STICHTING
MATHEMATISCH CENTRUM
2e BOERHAAVESTRAAT 49
AMSTERDAM

AFDELING MATHEMATISCHE STATISTIEK

S 393

Determination of the Chromatic Number of a Graph

by

Jac. M. Anthonisse

March 1968

I Introduction

In this report an algorithm is presented which may be used to determine such a partition of a given finite set that:

1. the number of subsets in the partition is minimized,
2. certain prescribed pairs of elements are not in the same subset.

If the elements of the set are identified with the vertices of a graph in which the prescribed pairs of elements are connected by an edge, the problem is equivalent with assigning a color to each vertex of the graph in such a way that:

1. the number of colors used in minimized,
2. connected vertices have different colors.

The terminology of 'graph', 'vertex', 'edge' etc. is used throughout the report.

The algorithm was programmed in ALGOL-60 and run on the ELECTROLOGICA X-8 computer. ALGOL text and numerical results are given.

II Definitions

Let a finite, non-oriented graph G without loops be given by

1. the set $N = \{1, 2, \ldots, n\}$ of indices of its constituent vertices ($n \geq 1$),
2. its associated matrix A,

thus

$$a_{ij} = \begin{cases} 1 \text{ if vertex } i \text{ and vertex } j \text{ are connected by an 'edge'} \\ \qquad \text{(directly connected),} \\ 0 \text{ otherwise.} \end{cases}$$

It follows that

$$a_{ij} = a_{ji} \qquad (1 \leq i, j \leq n),$$

$$a_{ii} = 0 \qquad (1 \leq i \leq n).$$

If $a_{ij} = 1$ vertex i and vertex j are said to be 'adjacent' vertices. A vertex i is adjacent to a subset $S \subset N$ if a vertex $j \in S$ exists which is adjacent to i.

A set $S \subset N$ is an 'internally stable' set if S contains no adjacent vertices:

$$S \text{ internally stable} \iff \sum_{i,j} (a_{ij} | i, j \in S) = 0.$$

If the elements of $C = (S_1, \ldots, S_m)$ are disjoint, non-empty, internally stable sets, these sets constitute a 'm-subcoloring' of G, if, moreover,

$$N = S_1 \cup S_2 \cup \ldots \cup S_m$$

the m sets constitute a 'complete m-coloring' of G.

The 'chromatic number' γ of G is defined as the smallest value of m for which a complete m-coloring of G exists.

The value of γ equals the smallest number of colors with which it is possible to color the vertices of G in such a way that adjacent vertices have different colors.

For each non-empty internally stable set S_h the 'representative' r_h is defined as the smallest index in S_h:

$$r_h = \min_i (i \mid i \in S_h).$$

The m representatives

$$(r_1, \ldots, r_m)$$

of a m-subcoloring of G form a 'representation' of that subcoloring.

Two m-subcolorings:

$$C = (S_1, \ldots, S_m)$$

and

$$C' = (S'_1, \ldots, S'_m)$$

are defined to be the 'same' subcoloring ($C = C'$) if a permutation $(p_1, \ldots, p_m)$ of the numbers $(1, \ldots, m)$ exists with:

$$S'_{p_i} = S_i \qquad (i = 1, \ldots, m).$$

A m-subcoloring $C = \{S_1, \ldots, S_m\}$ is defined to be in its 'normal' form if

$$r_1 < r_2 < \ldots < r_m.$$

In the sequal all colorings are supposed to be given in their normal form.

A m-subcoloring $C = (S_1, \ldots, S_m)$ is said to be 'included' in the m-subcoloring $C' = (S'_1, \ldots, S'_m)$, notation $C \subset C'$, if

$$S_i \subset S'_i \qquad (i = 1, \ldots, m).$$

A complete m-subcoloring C' is a 'completion' of C if C is included in C'.

III Outline of the procedure

Let $P(m)$ be a procedure which

1. either constructs a complete m-coloring of G,
2. or finds evidence that such a coloring does not exist.

