European Journal of Operational Research 51 (1991) 248–258
North-Holland

Theory and Methodology

Exact colouring algorithm for weighted graphs applied to timetabling problems with lectures of different lengths

Mirjana Čangalović
Faculty of Organizational Sciences, University of Belgrade, Belgrade, Yugoslavia

Jan A.M. Schreuder
Faculty of Applied Mathematics, University of Twente, Postbus 217, 7500 AE Enschede, Netherlands

Abstract: An exact algorithm is presented for determining the interval chromatic number of a weighted graph. The algorithm is based on enumeration and the Branch-and-Bound principle. Computational experiments with the application of the algorithm to random weighted graphs are given. The algorithm and its modifications are used for solving timetabling problems with lectures of different lengths.

Keywords: Vertex-colouring, timetable, graph, integer programming

1. Introduction

In order to model school timetabling problems with lectures of different lengths in terms of graph colouring the concept of vertex-composite graphs is introduced (Punter, 1976). This concept is a version of an interval colouring of a weighted graph (Golumbic, 1980). This idea is further elaborated (Clementson and Elphick, 1983; de Werra and Hertz, 1988).

A graph is said to be weighted if a positive integer is associated with each vertex of the graph.

Many practical problems can be formulated as a colouring problem of such kind of graphs. For example, problems of scheduling jobs with different processing times and with no preemption allowed, where each job needs some resources for its processing. Specially, if the jobs are school-lectures and the resources are teachers, class rooms or other school equipment, the problems represent a generalisation of the school timetabling problem (Krarup and de Werra, 1982; Clementson and Elphick, 1983; de Werra, 1985). These kind of assignment problems are hard to solve in the mathematical part as well in the real world (Eiselt and Laporte, 1987).

The general approach is to develop heuristic algorithms. We think however, it is better to use implicit enumeration whenever possible. This is due to the fact that in applications there must always be an acceptable — feasible — solution or evidence that such a solution does not exist (Schreuder and van der Velde, 1984). It boils down to reducing the solution space based on intimate knowledge of the problem, until it can be searched with an exact polynomial time algorithm.

Until now only heuristic colouring algorithms for weighted graphs have been proposed (Clementson and Elphick, 1983). Also some experiments with simulated annealing (Aarts and van

Received December 1987; revised August 1989

Laarhoven, 1989) have been executed, but the results were not encouraging (Čangalović, 1989).

In this article we propose an exact algorithm for determining the chromatic number of a weighted graph. Then we discuss some numerical experiments. Finally, we use the algorithm for solving timetable problems with lectures of different lengths.

2. Interval colouring of weighted graphs

We will denote by $G=(V, E, \mathbf{c})$ an undirected finite simple graph, where V is an non-empty set of vertices, E is a set of edges, and $\boldsymbol{c}$ is a vector containing the weights. Each number c_i of $\boldsymbol{c}$ is a positive integer associated with v_i of V. The c_i are called the *vertex chromaticities*.

A graph G has an *interval k-colouring* if c_i distinct and consecutive integers — colours — from the set $\{1, 2, \ldots, k\}$ are assigned to each v_i in such a way that no two adjacent vertices have a colour in common — a proper colouring.

An example of a weighted graph and an interval 7-colouring are given in Figure 1 and Table 1, respectively.

The *interval chromatic number* $\chi(G)$ is the smallest number k such that G has an interval k-colouring. The corresponding colouring is then called *optimal*.

In this article we denote by the chromatic number and colouring the interval version unless otherwise stated.

The colours assigned to a vertex are consecutive. Therefore, the colouring of G can be reduced to the determination of the first colour for each vertex: the *initial colour*.

It should be noted that, in the case when all the vertex chromaticities of G are equal — such a G is called a non-composite graph — the problem of finding $\chi(G)$ is equivalent to the well-known chromatic number problem. In this problem only one colour is associated with each vertex of the graph (Randall Brown, 1972; Sakaki et al., 1976; Brélaz, 1979; Krarup and de Werra, 1982; Kubale and Jackowski, 1985). In this case determining the chromatic number is known to be NP-complete for $k \geqslant 3$ (Karp, 1972).

