A Course Timetabling System in a Mexican University

Leovigildo Lopez-Garcia[1] and Adelita Posada-Bolivar[2]

[1]Actuarial Science and Statistics Department
Universidad de las Americas, Puebla
Santa Catarina Martir, Puebla
Mexico 72820
leolopez@mail.udlap.mx

[2]Industrial Engineering Department
Universidad de las Americas, Puebla
Santa Catarina Martir, Puebla
Mexico 72820
adela@mail.udlap.mx

Abstract.- A common problem that many universities around the world face is assigning classrooms to courses. The Universidad de las Americas, Puebla, in Mexico has to assign more than 2000 courses in fewer than 200 classrooms every term. The semi-automatic process that is currently being used requires around three weeks to generate the definitive assignment, which is far from an ideal assignment time. In this paper we present a system developed in Delphi that applies the Tabu Search metaheuristic to approach the solution to this difficult problem. The system is described and the results obtained are shown and compared to those obtained by other problem-solving methods.

Keywords.- Course timetabling, Metaheuristics, Tabu Search, Delphi.

1. INTRODUCTION

Every term the Universidad de las Américas, Puebla (UDLA) in México, must assign over 2000 courses in fewer than 200 classrooms. The current system that assigns classrooms to courses allows overlapping and also fails to assign classrooms to all the courses. The proposed "solution" has to be modified in a semi-automatic way, considering several types of constraints, such as permitted schedules, classroom capacity and professors' preferences, among others. Obtaining a satisfactory solution is a complex task that takes several weeks. The problem of assigning classrooms (PAC) is an NP-complete problem which has generated a great number of studies, since it is a real problem in every university in the world. In many universities this process is carried out manually and consumes a great deal of time and resources. At present we are seeking to solve this problem automatically by means of a computer and an efficient algorithm. The Tabu Search heuristic, introduced and developed in its current form by Fred Glover (1989), has proved to be highly efficient in solving a great variety of difficult combination problems. It explores a subset of solutions by seeking to escape from local optimums, allowing numerous new areas of solutions to be explored, and makes it possible to carry out intensive searches in any of these areas might be promising in terms of offering an adequate solution. These and other characteristics indicate that this technique may prove to be useful in approaching a solution to this difficult problem. The PAC has been called to the attention of many researchers because of its utility and complexity. In the last 30 years,

many algorithms have been proposed to solve this problem. In 1986, in Waterloo University in Canada, Michael Carter implemented a program to carry out room assignments efficiently, by means of a program complete with Langangian Relaxation (Carter 1989). Michael Carter and Gilbert Laporte (in Burke and Carter 1998) have achieved implementations and testing with real data. Rey and Carlevaro (2000) successfully solved the problem of assigning classrooms in the Universidad de la República de Uruguay, for one of their schools, using the Ant System heuristic. Angela Ahuactzin and Claudia Reyes (2000) used the Ant Colony Optimization heuristic to successfully solve the Course Assignment Problem at the Universidad de las Americas, Puebla, in Mexico.

2. THE COURSE ASSIGNMENT PROBLEM

Assigning courses to classrooms is a complex task, given the scarcity of classrooms and the assignment criteria that should be met in order to accomplish the task satisfactorily. The classrooms have different characteristics to be considered, such as: type of blackboard, television, VCR, computers, computer projectors, air conditioning, etc. An acceptable assignment should meet a variety of conditions, from the most basic ones (such as the need for each course to be assigned one classroom, that there be no overlapping of courses in classrooms, that the seating capacity be adequate, and that a class offered on several days of the week be assigned to the same classroom), to other conditions which can be considered preferences (for example, that there be a minimum distance between classrooms assigned to courses belonging to the same major, or that courses with few students not occupy classrooms with a large number of chairs). The simplest formula for this problem has been presented by Carter (1989) and can be described as a binary integer programming problem in which variable xij is equal to 1 if course i is assigned to classroom j and is equal to 0 in the opposite case. The time in which a classroom can be occupied in one day is divided into k periods:

$$\sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij}$$

$$s.t.$$

$$\sum_{j=1}^{n} x_{ij} = 1, \quad \forall i$$

$$\sum_{i \in P_k} x_{ij} \leq 1, \quad \forall j, \ \forall k$$

$$x_{ij} \in \{0,1\}, \quad \forall i, \quad \forall j$$

where P_k represents the set of all courses offered in period k, and c_{ij} is the cost of assigning course i to classroom j. The first restriction ensures that each course be assigned to a classroom and the second, that in a given period, at most one course ***i*** that is offered in period k is assigned to each classroom. Implicit in this formula is the

restriction that each course be assigned to only one classroom; that is, that students should not have to change classrooms halfway through class, or occupy a different classroom for the same course on another day of the week.

