An upper bound for the chromatic number of a graph and its application to timetabling problems

By D. J. A. Welsh and M. B. Powell*

This paper points out the connection between the basic scheduling or timetabling problem with the well known problem of colouring the vertices of a graph in such a way that (i) no two adjacent vertices are the same colour and (ii) the number of colours used is a minimum. We give an algorithm for colouring a graph subject to (i) and give a new easily determined bound for the number of colours needed. This same bound is also a new upper bound for the chromatic number of a graph in terms of the degrees of its vertices.

1. Introduction

The basic scheduling problem may be described as follows. We are given n jobs $\{J_i\}_{i=1}^{N}$, together with an incompatibility matrix $M \equiv \{m_{ij}\}$, where m_{ij} = zero or 1 depending on whether or not J_i and J_j can be carried out on the same day. The problem is to find the minimum number of days needed to carry out the n jobs.

By representing each job J_i by a point A_i and joining A_i to A_j by an undirected edge e_{ij} if and only if $m_{ij} = 1$, the above problem is clearly seen to be equivalent to finding a minimum colouring of the vertices of the graph G which has vertex set $V(G) \equiv \{A_i\}_{i=1}^{n}$ and edge set $E(G) \equiv \{e_{ij}\}$.

In the notation of Ore (1962), the minimum number of days needed is exactly the chromatic number $k(G)$, of the graph G. The problem of determining this chromatic number for an arbitrary graph G is a well known unsolved problem. In theory a solution can always be obtained by trial and error or by integer programming methods. However, in practical instances such as arise in large scale scheduling problems the labour involved often renders the method of solution inefficient, and as a compromise a solution is sought which blends the advantages of simplicity and reasonable accuracy.

2. Statement of results

The degree of a vertex A_i of the graph G is the number of edges having A_i as an endpoint, and we will denote it by d_i. Without loss of generality we assume that

$$d_1 \geqslant d_2 \geqslant \ldots \geqslant d_n. \tag{1}$$

It is easy to show (Ore 1962, Chapter 14) that if $k(G)$ denotes the chromatic number of G then

$$k(G) \leqslant d_1 + 1 \tag{2}$$

and provided G contains no d_1-simplices, then from (2) we know

$$k(G) \leqslant d_1. \tag{3}$$

The purpose of this paper is to point out that G may always be coloured in at most $\alpha(G)$ colours where

$$k(G) \leqslant \alpha(G) \leqslant \max_i \min \left[d_i + 1 \right]_i. \tag{4}$$

Furthermore, for those instances where this number is deemed sufficiently economical, a colouring employing no more than this number of colours may be obtained by a very simple algorithm described below.

The algorithm. Let $V(G)$ denote the vertex set of G. Let T_1 be a subset of $V(G)$ defined recursively as follows:

$$A_1 \epsilon T_1.$$

If $\{A_{i_k}\}_{k=1}^{m} \epsilon T_1$, where $i_1 = 1$ and

$$i_1 < i_2 < \ldots < i_m$$

then $A_j \epsilon T_1$ if and only if

$$j > i_m$$

and also A_j is not adjacent to any member of $\{A_{i_k}\}_{k=1}^{m}$.

Clearly T_1 is a finite, non-null subset of $V(G)$. Similarly we define T_2 to be a subset of $V(G) - T_1$ constructed recursively by:
If i is the least integer for which $A_i \notin T_1$, then $A_i \in T_2$. If $\{A_{j_k}\}_{k=1}^{P} \in T_2$ where $j_1 < j_2 < \ldots < j_P$ then $A_1 \in T_2$ if and only if

$$1 > j_P$$

and A_1 is not linked in G to any of the vertices $\{A_{j_k}\}_{k=1}^{P}$. Notice that T_2 may be null (when G is completely disconnected). Similarly we define T_3 to be a subset of $V(G) - (T_1 \cup T_2)$ such that: If i is the least integer for which $A_i \notin T_1 \cup T_2$, then $A_i \in T_3$, and then continuing as before.

In this way we define a sequence $\{T_i\}$ of disjoint subsets of $V(G)$ such that $V(G) = \bigcup_{i=1}^{\infty} T_i$ and there exists a finite integer $\alpha(G)$ such that

$$T_r = \phi \quad r > \alpha(G).$$

From the nature of the construction no two vertices in a given set T_i are adjacent in G and hence by assigning

* *Mathematical Institute, Oxford.*

to each vertex in T_i the colour C_i a colouring of G is achieved in $\alpha(G)$ colours.

Proof of (4). By construction every vertex in $V(G) - T_1$ is adjacent in G to some vertex of T_1. A standard inductive argument shows that every vertex in $V(G) - \bigcup_{j=1}^{m} T_j$ is adjacent to at least one vertex of each of the sets $\{T_i\}_{i=1}^{m}$.

It follows that

$$A_k \notin \bigcup_{j=1}^{m} T_j \rightarrow d_k \geqslant m$$

and so

$$A_i \in \bigcup_{j=1}^{d_i+1} T_j. \tag{5}$$

On the other hand by the nature of the algorithm

$$A_i \in \bigcup_{j=1}^{i} T_j. \tag{6}$$

Combining (5) and (6) we get

$$A_i \in \bigcup_{j=1}^{\phi(i)} T_j$$

where

$$\phi(i) = \min(i, d_i + 1).$$

Defining $\delta(G) = \max_i \phi(i)$, we see that

$$T_{\delta(G+1)} = \phi$$

and thus $\alpha(G) \leqslant \delta(G)$ which proves the required result.

3. Examples

Clearly the result (4) is usually a big improvement on (2) and for many graphs (scheduling problems) is quite a good upper bound which has the added advantage of being easily computed. It is important to notice, however, that the difference

$$\max_i \min \binom{d_i + 1}{i} - k(G)$$

can be arbitrarily large for certain graphs.

G a bipartite graph

G is said to be *bipartite* if the vertex set $V(G)$ can be partitioned into two non-null disjoint sets X, Y and the only edges of G join vertices of X to those of Y. The chromatic number of a bipartite graph is always exactly 2. In such cases, however, empirical evidence suggests that the actual number of colours used by the algorithm in the colouring is also very small.

The examination timetabling problem

As another example consider the problem studied by Cole (1964) of arranging a timetable for 34 examination papers. From the data given in Cole's Table 2 the upper bound given by (3) is 20, but the sequence $\{\phi_i\}_{i=1}^{34}$ is seen to be

1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 14, 12, . . .

This shows before even constructing a schedule that there exists a solution using at most 14 periods.

Conclusion

Apart from the improved upper bound for the chromatic number the algorithm presented above seems most useful for scheduling problems where there is a certain amount of "slack". It does not answer the much more difficult problem, which usually occurs in practice, when in addition to an incompatibility matrix we are given a preassignment matrix $P \equiv \{p_{ij}\}$ which specifies that certain jobs must be carried out on certain days ordained beforehand.

However, it would still be very informative to have some knowledge of the average value of the difference $\alpha(G) - k(G)$ over all graphs on n vertices. Because of the time needed to calculate $k(G)$ exactly, however, this seems almost insoluble theoretically or practically.

Acknowledgement

We would like to thank Professor C. Berge for some very useful advice.

References

ORE, O. (1962). "Theory of Graphs", *Amer. Math. Soc. Coloq. Publ.*, Vol. 38.
BROOKS, R. L. (1941). "On colouring the nodes of a network", *Proc. Camb. Phil. Soc.*, Vol. 37, pp. 194–197.
COLE, A. J. (1964). "The preparation of examination time-tables using a small-store computer", *The Computer Journal*, Vol. 7, pp. 117–121.