TABU SEARCH: AN EFFICIENT METAHEURISTIC FOR UNIVERSITY ORGANIZATION PROBLEMS

Ramón Alvarez-Valdés*, Enric Crespo**[1] and José M. Tamarit*
*Department of Statistics and Operations Research, University of Valencia, Valencia, Spain
**Department of Financial and Mathematical Economy, University of Valencia, Valencia, Spain

ABSTRACT
In the last 10 years we have been working on large scale academic organization problems. For the university we have worked on exam scheduling, student assignment to sections, course scheduling and a process of student self-registration. The developed algorithms are quite complex and are based on different techniques, but for all of them the Tabu Search procedures have been the essential tool for obtaining good solutions. In this paper we describe the common elements and the special characteristics of these algorithms.

Key words: timetabling, university, heuristic, tabu search.

MSC: 9008,90B40.

RESUMEN
A lo largo de los últimos 10 años hemos trabajado en problemas académicos de gran tamaño. En los aspectos referentes a la universidad hemos tratado con la programación de exámenes, la asignación de estudiantes a grupos, la programación de clases y un proceso de automatrícula de estudiantes. Los algoritmos desarrollados son bastante complejos y están basados en técnicas diversas, pero en todos ellos los procedimientos de Búsqueda Tabú han sido la herramienta esencial para obtener buenas soluciones. En este trabajo describimos los elementos comunes y las características especiales de esos algoritmos.

Palabras clave: horario, universidad, heurístico, búsqueda tabú.

1. INTRODUCTION

Academic organization problems are quite complex and involve many aspects and subproblems. Carter and Laporte [1995] propose a classification, based on scheduling aspects, in which they identify 5 different subproblems:

1. *Course timetabling.*
2. *Class-teacher timetabling.*
3. *Student scheduling.*
4. *Teacher assignment.*
5. *Classroon assignment.*

Some of these subproblems may appear together to varying degrees in the very real problems faced by academic institutions. However, in order to procedure a timetable, all such institutions have to solve at least one of the first two subproblems, depending on the institutions´ structure and objectives.

Course Timetabling is the most common problem at Spanish universities and other institutions offering postgraduate courses, where each student makes a particular choice of courses, and then has his/her specific curriculum configured. There are two types of solution methods: Master Timetable Systems and Demand Driven Systems, depending on the moment at which the students select the courses to attend. In the first case, the schedule is made first (Master Timetable) and, with this information, the students make their choice. In the second case, the students select first their courses and then a timetable is built, taking into account the choices made by the students.

Usually the university centres also have to solve the *Student Scheduling Problem*, because many courses are divided into sections and the students registered on these course have to be assigned to one or other of

[1]**E-mail**:enric.crespo@uv.es

the sections. The solution procedures range from simple alphabetical or registration order to assignment algorithms specifically designed for the problem.

When it is possible to group the students into stable set (sections), the problem is that of building timetables for these sections, and this is dependent on the availability of teachers, rooms and other necessary resources. This is *Class-Teacher Timetable*, which appears most frequently in high schools.

In our work we have studied four of these five subproblems, though in this paper we will only comment on those aspects directly related to universities. We have not dealt with the *Teacher Assignment Problem* because in our University this is an internal question for each Department. We have solved the *Course Timetabling* and *Classrom Assignment* problems simultaneously, in line with the University requirements. (Crespo [1999])

We have worked on the *Student Scheduling Problem* (Alvarez-Valdés **et al**. [1990]), which is embedded in a self-registration process in which the students select the courses, and the sections are the assigned by a global procedure that seeks to provide a good timetable for each student.

We have also solved the *Examination Timetabling* problem (Alvarez-Valdés **et al**. [1997]). It could be considered as a particular case of Course Timetabling with special characteristics, and it is possible to take advantage of these special features to design a specific algorithm.

In this paper we present our experience with Tabu Search Techniques applied to these academic organization problems. In Section 2 we describe the common elements of these problems and their specific objectives. The next Section is a brief introduction to Tabu Search. In Section 4 we expose the strategies developed in each case and the results obtained. Finally, in Section 5 we present the conclusions of our work up to the present date.