According to Carter and Tovey (1992), there are two versions of this problem, depending on the way the term "course" is interpreted. The first version is the "one-day or interval problem" in which classes occupy a classroom during a period of time, only one day a week. The optimal solution to this problem may be obtained by classic methods in polynomial time, but in real life, classes are given more than once a week and generally require the class to be offered in the same classroom. This version is known as "multi-day or non-interval problem", which is an NP-hard problem, even when $k \geq 2$, as Carter and Tovey have demonstrated.

The previous formulation would function if the length of all classes were the same and if they began exactly on the hour, each hour of the day. In many universities this does not normally occur. Class periods are different and may begin at any point during the hour. Rardin (1998) presents another way to handle three types of restrictions of the general assignment model. The first restriction ensures that each class be assigned to one classroom, the second avoids overlapping of classes, and the third ensures that the numbers attending the courses assigned to classroom j (s_{ij}) do not exceed the seating capacity of the classroom, b_j. The formulation is as follows:

$$\sum_{j=1}^{n} x_{ij} = 1, \quad \forall i$$

$$\sum_{i \in P_k} x_{ij} \leq 1, \quad \forall j, \ \forall k$$

$$\sum_{i=1}^{m} s_{ij} x_{ij} \leq b_j, \forall j$$

$$x_{ij} \in \{0,1\}, \quad \forall i, \ \forall j$$

3. THE COURSE TIMETABLING PROBLEM AT THE UNIVERSIDAD DE LAS AMERICAS, PUEBLA.

The Universidad de las Américas, Puebla (UDLA) is made up of five schools and 25 academic departments, which offer 41 majors, 31 masters' programs and 2 doctoral programs. Several of the courses offered each semester need to be taught in classrooms with special requirements, such as computers, computer projectors, overhead projectors, television and VCR, drawing boards, special rooms for dance and music, different types of laboratories, etc. For each course there are at least two groups known as sections, and these sections are offered at different times and sometimes on different days so that students can have several options to choose from in working out their schedules. Not all the courses have the same number of units or credits, some having 2 units, others 4, 6, 7, 8

or 10. This makes for variation in the number of weekly sessions for each course. There are courses that are offered only one day a week, and others may be taught in two, three, four and even five sessions a week. The days that a professor may choose to offer classes cannot be arbitrary. UDLA has established 14 valid modules (combinations of days). Each semester UDLA offers over 2000 courses and assigns them to fewer than 200 classrooms.

The classroom assignment system currently used at UDLA consists of using a technique developed in the 1980's, one which was adequate at the time, but which is now obsolete because UDLA has grown and changed over time. The solution offered by this system is not feasible because it allows courses to overlap and leaves many courses without an assigned classroom. To determine the final schedule, it is necessary to reassign nearly 80% of the courses manually as well as change the schedule of some courses, a reprocessing that normally takes weeks.

At least two months before a new semester starts, the academic departments have to prepare the programming schedule of the courses to be offered, and for which they must take into account the official time modules. Only in exceptional cases are courses allowed to be scheduled outside of these modules.

As respects classrooms, the university has 200 spaces available for courses, workshops, labs and other types of activities; however, it only has 93 classrooms per se. The Dean's department of each school reserves several classrooms with specific characteristics (such as computers, television, projectors) that are required for certain courses. Once these classrooms are eliminated, there remain 61 classrooms where the remaining 805 courses must be scheduled, as a result of eliminating other courses that require a lab, computer or other such facilities. In order that an assignment be acceptable, it should meet the following requirements:

- Each course should be assigned to a classroom.
- Each course should be offered in the same classroom, with the exception of laboratories.
- Courses should be assigned to classrooms with a seating capacity equal to or greater than the estimated number of students.
- Courses should be assigned in those hours established by academic departments.
- Classrooms should be used during the normal class schedule, which is from 7 a.m. to 10 p.m.
- There should be no overlapping: that is, one classroom should not be assigned to two courses at the same time.