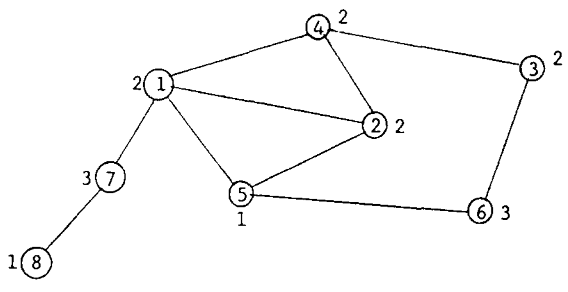

Figure 1. Weighted graph

Table 1
A colouring of the graph

Vertex	Chromaticity	Colours
1	2	5,6
2	2	3,4
3	2	4,5
4	2	1,2
5	1	7
6	3	1,2,3
7	3	1,2,3
8	1	4

Therefore, in general the determination of $\chi(G)$ is a NP-hard problem. The consequence is that a polynomial time algorithm for its solution is highly unlikely to exist. For this reason there are only heuristic algorithms developed for solving this problem.

3. Exact algorithm for chromatic number

There are a lot of exact algorithms for determining the value of $\chi(G)$ for non-composite graphs. Most of them are based on the implicit enumeration — backtracking — approach (Kubale and Jackowski, 1985).

Starting from the basic principles of such algorithms, we developed an exact algorithm for determining the chromatic number of a weighted graph. The algorithm consists of an enumeration procedure based on the branch-and-bound principle.

3.1. Reduction of dimensions

Sometimes the dimensions of G can be easily reduced as follows.

Let the Largest Common Divisor (LCD) of all c_i be greater than 1 and G' be a graph obtained from G by dividing each c_i through LCD. It can be proved that $\chi(G) = \text{LCD} \times \chi(G')$ and there is a connection between the optimal colouring of G and G' (Čangalović, 1989). Therefore, the determination of $\chi(G)$ can be reduced to a smaller problem: finding $\chi(G')$.

3.2. Vertex ordering

In order to determine an approximate value of $\chi(G)$, Clementson and Elphick have used several vertex orderings. According to their experiments the Largest First by chromaticity (LF) ordering is the most effective. Let $v_1, v_2, \ldots, v_n$ be a sequence of V ordered according to decreasing c_i. Let δ_i be the value defined as

$$\delta_i = \sum_{j:\ v_j \in S_i} c_j + c_i,$$

where S_i is the set of all the vertices adjacent to the vertex v_i. For those vertices where $c_i = c_j$, we suborder v_i such that $\delta_i > \delta_j$.

This LF vertex ordering is used in our algorithm. The order is fixed and it is not changed during the colouring process.

In the example from Section 2 LF gives 6, 7, 1, 4, 2, 3, 5, 8.

We mention that de Werra and Hertz have defined a so called Smallest Last vertex ordering (SL) for weighted graphs.

The average behaviour of our algorithm for both orderings has been investigated (Čangalović, 1989). The CPU-time for the determination of the lower and upper bound (see Sections 3.3 and 3.4) with SL was slightly better than with LF. In the enumeration part, however, there was no overall advantage for SL. Because LF is easier to implement, we choose that one.

3.3. Lower bound chromatic number

A lower bound *LBOUND* for $\chi(G)$ is defined as

$$\text{LBOUND} = \max_{H} \sum_{i:\ v_i \in H} c_i,$$

where the maximum in the above expression is established over all the cliques, i.e. maximal complete subgraphs H of G.

In our algorithm, LBOUND is determined by an exact procedure which represents a generalized version of a method suggested by Sakaki. The procedure is based on the branch-and-bound principle taking into account the LF-ordering (Čangalović, 1989).

For the example of Section 2 LBOUND is determined by the clique 1-2-4: LBOUND = 6.

3.4. Upper bound chromatic number

Clementson and Elphick proposed a so called interchange sequential colouring heuristic for finding $\chi(G)$. The heuristic colours the vertices of G successively according to a predestined order. It tries to associate with each vertex the smallest possible initial colour. An interchange procedure is used which allows to change colours for already coloured vertices in order to decrease the total number of used colours. The heuristic with the LF ordering (LFI) showed the best average behaviour.

We used LFI in our algorithm in order to determine an approximate value *APPROX* of $\chi(G)$ and the corresponding colouring of the graph.

If APPROX = LBOUND, the $\chi(G)$ = LBOUND. Otherwise, APPROX gives an initial value for the upper bound UBOUND of $\chi(G)$.

For the example of Section 2, APPROX = 7 and the corresponding colouring is given in Table 1 above.

3.5. Initial partial colouring

The following two propositions can be posed (Čangalović, 1989).

Proposition 1. *Let K be a complete subgraph of G which contains all v_i with $c_i > 1$ and let* **p** *be a proper colouring of K with all the colours from the set*

$$\left\{1, 2, \ldots, \sum_{i:\ v_i \in K} c_i\right\}.$$

Then there exists an optimal colouring of G which contains **p**.

Proposition 2. *If G is k-colourable, then there exists a k-colouring of G in which the initial colour of a vertex v_i belongs to the set*

$$\left\{1, 2, \ldots, \left\lceil \frac{k - c_i + 1}{2} \right\rceil\right\}.$$

$\lceil x \rceil$ is the smallest integer not smaller than x.