2. ELEMENTS, CONSTRAINTS AND OBJECTIVES OF ACADEMIC ORGANIZATION PROBLEMS

2.1. The elements

The problems have the following common elements:

- A set of *courses* {A_1, A_2,..., A_s}. Each course has a fixed number of hours per week which can be taught in one session or in several sessions during the week.
- According to the number of students and other considerations, each course A_i can be divided into *sections* {S_{i1}, S_{i2},..., S_{iij}} to be taught separately. For instance, our university is in a bilingual region (Spanish and Catalan) and some courses are offered in both languages.
- A set of *lesson* {L_1,...,L_2,...L_t}, which are the sessions of each section of a course. These lessons may have different lengths.
- A set of *students* registered on the courses.
- A set of *classes* {C_1, C_2,..., C_n}, sets of courses taken, basically, by the same group of students. These classes are designed by the planners, in line with the academic structure of the degrees offered and their past experience of the students´ choices. Each class will have quite a stable set of students and it is therefore very important that their lessons are not taught simultaneously.
- A set of *teachers* {T_1, T_2,..., T_t}. Each section of each course will have a teacher previously assigned to it.
- A set of *rooms* {R_1, R_2,..., R_r}, which can be of different types, including computer rooms and laboratories for specific subjects. Each room has a fixed capacity.
- A *week*, made up to a set of *periods* {P_1, P_2,..., P_p}, and divided into days. Each day has the same number of periods of equal length. The breaks and the non-teaching periods must be specified. The lessons lasting more than one period cannot be interrupted by a break.
- It is also possible to have some *preassignments* or some *forbidden periods* for classes, teachers or rooms.

In *Examination Timetabling* some of these elements do not appear. If there is one common examination per course, we do not have to consider the sections. There are no classes and there is only one lesson per course. The week is transformed into the examination period.

2.2. Objectives and constraints

Although the different problems may share some objectives, the main objectives of each problem are different.

1. **Course Scheduling**:

When we think of a good timetable we consider several types of conditions to be met. These conditions are not all of the same importance. We distinguish, on the one hand, *hard constraints*, that is, conditions that every acceptable timetable must satisfy. These constraints are:

(a) Every lesson has to be assigned to a period or a number of periods, depending on the lesson´s length.

(b) No teacher can give two simultaneous lessons.

(c) Two lessons of the same course cannot be asigned to the same day.

(d) No room can be assigned to two simultaneous lessons.

(e) The room assigned to a given lesson must be of the required type.

(f) All the preassignments and forbidden periods for classes, teachers and rooms must be re-spected.

On the other hand, there are some conditions that are considered helpful in a good timetable. The more these conditions are satisfied, the better the timetable will be. We call them *soft constraints* or objectives to be attained and therefore they will have a weight in the objective function. We group these objectives according to the elements they involve.

(a) For classes:

- Simultaneity of lessons of one class should be avoided. Usually this is the most important objective for users.
- Class timetables should be as compact as possible, eliminating idle times for students.
- The number of moves from one room to another of students belonging to one class should be minimized. Each class should have its lessons assigned to a minimum number of different rooms.

(b) For students, taken individually, the objectives are described in detail when we talk of the Student Assignment problem.

(c) For teachers, their non-desired periods should be avoided.

(d) For rooms, the objective is to match their capacity to the number of students enrolled in sections assigned to them.

(e) The section of a course should be balanced regarding the number of students assigned to them.

(f) Each university also has its specific objectives. We have tried to include in our program those which are more important and those which are usual.

2. **Examination Timetabling**:

The objectives are basically three and there is a clear hierarchy among them:

(a) No student should have two exams simultaneously. There *first order conflicts* have to be avoided. In fact, this conditions could be considered a constraint, but we cannot be sure of finding a feasible solution if we enforce it because of the multiple combinations of students' registrations and the reduced number of periods. Therefore, we put it as the main objective and we try to get a solution without first order conflicts.

(b) For each students, his/her exams should be as evenly spread as possible along the exam interval. This idea extends the usual objective of avoiding consecutive exams, back-to-back conflicts, and conditions of the type "Students should not write **x** or more examinations within any **y** periods" (Carter **et al**. [1994]).

This idea of evenly scattering the exams forms part of the non-written tradition of our university, specially in those faculties in which the examination timetabling has been done by hand after a negotiation process with teachers and students. Nowadays, the complexity of the process makes this difficult to maintain, but an acceptable computarized substitute has to include this objective.