Other characteristics that an optimal solution should have are the following:

- Courses should be located as closely as possible to the department which offers them.
- Nonuse of chairs should be minimized.
- Time during which classrooms are unoccupied between classes should be minimized.

Often the set of feasible solutions that satisfy all the requirements is small, and at times empty. For this reason, it is common to relax some restrictions, which then come to form

part of the objective function. Because many courses are often scheduled the same days at the same time, meaning that they exceed the number of existing classrooms at UDLA, it is impossible to assign all the courses at that particular hour, in which case there will be no feasible solutions to CAP. In order to avoid this absence of feasible solutions at UDLA`s problem, the first restriction was relaxed and it was introduced in the objective function as a penalty. In this paper we assume that the objective function is lineal and is the sum of four penalties, related to the aforementioned characteristics desirable for an adequate assignment, plus the relaxing of the restriction that all courses be assigned to a classroom. That is, the formula is as follows:

$$\sum_{i=1}^{nc}\sum_{j=1}^{ns}[CB_{ij} + CE_{ij}]x_{ij} + \sum_{j=1}^{s}T_j + C * CNA$$

where *nc* is the number of courses to be assigned, *ns* is the number of classrooms available; x_{ij} represents the decision variables, which have the value of one if course *i* is assigned to classroom *j* and zero in the opposite case; *CNA* is the number of courses not assigned; *C* is the penalty for not assigning a course; *CB* is the cost for the nonuse of chairs; *CE* is the cost of assigning a classroom in a building which is not the one preferred for the course, proportional to the distance between the preferred building and the assigned building, and T_j is the penalty for wasted time in classroom *j*, which is important during the process of trying to assign all courses. The values of the penalties were established subjectively by those responsible for the classroom assignment process. Several scenarios were presented to them, and they were questioned regarding the seriousness of each situation; based on their answer and their experience, they were asked to assign a weight from 0 to 100 to each unfavorable situation. The maximum penalty was given to the failure to assign a particular course. The penalty for the nonuse of chairs is proportional to the number of chairs not used; the penalty for distances is proportional to the distances. The penalty for class time wasted is proportional to the time a classroom remains unoccupied between classes.

4. THE TABU SEARCH HEURISTIC

Tabu Search heuristic (TS), as it currently stands, was developed by Glover (1989). It can be defined as an intelligent search procedure which leads to a solution outside of the local optimum, by means of mechanisms which allow the exploration of different areas for new solutions. This procedure carries out a search through difficult areas, by imposing restrictions on the search. In order for a solution to be considered acceptable, TS incorporates the concept known as *adaptive memory*, which helps one take advantage of information obtained previously.

Beginning with a function *f(x)* which needs to be optimized in a set of solutions, *X*, TS begins as a local search, going from one solution to another by means of iteration until a specific criterion is satisfied and the search is complete. A simple form of TS follows. The Tabu Search begins with an initial solution $x \in X$, which is generated by either a well-established or a random procedure and then continues to determine a neighborhood of *x*, $N(x) \subseteq X$, in which each neighboring solution $x' \in N(x)$ is arrived at by means of a *movement*. All the neighbors $x' \in N(x)$ are then evaluated and the best one is chosen, x^o, that is, the one that has the best objective function value. In each iteration the best solution, x^*,

is saved, and the same is then updated by the best neighbor if the latter improves the value of $f(x^*)$. The x^o solution is added to the *tabu list*, which keeps a record of the last solutions, to avoid returning to any of these in later iterations, in other words, to avoid cycling. This procedure is repeated until a termination criterion is reached and the best solution (x^*) obtained throughout all the iterations is reported. This procedure ensures that one goes beyond local optimum solutions, allowing the current solution to worsen from one iteration to another, or by means of strategies that modify the neighborhood $N(x)$ and that lead to areas away from the norm by means of diversification mechanisms and special memory structures.

