Invited Review

An introduction to timetabling

D. de WERRA
Swiss Federal Institute of Technology, Department of Mathematics, CH-1015, Lausanne, Switzerland

Abstract. A huge variety of timetabling models have been described in the OR literature; they range from the weekly timetable of a school to the scheduling of courses or exams in a university. Graphs and networks have proven to be useful in the formulation and solution of such problems. Various models will be described with an emphasis on graph theoretical models.

Keywords: Timetabling, education networks, graphs, heuristics

1. Introduction

During the last twenty years many contributions related to timetabling have appeared and it will probably continue with the same rate for years. One reason for this may be the huge variety of problems which are included in the field of timetabling; another reason is the fact that educational methods are changing, so that the models have to be modified. Finally, computing facilities are now available in most schools and, as a consequence, the approach to timetabling has to take this phenomenon into account. This means in particular that interactive methods are now becoming more important and, furthermore, they have to be adapted to microcomputers.

Usually one considers the timetabling process that consists of 2 distinct phases [7]:

a) First, the curricula are defined for each class or for each group of students and one has to assign the various resources (in manpower or in equipment) to the classes.

b) Second, when an agreement has been reached concerning these assignments of resources, then one tries to see whether a workable detailed timetable can be worked out which is compatible with all the previously defined requirements.

Generally the role of computers is to handle the second phase and we shall concentrate on this type of problem. However, one should mention that some attention has also been given to the first phase, both from a theoretical and from a computational point of view [15,20].

In this survey we cannot describe all special types of timetables which have been mentioned up to now. We shall refer the reader to the annotated bibliography [29] which covers the contributions which appeared before 1980 and to general surveys [5,7,18].

Our purpose will be to introduce the reader to some basic problems which occur in most of the real cases; we shall first describe the class-teacher problem and its variations. Then we will discuss the course scheduling model together with some closely related problems. For both types of timetabling problems, we shall mainly concentrate on models based on graphs and networks. As observed by Mulvey [24], such models are interesting, since network problems are efficiently solved even when their size is large; although, generally, real timetabling problems cannot be formulated with pure network models, in the man-machine iterative solution procedures pure network problems occur at various stages. One can therefore take advantage of the fact that they are relatively easy to handle to devise fast heuristic methods for timetabling.

Another argument for justifying our interest in models for simpel timetabling problems is the fact that heuristic methods (for handling real timetabling problems) are often derived and adapted

Received August 1984

North-Holland
European Journal of Operational Research 19 (1985) 151–162

from exact methods developed for the simple cases [30].

In this text we shall assume that the reader has some very elementary knowledge of graph-theoretical concepts. All terms related to graphs and networks which are not defined here can be found in [1,11].

2. The class–teacher model

2.1. A simple model

We first describe the basic class–teacher model without including all constraints which are usually present in real cases.

A *class* will consist of a set of students who follow exactly the same program. Let $C = \{c_1,\dots,c_m\}$ be a set of classes and $T = \{t_1,\dots,t_n\}$ a set of teachers. We are given an $m \times n$ requirement matrix $R = (r_{ij})$ where r_{ij} is the number of lectures involving class c_i and teacher t_j.

We shall assume that all lectures have the same duration (say one period). Given a set of p periods, the probelm is to assign each lecture to some period in such a way that no teacher (resp. no class) is involved in more than one lecture at a time. More precisely, if we define x_{ijk} to be 1 if class c_i and teacher t_j meet at period k and 0 otherwise, we have to solve problem CT1:

$$\sum_{k=1}^{p} x_{ijk} = r_{ij} \quad (i = 1,\dots,m; j = 1,\dots,n), \tag{1}$$

$$\sum_{j=1}^{n} x_{ijk} \leqslant 1 \quad (i = 1,\dots,m; k = 1,\dots,p), \tag{2}$$

$$\sum_{i=1}^{m} x_{ijk} \leqslant 1 \quad (j = 1,\dots,n; k = 1,\dots,p), \tag{3}$$

$$x_{ijk} = 0 \text{ or } 1 \quad (i = 1,\dots,m; j = 1,\dots,n; \; k = 1,\dots,p), \tag{4}$$

With this formulation we may associate a bipartite multigraph $G = (C, T, \hat{R})$: Its nodes are the classes and the teachers; node c_i and node t_j are linked by r_{ij} parallel edges. If each period corresponds to a color, the problem consists in finding an assignment of one among p colors to each edge of G in such a way that no two adjacent edges have the same color; so x_{ijk} will be 1 if some edge $[c_i, t_j]$ gets color k.

Proposition 2.1. *There exists a solution to* CT1 *iff*

$$\sum_{i=1}^{m} r_{ij} \leqslant p \quad (j = 1,\dots,n),$$

$$\sum_{j=1}^{n} r_{ij} \leqslant p \quad (i = 1,\dots,m).$$

This is the well-known theorem of König on edge colorings in bipartite multigraphs [1]; it says that there is a timetable in p periods if and only if no teacher (no class) is involved in more than p lectures.

Up to here we have made no distinction between the daily and weekly scheduling problem. In the daily problem, one has to assign each lecture to some hour of the day and R represents all lectures which have to be scheduled on one day.

In the weekly scheduling problem each lecture has to be assigned to some day of the week and R represents all lectures which must take place during the week. In such a situation for each class c_i (resp. teacher t_j) we have a positive integer a_i (resp. b_j) representing the maximum number of lectures in which c_i (resp. t_j) may be involved during each one of the p days. The assignment to days is now formulated as follows (x_{ijk} will be the number of lectures involving c_i and t_j, which are assigned to day k): Find values x_{ijk} satisfying (1) and problem CT2:

$$\sum_{j=1}^{n} x_{ijk} \leqslant a_i \quad (i = 1,\dots,m; k = 1,\dots,p), \tag{5}$$

$$\sum_{i=1}^{m} x_{ijk} \leqslant b_j \quad (j = 1,\dots,n; k = 1,\dots,p), \tag{6}$$

$$x_{ijk} \geqslant 0 \quad \text{integer}, \quad (i = 1,\dots,m, j = 1,\dots,n, \; k = 1,\dots,p). \tag{7}$$

Proposition 2.2 [38]. *There exists a solution to* CT2 *iff*

$$\sum_{i=1}^{m} r_{ij} \leqslant pb_j \quad (j = 1,\dots,n),$$

$$\sum_{j=1}^{n} r_{ij} \leqslant pa_i \quad (i = 1,\dots,m).$$

This problem also has an immediate formulation in terms of edge coloring in a bipartite multigraph: We have to assign one among p colors to each edge in such a way that no more than a_i

(resp. b_j) edges adjacent to node c_i (resp. t_j) have the same color.

