. Construction of nonlinear programming test problems. *Comm. ACM 8*, 2 (Feb. 1965), 113.
23. Saunders, M.A. Large-scale linear programming using the Cholesky factorization. Rep. STAN-CS-72-252, Stanford U., Stanford, Calif., 1975.
24. Wilkinson, J.H. *The Algebraic Eigenvalue Problem.* Oxford U. Press, 1965.

Management Science/ Operations Research
D.F. Shanno Editor

New Methods to Color the Vertices of a Graph

Daniel Brélaz
École Polytechnique Fédérale de Lausanne

This paper describes efficient new heuristic methods to color the vertices of a graph which rely upon the comparison of the degrees and structure of a graph. A method is developed which is exact for bipartite graphs and is an important part of heuristic procedures to find maximal cliques in general graphs. Finally an exact method is given which performs better than the Randall-Brown algorithm and is able to color larger graphs, and the new heuristic methods, the classical methods, and the exact method are compared.

Key Words and Phrases: NP-complete, graph structure, balancing, graph coloring, scheduling, comparison of the methods
CR Categories: 5.25, 5.32

Introduction

The coloration of the vertices in a graph G is often used for solving scheduling problems of the following type. We are given a set E of jobs (all of them have the same processing time) which have to be performed by some agents with some machines. The constraints to be taken into account are that for each job j in E there is a subset of jobs which cannot be performed at the same time as j because they have to be performed either by the same agent or by the same machine. This type of problem arises in the so-called school scheduling problem where teachers are "agents," classes are "machines," and lectures are "jobs."

Author's address: Ecole Polytechnique Fédérale de Lausanne, Département de Mathématiques, Avenue de Cour 61, 1007 Lausanne, Switzerland.

Unfortunately until now we have not found in the literature very good algorithms for coloring the vertices of a graph in a reasonable amount of computation time (*NP*-complete problem in the sense of Karp [4]).

In this paper we present new heuristic methods and improve an exact method proposed by Randall-Brown [6]. The purpose is to demonstrate that we can find good heuristic methods which are almost as good as the exact method, but which require very little execution time.

1. A Heuristic Method, Exact for Bipartite Graphs

Definition 1. Let G be a simple graph and C a partial coloration of G vertices. We define the saturation degree of a vertex as the number of different colors to which it is adjacent (colored vertices).

Dsatur Algorithm (so called because it uses saturation degree)

1. Arrange the vertices by decreasing order of degrees.
2. Color a vertex of maximal degree with color 1.
3. Choose a vertex with a maximal saturation degree. If there is an equality, choose any vertex of maximal degree in the uncolored subgraph.
4. Color the chosen vertex with the least possible (lowest numbered) color.
5. If all the vertices are colored, stop. Otherwise, return to 3.

THEOREM 1. *The Dsatur algorithm is exact for bipartite graphs.*

PROOF. Let G be a connected bipartite graph with at least three vertices. Assume that one vertex x has a saturation degree of two; in this case it has two neighbors with different colors and we can build two chains. Because G is finite, we find a common vertex y, hence a cycle. Either the cycle is even and the two neighbors of x have the same color or G is not bipartite. Therefore the Dsatur algorithm is exact for bipartite graphs. It is also a good method to determine if a graph is bipartite (polynomial time: $O(n^2)$).

If we successively color the vertices with the least possible color in a given order, we obviously begin by the coloration of a clique. Thus we obtain lower bounds on the chromatic number. There are at least two methods with given order that give cliques with an almost maximal dimension, because they use the graph structure: Dsatur algorithm and an algorithm created by Matula et al [5]. Thus we have two good heuristic methods to find a maximal clique.

In their paper Matula et al [5] give a method to improve their algorithm. This method SLI (smallest last with interchanges) seeks to make an interchange of colors in a bipartite subgraph for each case for which we must introduce a new color. This interchange method can be used for any algorithm with coloration of the vertices by the least possible color in a given order; therefore it can be used for the Dsatur algorithm. This new algorithm will be called DSI (Dsatur with interchanges). Finally if we use, at the beginning of the Dsatur algorithm, the biggest clique obtained by the comparison of the methods Dsatur and Matula, we have a new algorithm called Matula-Dsatur algorithm.