5. APPLICATION OF THE TABU SEARCH HEURISTIC TO THE COURSE ASSIGNMENT PROBLEM.

The Tabu Search heuristic (TS) has shown itself to be one of the most efficient heuristics to solve complex combinatorial problems. A brief explanation follows of how TS functions to solve CAP. First, an initial solution is generated in one of the following ways: randomly or by searching for a local optimum, which is by following a greedy procedure. The first algorithm orders the courses randomly and assigns them one by one to classrooms, which are then ordered by building preference each time a course is randomly chosen. At the moment a course is placed in a classroom, there is no intent to minimize the nonuse of chairs or time; the focus is on creating a feasible solution. The second method orders courses by capacity, starting with the largest capacity, and assigns them in this order, as well as ordering the buildings in the same way as the algorithm that assigns courses randomly. At the moment a course is designated, the program chooses the classroom where nonuse of chairs is minimal and where there is the least waste of time. The first method of construction for the initial solution, gives importance to the fact that courses are assigned in the preferred buildings, whereas the second method considers that it is more relevant to minimize the nonuse of chairs in classrooms so that the majority of courses could be assigned.

To generate a neighborhood, a course, let's say z, is randomly chosen. Then classrooms which are possible candidates for that course to be reassigned to are selected, that is, those classrooms whose seating capacity is equal to or above the number of students required by the course. Afterwards, it is necessary to verify that the courses currently scheduled in those classrooms can be exchanged for z; the feasibility of the exchange is considered before this takes place. Each exchange constitutes a movement and generates a new solution. The penalties of the new solution are then considered and the movement that involves the lowest cost of assignment is chosen; the associated solution is considered the *best neighbor*. The tabu list stores the opposite movement for 10 iterations.

If classrooms were not assigned to all courses in the initial solution, the program examines the list of unassigned courses, and if exchanging z generates the possibility of assigning a particular course of this list, it is assigned to the corresponding classroom. The objective function of the new solution is evaluated and saved as the current solution. If its cost is less than that of the best solution obtained up till that moment, it is saved as the best solution obtained. The *Aspiration Criterion* consists of allowing a tabu movement to take place if the cost of the assignment generated is lower than the cost of the best assignment found up to that moment. The previous procedure is then applied to the current solution for a specified number of iterations which constitute the *Stopping Criterion.*

Diversification strategies are those that lead the search towards new areas. The current TS algorithm diversifies by generating two kinds of initial solutions by means of a constructive algorithm.

The Tabu Search heuristic that was implemented to seek a solution to CAP was programmed in Delphi language, based on object-oriented Pascal, which has been used by the authors previously with excellent results. This language allows a user-friendly interface because it involves the use of windows and menus. Figure 1 Shows the initial screen of the developed system.

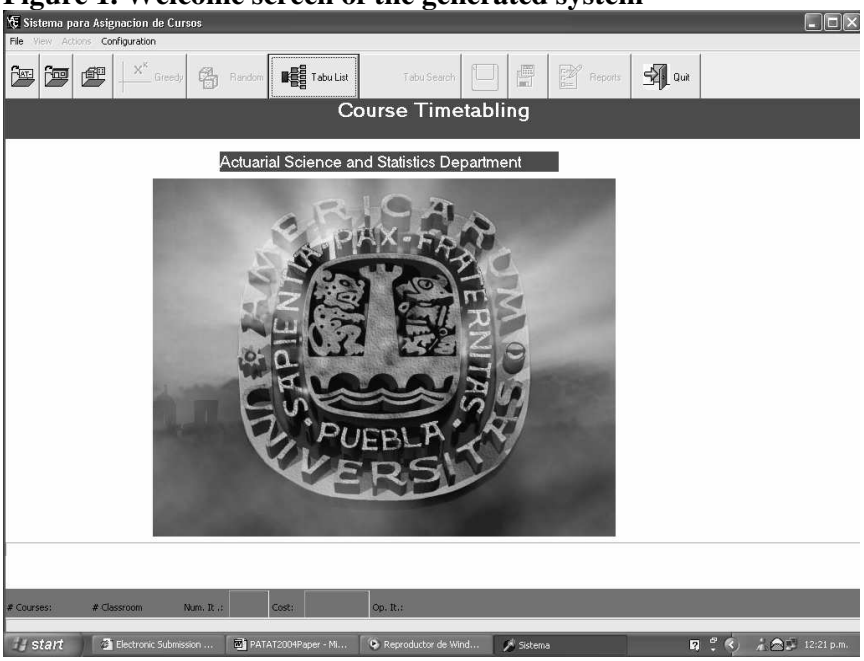
The menus generated in the program are the following:

- File.
  -Allows lists of classrooms and courses to be downloaded.
  -Records the optimal solution.
- View. Allows one to see:
  -The list of courses
  -The list of classrooms
  -The optimal solution
  -The report that was generated