(c) Classroom capacities should be respected. This is clearly a constraint, but not a hard one. The inclusion of some students above a classroom´s capacity is not physically impossible, although it may not be desirable. If this condition is included as a constraint, we may lose some good solutions with reduced capacity overflow that could be considered acceptable by the parties involved.

2. **Student Assignment to sections**:

The aim of the assignment procedure is to allocate students to course sections so as to obtain the best possible solution in accordance with the following criteria:

(a) student course selections must be respected.

(b) section maximum capacities should not be exceeded and section enrollments should be balanced.

(c) conflicts in student timetables (students having two classes at the same time) should be avoided.

(d) student timetables should be as good as possible. A *good* timetable is defined in several ways: the number of class periods per day, the number and length of *holes* (free time intervals between two classes), and moves between buildings and campuses.

(e) student language preferences should be respected. The students should be assigned to sections taking into account their language preferences.

The first criterion is the only hard constraint we impose. The balance between sections is considered as a constrains by using tolerances. An acceptable section enrollment balance is defined in terms of the tolerances considered admissible by the University. The other criteria are objectives to be attained and can be summarized in an overall objective of providing the best possible schedule for each student.

3. THE TABU SEARCH METHOD

Tabu Search is a metaheuristic formally introduced by Fred Glover in 1989, though he had previously written several papers on the subject. In recent years a large amount of work based on this technique has appeared, covering many different aspects of Operations Research. A good reference about Tabu Search is the book by Glover and Laguna [1997].

Unlike other metaheuristics, Tabu Search is based on the idea that an intelligent search has to perform a systematic exploration of the solution space. Therefore, the method is basically deterministic.

The basic paradigm of Tabu Search is to use the information about search history to guide local search approaches to overcome local optimality. In general this is done by a dynamic transformation of the local neighborhood, in which some possible moves are declared tabu, that is forbidden, in order to prevent cycling to formerly traversed solutions. This may lead to performing deteriorating moves when all improving moves of the current neighborhood are set tabu.

The basic elements of a Tabu Search implementation are:

- reduced neighborhoods and candidate lists: instead of considering all the non-tabu solutions, we limit the search to promissing solutions defining a candidate list. The criteria to consider a solution as promising depend on the problem characteristics and may be static or dynamic.

- short term or recency-based memory: it stores moves made in the recent past to prohibit visiting formerly searched solutions and prevent cycling. For computational efficiency, only some attributes of the solutions are stored in a tabu list.

- aspiration criteria: conditions in which a move is considered admissible in spite of its tabu status.

Besides these basic elements, the implementations of Tabu Search for complex problems include a long-term memory, keping information about the solutions visited along the search.

Tabu Search follows two basic strategies: intensification and diversification, which alternative according to the search history kept in memory. Intensification consists of examining thoroughly the neighborhood of good quality solutions. Diversifications consists of forcing the search to non visited regions.

In recent years, more sophisticated strategies have been developed (influence moves, strategic oscillation, path relinking). We have mainly used strategic oscillation. An introduction to this strategy may be found in Glover **et al**. [1993] and Glover [1996]. Strategic oscillation consists of cyclically varying some elements of the search process, for instance the objective function, thus creating an oscillating behaviour. Kelly **et al** [1993] give three reasons for using it:

- If the solution space is not connex, it provides a mechanism for crossing infeasible regions and getting new feasible solutions.
- When feasible solutions ase difficult to obtain, strategic oscillation allows us to vary the relative weights of the objective funciton and guide the search towards feasible solutions.
- Finally, strategic oscillation provides the diversification required for any heuristic which looks for high quality solutions.

4. SOME EXPERIENCES APPLYING TABU SEARCH TO ACADEMIC PROBLEMS

This section describes our experiences applying Tabu Search to the above mentioned academic problems. We discuss the different possibilities considered and explain the more successful strategies.

All the implementations of Tabu Search algorithms have a first phase in which we build an initial solution from which the search starts. This first phase usually consists of a simple greedy algorithm, as in Examination Timetabling of Course Timetabling. However, for the Student Assignment problem, a more complex method generates a solution space in which the search is performed.

4.1. Elements

For each element of the Tabu Search we analyze the developed procedures and report on their efficiency.

The solution space:

Whenever possible; it is highly convenient to define the solution space as the set of feasible solutions. If unfeasible solutions afre included, there must be a simple way of recorvering feasible.