Clearly, if a_i and b_j are given, then the minimum number of days p needed is given by

$$p = \max\left(\max_j \left\lceil \sum_{i=1}^{m} r_{ij}/b_j \right\rceil, \max_i \left\lceil \sum_{j=1}^{n} r_{ij}/a_i \right\rceil \right)$$

where $\lceil t \rceil$ is the smallest intever not less than t.

In most cases however, the number of days p is given and one wants to spread the lectures involving the same class c_i and the same teacher t_j as uniformly as possible throughout the p days. The formulation is then the following: Find integer values x_{ijk} satisfying (1) and problem CT3:

$$\left\lfloor \sum_{j=1}^{n} r_{ij}/p \right\rfloor \leqslant \sum_{j=1}^{n} x_{ijk} \leqslant \left\lceil \sum_{j=1}^{n} r_{ij}/p \right\rceil$$
$$(i = 1,\ldots,m,\ k = 1,\ldots,p), \tag{8}$$

$$\left\lfloor \sum_{i=1}^{m} r_{ij}/p \right\rfloor \leqslant \sum_{i=1}^{m} x_{ijk} \leqslant \left\lceil \sum_{i=1}^{m} r_{ij}/p \right\rceil$$
$$(j = 1,\ldots,n,\ k = 1,\ldots,p), \tag{9}$$

$$\lfloor r_{ij}/p \rfloor \leqslant x_{ijk} \leqslant \lceil r_{ij}/p \rceil$$
$$(i = 1,\ldots,m, j = 1,\ldots,n,\ k = 1,\ldots,p), \tag{10}$$

Here $\lfloor t \rfloor$ is the largest integer not larger than t. Constraints (8) and (9) simply express that the daily loads of all classes and of all teachers are prefectly balanced over the p days.

Proposition 2.3 [37]. *There exists a solution to* CT3 *for any p.*

This result is an extension of Proposition 2.1 and 2.2. It also has an obvious interpretation in terms of edge coloring in a bipartite multigraph.

Condition (10) expresses the fact that the lectures involving c_i and t_j are spread throughout the p days: In the family of so called parallel edges linking c_i and t_j there are at least $\lfloor r_{ij}/p \rfloor$ and at most $\lceil r_{ij}/p \rceil$ edges of each color.

Such an edge coloring can be found by network flow techniques as we shall now sketch [19].

In G each edge $[c_i, t_j]$ is oriented from c_i to t_j and becomes an *arc* (c_i, t_j); we introduce a node s and arcs (s, c_i) for each c_i as well as a node t and arcs (t_j, t) for each t_j. We have to solve p problems of compatible flow in this network; at step k we construct a flow which will give the numbers x_{ijk} for all c_i and t_j. Each arc (x, y) receives a lower bound $l^k(x, y)$ which is large enough to make sure that the left inequalities in (8)–(10) are satisfied and also that it will be possible to schedule the remaining lectures in the last $k - p$ days without violating the right inequalities in (8)–(10). With similar considerations we determine a capacity $c^k(x, y)$ for each arc (x, y).

An example of a weekly timetable (with $p = 3$ days) is given in Figure 1. It shows the construc-

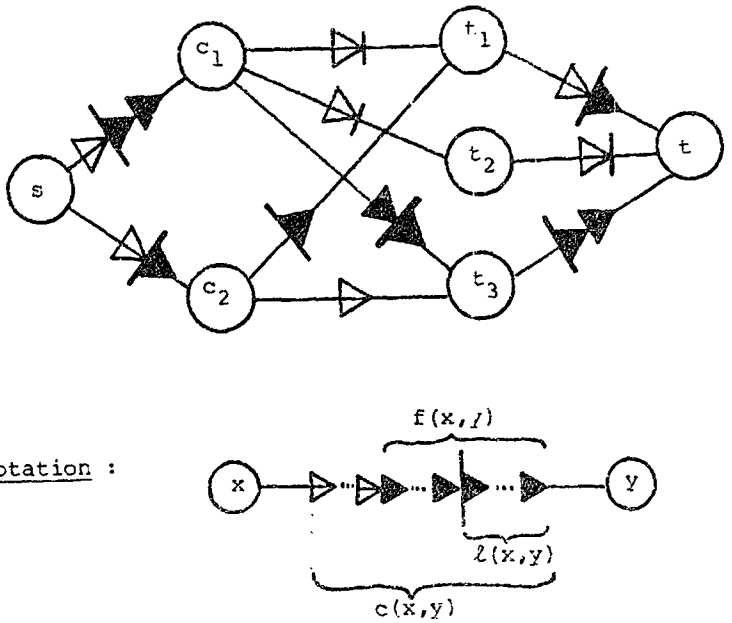

Figure 1. The construction of x_{ijl}. $x_{131} = 2$; $x_{211} = 1$; $x_{ijl} = 0$, otherwise.

tion of x_{ij1}. The requirements matrix is:

$$\begin{array}{c} \\ c_1 \\ c_2 \end{array} \begin{array}{c} \begin{array}{ccc} t_1 & t_2 & t_3 \end{array} \\ \begin{bmatrix} 1 & 2 & 4 \\ 3 & & 2 \end{bmatrix} \end{array}, \quad p = 3 \text{ days.}$$

The solution is (for x_{ij1}, x_{ij2} and x_{ij2}, respectively):

$$\begin{bmatrix} 0 & 0 & 2 \\ 1 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix}.$$

In general one may start by solving the weekly problem (assignment of lectures to days) and then solve separately the resulting daily problems.