-Actions. Allows:
  -An indication of the type of initial solution that will be generated
  -Repetition of the TS heuristic, asking for the number of iterations

-Configure.
  -Allows user to indicate the size of the Tabu list

Figure 1. Welcome screen of the generated system

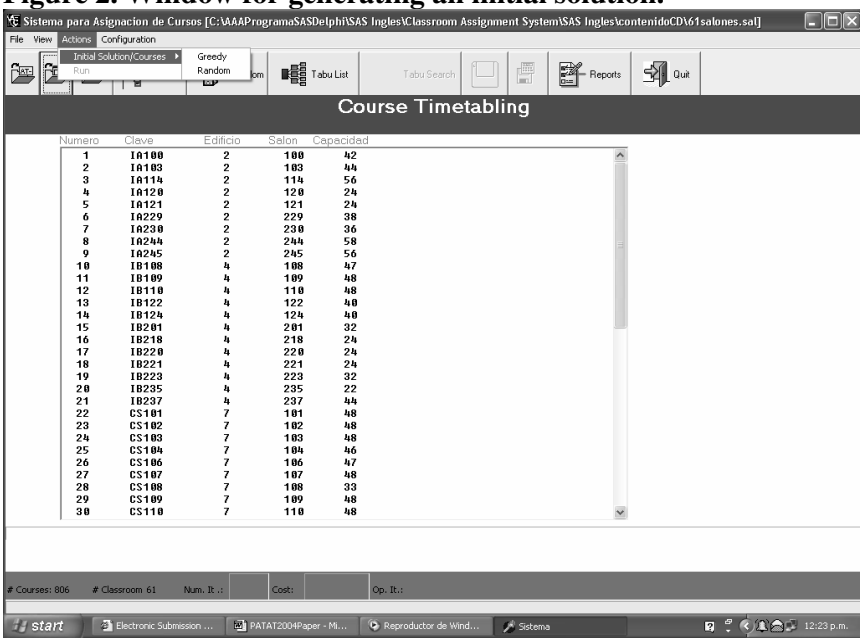


The windows that the program presents will now be described.

5.1 Window for generating an initial solution.

Once the lists of classrooms and courses are downloaded, the program offers two options to generate the initial assignment, optimal assignment and random assignment. This can be seen in figure 2.

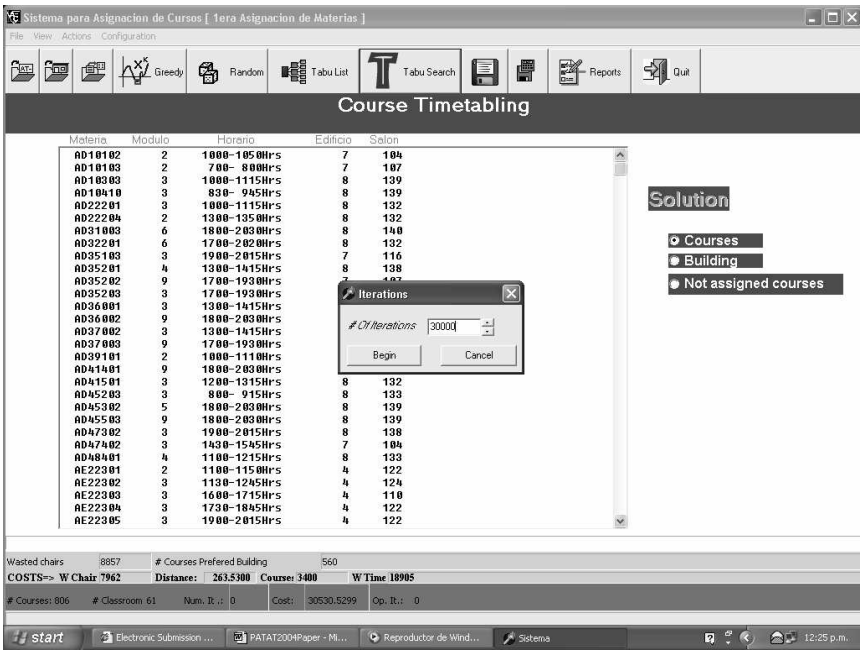
Figure 2. Window for generating an initial solution.



5.2 Window for indicating the number of iterations.

The maximum number of iterations that can be carried out depends on the memory and the processor of the computer on which the program is executed. In several of the tests carried out, 10,000 iterations were executed. Figure 3 shows the window that indicates this possibility.