Suppose we are interested in the balancing of the sets of colors. For all algorithms with coloration by the least possible color of the vertices in a given order (Welsh, Matula, Dsatur), we can use the method proposed by Brélaz et al [2]. It chooses for the next vertex of the classified list a color with a minimal number of vertices in the set of all already used possible colors. In [2] a comparison of the Welsh and Matula methods with and without balancing on 55 graphs shows that the balancing method is nearly as good as the original method and that the difference between the cardinalities of the colors is generally small (there is often perfect balancing and rarely more than two as the maximal difference).

In the last part of this paper we compare the new heuristic methods with the following algorithms: Matula's algorithm [5], Welsh's algorithm [8], Dunstan's algorithm [3] (Welsh's method with successive saturation of the colors, and reordering of the degrees), Tehrani's algorithm [7] (creation and coloration of bipartite subgraphs by examination of the successive sets of neighbors of the maximal degree vertex) and the SLI method. We will see in Section 3 that the Matula-Dsatur algorithm, used in Section 2, is a very good heuristic method.

2. A New Exact Method

Randall-Brown [6] has developed an algorithm which avoids redundancy in the solution enumeration of a graph vertex coloration problem. For this, he constructs the following tree.

Definition 2. A partial solution of level p is any assignment in which only the vertices $x_1, \ldots, x_p$ have been given a color.

Building of the Tree

Let K_{pq} be a coloration of $x_1, \ldots, x_p$ with q colors. From K_{pq} we obtain all the solutions with preservation of the $x_1, \ldots, x_p$ colorations by the following method:

1. Introduce a new vertex x_{p+1}.
2. Assign to x_{p+1} the colors $1, 2, \ldots, q+1$ successively.
3. Eliminate the cases, which are not usual colorations.
4. Continue the same procedure with all the new partial solutions.

All solutions which use the smallest number of colors are optimal.

After this formal definition of the solution tree, Randall-Brown gives an algorithm to go over this tree to reduce the number of solutions, which are examined while retaining only one at a time in memory. In this section we present an algorithm which uses Randall-Brown's algorithm, some ideas contained in his paper, and some new ideas.

To start, we note that the sequential coloration method, which gives at each vertex the least possible color, gives the path the most to the left of the solution tree with vertices in the order given by the coloration order. We also remark that we can arbitrarily color the vertices of a clique at the beginning of the coloration, without increasing the number of colors.

Definition 3. We define the rank of a vertex as the position of this vertex in the coloration order.

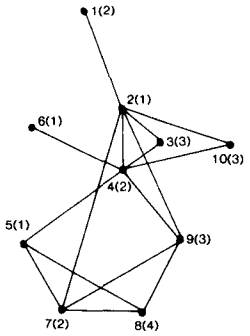
Randall-Brown's Modified Algorithm

1. Find a coloration order of the vertices, an initial clique with dimension w, and an initial coloration with r colors (an upper bound) by using the Matula-Dsatur algorithm or the DSI algorithm. If $w = r$, stop.
2. Color the vertices of the clique K with 1, 2, ... , w successively. For each other vertex x_k, write $U(x_k) = U(x_k) - (j)$, where j is the color of a K vertex, which is adjacent to x_k.
3. $k = w + 1$, $q = r$.
   Let $U(x_k)$ be the set of colors, which can color x_k ($U(x_k)$ is upper-limited by color $q - 1$ and each vertex is limited by the color with the same cardinality as its rank). Color x_k with the least possible color and remove this color from $U(x_k)$. Prohibit this color for all the vertices which are adjacent to x_k until there is a modification of the x_k color.
4. $k = k + 1$, determine $U(x_k)$.
5. If $U(x_k) = \varnothing$ go to 10.
   Otherwise let i be the minimal color of $U(x_k)$, color x_k with i and write $U(x_k) = U(x_k) - (i)$; prohibit this color for the vertices which are adjacent to x_k (with greater rank) until there is a modification of the x_k color. If $i \geq q$ go to 8. Otherwise go to 6.
6. If $k = n$, write $q = L$ where L is the number of colors used for this coloration, go to 7. Otherwise go to 4.
7. If $q = w$, stop. Otherwise: Let x_j be the q-colored vertex with minimal rank. If x_j has the rank $w + 1$, stop; otherwise write $k = j - 1$ and go to 5.
8. If $k = w + 1$, stop. Otherwise: Label all the unlabeled vertices which possess all the following properties: (i) smaller rank than k, (ii) adjacent to k, (iii) none of the colors of clique vertices adjacent to k, (iv) minimal rank among all the vertices of their color, which are adjacent to k. Write $v = k$. Those vertices are labeled with k; once we obtain the rank k or more in a partial coloration, we must remove this label.
9. Let $k = t$ where t is the maximal rank of labeled vertices, which have a smaller rank than k; for $k < i \leq v$, $U(x_i)$ is the set of colors defined in step 3. Go to 5.
10. None of the colors is possible for x_k (all are tested), we must backtrack. For this, go to 8.

