SEQUENTIAL ALGORITHMS FOR GRAPH COLORING

M. Mansour

ABSTRACT:

A survey of existing sequential algorithms for graph coloring is presented together with new methods developed by the author. The application of these methods to storage allocation, processor scheduling, scheduling of meetings and examinations, map coloring in computer graphics, etc. is shown.

I-INTRODUCTION:

Graph coloring is a problem which has been investigated by many mathematicians and computer scientists since the introduction of the four-color conjecture by Francis Guthrie in 1852. Despite the fact that no one till now could prove the correctness of this conjecture which states that any planar map can be colored by at most four distinct colors, most of the results in graph theory have been by-products of trying to prove this conjecture.

In this paper we are not concerned with coloring of planar graphs only, but with coloring other graphs as well. Only sequential methods which give near optimal solutions to the problem shall be considered.

(A) Problem formulation:

Given a simple graph $G(V,E)$ where $V$ is the set of vertices and $E$ is the set of edges, it is required to color the vertices of $G$ so that no two adjacent vertices have the same color. The number of distinct colors should be as small as possible. By a simple graph we mean a graph with vertices connected with undirected single edges and without self loops.

(B) Applications:

The solution of this problem can be applied to different areas in computer science and operations research, for example, storage allocation of scalar variables in computer programs. The different scalar variables are represented by the vertices of the graph. Two vertices are connected by an edge if the program does not allow the storage of the corresponding variables in the same location. Minimum coloring of the graph means minimum storage space for the variables. Another problem is processor scheduling in multiprocessor systems. Each program or task is represented by a vertex, and two vertices are connected by an edge if the corresponding tasks cannot run simultaneously. The net result of the graph coloring algorithm is a processor scheduling with quasi-minimum execution time. A third problem is the scheduling of meetings or examinations to reduce the total time eliminating or limiting conflicts. The meetings or examinations are represented by vertices, and two vertices are connected by an edge if two meetings or examinations should be attended by the same person. Other applications are in map coloring, state reduction of sequential machines, partitioning of logic, etc.

II-BASIC IDEAS OF SEQUENTIAL COLORING:

There exists different methods for the exact solution of the graph coloring problem [1], [2] which can be called chromatic partitioning methods. However, these methods are combined with extensive calculations for
graphs with modest structure, and cannot be applied to more complex graphs. We are not concerned in this paper with this class of methods. On the other hand, sequential methods give practical results which are not necessarily optimal, but the computation time is reasonable. Depending on the specific sequential method used we can expect results near the optimal value. In some cases optimal solutions can result. In a sequential algorithm we have two procedures, one for ordering the vertices and the other for coloring. The ordering procedure uses a certain criterion of the vertices which can be the degree, the density, the similarity, the 2-degree, the 2-density, ... etc.

(A) Definitions:

The degree of a vertex \( v_i \) of the graph \( G \) is the number of edges having \( v_i \) as an end point and is denoted by \( d_i \).

The density \( c_{ij} \) of a vertex \( v_i \) of the graph \( G \) is the sum of the degrees of the adjacent vertices. The densities of all the vertices can be obtained from \( c = A d \) where \( A \) is the adjacency matrix of the graph and \( d \) the degree vector. The adjacency matrix is a symmetric binary matrix whose element \( a_{ij} \) is 1 whenever the two vertices \( v_i \) and \( v_j \) are adjacent, and 0 otherwise.

The similarity \( s_{ij} \) between two nonadjacent vertices \( v_i \) and \( v_j \) is the number of vertices which are adjacent to both vertices. \( s_{ij} \) is an element of the symmetric similarity matrix \( S \). Two adjacent vertices have similarity zero. \( s_{ij} \) can also be considered as the number of paths of length 2 between \( v_i \) and \( v_j \).

The 2-degree \( d_{2i} \) of a vertex \( v_i \) is the number of paths of length two having \( v_i \) as end point.