In academic problems a detailed analysis of feasibility conditions is very important. There are constraints which are obviously hard and every feasible solution must satisfy them, for instance "every lesson has to be assigned to a period". Some other conditions are hard or soft depending on the problem. For instance, having two lesson simultaneously is a hard constraint for teachers, because it is physically impossible for them to attend two lessons at the same time, but maybe the individual timetable of a student would not have any feasible solution at all without some simultaneity conflicts. In this case we may accepted slightly unfeasible solutions, including those conditions in the objective function with a weight that makes their violation highly undesirable, though not impossible.

Imposing many conditions as hard constraints not only reduces the solution space, but makes it very difficult, if not impossible, to move within it, thus reducing the efficiency of search-based methods.

For example, in the definition of solutions for the Students assignment problem we relax the capacity and balance constraints. Therefore, we consider as a solution any global assignment of students to sections that respects the students' course registrations. However, the solutions defined in that way may include very bad individual assignments that are not considered admissible. In order to prevent the use of these assignments, in the first phase we generate a set of admissible assignments for each student and then each global solution will be composed only of admissible individual assignments.

The objective function:

In these problems we have several objectives to be attained. Therefore, the objective function is made up to several components, whose relative importance is controlled by using weights. The weights have been determined according to the information provided by the users. Moreover, in some cases, as in the application we have developed for Course Scheduling, the user may interatively modify them.

- Sometimes the assignment of weights has been static, that is, fixed at the beginning for the whole process. For instance, in Examination Scheduling the objective function was:

$$\sum_{i=1}^{N}\left(\sum_{j=1}^{N}\left(p_{|i-j|}\sum_{k\in E_i}\sum_{l\in E_j}x_{kl}\right)+w\,r_i\right)$$

We compute the cost of a solution by adding the costs of conflicts between exams for each pair of periods and the costs of exceeding the room capacities. In the first term of the function the $p_{|i-j|}$ is the cost of having a pair of exams at a distance |i - j|. In the second term, w reflects the relative weight of room shortages with respect to conflicts. All these weights are static.

- In other cases, the function varies dynamically. For instance, in the Student Assignment to sections, the function

$$\sum_{i\in I}\sum_{k\in K_i}c_{ik}x_{ik}+P\left(\sum_{j\in J}e_j\right)$$

has a firts term, reflecting the quality of individual timetables for students, and a second term that indicates the balance among the sections of a course. The relative importance of the second term is controlled by the weigth P. It is not easy to make a good estimation of P beforehand. If P is too high, the search will focus on solutions strictly respecting the balance between sections, but the students timetables would be of bad quality. On the contrary, if P is too low, the visited solutions would not attain an acceptable balance of sections. The *right* value of P will depend on the structure of the data and on the values of tolerance parameters. Therefore, the way to approximate to this value will be by adjusting it iteratively according to a strategic oscillation which will be explained in Section 4.2.

In Course Scheduling, the function to be minimized

$$\sum_{z}w_z f_z(x,y)$$

adds the cost of the several objectives describes in Section 2.2. Variable x represents the vector of assignments of lessons to periods and rooms. Vector y is the assignment of students to sections. Each objective has its corresponding weight w_z. The weights vary along the process in order to improve the search, as will be explained in Section 4.2.

The neighborhood:

The solution neighborhood, and the move defining it, is one of the most important elements, because the ability of the algorithm to search efficiently the solution space depends on it.

The type of move to define in each case depends on the solution space defined by the hard constraints.

- Sometimes the move may be simple. In Student Assignment to sections, where each solution is composed of one individual assigment for each student, the move is just changing one of these individual assignments. In Examination Scheduling, one of the moves we define consists of changing an exam from one period to another. In Course Scheduling a simple move is to change the period assigned to a lesson. Nevertheless, the constraints imposed on the problem may render this simple move almost useless. For instance, changing the period assigned to a lesson is very difficult to do if there has to be an empty room at this new period and the rooms are scarce.

- A more complex move is a swap, involving two items that interchange their assignments. In Examination Scheduling a swap consists of a pair of exams that change their periods. Another swap for the same problem is defined by the interchange of a complete list of exams assigned to a period with another complete list assigned to another period. In Course Scheduling, interchanging periods between lessons may eliminate the problem of finding rooms, because each lesson is assigned to the room previously occupied by the orher lesson.