THEOREM 2. *Randall-Brown's modified algorithm is an exact method for coloring the vertices of a graph.*

LEMMA. *Let r be an upper bound on the chromatic number and C be a partial coloration of a simple graph G in Randall-Brown's modified algorithm. Let k be a vertex of rank p, which has neighbors with r − 1 different colors; s(s ≤ r − 1) of which belong to the initial clique K. If s < r − 1, consider M the set of colors (from among 1 to r − 1), which does not contain vertices of the clique K which are adjacent to k. Let t_i be the vertex of color c_i with minimal rank, adjacent with k and $t = \max(\text{rank } t_i, i = 1, \ldots, v)$. Let c be the color of the corresponding vertex (with rank t); we can write $U(x_t) = U(x_t) - (c)$ and we can continue Randall-Brown's modified algorithm from this vertex without the risk of eliminating a solution with at most (r − 1) colors.*

Fig. 1. Graph with 10 vertices.



PROOF (Lemma). k is adjacent to the vertices of $r - 1$ different colors; if we want to obtain a $(r - 1)$-coloration we must change the color of at least one vertex with rank between $w + 1$ and $p - 1$ (because w is the dimension of the clique K and the clique vertices have the ranks 1, ... , w). On the other hand, if we change the coloration of a vertex adjacent to k and if there exists a vertex with the same color also adjacent to k, this color will still be prohibited for k. Finally, if we change the color of a vertex, which has not a minimal rank in the set composed of the vertices of the same color, which are adjacent to k, then k will not receive this color because the minimal rank adjacent vertex will still be of the prohibited color. Finally, if we do not want to risk losing solutions, we must take the maximal rank vertex from among all the minimal rank vertices of the different colors.

PROOF (Theorem 2). We already know that Randall-Brown's algorithm is an exact method for coloring the vertices of a graph. We have proved in the above lemma that we can go up to the labeled vertex with maximal rank without losing an interesting solution. If we arrive at step 10, all the permitted colors have been tested for the vertex; for this reason, we must backtrack. Therefore it follows that Randall-Brown's modified algorithm is an exact method to color the vertices of a graph.

Illustration of lemma. Consider the example given in Figure 1. The vertices 2, 4, 9 constitute a clique. Assume that we choose the following order to color the vertices: 2, 4, 9, 7, 5, 8, 3, 10, 1, 6. The sequential coloration method (least possible color for each vertex) gives the initial coloration: 2(1), 4(2), 9(3), 7(2), 5(1), 8(4), 3(3), 10(3), 1(2), 6(1). This coloration uses four colors, where $r = 4$ and $w = 3$. If we seek to improve this coloration we must change the color of vertex 8 with rank 6. 8 is adjacent to 9, which belongs to the clique; for this reason color 3 is prohibited for vertex 8. For the colors 1 and 2, 8 is not adjacent to a vertex of the clique. The minimal rank vertices are 7 (color 2) and 5 (color 1); the maximal rank is for 5. Let the color of 5 be 3 because 5 must not have color 2. Thus we have the coloration: 8(1), 3(3), 10(3), 1(2), 6(1). We have a three-colored coloration, which is the dimension of a clique; therefore this clique is maximal and the chromatic number is 3.

