On various algorithms for estimating the chromatic number of a graph

John Mitchem

Mathematics Department, California State University at San Jose, San Jose, California 95192, USA

The well known problem of colouring the vertices of a graph with the minimum number of colours such that adjacent vertices are coloured differently is used in a variety of scheduling and storage problems. This minimum number of colours is called the *chromatic number* of the graph G and is denoted by $\chi(G)$. A number of algorithms for finding a minimum colouring and thus the chromatic number of a graph are known (Christofides, 1971). However, the computer time required to implement such algorithms is often prohibitive. Thus faster algorithms which do not always yield a minimum colouring are frequently used. In this paper we discuss a number of such algorithms and construct graphs to show that each algorithm can give an arbitrarily bad estimate for the chromatic number of a graph.

(Received April 1973)

1. Introduction

For an ordering $v_1, \ldots, v_n$ of the vertices of G Matula, Marble, and Isaacson (1972) define a sequential colouring of G as a colouring of G using colours $1, 2, \ldots, k$ determined by colouring v_1 with colour 1 and each succeeding vertex with the least positive integer such that no pair of adjacent vertices have the same colour. Clearly the colouring derived in this manner depends on the ordering of the vertices. The first two algorithms considered use sequential colourings.

2. Two sequential algorithms

In the largest first (*LF*) algorithm the vertices are arranged in non-increasing order of their degrees and are then coloured sequentially. This algorithm is given by Welsh and Powell (1967) in a slightly different but equivalent form and is mentioned by Woods (1968), Christofides, and Matula *et al.* Welsh and Powell use it to show

$$\chi(G) \leqslant \max \min \{i, d_{i+1}\} \text{ where } d_i = \deg v_i \, . \qquad (1)$$

Also (1) follows from a theorem of Wilf and Skereres (1968).

To give a sequential algorithm developed Matula *et al.* we need the following notation. If $U \subset V(G)$, then $\langle U \rangle$ is the graph with vertex set U and all the edges of G which join two vertices of U and is called the *subgroup induced* by U. Now this smallest last (*SL*) algorithm colours the vertices sequentially after ordering them as follows: α. For $n = |V(G)|$, let v_n be a vertex of minimum degree in G.
β. For $i = n - 1, \ldots, 2, 1$, let v_i be chosen to have minimum degree in $\langle V(G) - \{v_n, \ldots, v_{i+1}\} \rangle$. The *SL* and *LF* algorithms frequently do not give the same ordering to $V(G)$ nor the same colouring of G.

3. Example

For each pair of integers $n > k > 3$, we construct a graph $G_{n,k}$ with $\chi(G_{n\,k}) = k$ such that using either the largest first or smallest last algorithm yields a colouring with at least n colours. For $i = 1, \ldots, k$, let $V_i = \{v_{i,1}; v_{i,2}; \ldots; v_{i,n}\}$ and $U_i = \{u_{i,1}; u_{i,2}; \ldots; u_{i,kn}2\}$ be mutually disjoint sets. Then $V(G_{n,k}) = V \cup U$ where $V = V_1 \cup \ldots \cup V_k$ and $U = U_1 \cup \ldots \cup U_k$. Vertices $v_{i,j}$ and $v_{r,s}$ are adjacent if $i \neq r$ and $j \neq s$; and $u_{i,j}$ and $u_{r,s}$ are adjacent if $i \neq r$. Furthermore for each $j > 1$ let $v_{i,j}$ be adjacent to each of $u_{2,(j-2)kn+1}; \ldots u_{2,(j-2)kn+(j-1)k}$. For each $j > 1$ and each $i > 1$ let $v_{i,j}$ be adjacent to each of $u_{1,(j-2)kn+1}, \ldots, u_{1,(j-2)kn+(j-1)k}$.

Since, for $i = 1, \ldots, k$, no two vertices of $G_{n,k} = G$ with first subscript i are adjacent we can colour such vertices with i and $\chi(G) = k$. The degree of each vertex of U is at least $kn^2(k - 1)$, and $\deg v_{i,j} = (k - 1)(n - 1) + (j - 1)k < kn^2(k - 1)$. Thus in using the *LF* algorithm on G the elements of U are coloured first. Also $\langle U \rangle$ is a complete k-partite graph, which according to a theorem by Matula *et al.* is k-coloured by any sequential colouring. Furthermore, the various $u_{1,j}$ are coloured 1 and $u_{2,j}$ are coloured 2. Then $v_{1,n}$ is coloured 1, and for $i = 2, \ldots, k$ $v_{i,n}$ is coloured 2. Also $v_{1,n-1}$ is coloured 1 and each $v_{i,n-1}$ is coloured 3. Similarly for $j = 2, \ldots, n - 1$; $v_{1,n-j}$ is coloured 1 and for $i = 2, \ldots, k$; $v_{i,n-j}$ is coloured $j + 2$. Thus the largest first algorithm yields an $n + 1$ colouring of G.