- In the solution structure is not easily modified with these moves, we van define a new move which is more complex still. For instance, in Course Scheduling we have used a multiswap, in which a lesson interchanges periods with a set of more than one lessons. This move is very useful when the lessons involved are of different length or they are linked in such a way that they have to be moved together.

A scheme of the different moves defined for Course Scheduling appears in Figure 1.

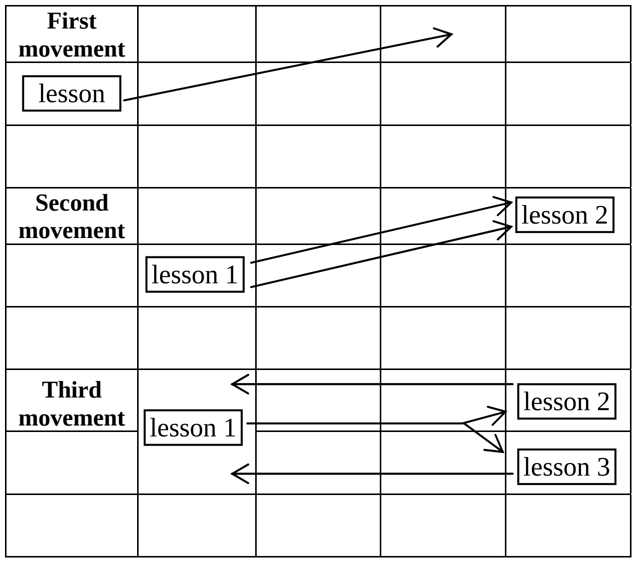

Figure 1. Scheme of the moves

The tabu lists:

In our experience, the use of simple attributes of moves to be kept in the tabu lists has always worked well. In Examination Scheduling for a simple move we keep the exam we move and its original period. For the interchange of exam lists we keep the list starting the move and its corresponding period. In Course Scheduling the tabu list stores the lesson starting the move and its original period. Nevertheless, the tabu status of all the elements involved in the move has to be checked in order to avoid cycling.

The length of tabu list was fixed in the algorithms at approximately $\sqrt{n}$, where n is the number of items in the problem. The length of the list has not been a relevant parameter and the performance of the algorithms has always been very robust for lengths in the neighborhood of this value.

In Course Scheduling we have tried two types of lists: static, that is of fixed length, and dynamic, of variable length, in which after a certain number of iterations without improvement, the new length is randomly chosen in the interval $(0.25\sqrt{n}, 2\sqrt{n})$. In this case, variable length list produce better results than fixed length lists.

The aspiration criterion:

We have always used the most simple criterion and it has worked well. If an explored move produces a solution with objective function lower than the best current solution, the move is made in spite of its tabu status.

Selection of moves:

The selection of the move to make may be done in different ways. Sometimes, the use of candidate lists is not necessary. For instance, in Examination Scheduling, if the move is interchanging periods for complete lists of exams, the whole neighborhood of a solution can be explored. If the move consists of chanching the period assigned to an exam, we always choose to move the exam which is producing more conflicts and explore all periods to which it could be assigned.

Our experience with cantidate lists has produced mixed results. Sometimes using candidate lists in which to choose the move has improved the search. On other occasions, the use of candidate lists has produced worse results than a random selection from the whole neighborhood.

An example in which the use of candidate lists leads to better results is Student Assignment of sections. We generate a candidate list of students. If the current solution has no violation of capacity, it includes the whole set of students. Otherwise, only those students involved in a violation are included in the list. Then we take a student at random from this candidate list. Finally, we generate and evaluate all possible exchanges for the student. We choose the move that improves the combined objective function the most (or degrades it the least) as long as it is not tabu or the aspirtation criterion can be applied.

For Course Scheduling we determine which lesson to move by two different methods. In the first one, we choose it randomly from among all the lessons. In the second, we make a list of candidates. If the solution has some conflicts of simultaneity among the lessons of a class, those lessons will constitute the list. If there are no conflicts of this type, all lessons such that when going out from their assigned periods produce an improvement of the objective function, make part of the list. Once the lesson has been chosen, all its possible periods are explored.

The computational results show that the average value of solutions with random selection of moves is significantly better (by 7 %) than the average value of solutions obtained using candidate lists. An illustration of the behaviour of the candidate list in Figure 2 which shows the evolution of the solutions of both strategies on an example. Although the implementation with a candidate list finds better solutions in the first iterations, in the long run it is overtaken by the implementation without canditate list. The reason may be that when working with an aggressive move, as is multiswap, the random election of the move allows the search to explore the solution space better if a large number of iterations is performed. However, the strategy without a candidate list may take many iterations to obtain solutions without simultaneity conflicts for classes, which are the largest component of the objective function. In the example of Figure 2, if we use a candidate lists, a solution without conflicts is obtained in the first 100 iterations while without a candidate list a conflict-free solution is not obtained in the first 12000. Therefore in hard problems, or when the user does not allow the algorithm to perform many iterations, we could end up without a solution free of conflicts, although they exist. As that situation would not be acceptable for the user, we designed a mixed strategy. We use a candidate list until a solution without conflicts is found and from this iteration onwards we do not use it. With this mixed strategy the results are almost as good as those obtained with the random selection and we guarantee conflict-free solutions.

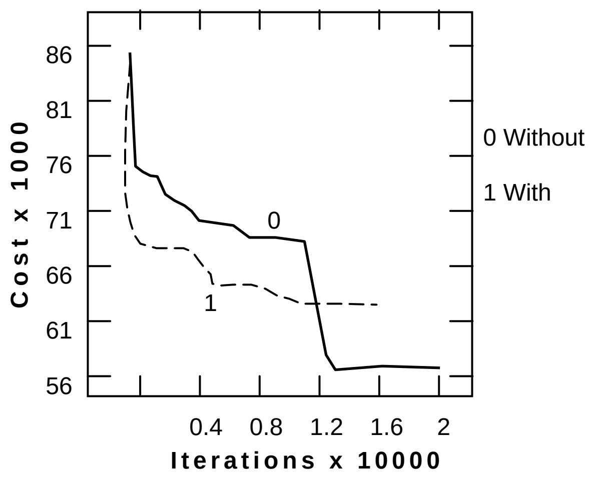

Figura 2: Evolution of the objective function with and without candidate list.

4.2. Search strategies

Among the strategies frequently used in Tabu Search applications, we focus on those which have been more important in our algorithmic implementations.

Intensificaton:

The descent method guided by the objective function is the intensification strategy used in each implementation.

Another intensification strategy is based on the use of candidate lists, restricting the search to promising solutions in order to improve the objective function faster.

The last strategy consists of recovering the best known solution after a certain number of iterations without improvement, and starting once again the search from it. We have used this strategy in Course Secheduling and the results show a slight improvement in the quality of the solution found.

Diversification:

A diversification strategy consists of alternating two types of completely different moves in the same search process. That allows us to reach points of the solution space which would not be attained by using only one of them. In Examination Scheduling we alternate simple moves, changing the period of an exam or swapping the period between two exams, with complex moves in which we interchange the periods for two complete lists of examinations. Each move, taken alone, performs an intensive search, but when we put them together produce a deep diversification in the search. the decision to change the type of move is taken after a given number of iterations with no improvement.

Another important diversification strategy is that of strategic oscillation, described below.

Strategic oscillation:

We have made two different types of strategic oscillation. In Course Scheduling we control the process, varying cyclically the weights of the objective function. More concretely, the weights corresponding to each aspect (classes, teachers, students, rooms) are set to zero for a given number of iterations. Therefore, the search is guided by the remaining components of the objective function and it explores new regions of the solution space. The computational results show a slight increase in the quality of solutions when applying this strategy.

Another implementation of a strategic oscillation corresponds to Student Assignment to sections, in which we have to adjust the value of P in such a way that we could obtain feasible solutions according to capacity and balance constraints without excessive deterioration in the quality of students' timetables. The value of P is very small at the beginning of the search and it increases linearly until a feasible solution is attained. The good moves made with low values of P will cause a very slight deterioration in the quality of the solutions for the students involved in the changes, while improving the feasibility of the solution. When moves with these features no longer exit (and the moves we make do not improve the vest solution found), we increase P looking for solutions nearer to feasibility, though at a greater cost for the students. We do many iterations at each step, trying to slow down the increase of P.