If a lower bound $b_l \geq 1$ and an upper bound $b_u \leq n$ of γ are known (e.g. $b_l = 1$, $b_u = n$) the following steps are sufficient to compute γ and a complete γ-coloring of G:

step:

```
1. define $l = b_l - 1$, $u = b_u + 1$,
2. if $u - l < 2$ proceed with step 7,
3. (re)define $m = l$ + integer part of $(\frac{u-l}{2})$,
4. perform $P(m)$,
   if a complete $m$-coloring does not exist proceed with step 6,
5. copy the coloring that was found in step 4,
   redefine $u = m$, proceed with step 2,
6. redefine $l = m$, proceed with step 2,
7. define $\gamma = u$, the last coloring that was copied in step 5 is a
   complete $\gamma$-coloring.
```

It is well-known that the problem to be solved by $P(m)$ is equivalent with a linear programming problem in $n \times m$ zero-one variables and $E \times m + n$ constraints, where $E = \sum_{i<j} a_{ij}$ = number of edges in G.

An alternative procedure will be given now.

IV The procedure P(m)

The relation $C \equiv C'$ is defined to hold between two m-subcolorings C and C' if and only if the representations of C and C' are identical. Obviously, the relation '$\equiv$' is an equivalence relation and there is a one-to-one correspondence between the equivalence classes and the representations. As each representation may be interpreted as a m-subcoloring with

$$S_i = \{r_i\} \qquad (i = 1, \ldots, m),$$

each equivalence class may be described as consisting of all m-subcolorings which include the representation corresponding with that equivalence class.

Let $Q(r_1, \ldots, r_m)$ be a procedure which

1. either constructs a complete m-coloring from the equivalence class corresponding with the representation $(r_1, \ldots, r_m)$,
2. or finds evidence that this equivalence class contains no complete m-coloring.

Let R(m) be a procedure which

1. either construct a representation $(r_1, \ldots, r_m)$ different from all representations that were constructed previously, and for which representation it is not evident that a complete coloring including this representation does not exist,
2. or finds that such a representation does not exist.

Now the procedure P(m) may be described by the following steps:

step

1. perform R(m),
   if no representation exists, proceed with step 4,
2. perform $Q(r_1, \ldots, r_m)$,
   where $(r_1, \ldots, r_m)$ in the representation found in step 1,
   if no completion of $(r_1, \ldots, r_m)$ exists proceed with step 1,
3. copy the complete coloring found in step 2, terminate P(m),
4. a complete m-coloring does not exist, terminate P(m).

V The procedure $Q(r_1, \ldots, r_m)$

$Q(r_1, \ldots, r_m)$ in a special case of the procedure $T(S_1, \ldots, S_m)$, where $(S_1, \ldots, S_m)$ is a m-subcoloring including $(r_1, \ldots, r_m)$, to be described now.

Define

$$F = N - \{S_1 \cup \ldots \cup S_m\}$$

and assume

$$F = \{f_1, f_2, \ldots, f_v\},$$

thus F contains all 'free' indices, the indices of the vertices which are left uncolored by $C = (S_1, \ldots, S_m)$.

If $v = 0$ C is a complete m-coloring of G.

Now suppose $v > 0$, then a $m \times v$ matrix B may be constructed with elements:

$$b_{ij} \begin{cases} = 0 & \text{if } f_j \text{ not incident with } S_i \text{ and } f_j > r_i, \\ \neq 0 & \text{otherwise.} \end{cases}$$

$$(i = 1, \ldots, m;\ j = 1, \ldots, v).$$

From B the quantities z_j $(j = 1, \ldots, v)$ may be computed:

z_j = number of elements = 0 in the j-th column of B.

Finally, j^* is defined by

$$z_{j^*} \leq z_j \qquad (j = 1, \ldots, v).$$

The rationale behind the definition of B is that $b_{ij} = 0$ if and only if the addition of f_j to S_i neither results in an unstable set $S_i \cup \{f_i\}$, nor results in a stable set that is not in the equivalence class corresponding to $(r_1, \ldots, r_m)$.

If $z_{j^*} = 0$ there exists no completion of C.

Suppose now $z_{j^*} > 0$, let $(i_1, \ldots, i_k, \ldots, i_w)$ be defined by:

$$b_{i_k j^*} = 0 \qquad (k = 1, \ldots, w),$$

where $w = z_{j^*}$.