Look-ahead algorithm. Randall-Brown [6] presents another method, called look-ahead algorithm, to improve his results. This algorithm looks ahead and determines whether a candidate c_i for $U(x_k)$ would, at some later time, cause an increase in the number of attributes needed. This information is also used to determine which attribute $c_i \in U(x_k)$ to assign to x_k.

In Section 3, the following exact methods are compared:

—Randall-Brown's method with the look-ahead algorithm (RB)
—The method described in this paper without the look-ahead algorithm (EM)
—The method described in this paper with the look-ahead algorithm (EMLA).

Randall-Brown's algorithm is slightly improved here because we stop it if the number of colors obtained is the same as the dimension of a known maximal clique.

The results are as follows. Randall-Brown's algorithm is only usable for the small graphs, where its time is not as good as the two others; for the rest of the graphs, its time is too large. For the two other algorithms, the look-ahead procedure is certainly useful for group 4 (density $\cong 0.3$). For the other groups and for the smallest graphs, the results for this algorithm are not so good, the same efficiency as without look-ahead. For large graphs, this result is obtained because the Dsatur heuristic algorithm uses ideas of the same type as the look-ahead procedure. Finally, the superiority of the improved Randall-Brown algorithm has three causes: (1) best quality initial heuristic procedure, (2) determination of a large clique and the comparison of its dimension and the number of colors actually used, (3) elimination of all the intermediary cases in the backtrack procedure.

These results are very interesting because this problem is *NP*-complete in the sense of Karp [4].

Table I. Summary of Results.

Best known total: 986 (+328), number of graphs 59 (+21)
Minimal theoretical total: (with lower bounds for un-finished cases): 1274
Total of the lower bounds (Matula): 1164
Total of the lower bounds (Dsatur): 1146
Total of the lower bounds (best of the two): 1181
Number of first places (best result between the heuristic methods): Welsh: 12 (G1: 4, G2: 2, G3: 1, G4: 5), Matula: 16 (6, 3, 3, 4), Dunstan: 23 (8, 3, 4, 8), Tehrani: 34 (11, 4, 7, 12), SLI: 42 (13, 7, 13, 9), Dsatur: 51 (13, 9, 14, 15), Matula-Dsatur: 55 (12, 10, 15, 18), DSI: 66 (14, 11, 19, 22)

Table II. Heuristic Methods (Error).

		Welsh	Matula	Dunstan	Tehrani	SLI	Dsatur	Matula-Dsatur	DSI
Group 1	Total error	23	11	10	5	4	4	5	3
	% 1	5.42	2.59	2.36	1.18	0.94	0.94	1.18	0.71
Group 2	Total error	25 (+15)	17 (+12)	16 (+8)	16 (+7)	8 (+4)	6 (+4)	5 (+4)	5 (+2)
	% 1	12.38	8.42	7.92	7.92	3.96	2.97	2.48	2.48
	% 2	13.38	9.70	8.03	7.69	4.01	3.34	3.01	2.34
	% 3	16.49	12.71	11.00	10.65	6.87	6.19	5.84	5.15
Group 3	Total error	38 (+21)	31 (+17)	24 (+13)	19 (+10)	16 (+6)	13 (+8)	12 (+7)	9 (+5)
	% 1	15.90	12.97	10.64	7.95	6.69	5.44	5.02	3.76
	% 2	15.73	12.80	9.87	7.73	5.87	5.60	5.07	3.73
	% 3	20.89	17.83	14.76	12.53	10.58	10.31	9.75	8.36
Group 4	Total error	23 (+19)	23 (+14)	15 (+11)	13 (+7)	11 (+9)	8 (+5)	5 (+5)	4 (+2)
	% 1	20.54	20.54	13.39	11.61	9.82	7.14	4.46	3.57
	% 2	19.44	17.13	12.03	9.26	9.26	6.02	4.63	2.78
	% 3	29.00	26.50	21.00	18.00	18.00	14.50	13.00	11.00
General case	Total error	109 (+55)	82 (+43)	65 (+32)	53 (+24)	39 (+19)	31 (+17)	27 (+16)	21 (+9)
	% 1	11.16	8.39	6.65	5.42	3.99	3.17	2.76	2.15
	% 2	12.48	9.51	7.38	5.86	4.41	3.65	3.27	2.29
	% 3	16.01	12.95	10.75	9.18	7.69	6.91	6.51	5.49

Table III. Heuristic Methods (Time).