We now order the vertices of G according to the smallest last algorithm. The vertices of the form $v_{i,1}$ will be the last k elements in the ordering. Then those of the form $v_{i,2}$, will be the next to last set of k elements in the ordering. Similarly the sets $\{v_{i,3}\}, \ldots, \{v_{i,n}\}$ are successively ordered. Then the elements of U form the first k^2n^2 vertices in the ordering and k colours are then used to colour U. Let the $u_{1,j}$ vertices be coloured s and the various $u_{2,j}$ be coloured t. We now consider two cases to complete the colouring of G.

Case 1

$s < t$. We successively colour the various $v_{i,n}$ with 1; $v_{i,n-1}$ with $2, \ldots,$ various $v_{i,n-s+2}$ with $s - 1$. Then $v_{1,n-s+1}$ is coloured s and each other $v_{i,n-s+1}$ is coloured $s + 1$. Subsequently for each $j < n - s + 1$ $v_{1,j}$ is coloured s and also for each $i > 1$ $v_{i,j}$ is coloured $n - j + 2$. Thus we have an $n + 1$-colouring of G.

Case 2

$t < s$. We successively colour the various $v_{i,n}$ with 1, $v_{i,n-1}$ with $2, \ldots,$ various $v_{i,n-t+2}$ with $t - 1$. Then $v_{1,n-t+1}$ is coloured $t + 1$ and each other $v_{i,n-t+1}$ is coloured t. Each remaining $v_{1,j}$ is coloured $t + 1$. For $i > 1$ and $n - s + 2 < j < n - t + 1$ each $v_{i,j}$ is coloured $n - j + 2$. For $i > 1$ and $j \leqslant n - s + 2$ $v_{i,j}$ is coloured $n - j + 3$. Thus we have an $n + 2$ colouring G in this case.

4. Sequential colourings with interchange

We have seen that both the *LF* and *SL* algorithms can give very poor colourings. The following recolouring procedure was introduced by Matula *et al.* to be used with sequential colouring algorithms and frequently results in a more efficient colouring of a graph.

If a vertex v_i is encountered which would require a new colour m in the sequential colouring we let $S \subset \{1, \ldots, m - 1\}$ consist of those colours which have only one adjacency to v_i. If for some $\alpha, \beta \in S$, $\alpha \neq \beta$, a component C of the subgraph induced by all vertices coloured α or β has only one vertex adjacent to v_i, then the colours α and β are interchanged in C. This frees one of α and β for colouring v_i. If the pair α, β does not exist, v_i uses the new colour m.

We observe that for a complete k-partite graph H this interchange procedure does not effect the sequential colouring of H. In adding the interchange to the LF algorithm for colouring G, the elements of U are coloured the same. In colouring V, each time some vertex v_i requires a new colour, v_i is adjacent with more than one vertex of each of the previous colours. This implies that the interchange has no effect on colouring G with the LF algorithm.

In applying the SL algorithm with interchange to G, each time a new colour is introduced the set S contains at most one element so no interchange occurs. Thus both of the algorithms with or without interchange yield colourings of G with more than n colours.

5. Colouring pairs algorithm

A procedure developed by Woods which we call the colouring pairs (CP) algorithm is now considered. The $p \times p$ similarity matrix $S = (S_{ij})$ of graph G with vertex set $\{v_1, \ldots, v_p\}$ is defined by $S_{ij} = 0$ if $i = j$ or v_i is adjacent to v_j; otherwise S_{ij} is the number of vertices adjacent with both v_i and v_j. The number S_{ij} is called the similarity of v_i and v_j.

The CP algorithm is started by colouring a pair of vertices of maximum similarity with colour 1. For each subsequent step, from the previously unconsidered pairs select one of maximum similarity and use the procedure given below:

1. Both v_i and v_j are already coloured.
   (*a*) Go to the next pair.
2. One vertex, say v_i, is already coloured r and v_j is uncoloured.
   (*a*) If the degree of v_j is less than the number of colours already used, proceed to the next pair.
   (*b*) If v_j is adjacent to a vertex already coloured r, proceed to the next pair.
   (*c*) If v_j is not adjacent to a vertex coloured r, then use r on v_j.
3. Neither v_i nor v_j is coloured.
   (*a*) If the degree of both v_i and v_j is less than the number of colours already used, proceed to the next pair.
   (*b*) Colour v_i and v_j with the first already used colour which has no vertices adjacent to v_i and v_j.
   (*c*) If v_i and v_j cannot both be coloured with a previously used colour, they become the first vertices with new colour s.