Starting from the above mentioned propositions, the following procedure of our algorithm can be defined.

First the procedure tries to find a clique of G which contains all the vertices v_i with $c_i > 1$. If such a clique exists, its vertices are properly coloured by consecutive colours starting from 1. All remaining vertices are ordered by the LF ordering. According to Proposition 1 the obtained partial colouring is not changed in further steps of the algorithm.

If such a clique does not exists, all the vertices of G are ordered by the LF ordering. Then, the set of feasible initial colours for the first vertex v_1 — with respect to UBOUND — is determined. According to Proposition 2, this set is equal to

$$\left\{1, 2, \ldots, \left\lceil \frac{\text{UBOUND} - c_1}{2} \right\rceil \right\}. \tag{1}$$

In further steps of the algorithm, the branching at v_1 is performed only over the set (1). Finally, v_1 is coloured by the initial colour 1.

The described procedure is called INITIAL. In our algorithm it is applied before any enumerative procedure.

In our example a clique which contains all the vertices with $c_i > 1$ does not exist. The vertices are LF ordered. The set of feasible initial colours for the first vertex 6 is equal to $\{1, 2\}$.

3.6. *Procedure* FORWARDS

Starting from some sequence of already coloured vertices, this procedure colours the remaining vertices of the graph sequentially, according to the LF ordering with the colours less than the current UBOUND.

At each step of FORWARDS the sets of feasible initial colours for all the still uncoloured vertices are determined. Then, the first uncoloured vertex is coloured with the smallest feasible initial colour.

Let the first m vertices be coloured with a total number of colours less than UBOUND. Let F_i be the set of feasible initial colours for some still uncoloured vertex v_i, $i \in \{m+1, \ldots, n\}$. Then F_i can be defined as the largest set which satisfies the following conditions:

(a) $F_i \subseteq \{1, 2, \ldots, \text{UBOUND} - c_i\}$.

(b) For any $a \in F_i$ and any $j \in \{1, 2, \ldots, m\}$ such that the vertices v_i and v_j are adjacent, the initial colour a is not in conflict with the initial colour of the already coloured vertex v_j.

(c) For any $a \in F_i$ and any $j \in \{m+1, \ldots, n\}$, $j \neq i$, such that v_i and v_j are adjacent, there exists a $b \in F_j$ such that a and b are not in conflict. Note that the initial colour a of v_i and b of v_j are not in conflict if

$$\{a, a+1, \ldots, a+c_i-1\} \cap \{b, b+1, \ldots, b+c_j-1\} = \emptyset.$$

FORWARDS terminates when either there exists an uncoloured vertex with an empty set of feasible initial colours or all the vertices have been coloured. In the last case a new complete colouring with the total number of colours COLNUM less than UBOUND has been found. Then a new value of UBOUND is COLNUM. If UBOUND = LBOUND, then $\chi(G) =$ LBOUND.

3.7. *Procedure* BACKWARDS

BACKWARDS goes sequentially back along the path constructed by the last FORWARDS. It stops at one of the already coloured vertices changing its initial colour.

Let v_m be the last vertex coloured in FORWARDS. While taking into account the LF ordering, BACKWARDS looks for the largest i, $i \in \{1, 2, \ldots, m\}$, for which v_i satisfies the following conditions:

(a) It does not belong to the clique (suppose there exists one) determined by INITIAL.

(b) The set of still unutilized feasible initial colours of v_i is non-empty.

(c) If v_i was coloured with the smallest colour from this set, then the corresponding partial colouring of $v_1, v_2, \ldots, v_i$ would have colours less than UBOUND.

(d) A recolouring of v_i might have an influence on a colouring of $v_{i+1}, \ldots, v_n$. The algorithm contains procedures to such an extent.

Exists such an i, then v_i is coloured with the smallest unutilized feasible initial colour and FORWARDS proceeds again. Otherwise, $i = 0$ and $\chi(G) =$ UBOUND.

In Figure 2 a pseudo-code of Pascal represents a reflection of all the procedures in the algorithm CHROMA. The enumeration tree for determining $\chi(G)$ by CHROMA is shown in Figure 3.

```
program CHROMA
begin
   reduce G to G';
   find LBOUND;
   find APPROX;
   if APPROX = LBOUND
      then χ(G') := LBOUND
      else begin (*enumeration*)
         UBOUND := APPROX;
         INITIAL;
         eoe := false;(*eoe = end of the enumeration*)
         while eoe = false do
            begin
               FORWARDS;
               if UBOUND = LBOUND
                  then eoe := true
                  else begin (*backtracking*)
                     BACKWARDS; (*to the vertex v_i *)
                     if i = 0 then eoe := true
                     end (*backtracking*)
               end;
            χ(G') := UBOUND
         end; (*enumeration*)
   χ(G) := LCD*χ(G')
end.
```

Figure 2. Pseudo-code of CHROMA

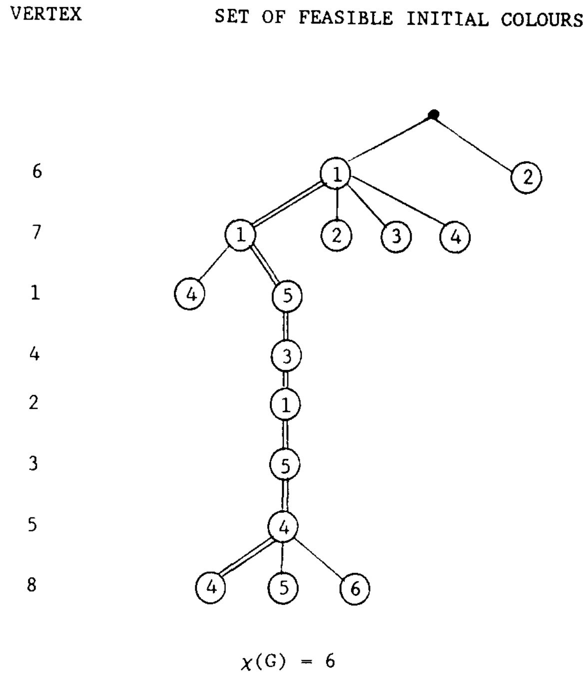

Figure. 3. Enumeration tree for $\chi(G)$

4. Numerical experiences

Random weighted graphs are used in order to investigate the average behaviour of the algorithm and to test its efficiency. Also the tightness of the bounds LBOUND and UBOUND can be approached.

A random weighted graph is defined as a graph on n vertices in which each of the possible edges occurs independently with a probability p (density of the graph), $0 < p < 1$.

The chromaticity of each vertex v_i is given by an independent truncated Poisson random variable with parameter q, $q > 0$, i.e.

$$P\{c_i = m\} = \frac{q^m}{(e^q - 1) \cdot m!}, \quad m = 0, 1, 2, \ldots$$

(Clementson and Elphick, 1983).

A Pascal program has been written which, for the triplet (n, p, q), generates five random weighted graphs and applies CHROMA to each of them. Further, the program calculates the minimal, the maximal and the average CPU-time of the algorithm (in seconds), the average number of backtrackings and the average value of χ-LBOUND and APPROX-χ.

We considered 45 groups for inputdata (n, p, q) in which $n = 20(5)40$, $p = 0.2$, 0.3, 0.4 and $q = 1.0$, 1.5, 2.0 (all in all 225 random weighted graphs).

The results obtained on a VAX11/750 are summarized in Table 2. For all the groups the average CPU-time for LBOUND and APPROX was less than 1 second. Only the times for the enumeration procedure are mentioned.

For greater values of n and p the difference between the maximal and minimal values increased. Therefore, the behaviour of CHROMA is much more difficult to predict in these cases. If the maximal time is given by *** it means that for the corresponding group of data there was at least one random graph with CPU-time larger than 10 minutes (time limit). For such graphs (4%) χ was determined as the last value of UBOUND.

For some groups of data a jump in the average CPU-time and the number of backtrackings can be noticed — for example $n = 30$. The reason is a large gap between χ and APPROX.

The values χ-LBOUND and APPROX-χ increase with larger p and n. For 87% of the groups χ-LBOUND < 1. Even better, for 78% χ-LBOUND < 0.5. It means that LBOUND, determined as in Section 3.3, represents a good lower