Group	Welsh	Matula	Dunstan	Tehrani	SLI	Dsatur	Matula-Dsatur	DSI
1A	15.40 (1.10)	16.27 (1.16)	24.63 (1.76)	15.83 (1.13)	51.34 (3.67)	17.54 (1.25)	18.18 (1.30)	47.90 (3.42)
1B	2.36	2.64	4.21	2.47	8.15	2.71	2.89	7.62
2A	15.25 (1.09)	16.19 (1.16)	21.52 (1.54)	15.59 (1.11)	47.92 (3.42)	17.27 (1.23)	17.91 (1.28)	45.45 (3.25)
2B	2.31	2.59	3.76	2.42	7.33	2.66	2.84	6.97
3A	12.98 (0.76)	14.28 (0.84)	18.48 (1.09)	15.79 (0.93)	26.46 (1.56)	14.77 (0.87)	15.48 (0.91)	27.84 (1.64)
3B	13.13 (1.88)	14.98 (2.14)	19.27 (2.75)	15.64 (2.23)	28.18 (4.03)	14.88 (2.13)	15.73 (2.25)	30.36 (4.34)
3C	2.03	2.30	3.05	2.60	4.66	2.37	2.58	4.82
4A	12.74 (0.75)	14.04 (0.83)	16.48 (0.97)	13.17 (0.77)	17.96 (1.06)	14.48 (0.86)	14.99 (0.88)	19.87 (1.17)
4B	13.86 (1.98)	14.68 (2.10)	17.32 (2.47)	13.48 (1.93)	19.74 (2.82)	14.51 (2.07)	15.36 (2.19)	21.93 (3.13)
4C	1.99	2.20	2.61	1.98	2.98	2.18	2.37	3.19

Note. When a group contains several graphs, the average CPU time is given in parenthesis.

Composition of the groups

1 A (2 A): The 14 smallest graphs of group 1 (2)
1 B (2 B): The biggest graph (100 vertices) of group 1 (2)
3 A (4 A): The 17 smallest graphs (from 10 to 85 vertices) of group 3 (4)
3 B (4 B): 7 graphs from 85 to 100 vertices of group 3 (4)
3 C (4 C): 1 of the greatest graphs (100 vertices) of group 3 (4)

Table IV. Comparison of Exact Methods.

No. vertices	Av. den.	No. problems	Time			Number of backtracks		
			RB	EM	EMLA	RB	EM	EMLA
10–50	0.7	5	59.74	5.54	5.41	3091	44	28
60	0.7	1	41.46	5.37	4.57	1022	80	40
65	0.7	1	382.06	9.50	8.40	11802	284	130
75	0.7	1	Inf.	90.49	75.39	–	3223	1637
85	0.7	1	Inf.	4.48	4.43	–	6	4
90	0.7	1	3.18	3.26	3.26	0	0	0
100	0.7	1	Inf.	6.41	6.20	–	13	10
10–55	0.5	8	19.18	5.54	5.70	628	31	27
60	0.5	1	Inf.	5.49	11.87	–	372	303
65	0.5	1	Inf.	42.09	41.00	–	1976	1265
70	0.5	2	Inf.	8.69	8.98	–	152	102
75	0.5	2	Inf.	59.72	95.63	–	2019	1998
80	0.5	2	Inf.	147.57	176.76	–	4488	3243
90	0.5	1	267.47	4.91	5.73	5609	8	8
100	0.5	1	Inf.	86.34	57.67	–	2448	1000
10–45	0.3	6	9.27	3.78	3.89	260	6	5
50	0.3	1	49.62	11.71	8.47	3458	870	339
55	0.3	1	9.23	9.91	4.89	516	652	160
60	0.3	1	5.32	2.74	2.52	197	58	33
65	0.3	1	94.64	7.62	7.26	4472	385	249
70	0.3	1	Inf.	3.49	3.20	–	49	36
75	0.3	1	Inf.	168.84	53.53	–	8849	1795
85	0.3	1	Inf.	5.15	5.80	–	48	42
90	0.3	1	Inf.	Inf.	365.05	–	–	10454
95	0.3	1	Inf.	Inf.	479.34	–	–	12396

Inf. = more than 512 seconds CPU.