Then w m-subcolorings $C^{(k)}$ $(k = 1, \ldots, w)$ each including C and each with $v - 1$ free indices are given by:

$$C^{(k)} = (S_1, \ldots, S_{i_k} \cup \{j^*\}, \ldots, S_m) \quad (k = 1, \ldots, w).$$

This leads to the following recursive description of the procedure T, where T is applied to the m-subcoloring $C = (S_1, \ldots, S_m)$ of G:

```
step
1. compute v,
   if v = 0 a complete coloring is found, proceed with step 5,
2. construct a matrix B, determine w and $j^*$,
   if w = 0 proceed with step 4,
3. apply, for k = 1, 2, ..., w consecutively, T to $C^{(k)}$,
4. terminate this instance of T,
5. copy the complete coloring of G,
   terminate all instances of T.
```

Thus, applied to $C = (S_1, \ldots, S_m)$, the procedure T

1. either constructs a completion of C,
2. or finds that a completion of C does not exist,

and $Q(r_1, \ldots, r_m)$ in equivalent with

$$T(\{r_1\}, \ldots, \{r_m\}).$$

It should be noted that a matrix $B^{(k)}$ associated with $C^{(k)}$ may be obtained from the matrix B associated with C by

1. deleting the j^*-th column,
2. re-computing the elements of the i_k-th row.

In the procedure to be given in section IX the elements of B are defined by

$$b_{ij} = \begin{cases} 1 & \text{if } f_j < r_i, \\ \sum_l (a_{f_j l} \mid l \in S_i) & \text{otherwise.} \end{cases}$$

VI The procedure R(m)

The normal form of a m-subcoloring $C = (S_1, \ldots, S_m)$ was chosen such that

$$r_1 < r_2 < \ldots < r_m.$$

The procedure R(m) should generate only representations for which it is not evident that no completion exists. This gives

$$r_1 = 1$$

for all representations to be considered. The maximum number of different representations is

$$\binom{n-1}{m-1}$$

as each set $\{1, r_2, \ldots, r_m\} \subset \{1, 2, 3, \ldots, n\}$ constitutes a representation.
This leads to the relation

$$i \leq r_i \leq n-m+i \quad (i = 1, \ldots, m)$$

which is easily verified.

The following rules with yield all $\binom{n-1}{m-1}$ representations in lexicographical order:

'first' representation: $r_i = i \quad (i = 1, \ldots, m)$

'next' representation:
let $(r_1, \ldots, r_m)$ be the current representation.

step
1. define i^* as the maximal index i with

$$r_i < n-m+i,$$

2. if $i^* = 1$ a next representation does not exist, otherwise it is defined by:

$$r'_i = r_i \qquad (i = 1, \ldots, i^* - 1)$$

$$r'_{i^*} = r_{i^*} + 1$$

$$r'_{i^*+l} = r'_{i^*} + l \quad (l = 1, \ldots, m-i^*).$$

Let $C = (S_1, \ldots, S_m)$ be a completion of the representation $(r_1, \ldots, r_m)$. If

$$r_l < j < r_{l+1}$$

for a vertex j and an index l $(1 \leq l < m)$, the definition of a representative gives

$$j \in \bigcup_{i=1}^{l} S_i.$$

Thus a completion of $(r_1, \ldots, r_m)$ does certainly not exist if there are a vertex j and an index l $(1 \leq l < m)$ such that

$$r_l < j < r_{l+1}$$

and

vertex j is adjacent to each of the vertices $r_1, r_2, \ldots, r_l$.

It is even true (and easily recognized) that:
if $r_l < j < r_{l+1}$ and j is adjacent to each of the vertices $r_1, \ldots, r_l$, none of the representations $(r'_1, \ldots, r'_m)$ with

$$r'_i = r_i \qquad (i = 1, \ldots, l),$$

can be completed.

This leads to the following extension of the rules to generate a 'next' representation:

let $(r_1, \ldots, r_m)$ be a 'next' representation obtained by the steps given above.

step
3. compute $c_{lj} = \sum_{i \leq l} a_{r_i j} \quad (l = 1, \ldots, m-1;\ r_l < j < r_{l+1})$
4. if $c_{lj} \neq l$ for all l and j proceed with step 7.
5. define l^* as the smallest index l for which a j exists with $c_{lj} = l$
6. define i^* as the maximal index i with

$$i \leq l^* \text{ and } r_i < n - m + i,$$

proceed with step 2.
7. copy the representation found in step 2, terminate this instance of R.