Table 2
Computational results of CHROMA to random graphs

P	q	Number of test	Number of vertices	Range in CPV-time	Average CPV-time	Average number of back-tracking	χ-LBOUND	APPROX-χ
0.2	1.0	5	20	0.00– 0.00	0.00	0.00	0.00	0.00
			25	0.00– 0.12	0.02	0.80	0.00	0.20
			30	0.00– 0.19	0.04	0.80	0.20	0.20
			35	0.00– 1.39	0.30	8.00	0.20	0.40
			40	0.00– 1.83	0.41	9.00	0.20	0.60
0.2	1.5	5	20	0.00– 0.03	0.01	0.40	0.20	0.00
			25	0.00– 0.09	0.02	0.20	0.00	0.20
			30	0.00– 9.11	1.91	88.80	0.40	1.00
			35	0.00– 0.69	0.28	2.60	0.00	1.00
			40	0.00–440.60	89.36	2618.40	0.00	1.00
0.2	2.0	5	20	0.00– 0.10	0.03	0.80	0.00	0.60
			25	0.00– 0.17	0.07	1.00	0.20	0.60
			30	0.00– 0.16	0.05	0.00	0.00	0.40
			35	0.00– 1.56	0.74	13.60	0.60	1.20
			40	0.30– 38.75	10.48	202.80	0.20	1.80
0.3	1.0	5	20	0.00– 0.06	0.03	0.20	0.00	0.60
			25	0.00– 0.20	0.10	3.00	0.20	0.80
			30	0.00– 10.77	2.46	140.40	0.00	0.80
			35	0.29– 2.99	1.35	32.00	0.40	1.20
			40	2.01– * * *	123.80	3822.40	0.40	1.00
0.3	1.5	5	20	0.00– 0.08	0.02	0.20	0.00	0.20
			25	0.00– 0.11	0.04	0.60	0.00	0.40
			30	0.00–230.86	52.86	1928.80	0.40	1.20
			35	0.00– 12.09	3.19	56.00	0.40	1.40
			40	0.00– * * *	127.91	3108.00	0.60	0.60
0.3	2.0	5	20	0.00– 0.81	0.23	10.80	0.00	1.00
			25	0.00– 5.47	2.59	89.00	0.20	1.00
			30	0.00– 55.93	12.33	297.20	0.20	1.40
			35	0.27– 15.44	6.64	112.40	0.80	1.80
			40	0.41– * * *	121.57	2817.60	0.40	1.60
0.4	1.0	5	20	0.00– 7.63	1.70	118.40	0.60	0.40
			25	0.00– 2.02	0.64	24.80	0.20	0.40
			30	0.00– * * *	122.82	4604.00	0.40	0.60
			35	0.45– * * *	132.39	3105.00	1.20	1.40
			40	0.35– * * *	145.76	2502.06	1.20	2.00
0.4	1.5	5	20	0.00– 3.08	0.72	72.60	0.00	0.80
			25	0.00– 0.69	0.14	4.40	0.20	0.20
			30	0.00– 11.73	4.24	121.60	0.20	0.80
			35	0.00– * * *	159.59	3689.00	0.80	1.60
			40	0.60–149.51	57.25	957.20	0.80	1.40
0.4	2.0	5	20	0.00– 2.34	0.50	24.80	0.00	0.60
			25	0.20– 4.07	2.03	63.00	0.40	1.20
			30	0.00– 85.42	21.02	504.20	0.80	1.00
			35	0.00– 29.03	14.17	253.40	0.40	1.20
			40	19.48– * * *	362.07	5678.60	1.80	2.00

approximation of χ. However, APPROX is not so good: for 49% of the groups APPROX-$\chi > 1$.

For 35% of the graphs LBOUND = APPROX. So, the enumeration procedure was not needed.

For 40% of the graphs χ = LBOUND, but only for 3% χ = APPROX.

The number of graphs for which LBOUND < χ < APPROX increases with greater p and n, and equals 18%.

The variations in q does not seem to have any influence on these results.

Table 3
Timetabling problem

Class	Teacher	Length of lecture
I	N_1	4
	N_3	3
II	N_1	2
	N_4	2
III	N_1	1
	N_3	2
	N_5	1

5. Application to timetabling problems

5.1. Problem definition

In the Operational Research literature (de Werra, 1985; Clementson and Elphick, 1983) the timetabling problem is described as follows:

– A set of classes who follow a fixed curriculum.

– A set of teachers who give subjects to these classes.

– A set of time periods.

– A set of lectures which consists of specific combinations of teachers and classes. Each lecture has a length which expresses the number of schoolhours or periods required for its completion.

The problem is to find a feasible timetable such that all the lectures are assigned to the given set of periods and each teacher or class has at most one lecture at a time. If there are no other conditions involved, such a timetable can be constructed as an edge-colouring of a bipartite multigraph (Schreuder and van der Velde, 1984).

In our problem the lectures can have different lengths and they should not be interrupted: the hours of one lecture are consecutive. Each teacher can teach in more than one class. The set of time periods consists of consecutive schoolhours for one day.

Let L_{ij} be the sum of the lengths of all those lectures which a teacher j gives to a class i. Then a value LT can be defined as

$$\mathrm{LT} = \max\left\{ \max_i \sum_j L_{ij},\ \max_i \sum_i L_{ij} \right\},$$

where the maxima and the sums are established over all classes i and all teachers j. Obviously, the necessary condition for the existence of a feasible daily timetable is that the total number of periods per day (TP) is not smaller than LT.