The 2-density \( c_{2i} \) of a vertex \( v_i \) is the sum of the 2-degrees of the vertices of distance 2 from \( v_i \).

The complimentary graph \( \hat{G} \) of \( G \) is the graph whose vertices are the same as those of \( G \) and whose edges are those which are not present in \( G \), i.e., two vertices are connected in \( \hat{G} \) if they are not connected in \( G \) and vice versa.

The maximal independent set \( MIS \) is the set of vertices of graph \( G \) which can be colored with the same color and cannot be added upon.

(B) Notions:

The graph \( G \) whose vertices \( v_1, v_2, v_3, \ldots, v_n \) is denoted by \( \langle v_1, v_2, \ldots, v_n \rangle \). A subgraph \( H \) of graph \( G \) which is obtained by eliminating one or more vertices together with the edges ending at them, is denoted by the remaining vertices; for example if \( v_1 \) is eliminated the remaining subgraph is denoted by \( \langle v_2, v_3, \ldots, v_n \rangle \) or \( \langle v \setminus \{v_1\} \rangle \).

(C) Nonrecursive sequential coloring: (Algorithm I)

1. The vertices of the graph are ordered according to a certain criterion: \( v_1, v_2, v_3, \ldots, v_n \)
2. \( v_1 \) is assigned color 1
3. If \( \langle v_1, v_2, \ldots, v_{i-1} \rangle \) has been \( j \) colored i.e. with \( j \) colors, then \( v_i \) is assigned color \( m \) where \( m \leq j + 1 \) is the minimum positive integer not occurring on adjacent vertices in \( \langle v_1, v_2, \ldots, v_i \rangle \). Thus \( \langle v_1, v_2, \ldots, v_i \rangle \) is \( j \) colored for \( m \leq j \) and \( j + 1 \) colored otherwise.
(D) **Recursive sequential coloring** *(Algorithm II)*

1. The vertices of the graph $G_1 = G$ are ordered according to a certain criterion: \( v_1, v_2, \ldots, v_n \)

2. \( v_1 \) is assigned color 1

3. Each vertex \( v_2, v_3, \ldots, v_n \) is tested in order for color 1. Vertex \( v_i \)

   is colored with color 1 if it is not adjacent to any of the vertices already colored with color 1. Thus a MIS(1) is colored with color 1.

Subgraph $G_2 = \left< V, \text{MIS}(G_1) \right>$ is the new graph to be colored.

4. Steps 1, 2, and 3 are repeated for subgraph $G_2$ to get a MIS(2) to be colored with color 2. Repeat for MIS(G3), MIS(G4), ... till all the vertices are colored.

II- **SEQUENTIAL COLORING WITH DEGREE AS CRITERION:**

(A) **Method of Welsh and Powell:** \([7]\)

The vertices \( v_1, v_2, \ldots, v_n \) are ordered so that \( d_1 \geq d_2 \geq \ldots \geq d_n \)

and Algorithm I is applied.

Welsh and Powell show that there is an upper bound for the number of colors \( \alpha(G) \) given by

\[
K(G) \leq \alpha(G) \leq \max_i \min [d_i+1, i]
\]

if their method is applied.

\( K(G) \) is the chromatic number of graph $G$, i.e., the minimum number of colors needed for $G$.

This result is better than the upper bounds obtained before:

\[
K(G) \leq 1 + \max_d \quad \text{(2)}
\]

or if $G$ is not a complete graph (i.e., no edges are missing)

\[
K(G) \leq \max d_i \quad \text{(3)}
\]

(B) **Method of Matula:** \([4]\)

A closer inspection of Algorithm I shows that for a given ordering \( v_1, v_2, \ldots, v_n \), the number of colors \( \alpha(G) \) has an upper bound given by

\[
\alpha(G) \leq \max \left[ 1 + d_i \left< v_1, v_2, \ldots, v_i \right> \right]
\]

where \( d_i \left< v_1, v_2, \ldots, v_i \right> \) is the degree of \( v_i \) in the subgraph \( \left< v_1, v_2, \ldots, v_i \right> \)