VII Optimal indexing

The efficiency of the algorithm described in the preceding sections depends on the number of representations that is generated by R(m) and presented to Q. This number is reduced by assigning the indices to the vertices of G in some 'optimal' way.

Define

$$d_i = \sum_{j=1}^{n} a_{ij} \quad (i = 1, \dots, n).$$

It was proved in [1] that

$$b_u = \max_i \min(d_i + 1, i)$$

provides an upperbound for the chromatic number of G if the vertices are indexed such that

$$d_1 \geq d_2 \geq \dots \geq d_n.$$

Now it will be assumed that the vertices are indexed in this way (result of step 1), in n-2 steps an 'optimal' indexing is obtained. Let the permutation of the original indices left by the s-th step be:

$$(t_1, t_2, \dots, t_n).$$

In the (s+1)-th step the smallest index i with

$$s + 1 \leq i \leq n$$

and

$$\sum_{l=1}^{s} a_{t_l t_i} \geq \sum_{l=1}^{s} a_{t_l t_j} \quad (s + 1 \leq j \leq n)$$

is computed. Let i^* be this smallest index:

$$(t_1, t_2, \dots, t_s, t_{s+1}, \dots, t_{i^*}, \dots, t_n),$$

then the new permutation is:

$$(t_1, t_2, \dots, t_s, t_{i^*}, t_{s+1}, \dots, t_{i^*-1}, t_{i^*+1}, \dots, t_n).$$

If the elements of A are permuted according to the indexing found in this way it may be expected that the upper left corner of A will contain most of the elements = 1. And due to the order in which the representations are generated it may be expected that after a few representations a complete coloring is found or that it is evident that a complete m-coloring does not exist.

The smallest index l with

$$\sum_{i=1}^{j-1} a_{ij} = j-1 \qquad (j = 1, \dots, l)$$

and

$$\sum_{i=1}^{l} a_{i,l+1} \leq l-1$$

provides a lower bound for the chromatic number of G.

VIII An example

Consider the graph given in figure 1, with associated matrix as in table 1.

	1	2	3	4	5	6	7	8	9	10
1	0	0	0	1	0	1	1	1	1	1
2	0	0	0	0	0	1	1	0	0	0
3	0	0	0	1	0	0	0	1	0	1
4	1	0	1	0	0	1	1	0	0	1
5	0	0	0	0	0	1	0	1	0	1
6	1	1	0	1	1	0	0	0	0	0
7	1	1	0	1	0	0	0	0	1	0
8	1	0	1	0	1	0	0	0	1	0
9	1	0	0	0	0	0	1	1	0	0
10	1	0	1	1	1	0	0	0	0	0

table 1.

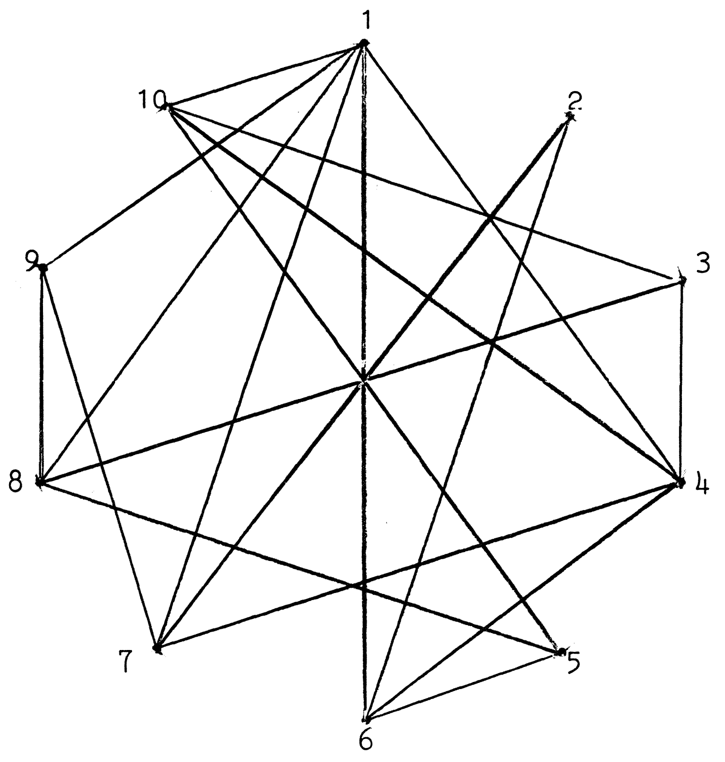

figure 1.