With the daily timetable we can associate a weighted graph G. Each lecture is represented as a vertex with a weight equal to the length of the lecture. Two vertices are connected with an edge (adjacent) if the corresponding lectures have either a class or a teacher in common. G always represents the edge-graph of a bipartite graph. Finding a feasible daily timetable can be reduced to a colouring of G with no more than TP colours — a TP-feasible colouring. The initial colour of a vertex represents the initial schoolhour for the corresponding lecture.

A daily timetabling problem with 3 classes, 5 teachers and 7 consecutive schoolhours per day is given in Table 3. The associated weighted graph is shown in Figure 4.

5.2. Minimal duration timetable

The determination of a feasible daily timetable with minimal time duration can be reduced to finding $\chi(G)$. If $\chi(G) > \mathrm{TP}$, then a feasible daily

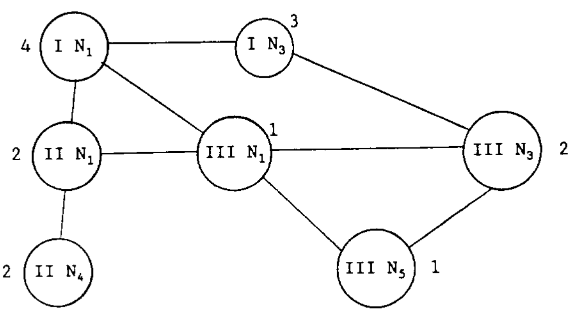

Figure. 4. Associated weighted graph

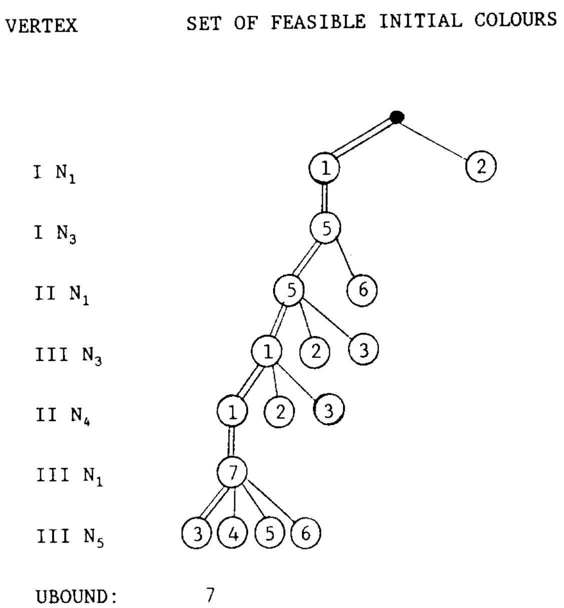

Figure. 5. Minimal duration timetable

timetable does not exist. Otherwise, a minimal duration timetable is determined by an optimal colouring of G. Therefore, if $\mathrm{LT} \leqslant \mathrm{TP}$ such a timetable can be found by applying CHROMA to G.

However, to be more efficient, CHROMA should be slightly modified as follows.

After the reduction of dimensions (see Section 3.1), instead of using the procedure mentioned in Section 3.3, LBOUND is determined as LT/LCD.

An initial value of UBOUND equals

$$\min\left\{\mathrm{APPROX}, \left[\frac{\mathrm{TP}}{\mathrm{LCD}}\right] + 1\right\},$$

($[x]$ is the integer part of x), where APPROX is calculated as in Section 3.4.

If BACKWARDS does not halt at any vertex, then

- if $\mathrm{UBOUND} = [\mathrm{TP}/\mathrm{LCD}] + 1$, a feasible timetable does not exist;
- otherwise, a minimal duration timetable is defined.

All the other steps of CHROMA remain untouched.

For our example, $\mathrm{TP} = \mathrm{LT} = 7$. The enumeration tree, obtained by the modified algorithm, is represented in Figure 5. A minimal duration timetable has been denoted by double lines.

5.3. Minimal free time timetable

If $\sum_j L_{ij} < \mathrm{TP}$ for at least one class i, then for each such a class there could exist free periods FT_i for a class i in a feasible daily timetable is equal to

$$\mathrm{FT}_i = z_i + 1 - p_i - \sum_j L_{ij},$$

where p_i and z_i are respectively the initial and the final schoolhour for class i. The problem is to find a feasible daily timetable for which the function

$$\max_i \mathrm{FT}_i$$

is minimum — a so called minimal free time timetable.

The determination of such a timetable can be reduced to a colouring problem as follows.