2.2. Preassignments

Let us now go back to the daily problem ($x_{ijk} = 0$ or 1); in practice there are additional requirements which have to be taken into account.

a) *Preassignments.* Some pairs c_i, t_j have to meet at fixed periods k; this means that for some triples i, j, k we will have a value $p_{ijk} = 1$. In CT1 we shall introduce new constraints

$$x_{ijk} \geqslant p_{ijk} \tag{11}$$

where p_{ijk} will be 0 if there is no preassigned meeting of class c_i and teacher t_j at period k.

b) *Unavailabilities.* At some periods k a teacher t_j (or a class c_i) may be unavailable. Such constraints may be reduced to preassignments by stating that if t_j is unavailable at period k, there is a meeting of t_j with a dummy class c_i^* at period k and similarly for a class c_i which is unavailable at some period.

Instead of keeping constraints (11), we shall give a different formulation for the daily problem with preassignments and unavailabilities. We define $\bar{x}_{ijk} = 1$ if class c_i and teacher t_j meet at period k for a lecture which is not a preassignment and $\bar{x}_{ijk} = 0$ otherwise. We have the following constraints CT4

$$\sum_{k=1}^{p} \bar{x}_{ijk} = \bar{r}_{ij} \quad (i = 1,\ldots,m;\ j = 1,\ldots,n), \tag{12}$$

$$\sum_{j=1}^{n} \bar{x}_{ijk} \leqslant \bar{b}_{ik} \quad (i = 1,\ldots,m;\ k = 1,\ldots,p), \tag{13}$$

$$\sum_{i=1}^{m} \bar{x}_{ijk} \leqslant \bar{c}_{jk} \quad (j = 1,\ldots,n;\ k = 1,\ldots,p), \tag{14}$$

$$\bar{x}_{ijk} = 0 \text{ or } 1 \quad (i = 1,\ldots,m;\ j = 1,\ldots,n;\ k = 1,\ldots,p). \tag{15}$$

Here

$$\bar{r}_{ij} = r_{ij} - \sum_{k=1}^{p} p_{ijk},$$

$\bar{b}_{ik} = 1$ if c_i is available and not preassigned at period k,
0 otherwise,

$\bar{c}_{jk} = 1$ if t_j is available and not preassigned at period k,
0 otherwise.

It is not difficult to see that one can by increasing the size of the problem reduce it to a so called canonical form where (13) and (14) are equalities.

Necessary conditions for the existence of solutions to (12)–(15) based on the Hall conditions for systems of distinct representatives [11] have been given by C.C. Gotlieb [14]. For instance, at each period k, given any subset of q classes, there must be at least q teachers who can each meet at least one among the q classes at period k. Such conditions are unfortunately not sufficient for the existence of a solution to (12)–(15).

In the case where (12)–(14) are equalities, these constraints together with $x_{ijk} \geqslant 0$ replacing (15) define a *three dimensional transportation polytope.* No simple necessary and sufficient conditions are known for the three dimensional polytope to be nonempty.

Several necessary conditions have been given [17,32,35]. Notice that finding a fractional point might be interesting although it does not define a solution of CT4. As mentioned in [6,31], $x_{ijk} = 1/2$ could mean that c_i and t_j meet every second week at period k.

Going back to the daily problem with unavailabilities and preassignments, it has in fact been shown that the problem of deciding whether a solution exists is NP-complete [9].

There is however a very special case of CT4 for which it is easy to decide whether a timetable exists or not.

Proposition 2.4 [9]. *If in* CT4 *every teacher is available during at most* 2 *periods, there is an* $O(n^2)$ *algorithm for finding a timetable (or showing that none exists).*

Proof. One may clearly assume that all teachers are available for 2 periods exactly (since a teacher who is available for 1 period has a fixed schedule);

so each teacher has exactly 2 possible schedules. We construct a graph G with $2n$ nodes (where n is the number of teachers) as follows: For teacher t_j introduce 2 nodes x_j, $\bar{x}_j$ corresponding to his possible schedules; we link 2 nodes in G if the corresponding schedules are not compatible (in particular, nodes x_j and $\bar{x}_j$ are linked). In our example (see Table 1) for each t_i we have the following schedules:

	x_i		$\bar{x}_i$	
t_1	c_1-1,	c_2-2	c_2-1,	c_1-2
t_2	c_2-2,	c_1-3	c_1-2,	c_2-3
t_3	c_3-1,	c_4-3	c_4-1,	c_3-3
t_4	c_1-1,	c_3-2	c_3-1,	c_1-2

Then a timetable exists iff there is in G a set of n pairwise non adjacent nodes (i.e. a stable set [1]). This construction is illustrated in Figure 2.

The idea is to use a limited backtracking method which runs as follows: Initially one arbitrary selects a schedule for some teacher (this is a decision); then one considers all the implications of this choice. This may fix the schedules of some other teachers. Then either we arrive to the conclusion that for some tacher no schedule can be found. In such a case, we go back *to the last decision* and we reverse it (if the reverse decision has not been examined yet).

