European Journal of Operational Research 76 (1994) 98–110
North-Holland

Theory and Methodology

A Tabu Search algorithm for computing an operational timetable

Daniel Costa

Ecole Polytechnique Fédérale de Lausanne, Département de Mathématiques, Chaire de Recherche Opérationnelle, CH-1015 Lausanne, Switzerland

Received December 1990; revised February 1992

Abstract: The many constraints of different types that must be taken into account and the volume of data make the timetabling problem very hard to solve. Various specific requirements are considered in this paper. After having described precisely the problem to be solved, we present a general technique based on tabu search for finding an acceptable timetable. The resulting computerized algorithm is used for constructing different real course schedules. The results we obtain are very satisfactory.

Keywords: Timetabling; Combinatorial optimization; Heuristic procedure; Tabu search

1. Introduction

Constructing a timetable by hand is a tedious and time-consuming task which easily involves one or two people for several days. Because of the many requirements that have to be taken into account, the timetabling problem (TTP) is particularly hard in educational institutions. A large amount of data, the diversification of teaching methods and the ever increasing requirements in curricula make educational organizations very difficult to schedule. The manual techniques which are still commonly used turn out to be insufficient. To master these difficulties, timetable planners are now ready to ask for computer aid.

The timetabling problem is not recent, many different models and approaches have been proposed in the literature (see for instance [6,16–19]). This is mainly due to the fact that the problem may be very different from one country or school system to another. Our goal in this paper is to present a general timetabling algorithm which can easily be adapted and used in various colleges and secondary schools.

In the next section, we formulate the timetabling problem: the data and the constraints that are taken into account will be pointed out. In Section 3 we indicate how we proceed to transform the timetabling problem in a combinatorial optimization problem. It has been shown that all common timetable problems are NP-hard [8]. Hence, the computing time required to find an optimal solution, or to prove the optimality of such a solution when it is found, becomes prohibitive when the problem gets large. It will be necessary to have recourse to heuristics in order to get a good solution of the problem in a

Correspondence to: Dr. D. Costa, École Polytechnique Fédérale de Lausanne, Département de Mathématiques, Chaire de Recherche Opérationnelle, CH-1015 Lausanne, Switzerland

reasonable amount of time. The tabu search technique, developed independently by Glover [10,11] and Hansen [12], is certainly one of the most efficient heuristics for handling large optimization problems. The method has already been successfully adapted to a large collection of applications [7,9,15,20]. In Section 4 we sketch the fundamental principles of tabu search, while in Section 5 we indicate how this general method has been adapted to the course scheduling problem we are interested in. Numerical results and some of our experiments with the corresponding computer code are reported in Section 6. Finally, possible extensions and concluding remarks are given in Section 7.

2. The timetabling problem

Many kinds of timetabling problems have been related in the literature during the last years, but most of them are very specific and do not take into account a lot of elements. The course scheduling problem is particular in the sense that it is generally based on a very large volume of data and different types of constraints. In Switzerland, since almost each canton has its own school system, a huge variety of timetables must be constructed for educational institutions. The situation in various colleges and secondary schools has led us to formulate the timetabling problem in a general way in order to take into account all the particular requirements we have encountered.

The following elements define the data of the problem. We have:

- a set $T = \{t_1, \ldots, t_{\mathrm{nt}}\}$ of teachers.
- a set $C = (c_1, \ldots, c_{\mathrm{nc}}\}$ of classes. A *class* is a group of students having the same curriculum.
- a set $S = \{s_1, \ldots, s_{\mathrm{ns}}\}$ of subjects.
- a set $\mathrm{SR} = \{\mathrm{sr}_1, \ldots, \mathrm{sr}_{\mathrm{nsr}}\}$ of special room types. A *special room* is an especially equipped teaching room, as for instance a laboratory or a sports hall.
- a subset $\mathrm{SS} = \{\mathrm{ss}_1, \ldots, \mathrm{ss}_{\mathrm{nss}}\}$ of S which contains all the subjects requiring special rooms. Such subjects will be called *special subjects*. With each special subject ss_j is associated a certain type of special room in which ss_j can be taught.
- a set $B = \{b_1, \ldots, b_{\mathrm{nb}}\}$ of distant buildings. Two buildings are considered to be distant if there is not enough time for moving from one building to the other during a break.
- a set $\mathrm{CO} = \{\mathrm{co}_1, \ldots, \mathrm{co}_{\mathrm{nco}}\}$ of courses. A *course* co_j is defined by:
  - a set of teachers,
  - a set of classes,
  - a set of subjects,
  - a number of single lectures (lasting one hour),
  - a number of double lectures (lasting two hours),
  - the building where the course must be given,
  - the number of rooms needed for each type of special room involved in the course.
  - the spacing of the course; there is need to mention if some teaching room must be prepared (resp. be tidied up) before (resp. after the end of) each lecture of the course.

  A course is called a *simple course* if it concerns only one teachers and one class. Otherwise, it is called a *grouped course*. A grouped course corresponds to the reunion of students from different classes who are divided into several groups in order to attend at the same time different lectures given by different teachers.

  With the set CO is associated the set $L = \{\ell_1, \ldots, \ell_{\mathrm{nl}}\}$ of all lectures (single or double).
- a set $P = \{p_1, \ldots, p_{\mathrm{nd}\cdot m}\}$ of one hour periods during which the various lectures must be scheduled. nd represents the total number of teaching days per week and m the number of periods in a day.
- a collection $\{P_{\ell_1}, \ldots, P_{\ell_{\mathrm{nl}}}\}$ where P_{ℓ_j} is a set of periods (*preassigned periods*) during which lecture ℓ_j has to be scheduled. $|P_{\ell_j}| = 1$ means that the j-th lecture must be planned at a given period. When there is no need to assign a lecture ℓ_j to certain periods, we set $P_{\ell_j} = P$.
- a collection $\{U_{t_1}, \ldots, U_{t_{\mathrm{nt}}}\}$ where U_{t_j} is the set of periods during which the j-th teacher is not available for teaching ($U_{t_i} \subset P$).

- a collection $\{U_{c_1},\ldots,U_{c_{nc}}\}$ where U_{c_j} is the set of periods during which the j-th class is not available for studying ($U_{c_j} \subset P$). The number of available periods for each class usually corresponds to the number of hours of lectures in its curriculum. In principle, these available periods are chosen in such a way that the schedule of each class does not contain any holes.
- a collection $\{U_{s_1},\ldots,U_{s_{ns}}\}$ where U_{s_j} is the set of periods during which the teaching of subject s_j is not wished (for some pedagogical reasons) ($U_{s_j} \subset P$).
- a ($\text{nsr} \times \text{nd} \cdot m$) matrix M of non-negative integers. M_{ij} gives the number of special rooms of type i available at period j.

Some hypotheses have been implicitly done above and are worth being pointed out. We have assumed that:

- the different simultaneous lectures of a grouped course have the same spacing constraints and are given in a same building (that is to say in rooms that are near one from another).
- at each period the number of non-special available rooms is not limited. Should this not be the case, we set $\text{SS} = S$ and consider an extra type of special room, that of the ordinary teaching rooms.
- for each course, the number of lectures and their length (one or two hours) are known in advance. A global approach has been presented by Hertz [14] for solving the course scheduling problem when the number and the length of lectures are not fixed.
- there is no need to repeat some courses during the week because they involve a large number of students. This requirement which concerns more directly large institutions as universities has been studied by Aubin and Ferland [2] and Hertz [13].

Based on the previous definitions and data, the timetabling problem consists in assigning each lecture of each course to a period while taking the following constraints into account:

(1) *teacher overlaps:* a teacher cannot be involved simultaneously in more than one lecture.
(2) *class overlaps:* a class cannot be involved simultaneously in more than one lecture.
(3) *room overlaps:* at every period and for each type of special room, the number of rooms required cannot be larger than the number of available rooms.
(4) *preassignment constraints:* certain lectures are preassigned to a set of specific periods and therefore have to be scheduled during these given periods.
(5) *teacher unavailabilities:* a lecture involving a teacher t_j cannot be scheduled at a period during which t_j is not available.
(6) *class unavailabilities:* a lecture involving a class c_j cannot be scheduled at a period during which c_i is not available.
(7) *subject unavailabilities:* a lecture of subject s_j should not be scheduled at a period during which the teaching of s_j is not allowed. For instance, timetable constructors try if possible not to schedule lectures of an important subject in the afternoon periods.
(8) *geographical constraints:* sufficient time must be provided to move from a building to another. Therefore two lectures given in two distant places and involving a same teacher or a same class cannot be scheduled consecutively.
(9) *compactness constraints:* each teacher wants a schedule with a minimal number of holes and isolated lectures. The schedules of classes are automatically compact since the number of available periods assigned to each class is exactly the number of periods required.
(10) *distribution constraints:* the identical lectures (i.e. the lectures of a same course) must be spread as uniformly as possible throughout the days of the week.
(11) *management of double lectures:* the two hours of a double lecture must be planned on two consecutive periods. For instance, they cannot be scheduled on the last period of a day and on the first of the next day or on some specified periods as just before and just after a lunch break.
(12) *precedence constraints:* a class schedule cannot contain some sequences of subjects. In practice, one generally avoids that a class has to attend consecutively two lectures of a same subject or for instance an English lecture just before a German lecture.
(13) *room preparing constraints:* a teaching room is not available when it is being prepared for a specific lecture. Thus no lecture can be scheduled in such a room (for instance a laboratory) just before a

lecture requiring a particular preparation. The same case occurs at the end of a lecture when the teaching room must be tidied up. These requirements concern only special rooms since the number of ordinary rooms is supposed to be unlimited.

(14) *variable lunch break:* when a lunch break must be planned daily during one among two consecutive periods (called *first* and *second break period*), teachers and classes cannot attend more than one single lecture during the two considered periods.

(15) *balanced lunch break:* during each variable lunch break the number of classes having the break at the first break period should not be very different from the number of classes having the break at the following period. This requirement must be taken into account in institutions where the dining hall is not large enough to contain all the students at a same time.

Some of the constraints formulated above do not occur frequently in a course scheduling problem. Nevertheless, we consider all of them and present in this paper the problem in its most general form. The computer program we have developed allows the timetable constructor to precise which constraints have to be taken into account in his school environment.

In practice, a timetable satisfying all the requirements may clearly not exist. We see for instance that constraints (9) and (10) are contradictory. On the one hand, we want to compact the schedules of teachers and on the other hand we want to spread the lectures of a same course throughout the week. A compromise between teacher and class convenience has to be found. In some cases, we need to find a timetable being as good as possible while admitting that some conflicts may happen. The number of these conflicts has to be minimized in a sense that will be explained later.

3. Mathematical formulation: A global approach

We give below some definitions needed in the mathematical formulation of the course scheduling problem. Let us consider a partition (C_a, C_b) of the set C of all the constraints described in Section 2. The constraints included in category C_a are called *essential* whereas those of category C_b are called *relaxed*.

- a timetable is *acceptable* if it satisfies all the constraints of the set C.
- a timetable is *feasible* if it satisfies the essential constraints. According to this definition a timetable is acceptable if it is feasible and satisfies in addition the relaxed constraints.

The timetabling problem described in the previous section will be assimilated to a combinatorial optimization problem in the following sense. Let us define a function $f(T)$ which gives a measure of the unacceptability of a timetable T. The principle of our approach consists in looking for the timetable T^* minimizing the value of f over the set X of all feasible timetables. In other terms the problem to be solved is of the form:

$$\text{(P)} \qquad \begin{array}{ll} \min & f(T) \\ \text{s.t.} & T \in X. \end{array}$$

How easy it is to explore the set X depends essentially on the partition (C_a, C_b). In the next section we will define an iterative procedure which starts from an initial feasible solution and tries to reach an optimum of the problem by moving successively from a solution to a neighbor solution. In order to apply such a principle for solving the problem (P), we must take care when defining the set X that one can reach an acceptable timetable from any feasible initial timetable. We need to find a compromise between the extent of the set X and the complexity of the objective function.

It was decided to include constraints (4)–(6), (10)–(12) and (14) in category C_a and to relax all other constraints (see Table 1). Though they are the most fundamental ones, constraints (1)–(3) were relaxed. This may be surprising but in this way we have a greater degree of freedom in moving in the search space X. The introduction of these constraints in the set C_a would have indeed reduced too much the size of

Table 1
Bipartition of the set C

Essential constrains	Relaxed constraints
a) preassignment constraints (4)	1) teacher overlaps (1)
b) teacher unavailabilities (5)	2) class overlaps (2)
c) class unavailabilities (6)	3) room overlaps (3)
d) distribution constraints (10)	3a) room preparing constraints (13)
e) management of double lectures (11)	4) subject unavailabilities (7)
f) precedence constraints (12)	5) geographical constraints (8)
g) variable lunch break (14)	6) compactness constraints (9)
	7) balanced lunch break (15)

X. We adopted the criterion presented in [14] to bipartition the set C. A conflict between two lectures is not a sufficient condition for hindering an assignment. However a lecture should never be assigned to a time period if such an assignment can induce nothing but an unacceptable timetable. According to this principle, constraints (10), (12) and (14) should be relaxed. Different tests were performed at this level and it turned out that the way of handling them does not affect significantly the quality of the results. In order to have an objective function which is not too complicated and to be sure to have constraints (10), (12) and (14) satisfied in the final solution, we decided not to relax them since their influence on the size of X is negligible.

We have modelled the distribution constraints in decreeing that no more than $\alpha(j)$ lectures of course co_j can be scheduled during the same day. The value $\alpha(j)$ depends on the number of lectures of course co_j. It is fixed by the timetable constructor.

Let us now define the objective function which measures the remoteness existing between a feasible timetable and the effective needs of the school:

$$f(T) = \sum_{i=1}^{7} w_i \cdot f_i(T).$$

This function is structured hierarchically: it is defined through a set of weights w_i giving a relative importance to the relaxed constraints. The component $f_i(T)$ computes the degree of violation of the i-th relaxed constraint in the timetable T. All components of $f(T)$ are described below. Some of the notations introduced in Section 2 will be used again.

Let us denote by

L_p:	The set of lectures given at period p.
$\mathrm{com\_}t(x,y)$:	The number of teachers common to lectures x and y; if x and y are single lectures, then $\mathrm{com\_}t(x,y) \in \{0,1\}$.
$\mathrm{com\_}c(x,y)$:	The number of classes common to lectures x and y.
$\mathrm{nra}(t,p)$:	The number of special rooms of types t available at period p.
$\mathrm{nrr}(t,p)$:	The number of special rooms of type t required at period p.
U_s:	The set of periods during which subjects s should not be taught.
$\mathrm{nls}(s,p)$:	The number of lectures of subject s scheduled at period p.
$\mathrm{ncp}(t,\mathrm{hd})$:	The number of components in the schedule of teacher t during the half-day hd. A *component* in a teacher schedule is a set of consecutive lectures.
$\mathrm{nfr}(t,\mathrm{hd})$:	The number of free periods in the schedule of teacher t during the half-day hd.
$\mathrm{ncl}(d,i)$:	The number of classes having a lecture at the i-th break period of the day d ($i \in \{1,2\}$).

- $f_1(T)$ evaluates how many teachers are involved simultaneously in two lectures:

$$f_1(T) = \sum_{p=1}^{\mathrm{nd}\cdot m} \sum_{x<y\in L_p} \mathrm{com\_}t(x,y).$$

• $f_2(T)$ evaluates how many classes attend simultaneously two lectures:

$$f_2(T) = \sum_{p=1}^{\mathrm{nd}\cdot m} \sum_{x<y\in L_p} \mathrm{com\_}c(x,y).$$

• $f_3(T)$ evaluates how many times the availability of special rooms is not respected:

$$f_3(T) = \sum_{t=1}^{\mathrm{nsr}} \sum_{p=1}^{\mathrm{nd}\cdot m} \max\{\mathrm{nrr}(t,p) - \mathrm{nra}(t,p),0\}.$$

• $f_4(T)$ evaluates how many lectures are scheduled at a period during which the teaching of a subject is not wished:

$$f_4(T) = \sum_{s=1}^{\mathrm{ns}} \sum_{p\in U_s} \mathrm{nls}(s,p).$$

• $f_5(T)$ evaluates the number of teachers and classes taking part in two lectures scheduled consecutively during the same day in two distant buildings. Let $\delta(x,y)$ be equal to 1 if lectures x and y are given at distant places; $\delta(x,y) = 0$ otherwise. Then we have

$$f_5(T) = \sum_{d=1}^{\mathrm{nd}} \sum_{p=(d-1)\cdot m+1}^{d\cdot m-1} \sum_{x\in L_p, y\in L_{p+1}} \delta(x,y)\cdot\{\mathrm{com\_}t(x,y) + \mathrm{com\_}c(x,y)\}.$$

• $f_6(T)$ measures the degree of compactness of the timetable:

$$f_6(T) = \sum_{t=1}^{\mathrm{nt}} \sum_{\mathrm{hd}=1}^{2\cdot\mathrm{nd}} \mathrm{ncp}(t,hd)\cdot\mathrm{nfr}(t,hd).$$

To fix ideas, let us examine the schedule of one teacher during three days of ten periods each (see Figure 1). The lunch break is fixed at the sixth period of the day. Inside a box, one can read the number of lectures given by the teacher at the considered period. Let us denote $f_{6,d}(T)$ the part of $f_6(T)$ which concerns the schedule of day d.

The timetable presented in Figure 1 is clearly not acceptable since the teacher is involved in two lectures at the third period of the first day. In the schedule of the first day, we observe two components and two free periods in the morning and one component and three free periods in the afternoon. Hence $f_{6,1}(T)$ is equal to $2\cdot 2 + 1\cdot 3 = 7$. Four consecutive lectures are scheduled in the afternoon of the second day and in the morning of the third day. But, on the contrary to $f_{6,2}(T)$, $f_{6,3}(T)$ is different from 0. This is due to the fact that there is a free period during the last morning. It is indeed possible to improve the compactness of this timetable in scheduling a lecture at the 5th period of the 3rd day.

• $f_7(T)$ sums everyday the difference in absolute value between the number of classes have a lecture at the first break period and the number of classes having a lecture at the following period:

$$f_7(T) = \sum_{d=1}^{\mathrm{nd}} |\mathrm{ncl}(d,1) - \mathrm{ncl}(d,2)|.$$

1	0	2	1	0	L	0	0	1	0	$f_{6,1}(T)=2\cdot 2+1\cdot 3=7$
0	0	0	0	0	L	1	1	1	1	$f_{6,2}(T)=0\cdot 5+1\cdot 0=0$
1	1	1	1	0	L	0	0	0	0	$f_{6,3}(T)=1\cdot 1+0\cdot 4=1$

Figure 1. Schedule of one teacher during three days of ten periods

Except for $f_6(T)$, the lower bound of each component of $f(T)$ is in principle equal to 0. The minimum value of $f_6(T)$ is strictly positive if the schedule of at least one teacher comprises a number of hours which is not a linear combination of the number of periods in a day and in a half-day. For solving the course scheduling problem described above, we will have recourse to the tabu search principle. A description of this method in a general framework is given in the next section.

4. Basic principles of tabu search

We give here a brief review of the basic ideas of the tabu search heuristic (TS). For a more detailed presentation of TS, the reader is referred to [10,11,20].

TS is a general search procedure devised for finding a (hopefully) global minimum of a function f defined on a feasible set X. For each solution s in X, we define a neighborhood $N(s)$ which consists of all feasible solutions that can be obtained by applying to s a simple type of modification m.

The procedure starts from an initial feasible solution and tries to reach a global optimum of the problem by moving step by step. Whenever a feasible solution s has been reached, we generate a subset V^* of $N(s)$ and we move to the best solution s^* in V^*. If $N(s)$ is not too large, it is possible to take $V^* = N(s)$. The use of the best move criterion in TS is based on the supposition that moves with higher evaluations have a higher probability of leading to an optimal (or near-optimal) solution or of leading to such a solution in a fewer number of steps. The notation $s^* = s \oplus m$ means that s^* is obtained from s by applying modification m. In order to be able to escape from local minima, the move to s^* is made even if s^* is worse than s (i.e. $f(s^*) > f(s)$). This strategy may clearly induce cycling of the algorithm. In order to prevent this, we introduce a *tabu list* T. This list contains the modifications which were made in the last $|T|$ steps of the algorithm. When constructing V^*, we forbid the generation of solutions that are obtained from s by applying the reverse of a modification memorized in the tabu list. This principle reduces the risk of cycling since it guarantees us not to return for a given number of iterations to a solution visited previously. Unfortunately, it may also forbid us to move to some solutions which have not been reached yet. Deciding that a modification m is, at a given step, tabu or not may be too absolute. In order to have a higher degree of freedom in generating the subset V^*, it should be possible to forget the tabu status of a modification when it seems reasonable to do this. This is why we introduce for every possible value z of the objective function an aspiration level $A(z)$: a solution s' in $N(s)$ which would be 'tabu' because of list T can nevertheless be taken into account if $f(s') < A(f(s))$. The function A is called the *aspiration function*. A simple example of application of this idea is obtained by setting $A(f(s)) = f(s^\circ)$ where s° is the best solution encountered so far. In this case, we accept a tabu modification only if it leads to a neighbor solution better than s°. This criterion can on no account introduce an additional possibility of cycling.

Two rules can be defined in order to interrupt the whole TS process. The first is to stop as soon as nimax iterations have been performed without decreasing the value of the best solution obtained. We may also interrupt the procedure when the value of the current solution is close enough to the minimum value $f_{\min}$ of f which is known in certain problems [5]. This second rule avoids us executing vainly nimax additional iterations after having reached an optimal solution or a solution judged sufficiently good. A general description of the TS procedure is given in Figure 2.

5. Adaptation of tabu search

TS is a very general heuristic search method. For a given problem, several ways of applying TS techniques may be considered. There exists in fact no absolute principle. In order to get an efficient version of TS, it is sometimes necessary to include a few more ingredients based on the structure of the specific problem to be solved. We describe in this section an adaptation of TS to the timetabling problem.

```
Initialization
    generate a random feasible solution s in X;
    $s° := s$;         (best solution reached so far)
    niter := 0;        (iteration counter)
    bestiter := 0;     (iteration at which the best solution has been found)
    $T := \emptyset$;
    initialize the aspiration function A;
While ($f(s) > f_{min}$) and (niter-bestiter < nimax) do
    niter := niter + 1;
    generate a set $V^*$ of solutions $s_i = s \oplus m_i$ in $N(s)$ such that either $m_i \notin T$ or $f(s_i) < A(f(s))$;
    choose the best solution $s^*$ in $V^*$;
    update the tabu list T and the aspiration function;
    If $f(s^*) < f(s°)$, then
        $s° := s^*$;
        bestiter := niter;
    $s := s^*$;
```

Figure 2. The general tabu search

5.1. Generation of an initial timetable

Let us call *availability* of a lecture the set of periods during which it can be scheduled if one considers the preassignment constraints and the teacher and class unavailabilities. To facilitate the generation of a feasible initial timetable, lectures are ordered according to their availability: we take first the ones with the smallest number of available periods.

Whenever we want to assign a lecture ℓ_i to a feasible period p_i (let us denote $\ell_i \to p_i$ such an operation), we take into account the highest possible among the six following overlap degrees:

degree 5: no overlaps authorized;
degree 4: no teacher and class overlaps authorized;
degree 3: no class overlaps authorized;
degree 2: no teacher overlaps authorized;
degree 1: no room overlaps authorized;
degree 0: no overlap restrictions.

The principle used to produce an initial timetable T is detailed in Figure 3.

Clearly, more sophisticated algorithms (having recourse for instance to graph coloring [19] or network flows [3]) could be developed to generate a better solution. But there is in fact little use to spend more time to find a good initial solution which is likely to be situated in a local minimum of $f(T)$ that can be left behind only by an especially strong effort.

5.2. Definition of the neighborhood of a timetable

The neighborhood $N(T)$ of a timetable T will consist of all feasible timetables which can be obtained from T by changing the period of exactly one lecture. According to this definition, two feasible timetables are neighbors if all lectures except one are scheduled at the same time in the two timetables.

```
sort lectures according to their availability;
$T := \emptyset$;
For $i := 1$ to nl do
    $d := 5$;
    While lecture $l_i$ is not assigned do
        generate randomly a period $p_i$ respecting the overlap degree d and
        such that $T \cup \{l_i \to p_i\}$ is feasible;
        If $p_i$ exists, then $T := T \cup \{l_i \to p_i\}$; else $d := d - 1$;
```

Figure 3. Generation of the starting solution

5.3. Definition of the tabu lists

In order to prevent cycling, two tabu lists T_1 and T_2 are considered. Whenever a lecture ℓ is moved from period p_1 to period p_2 ($p_1 \neq p_2$), we introduce ℓ in T_1 and the pair (ℓ, p_1) in T_2. This means that during $|T_1|$ steps of the tabu search procedure lecture ℓ is not allowed to be moved, and that during $|T_2|$ steps lecture ℓ cannot be replaced at period p_1. This second tabu list has a sense if an only if $|T_1| < |T_2|$.

5.4. Definition of the aspiration functions

Let us decompose the objective function in two parts as follows:

$$g_1(T) = \sum_{i=1}^{3} q_i \cdot f_i(T) \quad \text{where } q_i = \frac{w_i}{\text{g.c.d.}(w_1, w_2, w_3)},$$

$$g_2(T) = \sum_{i=1}^{7} w_i \cdot f_i(T).$$

The tabu status of a move from T to T^* will be dropped:

- if $g_1(T^*)$ is smaller than the lowest value of g_1 reached when we made any move from a timetable T' such that $g_1(T') = g_1(T)$; or
- if $g_1(T^*)$ is equal to the lowest value of g_1 reached when we made any move from a timetable T' such that $g_1(T') = g_1(T)$ and if $g_2(T^*)$ is smaller than the best value of g_2 reached during such a move.

Two aspiration functions $A_1(z)$ and $A_2(z)$ are considered. Initially we set $A_1(g_1(T)) = g_1(T)$ and $A_2(g_1(T)) = \infty$ for each value of g_1. Figure 4 shows how the aspiration functions are updated whenever we move from a timetable T to the best timetable T^* in V^*. The reverse move from T^* to T will be considered in this updating even though it has not been done explicitely [15].

The replacement of weights q_i by weights w_i in $g_1(T)$ does not alter the role of functions $A_1(z)$ and $A_2(z)$. Weights q_i have been considered in order to reduce significantly (if the weights w_i are chosen appropriately) the span of values $z = g_1(T)$ which we need to take into account in the two aspiration functions.

In summary, the tabu status of a move from T to T^* is cancelled if the following criterion is satisfied:

$$\{g_1(T^*) < A_1(g_1(T))\} \quad \text{or} \quad \{[g_1(T^*) = A_1(g_1(T))] \quad \text{and} \quad [g_2(T^*) < A_2(g_1(T))]\}.$$

```
$T_{1,1} := T$; $T_{1,2} := T^*$;
$T_{2,1} := T^*$; $T_{2,2} := T$;
For $k := 1$ to 2 do
If $g_1(T_{k,2}) \leq A_1(g_1(T_{k,1}))$, then
Begin
    If $g_1(T_{k,2}) < A_1(g_1(T_{k,1}))$, then
    Begin
        $A_1(g_1(T_{k,1})) := g_1(T_{k,2})$;
        $A_2(g_1(T_{k,1})) := g_2(T_{k,2})$;
    End
    else
        $A_2(g_1(T_{k,1})) := \min(A_2(g_1(T_{k,1})), g_2(T_{k,2}))$;
End;
```

Figure 4. Updating of the aspiration functions

5.5. Generation of a neighbor solution

A conflicting lecture in the current timetable T is a lecture which violates one of the relaxed constrains. From the compactness constraints point of view, a lecture ℓ_i will be conflicting if it is an isolated lecture (no lecture just before or just after ℓ_i) in the schedule of at least one of the teachers concerned by ℓ_i. For all the other relaxed constraints the notion of conflicting lecture is sufficiently clear and does not necessit further explanations. A fixed lecture (i.e. a lecture the availability of which is reduced to one period) is never considered as conflicting.

During the whole TS process, only conflicting lectures will be moved. There is little hope to improve the value of the objective function when moving a non-conflicting lecture. In fact, only the component f_6 measuring the compactness of the timetable could be reduced. This occurs in some specific situations that we do not intend to detail here.

A distinction will be made between the conflicting lectures. As long as the best solution reached so far does not satisfy the overlap constraints, we only allow the moves of lectures which do not respect at least one of the overlap constraints. This strategy allows us to reach faster hopefully good regions of the search space. As soon as an overlap-free solution has been visited, every conflicting lecture can be moved without any restriction.

When the number of conflicting lectures gets very small, the first tabu list defined in Section 5.3 may forbid all the moves leading to feasible neighbors of the current solution. In this case, we will forget the tabu status of a move if it is due to the first tabu list.

At each step of the iterative procedure, we attempt to generate nneigh different pairs (ℓ_i, d_i) where ℓ_i is a conflicting lecture in the current timetable and d_i a day of the week. Let us denote by ncand the number of conflicting lectures times the number of days per week. If ncand is smaller than nneigh, then only ncand pairs (ℓ_i, d_i) will be considered. We determine for every lecture ℓ_i the best feasible period p_i of the day d_i. If lecture ℓ_i is already scheduled during the day d_i, then p_i must be different from the current period of ℓ_i. The best of the moves $(\ell_i \rightarrow p_i)$ generated is retained for constructing the next timetable.

Since the minimum value of the objective function is not known, the search process will be stopped if no improvement of the best solution found has been made for nimax iterations.

5.6. Diversification of tabu search

An intelligent search technique should not only focus on regions of the set X that contain good solutions previously found but should also try to ensure that no region has been completely neglected. To achieve a diversification of the search procedure we will guide the process to regions of X that contrast with those examined so far in reducing drastically the weights w_1, w_2, w_3 ($w_i \rightarrow w_i'$ for $i = 1,2,3$). Since they are greater than the others, three weights force the search process to visit solutions which are relatively good from the overlap constraints point of view. Thus it is unlikely to leave completely the region of the search space in which we are. In reducing w_1, w_2, w_3, we concentrate on the less important relaxed constraints (the diversification is not completely random) and have consequently a higher degree of mobility. Such a diversification strategy will be performed for ndiviter iterations after having visited an overlap-free timetable and having made a certain number $k \cdot \eta(\text{nimax})$ of iterations without improving the best solution reached thus far ($k = 1, \ldots, \lfloor \text{nimax}/\eta(\text{nimax}) \rfloor$-1).

6. Computational results

We call COSTA (for Computing an Operational Schedule with a Tabu Algorithm) the TS method described in the previous section. The program was written in Pascal and implemented on a SILICON GRAPHICS personal workstation (1.6 Mflops). It has already been used for the construction of several course schedules. We present in this section the performance of COSTA and the results we have

w_1	w_2	w_3	w_4	w_5	w_6	w_7
100	100	50	3	10	1	1

w'_1	w'_2	w'_3
2	2	1

$\|T_1\|$	$\|T_2\|$	nimax	η(nimax)	ndiviter
2	20	8000	nimax div 4	100

Figure 5. Choice of parameters

obtained in computing the timetables of a high school and a secondary school. We will first briefly discuss the choice of the parameters used.

6.1. Choice of parameters

Several parameters need to be adjusted in the algorithm. To determine good values of these parameters we have run the program with different data sets. It appears that the various weights w_i and the lengths of the two tabu lists do not depend on the size of the problem and can consequently be kept constant. This is not the case for the value nneigh. We have observed that it should be proportional to the number nl of lectures. Good results have been obtained in setting nneigh = nl div 8. Figure 5 presents the values of the parameters we have chosen.

In the two following applications of COSTA, it has been decided to limit to two the number of hours of identical lectures that can be scheduled during the same day.

6.2. First application

COSTA has been successfully applied for scheduling 375 courses at the high school of Porrentruy, a town close to the French border in the north of Switzerland. We had to handle 780 lectures, 65 teachers, 32 classes, 37 subjects and 12 different types of special rooms. About 10% of the lectures were grouped lectures. All the lectures had to be scheduled on 50 periods (i.e. 5 days of 10 periods each). There were no subject unavailabilities, no geographical, no precedence and no room preparing constraints. All the other constraints had to be taken into account.

Our algorithm produced a timetable without overlaps after 20521 iterations and 41 minutes of CPU time. It has to be noted that the first overlap-free timetable has been reached at iteration 969. The values of the objective function components for the computerized schedule are pointed out in the first row of Table 2. The second row concerns the timetable which is actually applied at the high school. Such a schedule has been obtained by hand after about two weeks of work. We observed that this hand-built schedule is slightly less compact. The person responsible for the timetable told us that he would have been very satisfied with our timetable. He is looking forward to use COSTA in the future.

Table 2

	f_1	f_2	f_3	f_6	f_7
by computer	0	0	0	722	12
by hand	0	0	0	735	12

Table 3

f_1	f_2	f_3	f_4	f_5	f_6
0	0	0	1	0	380

6.3. Second application

In our second application of COSTA we have scheduled 162 courses for the secondary school of Sierre, a town in region Valais in the south of Switzerland. We have dealt with 355 lectures (15% of them were grouped lectures), 24 teachers, 12 classes, 27 subjects, 5 types of special rooms, 12 ordinary rooms and 2 distant buildings. As all the lectures to be given in the second building (the sports hall) were preassigned, no geographical requirements had to be taken into account. For pedagogical reasons, the various lectures of French, English, German and Mathematics (all in all 153 lectures) should not in principle be scheduled on the last period of every day. We needed not to balance the lunch break, because it was assigned daily to a fixed period. Except for the precedence and the room preparing constraints, all the other constraints had to be considered.

Though the volume of data was less important than in our first application, this second course scheduling problem was more difficult to solve. We had to handle 21 grouped lectures that required 4 ordinary rooms and involved 3 classes and 4 teachers. Thus everytime one of these grouped lectures is planned, it is necessary to schedule at least one lecture requiring a special room. This is due to the fact the number of classes is equal to the number of available ordinary rooms. This lack of rooms led us to consider an ordinary room as a new type of special room.

The lectures must be scheduled from Monday to Friday (50 periods) and once every fortnight on Saturday morning (4 periods). In agreement with the timetable constructor, we modelled this situation in considering 60 periods (6 days of 10 periods) and in decreeing that all the classes are unavailable for the last eight periods of the sixth day. The lectures scheduled on periods 51 and 52 will be doubled and given once every fortnight. This obliged us not to assign double lectures to these two periods.

Since no timetable has ever been produced with the considered data, we cannot compare our computerized schedule (the main characteristics of which are presented in Table 3) with a pre-existent one. The schedule we have obtained necessitated 48695 iterations and 79 minutes of CPU time, it is actually applied at the secondary school of Sierre. Only a few manual adjustments have been effected by the timetable planner. He managed (after about one hour of work) to diminish a little the number of isolated lectures in the teacher schedules. We have been told that the establishing by hand of a similar timetable would have required at least two weeks of work.

7. Concluding remarks

The present work shows that it is possible to find good solutions to the timetable problem using a procedure based on TS. We have seen that various special constraints can easily be taken into account. This is certainly an advantage over many other methods.

With little effort, COSTA can be modified to serve a variety of scheduling environments where there are competing objectives and multiple resources.

One has to admit that the computation time required for finding an operational schedule with COSTA is rather important. Nevertheless, the use of a computer saves a considerable amount of labour and permits to solve large and complex problems. Furthermore, the different computerized schedules we have produced are as good as (if not better than) manually constructed timetables. In practice, it is easy to compute a timetable satisfying the overlap, preassignment and unavailability requirements. Most of the CPU time is spent in order to satisfy the less important relaxed constraints.

It must be observed that the two most difficult requirements to satisfy are the distribution and in compactness constraints. In the previous section we decided that no class attend more than two hours of similar lectures in the same day. Clearly the ideal would be to limit to one the number of similar lectures allowed during the same day. Nevertheless, in this case there is little chance to obtain a very compact schedule. Teachers have to admit that individual convenience and the scheduling of all lectures are inconsistent goals and that it is sometimes impossible to guarantee both. Further research is being carried out along this line.

Since there are no standard test problems or measures of difficulty, it is difficult to judge the value of a given timetable algorithm. We had no opportunity to compare the efficiency of COSTA with other existing methods. Other general search heuristics such as simulated annealing [1] and genetic algorithms [4] have been applied to the timetabling problem. The use of TS in our approach was motivated by the fact that TS has proven to be one of the best algorithms among the iterative search procedures known today. Even though there is no convergence guarantee with TS, its superiority over simulated annealing when dealing with problems related to graph colorings was shown in [5]. Although we cannot claim to have found a general solution to the timetable problem, it should be noted that the results we have obtained are very satisfactory and encouraging for future research.

References

[1] Abramson, D., "Constructing school timetables using simulated annealing: Sequential and parallel algorithms", *Management Science* 37 (1991) 98–113.

[2] Aubin, J., and Ferland J.A., "A large scale timetabling problem", *Computers and Operations Research* 16 (1989) 66–77.

[3] Chahal N., and de Werra, D., "An interactive system for constructing timetables on a PC", *European Journal of Operational Research* 40 (1989) 32–37.

[4] Colorni, A., Dorigo, M., and Maniezzo, V., "Genetic algorithms: A new approach to the timetable problem", Research Report, Dip. di Elettronica, Politecnico di Milano, 1990.

[5] Costa, D., "On the use of some known methods for *T*-colorings of graphs", to appear in *Annals of Operations Research* on Tabu Search.

[6] Defrenne, A., "The timetabling problem: A survey" *Cahiers du Centre d'Études de Recherche Opérationnelle* (1978) 163–169.

[7] Dubois, N., and de Werra, D., "EPCOT – An Efficient Procedure for Coloring Optimally with Tabu Search", to appear in *Computers and Mathematics with Applications.*

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

[9] Fiechter, C.N., "A parallel Tabu Search algorithm for large Travelling Salesman Problems", Research Report 90-1, DMA, Ecole Polytechnique Fédérale de Lausanne, 1990.

[10] Glover, F., "Tabu Search – Part I", *ORSA Journal on Computing* 1 (1989) 190–206.

[11] Glover, F., "Tabu Search – Part I", *ORSA Journal on Computing* 2 (1990) 4–32.

[12] Hansen, P., and Jaumard, B., "Algorithms for the maximum satisfiability problem", RUTCOR Research Report 43–76, Rutgers University, 1987.

[13] Hertz, A., "Tabu Search for large scale timetabling problems", *European Journal of Operational Research* 54 (1991) 39–47.

[14] Hertz, A., "Finding a feasible course schedule using Tabu Search", *Discrete Applied Mathematics* 35 (1992) 255–270.

[15] Hertz, A., and de Werra, D., "The Tabu Search metaheuristic: How we used it", to appear in *Annals of Mathematics and Artificial Intelligence* 1 (1990) 111–121.

[16] Junginger, W., "Timetabling in Germany – A survey", *Interfaces* 16 (1986) 66–74.

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

[18] Tripathy, A., "School timetabling – A case in large binary integer linear programming", *Management Science* 30 (1984) 1473–1489.

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

[20] de Werra, D., and Hertz, A., "Tabu search techniques: A tutorial and an application to neural networks", *OR Spektrum* 11 (1989) 131–141.