When we reach feasibility for a given value of P we consider the existence of feasible solutions for smaller values of P to be very unlikely, but the possibility cannot be completely ruled out. On the other hand, we are not willing to increase this value of P because the quality of the solutions for students will be subject to larger decreases. Nevertheless, an oscillation around that value would diversify the search and obtain better solutions. Kelly **et al**. [1993] report that periods of 6 or 8 produced the best results in their problem. We have adopted their strategy of a period of 8 (reaching two levels up and two levels down the value of P for which feasibility was reached). We have no guarantee that the problem, such as we have formulated it, has any feasible solution. It depends on the structure of the data and on the values of the parameters. Therefore, we cannot increase the value of P indefinitely. As we cannot know if this is the case, we use an indirect indication. If after three increases of the value of P we have not obtained solutions which are nearer to feasibility, we consider that going on increasing P will not end with a feasible solution, and we then direct the search to solution with good values of the objective function, though not feasible.

5. CONCLUSIONS

Tabu Search contains several elements and strategies that should be explored to produce good solutions for complex problems. That means that an efficient implementation cannot be reduced to a simple move and an intensification process. For each problem a good implementation has to combine different strategies and draw on its special characteristics. As Dowsland [1997] points out, there is no general formula, but only solutions tailored to each type of problem.

Our experience during these years of applying Tabu Search algorithms to academic organization problems allows us to conclude that it is a powerful technique for solving this kind of problems, producing high quality results for large scale instances. This experience is shared by other authors. For instance, Boufflet and Sanegre [1996] have applied it to Examination Scheduling and Hertz has used it for Course Scheduling problems in 1991 (Herts [1991]) and 1992 (Hertz [1992]. Also, Tabu Search, as a heuristic framework, is highly flexible and can be successfully adapted to very different problems.

REFERENCES

ALVAREZ-VALDES, R.; E. CRESPO and J.M. TAMARIT (1997): "A Tabu Search algorithm to Schedule university examinations", **Qüestiió** 21(1-2), 201-215.

____ (1999): "Assigning students to course sections using Tabu Search", **Annals of Ops Res**. (in press).

BOUFFLET, J.P. and S. NEGRE (1996): "Three methods uses to solve an examination timetabling problem", **Practice and Theory of Automated Timetabling**, Eds., Burke, E. and Carter, M., Springer-Verlag, Lecture Notes in Computer Science 1153, Berlin, 327-344.

CARTER, M.W. and G. LAPORTE (1995): "Recent developments in practical examination timetabling", **Practice and Theory of Automated Timetabling**, Eds., Burke, E. and Carter, M., Springer-Verlag, Lecture Notes in Computer Science 1153, Berlin, 3-21.

_____________________________ (1997): "Recent developments in practical course time-tabling", **Practice and Theory of Automated Timetabling** II, Eds., Burke, E. and Carter, M., Springer-Verlag, Lecture Notes in Computer Science 1408, Berlin, 3-19.

CRESPO, E. (1999): "Un sistema informàtic per a la construcció d'horaris i l'assignació d'aules a la Universitat de València. Ph. D. Universitat de València. Spain.

DOWSLAND, K.A. (1997): "Off-the-Peg or Made-to-Measure?" Timetatabling and Scheduling with SA and TS. **Practice and Theory of Automated Timetabling** II, Eds., Burge, E. and Carter, M., Springer-Verlag, Lecture Notes in Computer Science 1408, Verlin, 37-52.

GLOVER, F. (1996): "Tabu Search and adaptive memory programming. Advances, applications and challenges", **Interfaces in Computer Science and Operations Research**, eds. Barr, Helgasson and Kennington (Kluwer Academic Publishers).

GLOVER, F. and M. LAGUNA (1997): "Tabu search", **Kluwer Academic Publishers**.

GLOVER, F.; E. TAILLARD and D. de WERRA (1993): "A user´s guide to tabu search", **Annals of O.R.** 41, 3-28.

HERTZ, A. (1991): "Tabu Search for Large Scale Timetabling Problems", JORS 54, 39-47.

________ (1992): "Finding a Feasible Course Schedule Using Tabu Search", **Discrete Applied Mathematics**, 35, 225-270.

KELLY, J.P.; B.L. GOLDEN and A.A: ASSAD (1993): "Large-scale controlled rounding using tabu search with strategic oscillation", **Annals of Ops Res**. 41, 69-84.