3. Comparison of the methods

3.1 Data Used

We generated groups of graphs. Each graph was generated randomly as follows: for each possible edge, a random number was generated from a uniform distribution between 0 and 1; if the number was less than the desired density, the edge was included.

	Group 1	Group 2	Group 3	Group 4
Number of graphs	15	15	25	25
Density	0.73–0.82	0.61–0.72	0.44–0.59	0.26–0.34

For groups 1 and 2, we have the following numbers of vertices: 10, 20, 30, 40, 50, 60, 65, 70, 75, 80, 85, 90, 95, 100, 100. For groups 3 and 4 we have the supplementary numbers 35, 45, 55, 70, 75, 80, 85, 90, 95, 100.

3.2 Results and Interpretations

Table I gives a summary of the totals. In Table II we show under the heading "Total error" the difference in the total number of colors used between each heuristic method and the exact method. In parenthesis we show the difference using the best known result for the cases where the exact method did not terminate in a time of 512 seconds CPU on a CDC Cyber 7326. Under "% 1" we show this difference as a percentage of the total number of colors, which have been obtained by the exact method for the terminated cases. Under "% 2" we show the difference as a percentage of the total number of colors which have been obtained by the best known results (exact method for the ended cases and best known result for the other graphs). Finally under "% 3" we use the same percentage, but we consider this time the best lower bound for the cases which are not ended by the exact method. This lower bound is obtained by the exact coloration of the subgraph, which contained the first 40 vertices of the order obtained by the Matula-Dsatur algorithm. The true percentage is between the figures in "% 2" and those in "% 3."

Tables I–III show a superiority of the DSI, Matula-Dsatur, and Dsatur methods (with the advantage going to DSI, which has a penalty of two colors only once in all the cases); next comes the SLI method, next Tehrani's method, next Dunstan's method, then far behind Matula's method (which has often a penalty of three or four colors), and very far behind Welsh's method (which has often a penalty of four colors and once five colors). The best exact method [see Table IV] has allowed us to resolve the 15 graphs of group 1, 11 out of 15 graphs in group 2, 18 out of 25 graphs in group 3, and 15 out of 25 graphs in group 4.

Received May 1977; revised September 1978

References

1. Berge, C. *Graphs and Hypergraphs.* North-Holland Pub. Co., Amsterdam, 1973.
2. Brélaz, D., de Werra, D., and Nicolier, Y. Compactness and balancing in scheduling. Zeitschrift für Operations Research, Band 21, Heft 1, Serie A, 1977, pp. 65–73.
3. Dunstan, F. Greedy algorithms for optimization problems. Presented at Euro I meeting, Brussels, Jan. 1975.
4. Karp, R.M. Reducibility among combinatorial problems. In *Complexity of Computer Computations,* R.E. Miller and J.W. Thatcher, Eds., Plenum Press, New York and London, 1972, pp. 85–103.
5. Matula, D., Marble, G., and Isaacson, J. Graph coloring algorithms. In *Graph Theory and Computing,* Academic Press, New York, 1972, pp. 109–122.
6. Randall-Brown, J. Chromatic scheduling and the chromatic number problems. *Management Science 19,* 4 (Dec. 1972), Part I, 456–463.
7. Tehrani, A. Un algorithme de coloration. Cahiers du centre d'études de Recherche Opérationnelle, Vol. 17, 2-3-4, 1975, pp. 395–398.
8. Welsh, D.J.A., and Powell, M.B. An upper bound for the chromatic number of a graph and its applications to timetabling problems. *The Comptr. J. 10* (1967), 85–86.