Table 1

Teacher t_i	Available at periods	Must meet classes
t_1	1, 2	c_1, c_2
t_2	2, 3	c_1, c_2
t_3	1, 3	c_3, c_4
t_4	1, 2	c_1, c_3

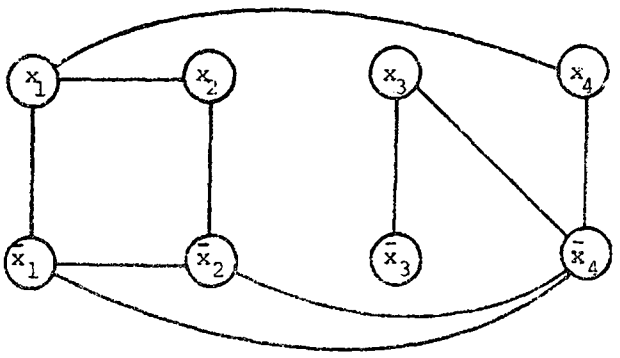

Figure 2. The corresponding 'conflict graph' G. A solution is obtained as follows: decision $\bar{x}_3$; decision $\bar{x}_4 \Rightarrow x_2 \Rightarrow$ impossible for x_1; decision $x_4 \Rightarrow \bar{x}_1$, x_2; feasible timetable.

If the reverse decision has been examined previously, then no timetable can exist as we shall see.

If we have taken all implications into account and there are still some teachers whose schedule is not fixed, then we consider the last decision as permanent and take a new decision and we continue as before.

Let us now show that when decisions x_j and $\bar{x}_j$ have been examined and when both imply that no timetable can be found, then none exists. Indeed there is no need to backtrack to some earlier decision: At some stage decision x_j and $\bar{x}_j$ had to be taken because there was no implication fixing the schedule of any of the yet unscheduled teachers. So the subproblem containing only tacher t_j and all those who were unscheduled when x_j and $\bar{x}_j$ were examined has no timetable. Hence the problem itself has no timetable.

It is not difficult to see that since we backtrack only to the last decision, there is an $O(n^2)$ algorithm to solve the problem.

Remark. The above proof in fact says that there is an $O(n^2)$ algorithm to determine whether for a graph $G = (X, E)$ with a perfect matching, there is a stable set of $|X|/2$ nodes.

2.3. Some methods

It follows from these considerations that the only available methods are heuristic procedures, which will try to produce a feasible solution but without guarantee of obtaining one whenever a solution exists. Many procedures have been suggested for dealing with this problem (see [5,7,18,29] for reviews).

Most of these methods combine in a clever way the use of *combinatorial models* for some structured subproblems with some more or less *intuitive* decision *rules*.

In the case of the daily problem for instance, one observes that if any index is kept fixed in x_{ijk}, then the determination of the values x_{ijk} can be performed by network flow techniques. For instance, if k if fixed, the problem of scheduling lectures at this period k is simply a *matching* problem in a bipartite graph; one has to assign classes which are available at period k to teachers who are available at period k.

Hence it seems natural to try to construct a schedule period after period while trying to sched-

ule as many 'urgent' lectures as possible at each period [25]. Such a method is heuristic since one has no guarantee that after having scheduled lectures at some periods, the remaining problem still admits a solution. At each step a maximum weight matching is constructed with a network flow algorithm.

In a similar way for a given class c_i the construction of a schedule consisting of all lectures involving c_i can be performed by network flows. We introduce one node k for each period and one node t_j for each teacher; there is an arc (k, t_j) if t_j is available for a lecture with class c_i at period k. We also introduce a node s with arcs (s, k) and a node t with arcs (t_j, t); the capacities $c(t_j, t)$ and the lower bounds $l(t_j, t)$ are r_j. For all previously introduced arcs (x, y) we set $c(x, y) = 1$ and $l(x, y) = 0$. Clearly any feasible integral flow will define a timetable for class c_i; this construction is illustrated in Figure 3. The row i of requirement matrix R is:

$$\begin{array}{cc} & \begin{matrix} t_1 & t_2 & t_3 \end{matrix} \\ c_i & \begin{bmatrix} 2 & 2 & 1 \end{bmatrix} \end{array}$$

The matrix with elements $\bar{c}_{jk}$ is:

$$\begin{array}{c} \\ t_1 \\ t_2 \\ t_3 \end{array} \begin{bmatrix} 1 & 1 & 1 & & \\ & 1 & 1 & 1 & \\ & & & 1 & 1 \end{bmatrix},$$

with

$$\bar{c}_{jk} = \begin{cases} 1 & \text{if } t_j \text{ available at period } k, \\ 0 & \text{otherwise.} \end{cases}$$

The timetable for c_i corresponding to the flow in Figure 3 is:

$$\begin{array}{cc} & \begin{matrix} 1 & 2 & 3 & 4 & 5 \end{matrix} \\ c_1 & \begin{bmatrix} t_1 & t_1 & t_2 & t_3 & \end{bmatrix} \end{array}$$

A schedule for c_i exists if and only if for any subset T of teachers, the number of periods where at least one teacher in T is available is at least as large as the total number of lectures involving class c_i and teachers in T (these are the conditions of Hall).

Again, the construction of the timetable class after class results in a heuristic procedure. In a similar way one could consider a procedure where the timetable would be constructed for each teacher consecutively. More generally methods have been

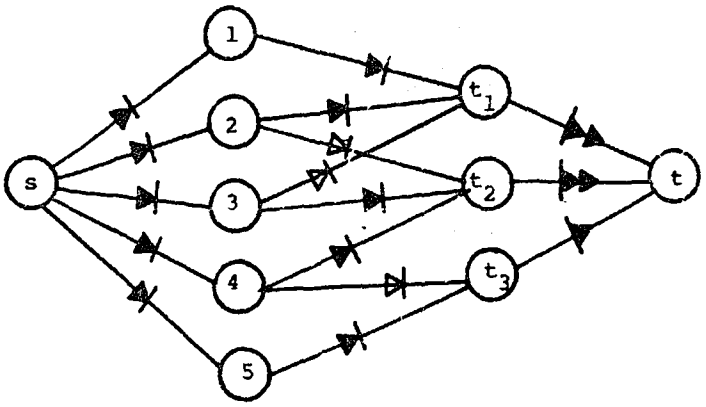

Figure 3. The flow for class c_i.

devised which consist in consecutively assigning lectures to some period while trying to keep satisfied some necessary conditions of existence of solutions for the remaining problem (see [32] for instance where various necessary conditions have been developed and also [18]).