Table 2 contains the value of d_i corresponding with each of the vertices.

i	1	2	3	4	5	6	7	8	9	10
d_i	6	2	3	5	3	4	4	4	3	4

table 2.

In table 3 such a permutation of the indices is given that the corresponding d_i's form a non-increasing sequence. Table 4 shows the associated matrix of the graph after the rows and columns were interchanged according to this permutation.

i	1	4	6	7	8	10	3	5	9	2
d_i	6	5	4	4	4	4	3	3	3	2

table 3.

	1	4	6	7	8	10	3	5	9	2
1	0	1	1	1	1	1	0	0	1	0
4	1	0	1	1	0	1	1	0	0	0
6	1	1	0	0	0	0	0	1	0	1
7	1	1	0	0	0	0	0	0	1	1
8	1	0	0	0	0	0	1	1	1	0
10	1	1	0	0	0	0	1	1	0	0
3	0	1	0	0	1	1	0	0	0	0
5	0	0	1	0	1	1	0	0	0	0
9	1	0	0	1	1	0	0	0	0	0
2	0	0	1	1	0	0	0	0	0	0

table 4.

It follows from table 3, that $b_u = 5$ provides an upperbound for the chromatic number of the graph.

Now, starting from table 4, the rules given in section VII will be applied to obtain an optimal order of the vertices. The indices 1, 4, 6 and 7 are in their proper position already, 8 and 10 must be interchanged, resulting in table 5.

	1	4	6	7	10	8	3	5	9	2
1	0	1	1	1	1	1	0	0	1	0
4	1	0	1	1	1	0	1	0	0	0
6	1	1	0	0	0	0	0	1	0	1
7	1	1	0	0	0	0	0	0	1	1
10	1	1	0	0	0	0	1	1	0	0
8	1	0	0	0	0	0	1	1	1	0
3	0	1	0	0	1	1	0	0	0	0
5	0	0	1	0	1	1	0	0	0	0
9	1	0	0	1	0	1	0	0	0	0
2	0	0	1	1	0	0	0	0	0	0

table 5.

Next, 8 and 3 are interchanged, giving table 6, an optimal indexing has been reached, the new indices are also shown in table 6. Throughout the computations the <u>new</u> indices will be used.

original indices

	1	4	6	7	10	3	8	5	9	2	
1	0	1	1	1	1	0	1	0	1	0	1
4	1	0	1	1	1	1	0	0	0	0	2
6	1	1	0	0	0	0	0	1	0	1	3
7	1	1	0	0	0	0	0	0	2	1	4
10	1	1	0	0	0	1	0	1	0	0	5
3	0	1	0	0	1	0	1	0	0	0	6
8	1	0	0	0	0	1	0	1	1	0	7
5	0	0	1	0	1	0	1	0	0	0	8
9	1	0	0	1	0	0	1	0	0	0	9
2	0	0	1	1	0	0	0	0	0	0	10
	1	2	3	4	5	6	7	8	9	10	

new indices

table 6.

From table 6 the lower bound $b_1 = 3$ for the chromatic number is computed.

According to section III the procedure P should be performed now with $m = 2 + \left[\frac{6-2}{2}\right] = 4$.

The first representation is (1, 2, 3, 4), the corresponding matrix B, and z_j are given in table 7.

color	vertices	5	6	7	8	9	10	free indices
1	1	1	0	1	0	1	0	
2	2	1	1	0	0	0	0	matrix B
3	3	0	0	0	1	0	1	
4	4	0	0	0	0	1	1	
		2	3	3	2	2	2	z_j

table 7.

First vertex 5 is colored with color 3, giving table 8.

color	vertices	6	7	8	9	10
1	1	0	1	0	1	0
2	2	1	0	0	0	0
3	3, 5	1	0	2	0	1
4	4	0	0	0	1	1
		2	3	3	2	2

table 8.