An ordering procedure which minimizes (4) is as follows:

1. \( v_n \) is chosen to have minimum degree in $G$

2. For $i = n-1, n-2, \ldots, 2, 1$ let \( v_i \) be chosen to have minimum degree in

\[ \left< v \setminus \{v_n, v_{n-1}, \ldots, v_{i+1}\} \right> \]

For this vertex ordering we have

\[
d_i \left< v_1, v_2, \ldots, v_i \right> = \min_{1 \leq j \leq i} d_j \left< v_1, v_2, \ldots, v_j \right>
\]

This ordering is termed smallest-last (SL) vertex ordering.
It can be shown that this ordering together with Algorithm I gives an $\alpha(G)$ given by

$$K(G) \leq \alpha(G) \leq 1 + \max \min \left[ d_v(H) \right] \text{ for any planar graph.}$$

Although the coloring procedure is nonrecursive, the ordering procedure is recursive.

(C) **Recursive sequential coloring w.r.t. III:**

The vertices of the graph $G = G_1$ are ordered according to degree:

$$d_1 \geq d_2 \geq \cdots \geq d_n$$

and Algorithm II is applied.

It can be easily proved that an upper bound of $\alpha(G)$ is given by

$$K(G) \leq \alpha(G) \leq \max_i \left[ d_{\alpha_i}(G) + m \right]$$

**IV-SEQUENTIAL COLORING USING OTHER CRITERIA:**

(A) **Using density as criterion:**

Here we use the density vector $d = \Delta d$ instead of the vector $d$. Non-recursive and recursive algorithms can be used, where in the non-recursive method a recursive ordering can be implemented as in III. A comparison between the two criteria can be obtained on statistical basis. $\Delta d$ was used by Williams, see [7].

(B) **Using similarity as criterion:**

1-Algorithm of Wood: [5]

The similarity matrix $S$ is given by

$$s_{ij} = \begin{cases} 0 & \text{if } a_{ij} = 1 \\ \sum_k (a_{ik} \wedge a_{jk}) & \text{if } a_{ij} = 0 \end{cases}$$

$\ a_{ij}$ is an element in the adjacency matrix $A$. In other words if $i$ and $j$ are not connected then the similarity is the number of other vertices which are connected to both $i$ and $j$.

The similarity matrix is scanned to find the greatest similarity. The first coloring group is started by a pair of vertices with the maximum similarity. Each pair of vertices with this similarity is then colored according to the algorithm:

1. If both $i$ and $j$ are colored then go to next pair
2. If $i$ is in coloring group $\gamma$ and the other vertex $j$ is uncolored then:
   1. If degree of $j$ is less than the number of groups then $j$ can always be colored and is ignored; go to next pair
   2. Try to add $j$ to $\gamma$, i.e., if $a_{jk} = 0$ for each vertex $k$ in group $\gamma$, then add $j$ to $\gamma$; go to next pair
   3. Go to next pair if $j$ cannot be added to group $\gamma$
3. Neither $i$ nor $j$ is colored
   1. If the degree of both $i$ and $j$ is less than the number of groups they are ignored
   2. Find the first group $\gamma$ to which $i$ and $j$ can be added i.e. $a_{ik} = 0$ and $a_{jk} = 0$ for each vertex $k$ in group $\gamma$
   3. If $i$ and $j$ cannot be added to an existing group, they become the first members of a new group
The similarity matrix is scanned repeatedly, reducing the similarity level by one each time until all vertices have been colored. Vertices left uncolored can always be added to one of the existing groups. Wood has given statistical results comparing his with the method of Welsh and Powell. The results depend on the number of vertices and the density of the graph.