The process is continued until either all lectures have been assigned or until no more can be assigned and some lectures are left unassigned. At this point, some methods will stop and some others will cancel some of the last assignments and start again at this stage.

We shall not go into the details of these procedures.

Let us mention however that in practice there are often requirements concerning sequences of lectures which have to be avoided or lectures consisting of two or more consecutive periods or even special lectures in which several classes and several teachers are involved. If the density of such occurrences is not too high, one may often preassign such lectures and then use the previous models with preassignments.

However, in the situation where special cases are frequent, different models have to be developed where the apparent symmetry between classes and teachers can no longer be exploited.

2.4. Feasibility vs optimality

Most timetabling problems which were discussed up to now have been formulated as feasibility problems (find a feasible solution) rather than optimality problems (find an optimal solution).

In fact, both formulations are very similar since

finding a feasible solution may be regarded as minimizing a kind of 'distance to feasibility'.

Here we generally do not have a well defined objective function to optimize; there are many requirements which occur as constraints in the problem and a collection of wishes which are not always completely explicited.

This situation however does not prevent us from using heuristic methods to generate timetables satisfying as well as possible the various wishes occurring in the problem.

3. Course scheduling

3.1. Basic formulation

While the problems CT1–CT3 could be formulated as edge coloring problems with some additional restrictions on the colors assigned to the edges (because of preassignments or unavailabilities), we shall have to use a more general model for tackling the following timetabling problems.

The *course scheduling problem* arises when a university (or even a school) offers a collection of courses (each one consisting of a given number of lectures) and there is no fixed curriculum: Each student may choose a certain number of courses. The problem consists in assigning each lecture to some period of the week in such a way that no student is required to take more than one lecture at a time. The situation is quite similar for the examination scheduling problem.

For the course scheduling problem, the following graph-theoretical model is used: We associate with each lecture l_a of each course K_b a lecture-node m_{ab}; for each course K_b we introduce edges between all pairs of lecture-nodes in K_b. Also whenever there is a student taking courses K_b and $K_{\bar{b}}$ we introduce an edge between each pair of lecture-nodes m_{ab}, $m_{a\bar{b}}$.

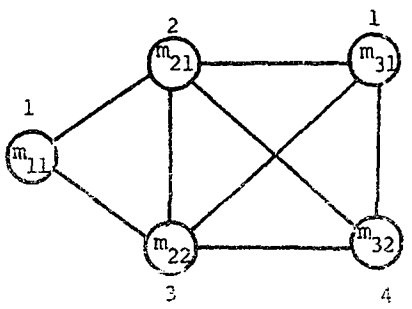

Figure 4.

A feasible course schedule in p periods will correspond to a node coloring of the above graph with p colors: Each node receives come color and no two adjacent nodes are allowed to have the same color.

An example of this construction is given in Figure 4; we have the following data:

course	# of lectures
K_1	1
K_2	2
K_3	2

$p = 4$ periods; student 1 takes K_1, K_2; student 2 takes K_2, K_3.

The schedule corresponding to the coloring in Figure 4 is:

period 1 : 1 lecture of K_1 and 1 lecture of K_3;
period 2 : 1 lecture of K_2;
period 3 : 1 lecture of K_2;
period 4 : 1 lecture of K_3.

Depending on the number p of periods,, it will not always be possible to find a schedule in p periods. In fact, deciding whether a graph has a node coloring with p colors is known to be an NP-complete problem when $p \geq 3$ [13].

As before, we have to include in our model a feature which enables us to deal with unavailability or preassignment constraints: For instance some courses cannot take place at certain prescribed periods (because a teacher or a specific classroom may not be available). Another requirement might be that a course has to be scheduled at certain prescribed periods.

Such a course scheduling problem with unavailabilities and preassignments (CSUP) can be reduced to a node coloring problem.

Proposition 3.1 [32]. *For any* CSUP *one can construct a graph* $\hat{G}$ *such that* CSUP *has a solution in* p *periods iff* $\hat{G}$ *has a node coloring with* p *colors.*

Proof. Let us start with the graph constructed above. We introduce for each period k a period-node k and we link all pairs of period-nodes.

If no lecture of a course K_b can be scheduled at period k, we link all lecture-nodes m_{ab} of course K_b to period-node k.

If a lecture l_a of course K_b has to be scheduled

at period $\bar{k}$, we link lecture node m_{ab} to all period-nodes $k \neq \bar{k}$. (Observe that we may as well remove node m_{at} and link all its neighbours with period-node k). The interpretation of the colored graph $\hat{G}$ is the following: Lecture l_a of course K_b will take place at period k if and only if node m_{ab} received the same color as period-node k. Clearly, a node coloring of $\hat{G}$ will correspond to a feasible schedule and conversely. The construction of $\hat{G}$ is illustrated in Figure 5. The data are the same as in Figure 4 with the additional constraints
- K_1 not scheduled at period 1;
- 1 lecture of K_3 at period 1 or 2.

It is interesting to observe here a difference between edge coloring and node coloring in a graph. Problem CT1 in Section 2 is an easy edge coloring problem solvable by network flow techniques, but as soon as preassignments are introduced (i.e. when some edges are precolored), finding whether an edge coloring with p colors exists is a difficult problem (NP-complete). For node coloring problems, preassignments do not make the problem more difficult; more precisely we have the following.

Proposition 3.2. *Let $\hat{G}$ be a course scheduling graph and assume some lecture-nodes have been precolored. Then there exists a graph $\hat{G}'$ (with no precolored node) such that $\hat{G}$ has a node coloring with p colors respecting the precoloring iff $\hat{G}'$ has a node coloring with p colors.*

Proof. As indicated in the proof of Proposition 3.1 each precolored node can be removed provided some additional edges are introduced. By doing this for each precolored node we get $\hat{G}'$ and it is obvious that there is a 1–1 correspondence between the colorings of $\hat{G}'$ and the colorings of $\hat{G}$ respecting the precoloring.