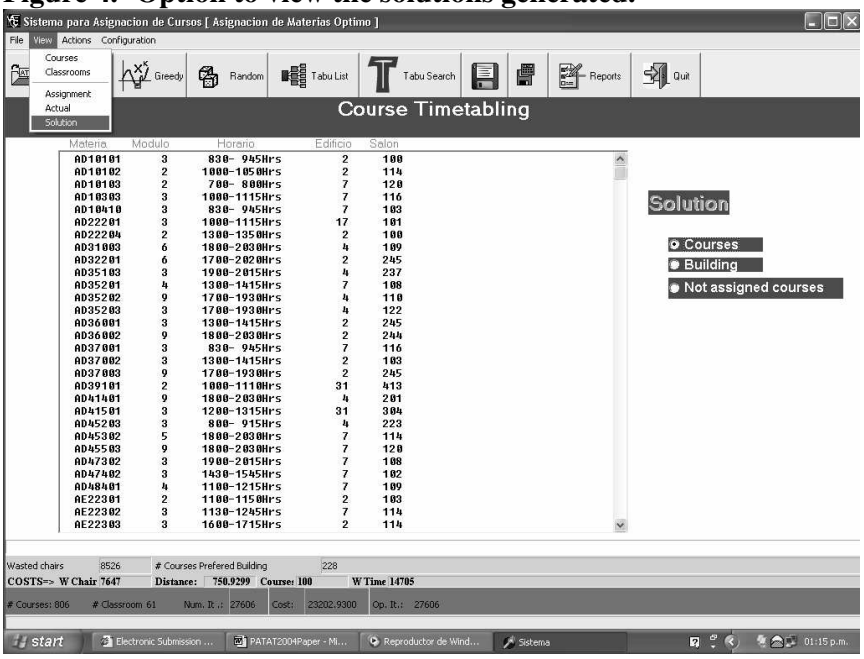
Figure 3. Option of the number of iterations.



5. 3 Window to see assignments.

The *View* window offers us the option of visualizing the best solution (*Solution*), the initial solution (*Assignment*) and the current solution (*Current*), the assignment generated in the last iteration. This is shown in figure 4.

Figure 4.- Option to view the solutions generated.



6. RESULTS OBTAINED.

Here we will describe some of the tests that have been carried out to assess the performance of the heuristic that was developed, and the results obtained. The tests consisted of varying the number of iterations and the size of the tabu list, and generating different initial solutions. The program was run on a personal computer with an AMD Athlon XP processor, 1.4 GHz, 256 Mb's of RAM memory and 40 Gb's on the hard disk.

6.1 Tests with Different Sizes of the Tabu List

To determine the extent to which the size of the tabu list influenced the quality of the solutions, 8 runs were effected, each of 10,000 iterations, all with different sizes of Tabu lists, starting with the same initial solution. The results can be seen in Table 1. In this table the first column shows the sizes of the tabu list (TLS), the second indicates in which iteration the best solution was obtained (It), the third shows the total assignment cost (TAC), the fourth gives the number on courses not assigned (NCNA), the fifth gives the number of courses assigned in preferred buildings (NCPB), the sixth indicates the percentage of courses assigned to preferred buildings (%NCPB), the seventh shows the number of nonuse chairs (NUC), and the eighth presents the percentage of reduction in the Total Assignment Cost with respect to the initial solution (%Red).

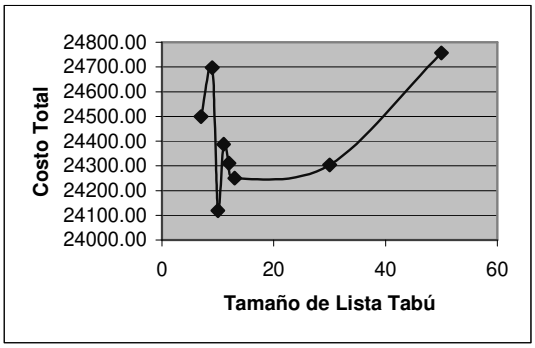
Table 1.- Costs of the best assignment with different sizes of the tabu List.