2-Block similarity as criterion:
Here we can use either a nonrecursive procedure or a recursive one. In the nonrecursive procedure we choose the two vertices of highest similarity and give them the same color. Then we determine the vertex which has the highest similarity to the block of the two colored vertices i.e. the highest number of paths of length two between the vertex and the block. This vertex is colored with the same color. Repeat for the block of the three vertices and so on till the first color is finished. The second and other colors are obtained by repeating the procedure. In the recursive procedure the vertices of the first color are eliminated to get a subgraph which can be treated in the same way as the original graph.

3-Algorithm of Wagner: [G]

The criterion used in this method is not exactly the same but a criterion analogous to it. Wagner uses the successive reduction of a complimentary graph to get the coloring of the original graph. First determine the complimentary graph \( \overline{G} \) of \( G \). A graph \( C_{ij}(\overline{G}) \) is derived from \( \overline{G} \) by coalescence of vertices \( i,j \) according to:

\[ V(C_{ij}(\overline{G})) = V(\overline{G}) \setminus \{j\} \]

\[ E(C_{ij}(\overline{G})) = E(\overline{G}) \setminus \{(i,j)\} \cup \{(i,m),(j,m)\} \]

where \( R(j,E(\overline{G})) \) = set of edges in \( E(\overline{G}) \) which are incident to vertex \( j \) and \( J(i,j,E(\overline{G})) = \) set of edges of \( E(\overline{G}) \) which join \( i \) to any vertex \( m \) of \( \overline{G} \) which is not also joined to \( j \) by an edge in \( E(\overline{G}) \).

It can be easily proved that a valid coloring for \( C_{ij}(\overline{G}) \) is a valid coloring for \( G \). It is clear that the number of colors is reduced when the process of coalescence is prolonged. This means that in each step the smallest number of edges should be eliminated (the result of the coalescence procedure is the elimination of all edges in \( \overline{G} \)). Therefore, \( i,j \) are chosen to minimize the number of edges of \( G \) deleted in forming \( C_{ij}(\overline{G}) \). A vertex \( i \) whose set of adjacent vertices is a subset of the set of adjacent vertices of another vertex \( j \) should have the same color as \( j \). They are coalesced first. This is called free coalescence. The algorithm is as follows:

(a) Find \( G \)

(b) Find \( i,j \) such that \( e_{ij} \in E(\overline{G}) \) and \( W(j,E(\overline{G})) \cup \{j\} \cap W(i,E(\overline{G})) \cup \{i\} \)

where \( W(j,E) = \) set of all vertices which are joined to \( j \) along an edge in \( E \).

replace \( \overline{G} \) by \( C_{ij}(\overline{G}) \) (free coalescence)

(c) repeat (b) until no such \( i,j \) exist in \( \overline{G} \)

(d) choose \( e_{ij} \in E(\overline{G}) \) such that the minimum number of edges in \( \overline{G} \) is eliminated and replace \( \overline{G} \) by \( C_{ij}(\overline{G}) \)

(e) repeat (d) until no edge remains in \( \overline{G} \)
(f) color each vertex in the final graph $C_i^j(\bar{G})$ with one color
(g) go back and color last coalesced vertices $i,j$ with the same color
(h) repeat (g) until original graph $\bar{G}$ is recovered and colored

(C) Using 2-degree as criterion:
Vertices whose distance =2 are suitable candidates for the same color. Choosing the vertex with the highest 2-degree to color first means that a quasi maximum number of vertices can be obtained for the first color as a means of reducing the total number of colors. This criterion can be used nonrecursively by ordering the vertices according to the 2-degree and using Algorithm I, or it can be used recursively with Algorithm II. A nonrecursive coloring with a recursive ordering can also be used similar to the coloring with degree as criterion.

(D) Using 2-density as criterion:
This criterion can be used analogous to the density.