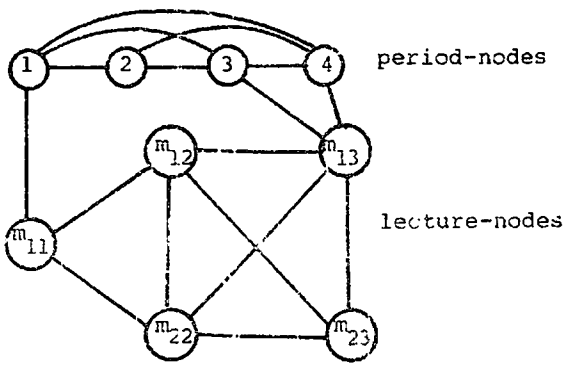

Figure 5

It is easy to see that problem CT4 can be formulated as a node coloring problem in $\hat{G}$; one should bear in mind that one can always transform an edge coloring problem in a graph G into an equivalent node coloring problem in a graph $\hat{G}$, but the converse is not true. So the formulation in terms of node coloring is more general and allows us to handle more general timetabling problems. Not surprisingly, many heuristic methods have been proposed in the timetabling literature for coloring the nodes of graphs; we shall describe some basic ideas of these procedures.

3.2. Node coloring methods

Some methods have been devised which will find a node coloring of a graph with p colors whenever such a coloring exists [3,26,27]. These procedures do in fact enumerate implicitly all possible colorings with p colors and so they cannot handle very large graphs; presently one cannot go beyond a few hundred nodes, which is a strong limitation since most timetabling graphs have many more nodes.

Experiments on randomly generated graphs having up to 1000 nodes have been carried out [16] and very good estimates of the minimum number of colors needed have been obtained. One may wonder however whether timetabling graphs are close to randomly generated graphs.

Besides the so called exact algorithms which can find a node coloring with p colors whenever there is one, various heuristic methods have been developed for coloring the nodes of a course scheduling graph.

Most of them color one node at a time; so one may consider that they are based on an *assignment strategy*: At each step one chooses according to some rule a lecture which has to be scheduled and then one chooses according to some criterion a period where this lecture will be scheduled.

These methods can be classified in two groups.

a) Those which choose the period independently of the lecture.

b) Those which choose the period and the lecture simultaneously.

In most of these methods, one chooses first a

lecture-node corresponding to a lecture which is 'difficult' to schedule. The period is chosen in such a way that an assignment of the lecture to this period will decrease as little as possible some *degree of freedom* of the classes, teachers, classrooms and courses.

For instance, we may take as degree of freedom of a course K_b the remaining number of periods where a lecture of K_b could be scheduled minus the remaining number of lectures of K_b to be scheduled. Various assignment strategies are reviewed in [32]. Some of them have simple interpretations in terms of graphs; in case a) we may have sequential coloring methods [3] where an order of the nodes is determined first; then the nodes are consecutively colored in this order and one tries to give each node the smallest possible color.

A well-known method consists in ordering the nodes according to the number of their neighbours, i.e. one takes first the nodes with the largest number of neighbours [36].

Among the simple heuristic methods, the most efficient in average seems to be the DSATUR method [3]. The order is constructed as follows: Initially take any node with the largest possible number of neighbours and color it with the smallest possible color. In general, for each node x there is a set $F(x)$ of forbidden colors (these may be colors which have already been assigned to neighbours of x); at each step we choose a node y for which the cardinality $|F(y)|$ of $F(y)$ is maximum and we color it with the smallest possible color.

The efficiency of this procedure is increased (i.e. the number of colors may be reduced) when it is combined with a bichromatic interchange procedure [23].

It is interesting to notice that the DSATUR method is an example of a heuristic procedure for general graphs which is derived from an exact method designed for coloring bipartite graphs (for these graphs, we have $|F(x)| \leq 1$ at each step). Another such method is decribed in [8].

Experiments with sequential coloring methods have been reported by numerous authors [5,21,33] and graphs having a few hundred nodes have been colored almost optimally; such tests are described in [33] where a procedure is developed for generating random graphs which can be colored with exactly p colors and not less.

Methods of type b) are different from the above described procedures in the sense that one chooses at the same time the next node to be colored as well as the color which it will receive.

At this point we should mention that a practically and theoretically promising area of research consists in determining the largest possible classes of graphs for which simple heuristics of type a), or b) are exact methods (i.e. they find a coloring with p colors whenever there exists one). If such classes were entirely characterized, one could try to decompose a course scheduling graph $\hat{G}$ into smaller subgraphs $\hat{G}_1,\ldots,\hat{G}_r$, which would be easier to color. It would then suffice to recombine them in an adequate way to obtain a coloring of the initial graph $\hat{G}$. Partial encouraging results have been obtained in this direction [4,22]; some of the classes of graphs which have been characterized belong to the so called perfect graphs [1].

3.3. Course scheduling vs exam scheduling

In Section 3.1 we mentioned that course scheduling problems and exam scheduling problems were quite similar; both types could be formulated in terms of node coloring in a graph. There are in fact a few differences which should be underlined.

a) There is usually only one exam in each subject (while we had several lectures in a course). Exam graphs will so be generally smaller in size than course scheduling graphs.

b) In a weekly course schedule, the only objective is to avoid conflicts (i.e. the situation where 2 courses chosen by a student are scheduled at the same period). For the exams, it is usually required to have at most one exam per day for each student or even to avoid having exams on consecutive days if the time interval for exams allows it.

Such constraints are in some sense not naturally included in coloring models; one has to take special care of them in order to get a feasible exam schedule (see [23,40]).

3.4. Mathematical programming approach to course scheduling

Several models of course scheduling based on mathematical programming have been proposed [10,34,39]. Here we shall present a formulation which exploits the underlying network structure of the problem [34].

We are given a collection of q courses $K_1,\ldots,K_q$, course K_i consists of k_i lectures of one period each.

The total number of periods is p. The students are divided into r groups $S_1,\ldots,S_r$ such that in each S_i all students take exacity the same courses; l_k is the maximum number of lectures which can be scheduled at period k (this may correspond to the number of available classrooms).

Then we define $y_{ik}=1$ if a lecture of course K_i is scheduled at period k and $y_{ik}=0$ otherwise.

$$\text{Max} \quad \sum_{i=1}^{q}\sum_{k=1}^{p} C_{ik}y_{ik}, \tag{16}$$

$$\text{s.t.} \quad \sum_{k=1}^{p} y_{ik}=k_i \qquad (i=1,\ldots,q), \tag{17}$$

$$\sum_{i=1}^{q} y_{ik}\leqslant l_k \qquad (k=1,\ldots,p), \tag{18}$$

$$\sum_{i\in S_l} y_{ik}\leqslant 1 \qquad (l=1,\ldots,r,\ k=1,\ldots,p), \tag{19}$$

$$y_{ik}=0 \text{ or } 1, \tag{20}$$

The costs C_{ik} occuring in the objective function (16) are a measure of the desirability of $y_{ik}=1$; one can give a high value if it is desired to have a lecture of course K_i in period k, or 0 otherwise.

Problems with $p=70$ periods, $q=22$ courses and $r=40$ groups of students have been successfully solved [34]. The idea was to apply a Lagrangean relaxation technique; the constraints (19) were incorporated in the objective function which became

$$\max \sum_{i=1}^{q}\sum_{k=1}^{p} C_{ik}y_{ik}+\sum_l\sum_k \lambda_{kl}\left(1-\sum_{i\in S_l} y_{ik}\right) \tag{21}$$

The constraints are now (17), (18), (20) so that one has a capacitated transportation problem; the values λ_{kl} were obtained with a subgradient optimization method combined with a Branch and Bound procedure [34]. Larger problems corresponding to university exam schedules were also solved in a few minutes.

A different formulation leading to a quadratic assignment problem was used for course scheduling as well as for classroom assignment at the University of Montreal [10]. Up to 174 courses could be scheduled in a few minutes; an interactive code has been developed for this purpose.

3.5. Classroom assignment

In the model presented in Section 3.4 we met constraints imposed by the number of simultaneously available classrooms. Some models take such requirements into account in a heuristic way and reduce each step of the procedure to a network flow problem (see, for instance, [24]). If we want to formulate such constraints in a global way, we need to extend teh concepts of graphs and of colorings. First a *hypergraph* $H=(X, E)$ will consist of a finite set X of nodes and a finite collection E of edges E_i; each edge will be a subset of the node set; if every E_i contains exactly 2 nodes we have a graph. Now, for each edge E_i we also give a positive integer e_i. An assignment of colors to the nodes of H will be a *node coloring* if in each E_i no more than e_i nodes can have the same color. Clearly, if H is a graph and if $e_i=1$ for each edge E_i, we have the usual concept of node coloring in a graph.

Let us consider a course scheduling problem and its corresponding graph G (with lecture nodes and period nodes). Assume that the courses $K_1,\ldots,K_r$ have to be given in a special type of room and that there are s such rooms in the school. Then we transform G into a hypergraph $\hat{H}$ by introducing an edge E_1 containing all lecture nodes of $K_1,\ldots,K_r$ with $e_1=s$. By introducing such edges for all similar constraints we get a hypergraph, the node colorings of which are in 1–1 correspondence with the feasible schedules.

As can be expected finding node colorings in hypergraphs is not easier than in graphs. There are however interesting special cases where efficient techniques are available [19].

4. General remarks

4.1. The real case

Until now we have concentrated our attention on some models which can formalize various types of constraints occurring in timetabling problems. In spite of our efforts to introduce more and more ingredients in the models, there are still many other types which have not been mentioned yet: Compactness requirements (as few 'holes' as possible in the timetable of a class or of a teacher), lectures of different lengths (1 or 2 periods), se-

quences of subjects to be avoided , etc.

Although such constraints are not obviously translated in the previous models, they can be and they are in fact included among the requirements which are accepted by most available timetabling codes. As pointed out before, heuristic methods constructing a schedule step by step are usually able to handle all kinds of requirements. Successful applications in various types of schools or universities have been described together with critical and constructive remarks [2,7,12,18,25,28]. We summarize some observations which are scattered in the above references.

a) With a few exceptions (see [7]), most of the early computer codes tried to construct a timetable from scratch; the constraints, requirements and wishes were given with a set of priorities as input data. Then the program managed either to assign each lecture to some period in such a way that the number of 'conflicts' was kept as low as possible or it left a few lectures unassigned. It turned out that with such codes, a feasible timetable could be produced (possibly after several runs) with relatively minor manual adjustments.

It appeared that the use of computers was not widely accepted by teachers who did not like the idea of being scheduled by a machine. This is a reason why the next generation codes are highly interactive; crucial decisions can be taken by the course schedular himself during the construction process [10,28].

b) Besides the savings in time, the computer has the advantage that the timetables produced do not violate any constraints. Furthermore the computer facilities can be used to print the timetables of classes, of teachers, of classrooms, etc.

c) Of course the use of a computer is interesting only for large schools or universities; for instance as soon as there are at least 30 classes, one should try to construct a timetable with a computer.

d) Since timetabling problems can be very different between one school and another one (even in the same educational system), it does not seem reasonable to develop a universal timetabling program which could be used everywhere. To be efficient heuristic methods have to be adjusted not only to the nature of the constraints and requirements but also to their density. So a heuristic which is good for some school may perform very badly in another one which looks similar.

e) It is essential that the codes be made easy to use, that the method be almost transparent to the user; feeling and understanding how the procedure works may help the schedular to reach a good solution in a reasonable number of runs. To increase the chances of survival of such a program, the user should have a direct access to it; he should definitely not have to go to a computing center outside of the school.

4.2. Conclusions

Our purpose here was not to recommend the use of such and such available code for timetabling. Instead we simply described some basic models and we briefly mentioned how they could be inserted in a general program which produces usable timetables. As a conclusion, the field of timetabling is a nice topic for the specialist in OR for several reasons: there are many problems to be solved which include pure combinatorial mathematics, computer science (data basis, management, implementation of algorithms, etc.) and applied OR (constructing a good timetable may be viewed as a multicriteria decision problem or a group decision problem in a human context).

Much research in all these fields will be needed for coping with the new problems which will undoubtedly arise in the future in the area of timetabling.

References

[1] Berge, C., *Graphes*, Gauthier-Villars, Paris, 1983.

[2] Bloomfield, S.D., and McSharry, M.M., "Preferential course scheduling", *Interfaces* 9 (1979) 24–32.

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

[4] Carter, M., "A decomposition algorithm for practical timetabling problems", Working Paper 83-06, Dept. of Industrial Engineering University of Toronto, 1983.

[5] Carter, M., "A survey of practical applications on examination timetabling", Working Paper 83-07, Dept. of Engineering University of Toronto, 1983.

[6] Clementson, A.T., and Elphick C.H., "Continuous timetabling problems", *Journal of the OperationalResearch Society* 33 (1982) 181–183.

[7] Dempster, M.A.H., Lethridge, D.G., and Ulph, A.M., *School timetabling by Computer – A Technical History*, Oxford University, 1973.

[8] Dutton, R.D., and Brigham, R.C., "A new graph coloring algorithm", *Computer Journal* 24 (1981) 85–86.

[9] Even, S., Itai, A., and Shamir, A., "On the complexity of timetable and multicommodity flow problems", *SIAM Journal on Computing* 5 (1976) 691–703.

[10] Ferland, J.A., and Roy, S., "Course scheduling and classroom assignment in a university", Publication 459, University of Montreal, 1982.

[11] Ford, L.R., and Fulkerson, D.R., *Flows in Networks*, Princeton University Press, Princeton, 1962.

[12] Gans, O.B. de, "A computer timetabling systemf or secundary schools in the Netherlands", *European Journal of Operational Research* 7 (1981) 175–182.

[13] Garey, M.R., and Johnson, D.S., *Computers and Intractability, a Guide to the Theory of NP-completeness*, Freeman, San Francisco, 1979.

[14] Gotlieb, C.C., "The construction of class-teacher timetables" *Proceeding IFIP Congress* 1962, Amsterdam, 1963, 73–77.

[15] Hilton, A.J.W., "School timetables", *Annals of Discrete Mathematics* 11 (1981) 177–138.

[16] Johri, A., and Matula, D.W., "Probabilistic bounds and heuristic algorithms for coloring large random graphs", Dept. of Computer Science Engineering, Southern Methodist University Dallas, Texas, 1982.

[17] Junginger, W., "Zurückführung des Studenplanproblems auf ein dreidimensionales Transportproblem", *Zeitschrift für Operations Research* 16 (1972) 11–25.

[18] Junginger, W., "Zum aktuellen Stand der automatischen Studenplanerstellung", *Angewandte Informatik* 1 (1982) 20–25.

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

[20] Lawrie, N.L., "An integer programming model of a school timetabling problem", *Computer Journal* 12 (1969) 307–316.

[21] Leighton, F.T., "A graph coloring algorithm for large scheduling problems", *Journal of Research of the National Bureau of Standard* 84 (1979) 489–506.

[22] Mahadev, N.V.R., and de Werra, D., "On a class of perfectly orderable graphs", Report CORR 83/27, University of Waterloo, Ontario, 1983.

[23] Mehta, N.K., "The application of a graph coloring method to an examination scheduling problem", *Interfaces* 11 (1981) 57–64.

[24] Mulvey, J.M., "A classroom/time assignment model", *European Journal of Operational Research* 9 (1982) 64–70.

[25] Ostermann, R., and de Werra, D., "Some experiments with a timetabling system", *OR Spektrum* 3 (1982) 199–204.

[26] Peemöller, J., "A correction to Brelaz's modification of Brown's coloring algorithm", *Communications of the ACM* 26 (1983) 595–597.

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

[28] Romero, B.P., "Examination scheduling in a large engineering School: A computer-assisted participative procedure", *Interfaces* (1982) 17–23.

[29] Schmidt, G., and Ströhlein, T., "Timetable construction —an annotated bibliography", *The Computer Journal* 23 (1979) 307–316.

[30] Silver, E.A., Vidal, R.V., and de Werra, D., "A tutorial on heuristic methods", *European Journal of Operational Research* 5 (1980) 153–162.

[31] Smith, G., and Sefton, I.M., "On Lion's counter example for Gotlieb's method for the construction of school timetables", *Communications of the ACM* 17 (1974) 197.

[32] Smith, G., "Computer solutions to a class of scheduling problems", Ph.D. Thesis, University of New South Wales, 1975.

[33] Thomson Leighton, F., "A graph coloring algorithm for large scheduling problems", *Journal of Research of the National Bureau of Standards* 84 (1979) 489–506.

[34] Tripathy, A., "A Lagrangean relaxation approach to course timetabling", *Journal of the Operational Research Society* 31 (1980) 599–603.

[35] Vlach, M., "On the three planar sums transportation polytope", Report 83312-OR, University of Bonn, 1983.

[36] Welsh, D.J.A., and Powell, M.B., "An upper bound for the chromatic number of a graph and its application to timetabling", *Computer Journal* 10 (1967) 85–86.

[37] Werra, D. de, "A few remarks on chromatic scheduling", in: B. Roy, (ed.), *Combinatorial Programming: Methods and Applications*, Reidel, Dordrecht 1975, 337–342.

[38] Werra, D. de, "Some comments on a note about timetabling", *INFOR* 16 (1978) 90–92.

[39] White, D.J., "A note of faculty timetabling", *Operations Research Quarterly* 26 (1975) 875–878.

[40] White, G.M., and Chan, P.W., "Towards the construction of optimal examination schedules", *INFOR* 17 (1979) 219–229.