TLS	It	TAC	NCNA	NCPB	%NCPB	NUC	% Red
7	9945	24499	5	444	55.08	7861	15.53
9	8088	24697	3	450	55.83	7891	14.85
10	**9823**	**24118**	**2**	**460**	**57.07**	**7929**	**16.39**
11	9924	24387	3	457	56.69	7915	15.92
12	9075	24310	4	459	56.94	7871	16.18
13	9586	24250	5	449	55.70	7873	16.85
30	5325	24303	2	456	56.57	7946	16.21
50	9862	24756	3	466	57.81	7891	15.53

Table 1 shows that the difference in the total cost of assignments among different runs is quite small, almost nil; however, since the lowest cost is obtained with a Tabu List size of 10, in the program that was developed, the value of this parameter was set with 10 as a default, in order to carry out the following tests. It should be noted that the user can modify this value easily in the *Configure* (*Configure*) menu of the program.

Figure 5 shows the values of the objective function (cost of the assignment) for each run of 10,000 iterations for each value of Tabu List size.

Figure 5. Values of the objective function for different sizes of Tabu List.



6.2 Comparison of the best solutions against the initial solution.

The program was run, generating the initial solution in an optimal way, and afterwards a run of the TS heuristic in two stages was requested. In the first stage, 30,000 iterations were requested and in the second 80,000 more; the effect of the penalty values on the search process was then observed. Table 2 shows the results obtained. The first columns (TL) indicates the size of the Tabu list, the second shows the total cost of the assignment (TAC), the third column indicates the number of iterations, the fourth column indicates which iteration gave the best solution (IBS), the fifth column shows the number of courses not assigned (NCNA), the sixth column indicates the number of nonuse chairs (NUD), the seventh gives the cost per time wasted (CWT), and the eighth column presents the cost per distance between buildings (CD).

Table 2. Comparison of the best solution and the initial solution.

TL	TAC	Iterations	IBS	NCNA	NUC	CWT	CD	Red
10	29010	0	0	8	6855	21145	210	-
10	24407	30,000	28,851	1	6515	17385	407	15.9%
10	24244	80,000	79,481	1	8199	15235	710	16.4%

In the first run the best assignment was reached in iteration 28,851, and in the second run the best assignment was found 79,481 iterations afterwards. The total cost of assignment for the first run decreased 15.9%, and for the second run it was 16.4%; that is, the additional 80,000 iterations after the first solution did not contribute much to improving the solution. The number of unassigned courses on the first run was 8, whereas as after applying the heuristic there was only one unassigned course. Upon analyzing the reasons for which that course was unassigned, it was discovered that the course had requested a classroom with a 75-student capacity. There is no classroom at UDLA with that capacity. The conclusion was that the heuristic was able to assign 100% of the assignable courses.

Table 2 also shows that the costs for unused chairs and distances between buildings increase upon increasing the number of iterations, and that the cost of classroom “dead time” is reduced. This is due to the cost structure of the objective function, that is, the amount of time wasted is greater than the waste in chairs and the distance between

buildings. This was determined by the individuals responsible for assigning classrooms. However, if necessary, the current cost structure can be changed. For example, if we want now to assign more courses near the academic departments that offers the courses, we need to give a higher penalty to that element in the objective function.

6.3 Comparison with other methods.

In 2001 another project was developed at UDLA to solve the same problem, with very good results (Ahuactzin and Reyes, 2001). This project was developed with a heuristic based on the Ant Colony Optimization (ACO) algorithm and the computer program was developed in C++. In Table 3 the results obtained in this research are compared with those generated using the ACO heuristic and with the results of the procedure currently used at UDLA. The data used in this comparison were those of the 2001 Spring semester, in which 806 courses were assigned to 61 classrooms.
Comparing the results (given in both absolute and relative terms), one can observe that with manual assignment, three weeks are required to finish the task; 100 courses could not be assigned, 105 courses were assigned classrooms with a seating capacity lower than that required, many chairs were underused and it offers a good number of courses assigned to preferred buildings. The Ant Colony Optimization heuristic improves the current procedure dramatically, in that within an hour and a half it generates a solution which leaves no course unassigned, and leaves fewer chairs unused, but also assigns fewer courses to preferred buildings. The only course that could not be assigned requested space for 75 students, and as was mentioned previously, there are no classrooms at UDLA with this capacity.