COMBINATION OF 2-DEGREE AND BLOCK SIMILARITY IN A RECURSIVE ALGORITHM:
The above mentioned criteria can be arranged in the following diagram

- sequential method
- degree as criterion
  - nonrecursive (Welsh and Powell)
  - density as criterion
    - nonrecursive (Williams)
      - similarity as criterion
        - nonrecursive (Wood)
          - quasi similarity, recursive (Wagner)
            - similarity with blocks
              - nonrecursive recursive w.r.t. MIS
    - 2-degree as criterion
      - nonrecursive recursive w.r.t. MIS
        - recursive ordering
          - nonrecursive recursive w.r.t. MIS
            - recursive ordering
It is possible to combine some of these criteria to get new algorithms. A powerful algorithm can be obtained as follows:

Begin with the vertex of highest 2-degree and use block similarity to get the vertices of the first color. Use 2-degree of the block as criterion if more than one vertex have the same similarity. The vertices of the second color can be obtained by the same method applied to the subgraph obtained after deleting the vertices of the first color. This is repeated till all the vertices are colored. The method uses matrix manipulation.

Given $A = \text{adjacency matrix}; \quad \tilde{A}$ is the compliment of $A$

$A^* = A + I$ where $I$ is the unit matrix

$A^2$ (normal matrix multiplication):

- $ij$ entry = number of paths of length 2 between $v_i$ and $v_j$
- $ii$ entry = $d_i = \text{degree of } v_i$

$d = \text{degree vector whose elements are the sums of the rows of } A$

$i^* = \text{operator for element by element multiplication of matrices}$

- $i.e. \quad A^* \cdot B = C , c_{ij} = a_{ij} \cdot b_{ij}$

$A^* \cdot A^2$: $ij$ entry = number of paths between nonadjacent $v_i$ and $v_j$ whose length is equal 2

$ii$ entry = 0

This is the similarity matrix $S$

(A) Coloring algorithm:

1. From $A$ get $A^* , A^* ^2 , A^* A^2 = S$

2. Get diagonal of $A^* \cdot S = f_i = 2$-degree vector

3. Determine maximum $f_i$; $v_i$ gets color 1

4. Determine largest element in the $i$th row of $S$; $s_{ij}$ is largest in row

$s_{ij}$. If more than one value of $s_{ij}$ has been obtained get $j$ from

max. sum of elements of $s_{ij} A^* + s_{ij} A^* A^2 = A^*$

$v_j$ has the same color as $v_i$ i.e. color 1

5. Determine largest element $g_k$ in $g_i$. If more than one value of $k$ is present then get $k$ from

max. sum of elements of $g_k A^* + [S_k A^* A^2] \cdot 1 = h$

color $v_k$ with color 1

6. Repeat 5 till you get zero vector, thus a MIS is obtained for color 1

7. Eliminate the vertices already colored to get subgraph and new adjacency matrix. Repeat 1, 2, ..., 5 for color 2.

8. Repeat 7 for the other colors till you get a zero adjacency matrix for the last color.

* In repeating 5 $g$ is replaced by $h$ and expression is extended to include the colored vertices
B) Example:

```
0 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0
1 0 1 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0
1 1 0 1 1 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0
0 0 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 1 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 1 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 1 0 1 1 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 1 0 1 1 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 1 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1
```

= A
\[ \begin{array}{cccccccccccccccccccc}
1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
2 & 0 & 0 & 0 & 2 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 3 & 3 & 1 & 1 & 0 & 0 & 0 \\
3 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
5 & 1 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
6 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
7 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
10 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
11 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
12 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
13 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
14 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
15 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
16 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
17 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
18 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
19 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
20 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array} \]

\[ f : \quad \begin{array}{cccccccccccccccccccc}
\end{array} \]