The following tables are self-explanatory.

color	vertices	7	8	9	10
1	1, 6	2	0	1	0
2	2	0	0	0	0
3	3, 5	0	2	0	1
4	4	0	0	1	1
		3	3	2	2

table 9.

color	vertices	7	8	10
1	1, 6	2	0	0
2	2, 9	1	0	0
3	3, 5	0	2	1
4	4	0	0	1
		2	3	2

table 10.

color	vertices	8	10
1	1, 6	0	0
2	2, 9	0	0
3	3, 5, 7	3	1
4	4	0	1
		3	2

table 11.

color	vertices	8
1	1, 6, 10	0
2	2, 9	0
3	3, 5, 7	3
4	4	0
		3

table 12.

color	vertices	
1	1,6,10,8	
2	2, 9	
3	3, 5, 7	
4	4	

table 13.

Thus a complete 4-coloring exists, procedure P is performed again, with $m = 2 + \left\lceil \frac{4-2}{2} \right\rceil = 3$.

color	vertices	4	5	6	7	8	9	10
1	1	1	1	0	1	0	1	0
2	2	1	1	1	0	0	0	0
3	3	0	0	0	0	1	0	1
		1	1	2	2	2	2	2

table 14.

color	vertices	5	6	7	8	9	10
1	1	1	0	1	0	1	0
2	2	1	1	0	0	0	0
3	3, 4	0	0	0	1	1	2
		1	2	2	2	1	2

table 15.

color	vertices	6	7	8	9	10
1	1	0	1	0	1	0
2	2	1	0	0	0	0
3	3, 4, 5	1	0	2	1	2
		1	2	2	1	2

table 16.

color	vertices	7	8	9	10
1	1, 6	2	0	1	0
2	2	0	0	0	0
3	3, 4, 5	0	2	1	2

table 17.

color	vertices	7	8	10
1	1, 6	2	0	0
2	2, 9	1	0	0
3	3, 4, 5	0	2	2
		1	2	2

table 18.

color	vertices	8	10
1	1, 6	0	0
2	2, 9	0	0
3	3,4,5,7	3	2
		2	2

table 19.

color	vertices	10
1	1, 6, 8	0
2	2, 9	0
3	3,4,5,7	2
		2

table 20.

color	vertices	
1	1,6,8,10	
2	2, 9	
3	3,4,5,7	

table 21.

As a complete coloring with $b_1 = 3$ colors has been found it follows that $\gamma = 3$, the coloring is given in table 22.

original index	1	2	3	4	5	6	7	8	9	10
color	1	1	1	2	1	3	3	3	2	3

table 22.

IX ALGOL-60 procedures

Two procedures are given:
PERMUTE rearranges the rows and columns of the associated matrix of the graph and computes an upper and lower bound for the chromatic number, as described in section VII.
CHROMATIC NUMBER computes the chromatic number and a coloring of the graph using the minimum number of colors, as described in sections III - VI.

A call of PERMUTE followed by a call of CHROMATIC NUMBER is sufficient to compute the chromatic number.

Both procedures contain calls of the non-local procedure SUM which may be defined as:

```
real procedure SUM (i, a, b, x); value b; integer i, a, b; real x;
begin real s; s := 0;
      for i := a step 1 until b do s := s + x;
      SUM := s
end;
```

```
procedure PERMUTE(A, I, n, upper, lower); integer n, upper, lower;
integer array A, I;
comment A[1 : n, 1 : n] should contain the associated matrix of a
symmetric graph without loops, I[1 : n] should contain the indices
of the vertices of this graph. PERMUTE interchanges the rows and
columns of A in accordance with an optimal re-indexing of the
vertices of the graph. The elements of I are permuted such that
I[i] contains the original index corresponding with the new index
i. Such values are assigned to upper and lower that these variables
provide an upper bound and a lower bound for the chromatic number
of the graph;

begin integer g, h, i, j, k, l;
      integer array D[1:n];

      procedure wsl(i, j); value i, j; integer i, j;
      begin integer g, h;
            for h:= 1 step 1 until n do
            begin g:= A[h,i]; A[h,i]:= A[h,j]; A[h,j]:= g end;
            for h:= 1 step 1 until n do
            begin g:= A[i,h]; A[i,h]:= A[j,h]; A[j,h]:= g end;
            g:= I[i]; I[i]:= I[j]; I[j]:= g; g:= D[i]; D[i]:= D[j];
            D[j]:= g
      end wsl;

      for i:= 1 step 1 until n do D[i]:= SUM(j, 1, n, A[i,j]);
      for i:= 1 step 1 until n do
      begin g:= - 1;
            for j:= i step 1 until n do
            begin h:= D[j]; if h > g then
                  begin g:= h; k:= j end
            end;
            if k ≠ i then wsl(i, k)
      end;
      upper:= 1;
```

```
    for i:= 1 step 1 until n do
    begin h:= D[i] + 1; if h > i then h:= i;
          if h > upper then upper:= h
    end;
    lower:= 1;
    for i:= 2 step 1 until n do
    begin g:= - 1;
          for j:= i step 1 until n do
          begin h:= SUM(l, 1, i - 1, A[j,l]); if h > g then
                begin g:= h; k:= j end
          end;
          for j:= k - 1 step - 1 until i do wsl(j, j + 1);
          if g = i - 1 ∧ lower = i - 1 then lower:= lower + 1;
    end
end PERMUTE;
```

```
integer procedure CHROMATIC NUMBER(A, n, C, upper, lower);
value n, upper, lower; integer n, upper, lower; integer array A, C;
comment CHROMATIC NUMBER:= chromatic number of the symmetric
graph without loops with associated matrix A[1 : n, 1 : n]. upper and
lower should provide an upper and lower bound for this chromatic
number. A coloring of the graph using the minimum number of colors
is stored into C[1 : n], C[i]:= j if vertex i is to be colored with
color j ;

begin integer m;
      integer array CC[1:n];

      procedure P(m); value m; integer m; if m = n then
      begin integer i;
            for i:= 1 step 1 until n do C[i]:= i; upper:= m
      end
      else
      begin integer array B[1:m,1:n - m], rep[1:m], I, Z[1:n - m];
            integer col;
            Boolean first;

            procedure R;
            begin integer h, i, j, k, l, a, z, f, g;
                  if ¬first then goto iii; first:= false;
                  for i:= 1 step 1 until m do rep[i]:= CC[i]:= i;
                  goto BCC;
            iii:  i:= m + 1;
            im1:  i:= i - 1; if i = 1 then goto step3;
                  h:= rep[i]:= rep[i] + 1;
                  if h > n - m + i then goto im1;
                  for j:= i + 1 step 1 until m do rep[j]:= rep[j - 1]
                  + 1;
                  for i:= 2 step 1 until m do
                  begin h:= rep[i - 1] + 1; k:= rep[i] - 1;
```

```
                for j:= h step 1 until k do if SUM(1, 1, i -
                1, A[j,rep[l]]) = i - 1 then goto im1
            end;
            for i:= 1 step 1 until m do CC[rep[i]]:= i;
   BCC:     g:= 0; f:= m + 1; col:= 0;
            for i:= 2 step 1 until m do
            begin h:= rep[i - 1] + 1; k:= rep[i] - 1;
                for j:= h step 1 until k do
                begin g:= g + 1; I[g]:= j; CC[j]:= 0; z:= 0;
                    for l:= 1 step 1 until i - 1 do
                    begin a:= B[l,g]:= A[rep[l],j];
                        if a = 0 then z:= z + 1
                    end;
                    if z < f then
                    begin f:= z; col:= g end;
                    Z[g]:= z;
                    for l:= i step 1 until m do B[l,g]:= 1
                end
            end;
            for j:= rep[m] + 1 step 1 until n do
            begin g:= g + 1; I[g]:= j; CC[j]:= 0; z:= 0;
                for l:= 1 step 1 until m do
                begin a:= B[l,g]:= A[rep[l],j];
                    if a = 0 then z:= z + 1
                end;
                if z < f then
                begin f:= z; col:= g end;
                Z[g]:= z; if f = 0 then goto iii
            end
        end R;


        procedure T(col); value col; integer col;
        begin integer k, i, j, loc, f, z, a, b;
            for i:= 1 step 1 until m do if B[i,col] = 0 then
            begin j:= I[col]; I[col]:= - j; CC[j]:= i;
                f:= m + 1; loc:= 0;
```

```
                    for k:= 1 step 1 until n - m do if I[k] > 0
                    then
                    begin a:= A[I[k],j]; b:= B[i,k]; z:= Z[k];
                          B[i,k]:= b + a;
                          if b = 0 ∧ a ≠ 0 then z:= Z[k]:= z - 1;
                          if z < f then
                          begin f:= z; loc:= k end;
                     end;
                    if loc = 0 then goto step4; T(loc);
                    j:= - I[col]; CC[j]:= 0;
                    for k:= 1 step 1 until n - m do if I[k] > 0
                    then
                    begin a:= A[I[k],j]; b:= B[i,k]; z:= Z[k];
                          B[i,k]:= b - a;
                          if b = a ∧ b ≠ 0 then Z[k]:= z + 1
                    end;
                    I[col]:= j
              end
        end T;

        first:= true;
  next: R; T(col); goto next
  end P;

  if upper > n then upper:= n; if lower < 1 then lower:= 1;
  upper:= upper + 1; lower:= lower - 1;
step1: if upper - lower < 2 then goto exit;
  m:= lower + (upper - lower) : 2;
step2: P(m);
step3: lower:= m; goto step1;
step4: upper:= m;
  for m:= 1 step 1 until n do C[m]:= CC[m]; goto step1;
exit: CHROMATIC NUMBER:= upper
end ;
```

X Numerical Results

The efficiency of the algorithms given in the previous section was studied by generating associated matrices with the help of random numbers and measuring the time that was used to compute each chromatic number.
The matrices were of order $n = 10, 20, 30$, the elements were drawn from the distribution

$$\text{Prob}\ (\underline{x} = 0) = 1 - \rho$$

$$\text{Prob}\ (\underline{x} = 1) = \rho$$

with $\rho = .2, .4, .6, .8$.

For $n = 10$ (20, 30) and each ρ, 10(20, 25) matrices were generated.
The results of these 220 experiments are given in tables 23-26.

Table 23 shows the observed frequencies of δ = upperbound - lowerbound.

Table 24 contains the frequencies of γ = chromatic number.

In table 25 the computing times are given.

Table 26 shows the means of the computing times (in seconds) for each combination of n and ρ.

n	10				20				30			
ρ	.2	.4	.6	.8	.2	.4	.6	.8	.2	.4	.6	.8
δ												
0	3											
1	6	4	3	7	1							
2	1	6	6	2	8							
3			1	1	10							
4					1	3			7			
5						16	1	5	11			
6						1	11	4	7			
7							6	8		4		
8							2	3		14		
9										7		
10											8	3
11											12	3
12											4	9
13											1	6
14												4

table 23.

n	10				20				30			
ρ	.2	.4	.6	.8	.2	.4	.6	.8	.2	.4	.6	.8
γ												
2	3											
3	7	8			10				1			
4		2	4		10	7			23			
5			6	1		13			1	2		
6				4			7			23		
7				5			10					
8							3	1			18	
9								7			7	
10								2				
11								10				3
12												15
13												7

table 24.

n	10				20				30			
ρ	.2	.4	.6	.8	.2	.4	.6	.8	.2	.4	.6	.8
time (seconds)												
$\leq .5$	6	1	5	8								
.5 - 1	4	9	5	2								
1 - 2												
2 - 3					3			11				
3 - 4					12	13	11	6				
4 - 5					5	5	6	1				
5 - 10						2	3	2	2	1	1	6
10 - 15									21	13	9	6
15 - 20									2	6	2	6
20 - 30										3	10	3
30 - 40												
40 - 50											1	
50 - 60												
> 60										2	2	4

table 25.

ρ	.2	.4	.6	.8
n				
10	.50	.58	.51	.46
20	3.54	4.04	4.17	3.29
30	12.14	22.68	29.04	43.46

table 26.

XI Literature

1. D.J.A. Welsh, M.B. Powell, An upperbound for the chromatic number of a graph and its application to time tabling problems.
The Computer Journal, 10 (1967) 85-86.

2. C. Berge, The Theory of Graphs, translated by A. Doig.
London, Methuen, and New York, John Wiley, 1962.