Table 3.-Comparison of assignments at UDLA

	Current	**ACO**	**Tabu Search**
Required time	3 semanas	90 min	120 sec
Iterations	-	10,000	30,000
Non use chairs	10,087	6,688 (66%)	7,932 (79%)
Prefered building	390	315 (81%)	415 (106%)
Non-assigned courses	**100**	1	1
Courses with lower capacity	**105**	0	0

The solution offered by the Tabu Search heuristic is the one that assigns more courses to preferred buildings, leaves more chairs unused than the algorithm used by ACO but much less than the current method, leaves only one course unassigned (the one requiring 75 seats) and does not assign classrooms with a lower capacity than that required. The great difference is in the fact that it needed only 40 seconds to generate the best solution, using 30,000 iterations. This reduction in execution time is one of the most valuable contributions of implementing Tabu Search to solve the Problem of Assigning Classrooms.

7. CONCLUSIONS

The goal of this research was to implement the Tabu Search heuristic to solve the Classroom Assignment Problem at the Universidad de las Américas, Puebla, and to develop a user-friendly interface. Thanks to the simplicity of defining the movements and structures used to implement the algorithm, this process proved to be very rapid. The two principal contributions of this paper are the presentation of a user-friendly interface and the reduction of the computer time needed to generate an excellent solution.

In the program, a criterion of diversification or of multiple beginnings was implemented, in which a specific number of initial solutions is generated, and to each the process of improving the solution with the Tabu Search heuristic is applied. In this sense it was possible to observe that upon selecting the courses randomly to generate the initial solutions, fewer courses were assigned than with the option of generating the initial solution in a greedy way. Despite this, it reaches the point in which all or the majority of the courses are assigned more rapidly. Although the assignment procedure and that of improving TS solutions is simple, it is effective and offers excellent solutions in very little time. Another great advantage in the procedure developed is that it is very easy to design a variety of patterns for the structure of the objective function; that is, when the preferences change, in relation to the distances between preferred classrooms, the nonuse of chairs and dead time in classrooms, the objective function is adjusted and tests can be carried out to analyze the results that are generated.

The combination of the Tabu Search heuristic with a programming language that allows a very attractive and user-friendly operation has allowed, once again, the development of an application which culminates with a successful approach to the solution of a difficult combinatorial problem in very little time.

REFERENCES

Ahuactzin Ortega, Angela María y Reyes Montiel, Claudia Gabriela, "Estudio de la heurística Sistema Hormiga y su aplicación al problema de asignación de cursos a salones en la UDLA". Tesis para obtener el grado de Actuaría en la UDLA. 2001

Burke, Edmund and Ross, Peter, *Practice and Theory of Automated Timetabling*, Vol. I, Selected Papers of the First International Conference in Edinburgh, UK. Ed. Springer, 1996

Burke, Edmund and Carter, Michael, *Practice and Theory of Automated Timetabling*, Vol. II, Selected Papers of the Second International Conference in Toronto, Canada. Ed. Springer, 1998

Carter, Michael, "A Langangian Relaxation Approach to the Classroom Assignment Problem". INFOR Vol. 27, No. 2, Mayo 1989

Carter, Michael and Tovey, Craig. "When is the Classroom Assignment Problem Hard?". Operations Research. Vol. 40, Supp. No. 1. 1992.

Carter, Michael and Laporte, G. "Recent Developments in Practical Course Timetabling", in Burke, Edmund and Carter, Michael, *Practice and Theory of Automated Timetabling*,

Vol. II, Selected Papers of the Second International Conference in Toronto, Canada. Ed. Springer, 1998

Diaz, Adenso (coordinador), *Optimización Heurística y Redes Neuronales en Dirección de Operaciones e Ingeniería*. Editorial Paraninfo. 1996

Glover, F., "Tabu Search-Part I". ORSA Journal on Computing, Vol. 1, No. 3, 1989

Glover, F. and Laguna, Manuel. Tabu Search, Kluwer Academic Publishers,

Hertz, A., "Tabu Search for large scale timetabling problems". European Journal of Operational Research. Vol. 54. pp. 39-47. 1991.

Rardin, Ronald L *Optimization in Operations Research*. Prentice Hall. 1998.

Reeves, Colin R. *Modern Heuristic Techniques for Combinatorial Problems*. Wiley. 1993.

Rey, Fernando y Carlevaro, Marcos, "Problema de asignación de horarios y salones a cursos". Tesis para obtener el grado de Ingeniero en la Universidad de la República de Uruguay. 2000.