Let G_i be a complete subgraph of G determined by all the vertices corresponding to a class i. For a colouring of G we define its *range* as the value equal to

$$\max_i \left\{ \max_{j:\, v_j \in G_i} (f_j + c_j) - \min_{j:\, v_j \in G_i} f_j - \sum_{j:\, v_j \in G_i} c_j \right\},$$

where f_j is the initial colour of v_j in the colouring.

Now the timetable can be reduced to finding a TP-feasible colouring of G with minimal range. Proposition 2 from Section 3 can be extended to such a colouring problem (Čangalović, 1989).

Proposition 3. *If there exists a* TP-*feasible colouring of* G_i, *then there exists a* TP-*feasible colouring of* G *with the same range in which the initial colour of a vertex* v_i *belongs to the set*

$$\left\{1, 2, \ldots, \left\lceil \frac{\mathrm{TP} - c_i + 1}{2} \right\rceil \right\}.$$

With the free time timetable we can not use the reduction of dimensions and Proposition 1.

The colouring problem which is the result of the transformation of the free time timetable can be solved by a modification of CHROMA called MINFT, see Figure 6.

In INITIAL all the vertices of G are ordered by the LF ordering. Then, according to Proposition 3, the set of feasible initial colours for v_1 is determined as

$$\left\{1, 2, \ldots, \left\lceil \frac{\mathrm{TP} - c_1 + 1}{2} \right\rceil \right\}.$$

As $F_i \leqslant \mathrm{TP} - 2$ for each class i, then $\mathrm{TP} - 1$ is an initial value of the upper bound *UBRAN* for the range of a colouring.

```
program MINFT;
begin
   INITIAL;
   UBRAN := TP - 1
   eoe := false;
   while eoe = false do
      begin
         FORWARDS;
         if UBRAN = 0
            then eoe := true
            else begin
               BACKWARDS;
               if i = 0 then eoe := true
               end
      end;
   if UBRAN < TP
      then min free time timetable is obtained
      else a feasible timetable does not exist
end.
```

Figure 6. Pseudo-code of MINFT

The procedure FORWARDS used here differs from the one in Section 3.6 in several ways.

The set F_i of the feasible initial colours for v_i is determined as the largest set for which $F_i \subseteq \{1, 2, \ldots, TP - c_i + 1\}$ and the conditions (b) and (c) of Section 3.6 are satisfied.

The initial colour a of v_s and b of v_t are not in conflict if they satisfy the condition from Section 3.6 and the following additional condition. If v_s, $v_t \in G_i$, then

$$\max\{a + c_s, b + c_t\} - \min\{a, b\} - \sum_{j:\ v_j \in G_i} c_j < \text{UBRAN}.$$

If the first uncoloured vertex is adjacent to at least one of the vertices next in the ordering, then it is coloured with the smallest feasible initial colour. Otherwise, the vertex is coloured with a feasible initial colour for which the corresponding partial colouring has the minimal range (see also the remark at the end of this section).

If in FORWARDS a new complete colouring with the range RAN is found, then UBRAN = RAN. If UBRAN = 0, then the colouring represents a minimal free time timetable.

BACKWARDS looks for the largest i for which v_i satisfies the conditions (b) and (d) of Section 3.7 and the following condition. If v_i was coloured with the smallest unutilized initial colour, then the corresponding partial colouring of $v_1, v_2, \ldots, v_i$ would have a range less than UBRAN. If such a vertex does not exist, then

– for UBRAN = TP, a feasible daily timetable does not exist;

– otherwise, a minimal free time timetable is obtained.

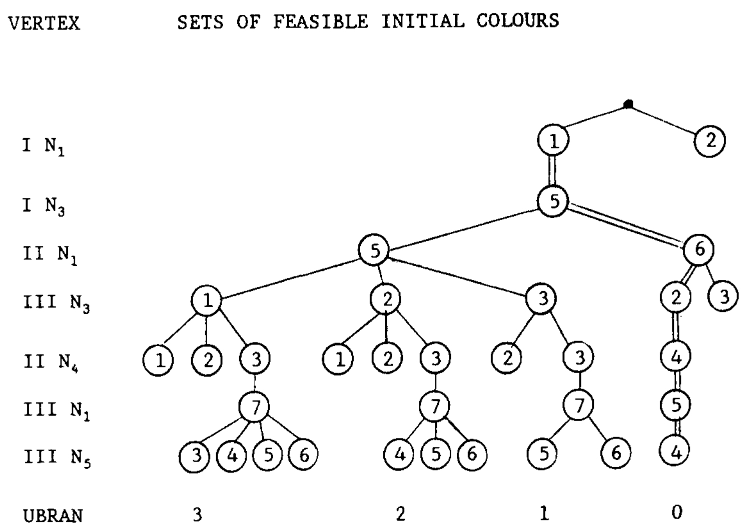

Figure 7. Minimal free time timetable

Table 4
Computational results of CHROMA to random graphs representing timetabling problems

p	l	Number of test	Number of vertices	Range in CPV-time	Average CPV-time	Average number of backtracking
0.2	0.02	5	20	0.00– 0.01	0.00	0.00
			30	0.00– 0.15	0.04	0.60
0.3	0.02	5	20	0.00– 0.09	0.03	1.80
			30	0.00– 3.03	1.03	27.20
0.4	0.02	5	20	0.00– 0.03	0.02	0.00
			30	0.13–15.86	4.60	170.40

In Figure 7 the enumeration tree for finding such a timetable for our example by MINFT, is represented. The timetable is denoted by the double lines.

We mention that during MINFT the vertex v_5 (representing class II and teacher N_4) was not adjacent to any vertex next in the ordering. Therefore, it was coloured with colour 3 instead of its smallest unutilized colour. In this case, the range of the partial colouring of $v_1, v_2, \ldots, v_5$ was 0. If v_5 was coloured with 1 or 2, the range would be greater than 0.

6. Conclusions

The problem of finding the chromatic number of a weighted graph is NP-hard. Therefore, an exact algorithm based on implicit enumeration has been proposed in Section 3. The algorithm has exponential time complexity, and, consequently, it is applicable only to graphs with a rather small number of vertices, density and chromaticities. Fortunately, there are real-world problems of such a size which can be reduced to a colouring problem of a weighted graph of proper dimensions.

For example, the associated weighted graph of the daily timetable problems of Section 5 usually consists of several isolated subgraphs. Some of them are complete and, therefore, trivial to colour. The remaining ones have 20 to 30 nodes with weights not more than 3 ($q = 0.02$) and with a rather low density ($p \leqslant 0.4$). In these cases CHROMA and its modifications can be successfully applied as demonstrated in Table 4.

Further, we mention that the translation of the minimal free time timetable problem to an interval colouring as defined in Section 2, could simplify MINFT. However, we think that without adding additional constraints like in Section 5.3, it can not be realized.

Acknowledgement

This research was established during the first author's stay at the Department of Applied Mathematics, University of Twente. The authors would like to thank their colleagues in Twente for their stimulating discussions and helpful remarks.

References

Aarts, E.H.L., and Laarhoven, P.J.M. van (1989), "Simulated annealing: An introduction", *Statistica Neerlandica* 43/1, 31–52.

Brélaz, D. (1979), "New methods to color the vertices of a graph", *Communications of the ACM* 22, 251–256.

Čangalović, M. (1989), "Some new combinatorial optimization algorithms applied to timetabling problems" (in Serbo-Croatian), Ph.D. thesis, University of Belgrade, Yugoslavia.

Clementson, A.T., and Elphick, C.H. (1983), "Approximate colouring algorithms for composite graphs", *Journal of Operational Research Society* 34/6, 503–509.

Eiselt, H.A., and Laporte, G. (1987), "Combinatorial optimization problems with soft and hard requirements", *Journal of Operation Research Society* 38/9, 785–795.

Golumbic, M. (1980), *Algorithmic Graph Theory and Perfect Graphs*, Academic Press, New York, 203–207.

Karp, R.M. (1972), "Reducibility among combinatorial problems", in: R.E. Miller and J.W. Thatcher (eds.), *Complexity of Computer Computations*, Plenum, New York.

Krarup, J., and Werra, D. de (1982), "Chromatic optimisation: Limitations, objectives, uses, references", *European Journal of Operational Research* 11, 1–19.

Kubale, M., and Jackowski, B. (1985), "A generalized implicit enumeration algorithm for graph colouring", *Communications of the ACM* 28, 412–418.

Punter, A. (1976), "Systems for timetabling by computer based on graph colouring", Ph.D. Thesis, C.N.A.A., Hatfield Polytechnic.

Randall Brown, J. (1972), "Chromatic scheduling and the chromatic number problem", *Management Science* 19, 456–463.

Sakaki, T., Nakashima, K., and Hattori, Y. (1976), "Algorithms for finding in the lump both bounds of the chromatic number of a graph", *The Computer Journal* 19, 329–332.

Schreuder, J.A.M., and Velde, J.A. van der (1984), "Timetables in Dutch High Schools", in: J.P. Brans (ed.), *Operational Research '84*, North-Holland, Amsterdam, 601–612.

Werra, D. de (1985), "Introduction to timetabling", *European Journal of Operational Research* 19, 151–162.

Werra, D. de, and Hertz, A. (1988), "Consecutive colourings of graphs", *Zeitschrift für Operations Research* 32/1, 1–8.