\( v_3 \) has color 1
from \( S_2 \) get \( j = 7 \) or 13
\[ g(7) : 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \]
\[ \Sigma g(7) = 20 \]
\[ g(10) : 0 \ 0 \ 0 \ 0 \ 2 \ 3 \ 0 \ 0 \ 0 \ 0 \ 0 \ 2 \ 1 \ 1 \ 0 \ 1 \ 3 \ 3 \ 2 \]
\[ \Sigma g(10) = 18 \]
\( v_7 \) has color 1
from \( g(7) \) get \( k = 9 \) or 10 or 12 or 13 or 17 or 18
\[ h(9) : 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 3 \ 3 \ 2 \ 1 \]
\[ \Sigma h(9) = 14 \]
\[ h(10) : 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 3 \ 4 \ 3 \ 2 \]
\[ \Sigma h(10) = 16 \]
\[ h(12) : 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 3 \ 6 \ 0 \ 3 \]
\[ \Sigma h(12) = 12 \]
\[ h(13) : 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 5 \ 4 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 5 \ 0 \ 0 \ 3 \]
\[ \Sigma h(13) = 17 \]
\[ h(17) : 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 3 \ 0 \ 0 \ 3 \ 0 \ 5 \ 0 \ 0 \ 0 \ 2 \ 2 \]
\[ \Sigma h(17) = 18 \]
\[ h(18) : 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 3 \ 4 \ 0 \ 6 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \]
\[ \Sigma h(18) = 13 \]
\( v_{17} \) has color 1
from \( h(17) \) \( v_{13} \) has color 1

151
new \( h(13) \) after considering \( v_1 \); in the block similarity is given by:

\[
\begin{align*}
h(13) & : 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 4 \\
v_9 \text{ is colored with color } 1 \\
h(9) & : 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 4 \\
v_{20} \text{ is colored with color } 1 \\
\end{align*}
\]

Color 1: \( v_3, v_7, v_{17}, v_{19}, v_9, v_{20} \)

The vertices of color 2 are obtained by the same method using subgraph.

Color 2: \( v_{12}, v_{18}, v_{14}, v_{15}, v_5 \)

similarly, for color 3: \( v_2, v_4, v_{11}, v_6 \)

The remaining vertices have zero adjacency matrix so that they are colored with color 4. Color 4: \( v_8, v_{10}, v_{16}, v_9 \)

This result is equivalent to using a chromatic permutation matrix \( P_c \), which is a binary matrix having a single 1 in each row and column. \( P_c^{-1} \) = \( P_c \) (orthogonal transformation). The resulting adjacency matrix \( A_c = P_c^{-1} A P_c \) has few blocks of zeros in the diagonal.

This leads to another formulation of the four color conjecture.

VI-BICHROMATIC INTERCHANGE:

Bichromatic interchange was used by Kempe in his attempt to prove the four color conjecture by induction [7]. The same method was used by Heawood in his proof that any planar graph is 5-colorable. The definition of the bichromatic interchange is as follows: [4]

Given a graph \( G \) with a \( k \)-coloring \( k \), the colors \( 1, 2, \ldots, k \), let \( V \) be the set of vertices of \( G \) colored \( i \). For \( i \neq j \), the \( i,j \) bichromatic subgraph of \( G \) is the subgraph \( \langle V_i, V_j \rangle \) and the components of \( \langle V_i, V_j \rangle \) are \( i,j \) components (components of a graph is a connected subgraph which is disconnected to the other components). If the distinct vertex colors \( i \) and \( j \) are interchanged on an \( i,j \) component of the \( k \)-colored graph \( G \) then another \( k \)-coloring of \( G \) is obtained. The bichromatic interchange can be used in order to free a color for a new vertex which is adjacent to all colors already used. It can be used together with the sequential methods described before in order to get a better result. In [4] it was proved that the recursive -smallest-vertex-degree-last-ordering with interchange coloring algorithm will utilize at most five colors in coloring any planar graph.

VII-CONCLUSION:

Different sequential methods have been presented which utilize different criteria for the ordering of the vertices. The algorithm discussed in section V was found effective in giving good results. Its merits relative to the other criteria should be investigated on a statistical basis.
ACKNOWLEDGEMENT

The author should like to thank IBM for the support obtained during the execution of this work.

REFERENCES:


