A Tabu Search algorithm for assigning teachers to courses

Ramón Alvarez-Valdés, Francisco Parreño, José Manuel Tamarit

Departamento de Estadística e Investigación Operativa

Universitat de Valencia, Spain.

ABSTRACT

In this paper we deal with the problem of assigning teachers to courses in a secondary school. The problem appears when a timetable is to be built and the teaching assignments are not fixed. We have developed a tabu search algorithm to solve the problem. The parameters involved in the algorithm have been estimated by using multiple regression techniques. The computational results, obtained on a set of Spanish secondary schools, show that the solutions obtained by this automatic procedure can be favourably compared with the solutions proposed by the experts.

Keywords: Assignment, Timetabling, Heuristics, Tabu Search
AMS subject Classification: 90C27
Running title: Assigning teachers to courses

1.- INTRODUCTION

The school timetabling problem consists of scheduling a set of meetings between teachers and groups of students over a set of time periods. The schedule has to take into account the characteristics of the school week, the limited availability of teachers and rooms, the requirements of each subject to be taught, preassignments and other constraints, as well as the characteristics and regulations of the education system, which varies from country to country.

When the planner starts to deal with a timetabling problem, he/she may find one of the following two situations. Sometimes, the teachers' assignments, that is, the sets of subjects and the sets of groups of students to teach, are fixed. More frequently, there is some flexibility in these assignments. For instance, a teaching duty can be expressed as "Teacher A has to teach 3 out of 5 groups of Chemistry at level 3 and 2 out of 5 groups of Physics at level 4", without stating explicitly which groups. This flexibility adds a further degree of complexity to the timetabling problem. In order to make it more simple we can decompose it into two phases, solving first the assignment of teachers to subjects and groups, and then the resulting timetabling problem. This decomposition also allows the planner to use some of the existing algorithms for the standard school timetabling problem (see Carter and Laporte (1998) for a review).

The problem of assigning teachers to subjects and groups for the University timetabling problem has been studied by many authors. Tillet (1975) was the first to address it, using an integer linear programming model. Then other authors like Shih and Sullivan (1977), Schniederjans and Kim (1987) and also Badri et al. (1998) developed linear programming models. A different approach is followed by Dyer and Mulvey (1976), Bloomfield and McSharry (1979), Mulvey (1982) and Dinkel et al. (1989) who developed capacitated flow models for the problem. However, we have not found any paper dealing with this assignment problem for secondary school timetabling problems.

In this paper we develop an algorithm for assigning teachers to subjects and groups as a first phase for solving the Spanish school timetabling problem. This algorithm combines a constructive procedure to build an initial solution with an improving procedure based on Tabu Search techniques.

The remainder of the paper is organised as follows. First, we describe the problem and provide an integer programming formulation. Then, we develop the algorithms and present the computational results obtained on a set of Spanish secondary schools. Finally, we draw some conclusions and indicate future developments.

2.- DESCRIPTION OF THE PROBLEM

Our school timetabling problem has the following elements:

- *A set of levels*, defined by the education system. A Spanish secondary school usually has 4 or 6 levels.
- *A set of classes,* the groups into which each level is divided. Each class is a set of students with essentially the same teaching programme. There may be some optional subjects for which the class splits and /or merges with others, but that does not modify the class structure substantially.
- *A set of subjects,* to be taught at each level. Some subjects are compulsory and are taught at all the classes of the level. Others are optional and are taught only at one or more classes of the level.
- *A set of courses,* the subjects taught at each class. Each course has a fixed number of periods per week.
- *A set of blocks*. A block is a set of courses taught simultaneously, involving several classes and teachers. A simple case would be a block of languages (English, French, German). The students of two classes merge and then split into three subgroups to receive one of these three courses, with three different teachers.
- *A set of teachers*. For some teachers the courses to teach may be fixed. For others, their requirements are expressed as the number of times they teach a subject at a level, for instance, "Mathematics to 2 classes of level 1 and 3 classes of level 2".
- *A set of periods* of the school week, including specifications of the breaks and morning/afternoon sessions.
- *A set of non-available periods* for each class, teacher or subject.

The problem considered in this paper consists of assigning courses to teachers in such a way that all the courses are assigned and the requirements of teachers are satisfied. The objective is to obtain an assignment that facilitates the solution of the second phase of the timetabling problem in which the periods to teach each course are assigned. If all the teachers, classes and subjects were available at all the periods, any assignment would do it. However, in real problems there are non-available periods for classes, subjects and teachers. Hence, a careless assignment of teachers to courses may mean that in a given period very few, if any, courses can be taught to a class, thus making the solution of the timetabling problem very hard, or even impossible. Therefore, when we build an assignment, we try to maximise the minimum number of courses available for each class at each period.

3.- AN INTEGER PROGRAMMING FORMULATION

We have the following components:

- Let **L** be the set of school levels
- Let **C** be the set of classes. For each class *c,* we have $c \in C_l$, where C_l is the set of classes of level *l*.
- Let **T** be the set of teachers.
- Let **B** be the set of blocks.
- Let **P** be the set of periods of the school week.
- Let **S** be the set of subjects.
- Let **SC**={*(c,s)/ s* is a subject being taught at class *c*}. The elements of **SC** are *simple courses*.
- Let **SB**={*(b,s)/s* is a subject involved in block *b*}. The elements of **SB** are *block courses.* For each block course *(b,s,)* we have $(b,s)\in SB_l$, where SB_l is the set of block courses in level *l*.
- Let t_b = number of teachers who have to be assigned to block *b*
- Let **R**=$\{r_{tls} / \ l \in L,\ s \in S,\ t \in T\}$ be the requirement matrix of teacher *t*, where r_{tls} is the number of courses of subject *s* teacher *t* teaches at level *l.*
- Let **AT**=$\{at_{tp} / \ t \in T,\ p \in P\}$ be the availability matrix of teacher *t*. That is:

$$at_{tp} = \begin{cases} 1 \text{ if teacher } t \text{ is available at period } p \\ 0 \text{ otherwise} \end{cases}$$

- Sea **AC**=$\{ac_{csp}/(c,s) \in SC,\ p \in P\}$ be the availability matrix of course *(c,s):*

$$ac_{csp} = \begin{cases} 1 \text{ if simple course } (c,s) \text{ can be taught at period } p \\ 0 \text{ otherwise} \end{cases}$$

- Sea **AB**=$\{ab_{bp}/ \ b \in B,\ p \in P\}$ be the availability matrix of block *b*:

$$ab_{bp} = \begin{cases} 1 \text{ if block } b \text{ is available at period } p \\ 0 \text{ otherwise} \end{cases}$$

We define the variables:

$$x_{tcs} = \begin{cases} 1 \text{ if teacher } t \text{ is assigned to simple course } (c,s) \\ 0 \text{ otherwise} \end{cases}$$

$$y_{tbs} = \begin{cases} 1 \text{ if teacher } t \text{ is assigned to block course } (b,s) \\ 0 \text{ otherwise} \end{cases}$$

$$u_{bp} = \begin{cases} 1 \text{ if block } b \text{ can be taught at period } p \text{ because the assigned teachers are available} \\ 0 \text{ otherwise} \end{cases}$$

h_{cp} = number of available teachers for teaching class c at period p.

h_c = minimum number of available teachers for class c over all periods

z = minimum number of available teachers over all classes and all periods.

v_b = number of available periods for block b

w = minimum number of available periods over all blocks.

Then, solving the assignment problem consists of solving the following integer programming problem:

$$Max\ z + w \qquad (1)$$

Subject to the constraints:

$$\sum_{t\in T} x_{tcs} = 1 \qquad \forall (c,s) \in SC \qquad (2)$$

$$\sum_{t\in T} y_{tbs} = 1 \qquad \forall (b,s) \in SB \qquad (3)$$

$$\sum_{c\in C_l} x_{tcs} + \sum_{(b,s)\in SB_l} y_{tbs} = r_{tls} \qquad \forall t \in T, \forall l \in L, \forall s \in S \qquad (4)$$

$$1\text{-}u_{bp} \le t_b - \sum_{t\in T} at_{tp} ab_{bp} y_{tbs} \le t_b (1 - u_{bp}) \qquad \forall p \in P, \forall b \in B \qquad (5)$$

$$\sum_{t\in T} \sum_{(c,s)} at_{tp} ac_{csp}\ x_{tcs} = h_{cp} \qquad \forall p \in P, \forall c \in C \qquad (6)$$

$$h_c \le h_{cp} \qquad \forall p \in P, \forall c \in C \qquad (7)$$

$$z \le h_c \qquad \forall c \in C \qquad (8)$$

$$\sum_{p} u_{bp} = v_b \qquad \forall b \in B \qquad (9)$$

$$w \le v_b \qquad \forall b \in B \qquad (10)$$

Constraints (2) and (3) ensure the assignment of all simple and block courses. Constraints (4) are the teacher requirements. Constraints (5) relate the teacher assignment to blocks, y_{tbs}, with the block availability, u_{bp}. Constraints (6)–(10) define the auxiliary variables.

The objective function has two terms. First, we want to maximise z, the minimum number of available teachers for all classes and periods, trying to avoid classes for which, in some periods, the number of available teachers is too low and building the timetable becomes very difficult. Second, we give special treatment to blocks, because they are the most difficult pieces to fit into a timetable. This difficulty would be lower if the teacher assignment gave them the maximum number of available periods.

We have not solved this integer programming formulation for two reasons. On the one hand, the problem can be quite large and solving it by using an integer programming code can take too long. On the other hand, we would like to develop an algorithm which could be embedded as a module in a more general timetabling package, and these packages do not use linear programming techniques. This algorithm is described in the next section.

4.- A HEURISTIC ALGORITHM

4.1.- BUILDING AN INITIAL SOLUTION

The teacher assignment problem can be decomposed into two parts: the assignment of teachers to simple courses and their assignment to block courses. Hence, we solve it in two steps. First, we assign the block courses and then the simple courses. When assigning teachers to a block we try to select a set of teachers whose sets of non-available periods are as similar as possible, because the set of non-available periods for the block will contain the union of the those sets. The algorithm scheme appears in Figure 1.

FIGURE 1.Teacher assignment to blocks

```
Order the blocks by non-increasing number of non-available periods
While there are non-assigned blocks.
   {
   Take the next block.
        {
        While there are non-assigned block courses:
            {
              Select the teacher producing the lowest increase in
             the number of non-available periods of the block.
              Update the set of non-available periods of the block.
             }
        }
   }
```

As a second step, we assign the simple courses. We define the *frequency vector* of each class as the vector whose *i-th* component indicates the number of periods this class has *i* available teachers. The dimension of the vector of each class is the number of courses of the class.

When deciding which teacher to assign to a simple course we select the teacher producing the lexicographically minimum frequency vector for the class. The first components of the frequency vector correspond to the most conflictive periods and their value should be kept to a minimum.

FIGURE 2. Teacher assignment to simple courses

```
Build the initial frequency vectors for the classes
   Put the classes in order by lexicographically non-increasing
frequency vectors
   While some classes still have courses to be assigned to them
   {
   Take the next class.
        {
        While all the simple courses are not assigned:
            {
            Select the teacher producing the lexicographically
            minimum frequency vector for the class.
            }
        }
   }
```

4.2.- A TABU SEARCH ALGORITHM

Starting from the solution built in Phase I, we try to improve it by using an algorithm based on Tabu Search techniques. A general introduction to Tabu Search may be found in Glover and Laguna (1997). The basic components of our algorithm are the following:

- *The solution space X.*

The solution space in which we move is the set of assignments satisfying all the problem constraints.

- *The objective function f.*

The objective is to minimise the sum of *f(c)* over the classes $c \in C$, where, for each class *c* with frequency vector *v*:

$$f(c)=10^6*v[1]^2+10^5*v[2]^2+\dots +10*v[5]^2+v[6]^2+v[7]^2/2+v[8]^2/3+\dots+v[n]^2/(n-5)$$

The coefficients, 10^6, 10^5,...10,1, 0.5, 0.33,..., reflect the importance given to the first vector components, which should be kept to zero or very low values. The vector values, *v[i],* are squared looking for a balance between classes. For instance, we would prefer a solution with two classes, *i* and *j,* with *v[i]=1* and *v[j]=1* to another with *v[i]=2* and *v[j]=0*.

- *The neighbourhood N(s)*

A solution *s'* is a neighbour of a solution *s*, if *s'* can be obtained from *s* by making a move. The moves are defined as follows: Let us assume that in solution *s*, teacher *P* is assigned to class *c* and teacher *P'* to class *c'*. Solution *s'* is obtained by changing the assignment of *P* to *c'* and *P'* to *c*. There are three possible moves:

- Changes between teachers assigned to simple courses (Figure 4).
- Changse between a teacher assigned to a simple course and a teacher assigned to a block. (Figure 5)
- Changes between teachers assigned to blocks.(Figure 6)

FIGURE 4.

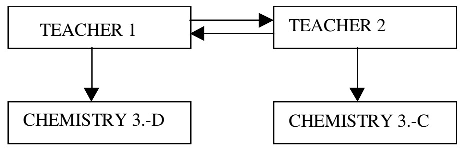


FIGURE 5.

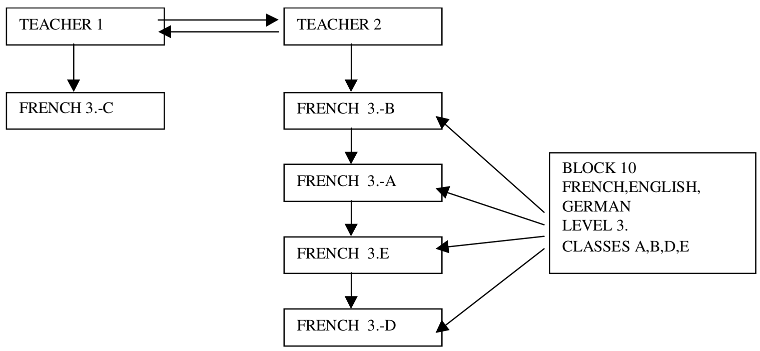
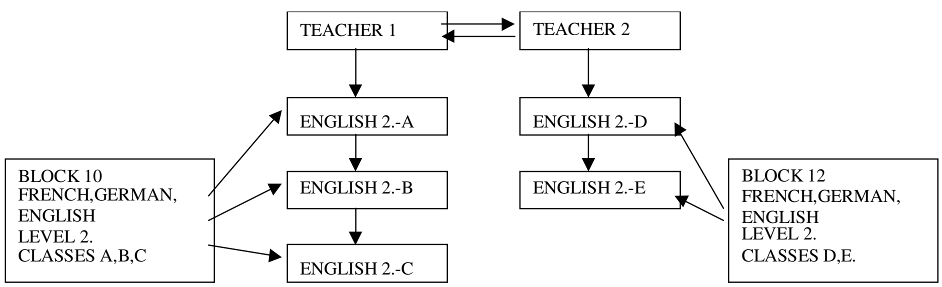


FIGURE 6.



- *The tabu list*

The tabu list contains the original assignments of teachers to classes. If, before the move, in solution *s* teacher *P* is assigned to class *c* and teacher *P'* is assigned to class *c'*, the list will include two pairs *(P,c)* and *(P',c')*. A move is considered tabu if the assignment resulting from the move contains any pair from the tabu list.

- *Aspiration criterion*

If a move produces a solution s' improving the value of the best known solution, this move is made in spite of its tabu status.

- *Selection of moves*

In order to get a fast improvement of the objective function, we will focus our attention on the class with largest weight in the objective function. We consider all the moves involving an assignment in that class and select the move producing the best value for the objective function. Based on that idea, we have developed two strategies for selecting the move:

- *Strategy A*

This strategy consists of always selecting the class with the largest weight in the objective function in which a move can be performed. We only evaluate moves involving that class and make the best one from among those which are non-tabu or satisfy the aspiration criterion.

FIGURE 7

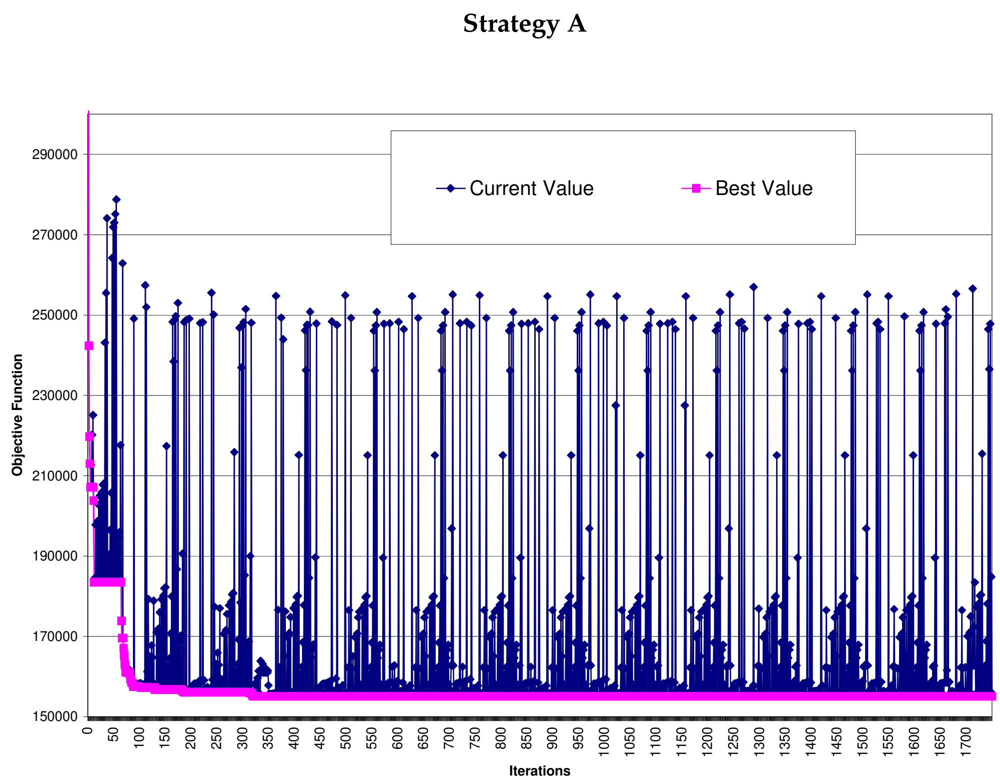


Figure 7 shows the performance of the algorithm with Strategy A on an example. In this example, the moves are always selected in the same class, because this class always has the largest weight in the objective function. As the search goes on, more moves are considered tabu, the number of possible moves decreases and the moves finally selected produce large increases in the objective function. This objective function decreases again when some good moves lose their tabu status and are available again, but the search keeps exploring a small area of the solution space and may even cycle.

The risk of cycling can be avoided by increasing the tabu list size, but some studies have shown that long tabu lists prevent the search from obtaining good quality solutions (Skorin-Kapov (1990).

- *Strategy B.*

In order to diversify the search, we introduce a random element. The class in which to study the possible moves is selected at random, with a probability proportional to its weight in the objective function. Classes with large weights are selected more often, but other classes can also be considered.

FIGURE 8

Strategy B

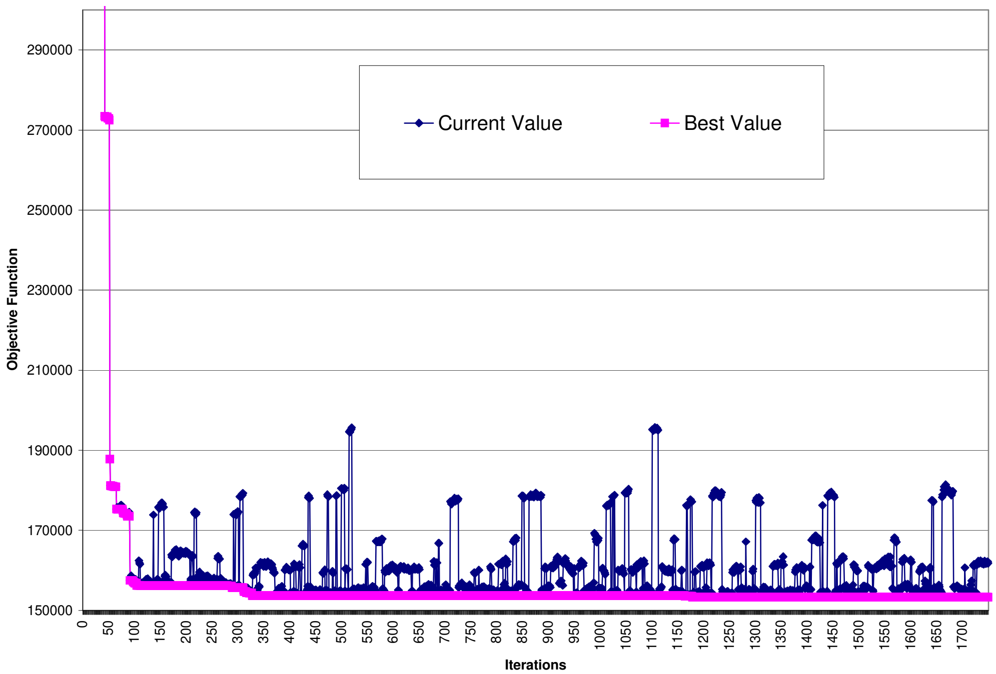


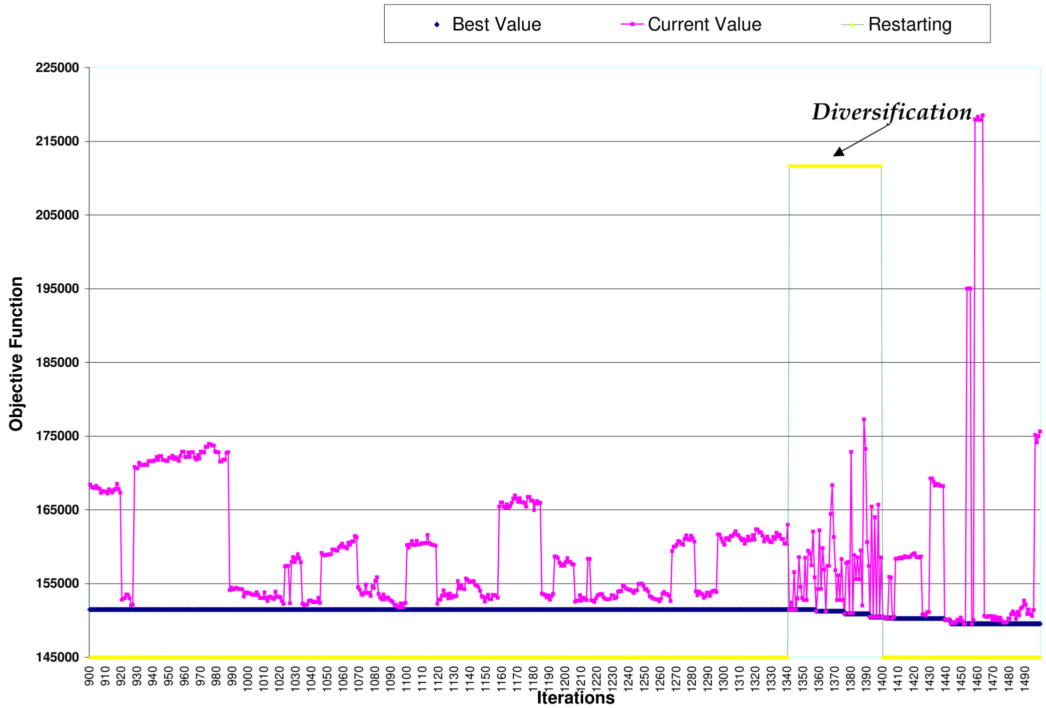
Figure 8 shows the performance of the algorithm with Strategy B. At each iteration there are more choices and the objective function does not undergo the large increases of Figure 7. Also, the possibility of cycling is avoided.

- *Long term memory*

Apart from the short term memory kept in the tabu list, we use a long term memory based on frequencies, the numbers of times each class is involved in a move during the search process.

When we have performed a given number of iterations without improving the best known solution, we recover this solution and start a new phase of the search from it. In this phase we take the frequencies into account. The class in which to make a move is selected at random with a probability defined as the ratio between its weight in the objective function and its frequency. Hence, classes that have been considered many times in previous iterations will have lower probabilities of being considered now and the search will explore new regions of the solution space. This diversification phase is maintained for a given number of iterations and then we go back to the usual selection process based on strategy B.

FIGURE 9.



In Figure 9 we observe an example in which the diversification phase produces new best solutions after restarting.

4.3. ESTIMATING THE SEARCH PARAMETERS

As Barr et al. (1995) point out, if a heuristic algorithm depends on some parameters, it is very important to select the best values for them.

In our case, we have to estimate 3 parameters:

- *Tabusize*, the size of the tabu list.
- *Nonimprove,* the number of iterations we allow the algorithm to perform without improving the best known solution before we start a diversification phase.
- *Iterfre,* the number of iterations in a diversification phase.

The value of these parameters depend on the problem size n, the number of courses to be assigned. We have selected the following possible values:

- *Tabusize* (14 values):

$$\left\{\frac{n}{4},\frac{n}{4}+\left(\frac{n}{4}\Big/13\right),\frac{n}{4}+2\cdot\left(\frac{n}{4}\Big/13\right),\frac{n}{4}+3\cdot\left(\frac{n}{4}\Big/13\right),\ldots,\frac{n}{2}\right\}.$$

- *Nonimprove* (5 values):

$$\{2\cdot n, 2\cdot n+(2\cdot n/4), 2\cdot n+2\cdot(2\cdot n/4), 2\cdot n+3\cdot(2\cdot n/4), 4\cdot n\}.$$

- *Iterfre* (14 values):

$$\left\{\frac{n}{8},\frac{n}{8}+\left(\frac{3\cdot n}{8}\Big/13\right),\frac{n}{8}+2\cdot\left(\frac{3\cdot n}{8}\Big/13\right),\frac{n}{8}+3\cdot\left(\frac{3\cdot n}{8}\Big/13\right),\ldots,\frac{n}{2}\right\}.$$

We then have the 3 parameters as 3 independent variables and the objective function as a response variable Y, and the model is:

Y=ϕ(*Tabusize, Nonimprove, Iterfre*)+ε

where ϕ(*Tabusize, Nonimprove, Iterfre*) is a regression surface that we will assume can be adequately approximated by a polynomial function in a neighbourhood of (*Tabusize, Nonimprove, Iterfre*) and ε is a random error. The most common situation is that $\varepsilon \sim N(0,\sigma^2)$. For the local estimation of the polynomial we use weighted estimation with weights ω. Once we determine the most parsimonious model, that is, the model with fewest parameters producing a good fit, we determine the values of the corresponding parameters producing a minimum for the mean response.

The local regression allows us to compare nested models. We compared the possible models in order to know which parameters are the most influential. *Tabusize* appeared to be the most important parameter, though the other two parameters also have influence and are included in the model.

Figure 10 shows the level curves of the response surface adjusted to our data. Dark zones correspond to low values of the objective function and light zones to large values. Each square represents a value of parameter Nonimprove. For each value of Nonimprove we observe a central range of values of Tabusize and Iterfre producing the minimum values.

Concretely, the best values for the three parameters are:

$$Tabusize = \frac{n}{4} + 8 \cdot \left(\frac{n}{4} \middle/ 13 \right) \text{ ; } Nonimprove = 5\,\frac{n}{2}\text{; } Iterfre = \frac{n}{8} + 7 \cdot \left(\frac{3n}{8} \middle/ 13 \right)$$

5.- COMPUTATIONAL RESULTS

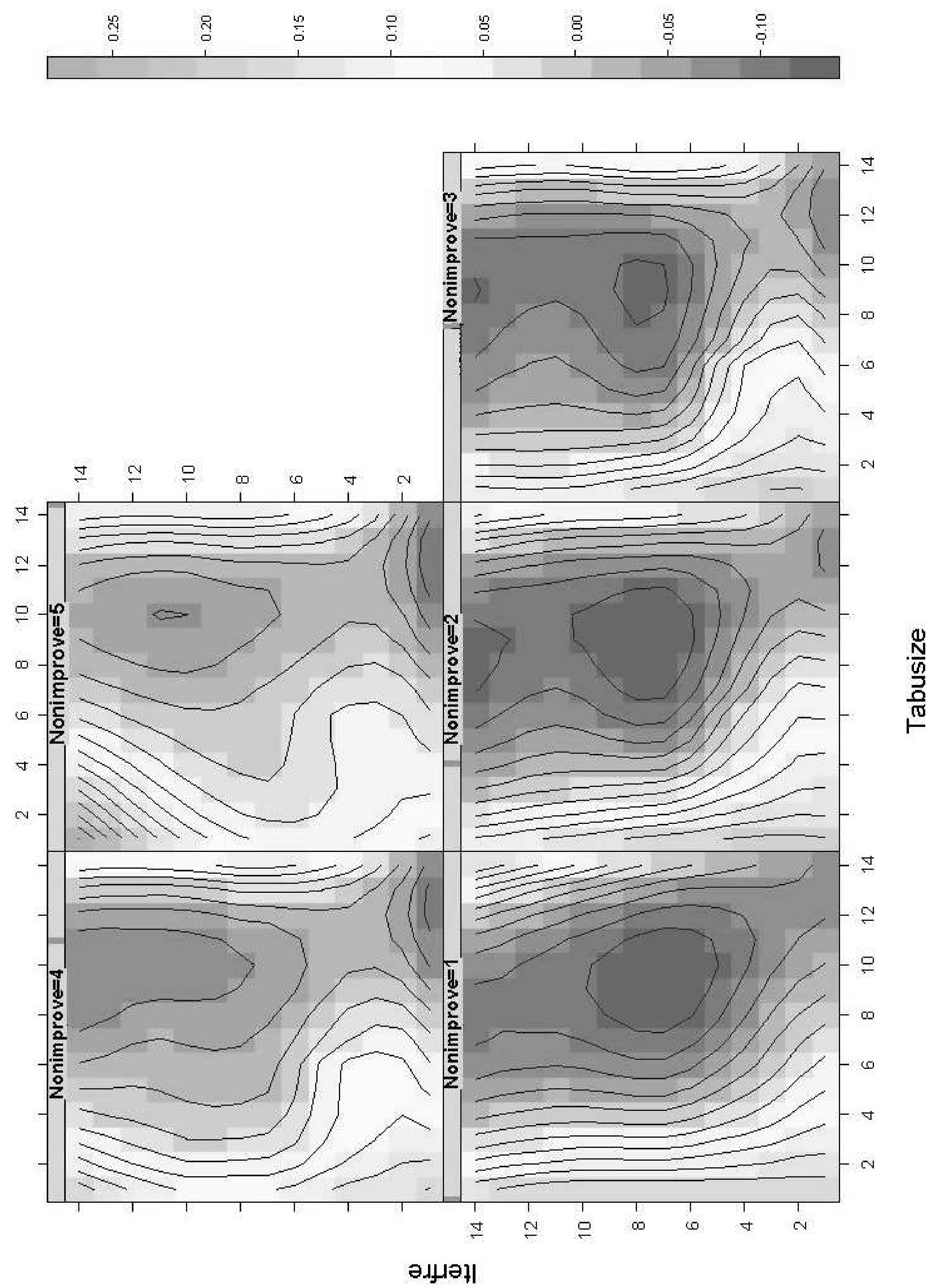
Tables 1 and 2 show the characteristics of the Spanish secondary schools used to assess the efficiency of our algorithms. Table 1 shows the size of the instances and Table 2 contains information about non-available periods. In Table 1 we have:

- *SCourses*: Simple courses.
- *STeach*: Sum of possible teachers over the set of simple courses
- *BCourses* Block courses.
- *BTeach*: Sum of possible teachers over the set of block courses

TABLE 1. Size of schools

Problem	Levels	Classes	Courses	Teachers	Blocks	NOT ASSIGNED			
						SCours	STeach	BCours	BTeach
1	4	8	98	21	11	12	24	8	16
2	4	18	208	37	10	125	340	15	19
3	4	17	218	45	21	111	256	25	24
4	4	17	218	45	21	111	256	25	24
5	4	26	342	60	27	190	684	50	127
6	4	19	244	47	23	146	392	28	44
7	4	27	1625	67	40	137	457	38	167
8	4	17	243	33	16	80	177	22	83
9	4	15	154	25	7	61	130	9	9
10	4	17	314	44	42	99	278	58	134
11	6	29	389	65	36	149	415	41	87
12	6	31	425	61	44	167	484	64	74

FIGURE 10. Response surfaces



In Table 2 we have:

- *NAClass*: Average number of non-available periods for classes.
- *NACourse*: Average number of non-available periods for simple courses .
- *NABlock*: Average number of non-available periods for block courses.
- *NATeacher*: Average number of non-available periods for teachers.

TABLE 2. Non-available periods.

Problem	NAClass	NACourse	NABlock	NATeacher
1	11.00	11.00	11.00	20.43
2	10,00	12,60	10,00	15,95
3	8,76	10,47	9,36	11,60
4	8,76	10,47	9,36	11,58
5	12,00	12,00	12,00	16,43
6	11,74	11,76	11,64	22,98
7	14,89	15,51	15,22	10,18
8	10,94	12,79	33,04	14,88
9	13,40	13,32	13,55	11,52
10	13,00	13,04	13,57	16,25
11	15,03	15,04	15,02	15,02
12	15,81	15,93	15,48	15,72

The average number of non-available periods for the different elements of the problem cannot give an accurate measure of the difficulty of a problem, because the main difficulty lies in the interaction among these elements. If the non-available periods are evenly distributed the problem is easier than the case in which the non-available periods are concentrated on some classes, on courses to be taught to those classes and on the teachers to be assigned to those courses. Some local combinations of non-available periods can make a problem very difficult or even impossible to solve.

In order to evaluate the quality of our procedure, we needed a program for solving the resulting timetabling problem. We have used the program HORES developed by Alvarez-Valdés et al. (1995, 1996). We have compared the original assignment made by the planner at each school, the assignment provided by our algorithm and a third assignment randomly generated. It should be noted that the first assignment had been made by a person with a deep knowledge of the problem and, usually, after a lengthy trial-and-error-process. We measure the quality of the solutions with three criteria: the value of our objective function *(Fobj*), the iteration in which HORES obtained the best timetable *(Iter*), and the value of the best solution obtained by HORES, which is a measure of the quality of the timetable *(Best).*

TABLE 3. Computational results.

	Original Assignment			Algorithm Assignment			Random Assignment		
Probl	*Fobj*	*Iter*	*Best*	*Fobj*	*Iter*	*Best*	*Fob*	*Iter*	*Best*
1	96364	33165	3320	21008	13903	3280	959	4824	3230
2	82568	77727	3700	8791	3850	1460	698	2101	1440
3	59285	30885	7110	16060	46589	9790	147	89473	10880
4	11835	6123	4690	2858	51079	4440	112	33905	6430
5	45474	59826	2360	5470	72878	2320	111	52088	4280
6	513737	73916	12500	32195	68474	11220	100	72064	24100
7	470849	37728	96930	147718	56875	94440	495	54030	112130
8	6863	23752	11150	3493	27050	5720	728	49924	7130
9	17965	11268	4410	14795	9975	4130	251	99988	5430
10	442414	31764	1680	5704	7123	1980	114	7855	2060
11	4414	99686	3340	4414	99246	5120	442	19807	2400
12	3545	99747	22070	3377	94677	16640	360	93289	16920
Mean	*146276*	*48799*	*14438*	*22157*	*45977*	*13378*	*166*	*48279*	*16369*

Problems with not feasible solution.

On average, our assignment is better than the other two alternatives. Nevertheless, our algorithm has not been able to provide assignments leading to feasible timetables for problems 6, 7, 12. We feel that for these problems there is no any assignment producing feasible timetables, but this cannot be proved.

The most important criterion for comparison is the quality of the resulting timetable, because the assignment phase only makes sense as a way of facilitating the timetabling phase. If we look at the results of the corresponding columns, we observe that our algorithm gets better results than the original assignment for 8 out of the 12 problems, with an average improvement of 7.5%. This improvement may seem small at first glance, but it shows that an automatic procedure can, in a very short time, obtain assignments that work similarly or even better than the assignments provided by the experts in a much more lengthy process.

The random assignment is not reliable. It can obtain good solutions in some cases (see Problem 11), but it can fail to produce an assignment resulting in a feasible solution, as in Problem 9.

6.- CONCLUSIONS.

We have developed a tabu search algorithm for assigning teachers to courses, as a first step in solving the timetabling problems for Spanish secondary schools. Tabu Search appears to be a flexible and powerful tool for obtaining high quality solutions in very short computing times. Particular attention has been given to the selection of the best values for the parameters involved in the algorithm, which have been estimated using polynomial multiple regression.

The computational results show that the automatic procedure developed here produces similar or better results than the assignment provided by the experts. Therefore, it can be used jointly with a timetabling program to solve the whole problem faced by the planners each academic year.

REFERENCES

ALVAREZ-VALDÉS R., G. MARTÍN and J.M. TAMARIT (1995). HORES: A Timetabling System for Spanish Secondary Schools. TOP 3, 137-144.

ALVAREZ-VALDÉS R., G. MARTÍN and J.M. TAMARIT (1996). Constructing good solutions for the Spanish School Timetabling Problem. JORS 47, 1203-1215.

BADRI, M.A , DAVIS D.L., DONNA F.DAVIS and HOLLINGSWORTH J (1998). A multi-objective course scheduling model: combining faculty preferences for courses and times. Computers & Operations Research. Vol. 25, No.4, 303-316.

BARR R.S., GOLDEN B.L., KELLY J.P., RESENDE G.C., and STEWART, JR W.R (1995). Designing and reporting on Computational Experiments with Heuristic Methods. Journal of Heuristics.Vol 1, 9-32.

BLOOMFIELD S.D. and MCSHARRY M.M (1979). Preferential Course Scheduling. Interfaces. Vol. 9, No. 4, 24-31.

CARTER M.W and LAPORTE G. (1998). Recent developments in practical course timetabling. In Practice and Theory of Automated Timetabling II. (E. Burke and M. Carter, eds.) , pp. 3-19. Springer-Verlag, Berlin.

DE WERRA D and HERTZ A (1989). Tabu Search Techniques: a tutorial and an application to neural networks. OR Spektrum 7,131-141.

DINKEL J., MOTE J. and VENKATARAMANAM M (1989). An efficient decision support system for academic course scheduling. Operations Research. Vol. 36, No 6, 853-864.

DYER J. and MULVEY J (1976). An integrated optimization/information system for academic department planning. Management Science. Vol. 22, No 12, 1332-1341.

GLOVER, F. and LAGUNA, M. (1997). Tabu Search. Kluwer Academic Publishers,Boston.

MULVEY, J.M (1982). A classroom/time assignment model. European Journal of Operations Research. Vol. 9, 64-70.

SCHNIEDERJANS M.J.and KIM G.K. (1987). A goal programming model to optimize departmental preference in course assignments. Computers & Operations Research. Vol. 14, No.2, 87-86.

SHIH W, and SULLIVAN J.A. (1977). Dynamic course scheduling for college faculty via zero-one programming. Decision Sciences. Vol. 8, 711-721.

SKORIN-KAPOV, J. (1990). Tabu Search Applied to the Quadratic Assignment Problem. ORSA. Vol. 2, No. 1. Winter 1990.

TILLET, P. (1975). An operations research approach to the assignment of teachers to courses. Socio-Economic Planning Sciences. Vol. 9, 101-104.