After all pairs are considered in this way we colour the vertices previously omitted because of their small degrees with colours already used. However some vertices, for example a vertex of degree $p - 1$, still may be uncoloured and, if necessary, new colours may be used to complete the colouring.

We now use the CP algorithm on graph $G_{n,k}$ from Section 3. We let $S(u, v)$ denote the similarity of distinct vertices u and v in $G_{n,k}$.

Then

for $i - r$, $S(u_{i,j}, u_{r,s}) \geqslant (k - 1)\,kn^2$
for $i \neq r$, $S(u_{i,j}, u_{r,s}) = 0$
for $i \neq 1$, $S(v_{1,j}, v_{i,j}) = (k - 2)(n - 1)$
for $i > r > 1$, $S(v_{i,j}, v_{r,j}) = (k - 2)(n - 1) + (j - 1)k$
for $j \neq s$, $S(v_{i,j}, v_{i,s}) = (k - 1)(n - 2)$
for $i \neq r, j \neq s$, $S(v_{i,j}, v_{r,s}) = 0$
and $S(v_{i,j}, u_{r,s}) \leqslant (j - 1)\,k + k - 1$.

Thus the CP algorithm colours the vertices of U with k colours before colouring any vertices of V. Let t be the colour assigned to the various $u_{1,j}$. From the fact that $(k - 2)(n - 1) + (j - 1)k \geqslant (k - 1)(n - 2)$ if and only if $j \geqslant n/k$, it follows that for $i > 1$, the various $v_{i,n}$ are coloured 1; $v_{i,n-1}$ are coloured 2; ...; $v_{i,n-t+2}$ are coloured $t - 1$; $v_{i,n-t+1}$ are coloured $t + 1$; ..., $v_{i,r}$ are coloured $n - r + 2$ for $r = \{n/k\}$. Then the remainder of the vertices are coloured and the CP algorithm uses at least $n - r + 2 > n - (n/k - 1) + 2 > n(k - 1)/k$ colours. Thus for $n > k > 3$ the graph $G_{m,k}$, $m = \{kn/(k - 1)\}$ has chromatic number k and the CP algorithm gives a colouring with at least $m(k - 1)/k \geqslant n$ colours.

6. Conclusions

For each of the algorithms above we have shown there exist graphs for which the algorithm yields arbitrarily bad estimates for the chromatic number. The user of any of these algorithms should be aware that there is a possibility that his estimate for a chromatic number is not good.

Wood states that in implementing the CP algorithm on large examples, the lists for small similarities, e.g. less than five, would be enormous and of little value, so they are not recorded. The vertices left uncoloured can then be easily coloured individually. Kelly and Kelly (1954), Erdös (1961), and Lovász (1968) have shown that there are graphs with arbitrary chromatic numbers which contain no 4-cycle. In such graphs each pair of points have similarity 0 or 1. Thus this type implementation of the CP algorithm would be worthless for such graphs.

References

Christofides, N. (1971). An algorithm for the chromatic number of a graph, *The Computer Journal*, Vol. 14, No. 1, pp. 38-39.

Erdös, P. (1961). Graph theory and probability II, *Canada Journal of Math.*, Vol. 13, pp. 346-352.

Kelly, J. B., and Kelly, L. M. (1954). Paths and circuits in critical graphs, *Amer. J. Math.*, Vol. 76, pp. 786-792.

Lovász, L. (1968). On chromatic number of finite set systems, *Acto. Math. Acad. Sci. Hungar.*, Vol. 19, pp. 59-67.

Matula, D., Marble, G., and Isaacson, J. (1972). Graph colouring algorithms, *Graph Theory and Computing* (R. Read, Ed.), Academic Press, New York, pp. 109-122.

Welsh, D., and Powell, M. (1967). An upper bound to the chromatic number of a graph and its application to time-tabling problems, *The Computer Journal*, Vol. 10, No. 1, pp. 85-86.

Wood, D. C. (1969). A technique for colouring a graph applicable to large scale timetabling problems, *The Computer Journal*, Vol. 12, No. 4, pp. 317-319.