İstanbul Ticaret Üniversitesi Fen Bilimleri Dergisi Yıl: 6 Sayı:12 Güz 2007/2 s. 1-20

TIMETABLING OF FACULTY LECTURES USING SIMULATED ANNEALING ALGORITHM

Tunçhan CURA[*]

ABSTRACT

In this study, a faculty-course timetabling problem is solved by using a Simulated Annealing based algorithm. In this sort of problems, both the objectives and the constraints are usually highly institution-specific. Thus, there is not a single commonly used tool to solve this planning problem. Since the problems are institution-specific, the results of this study have not been compared to those of the studies which are published already. Comparing with the many of the studies, the most important difference of this study is to take the lecturer seniority into consideration. This study separates the problem into two main components in the solution progress. While the first one is dealing with searching of the lectures which can be located into the same time interval, the second one is dealing with assigning the lectures to the most suitable place in the timetable. That algorithm is experimented with 2006-2007 academic year first term data of Faculty of Business Administration at Istanbul University. The results of proposed algorithm is compared to those of genetic algorithms and tabu search. Thus, the genetic algorithms approach can not even find a feasible solution. And the tabu search approach finds worse solutions than the proposed algorithm.

Keywords: Simulated Annealing, Timetabling, Heuristics Techniques

TAVLAMA BENZETİMİ ALGORİTMASINI KULLANARAK FAKÜLTE DERSLERİNİN ÇİZELGELENMESİ

ÖZET

Bu çalışmada fakülte derslerinin çizelgelenmesi problemi Tavlama Benzetimi temelli bir algoritma ile çözülmüştür. Bu tür problemlerde hem amaçlar hem de kısıtlar genellikle kuruma özgüdür. Bu nedenle böyle bir planlama problemini çözecek ortak bir araç bulunmamaktadır. Problemlerin kuruma özgü olması nedeniyle çalışmanın sonuçları literatürdeki bir çalışmanın sonuçlarıyla karşılaştırılamamıştır. Bu çalışmanın literatürde yer alan pek çok çalışmadan en önemli farklarından birisi öğretim üyesi kıdemlerinin dikkate alınmış olmasıdır. Çözüm sürecinde problem iki ana parçaya ayrılmıştır. Bunlardan birincisi aynı zaman dilimine yerleştirilebilecek dersleri aramakla ilgilenirken, ikincisi derslerin zaman çizelgesinde en uygun yerlere yerleştirilmesiyle ilgilenmektedir. Algoritma İstanbul Üniversitesi İşletme Fakültesi' nin 2006-2007 Akademik takvimi birinci yarıyıl verileriyle denenmiştir. Önerilen algoritmanın sonuçları ile genetik algoritmalar ve tabu arama algoritmalarının sonuçları kıyaslanmıştır. Buna göre, genetik algoritmalar yaklaşımı uygun çözüm dahi bulamamaktadır. Tabu arama yaklaşımı ise daha başarısız çözümler bulmaktadır.

Anahtar Kelimeler: Tavlama Benzetimi, Çizelgeleme, Sezgisel Teknikler

[*] *İstanbul Üniversitesi, İşletme Fakültesi, Sayısal Yöntemler Anabilim Dalı, Avcılar-İstanbul*

1. INTRODUCTION

University course timetabling problems are combinatorial problems, which consist of scheduling a set of courses within a given number of rooms and time periods. Solving a real world timetabling problem manually often requires a significant amount of time, sometimes several days or even weeks (Abdennadher and Marte, 2000). Although many of them are not about specifically university course timetabling, there have been a lot of research for timetabling. One of the examples for course timetabling is the study of Henz and Würtz (1996). They suggested the constraint logic programming approach which is, according to them, competitive or better than traditional operations research algorithms for many real-world problems. Burke et al. (2007) investigated a simple generic hyper-heuristic approach upon a set of widely used constructive heuristics in timetabling. Within the hyper-heuristic framework, they developed a tabu search approach to search for permutations of graph heuristics which were used for constructing timetables in exam and course timetabling problems. Head and Shaban (2007) combined two problems, student scheduling and course scheduling, which are typically treated as separate tasks. They build the schedule based on heuristic functions and place the students into classes simultaneously.

In this study, it has been tried to solve university course timetabling problem using simulated annealing (SA) algorithm which is a stochastic heuristic algorithm, and searches the solution space using a stochastic hill climbing process. In this sort of problems, both the objectives and the constraints are usually highly institution-specific. Thus, the results of this study can not be compared to those of any other study in that area. Note that, almost all of the studies similarly focused on satisfying lecturer preferences. However, none of them has dealt with lecturer seniorities. In order to mention the differences of the problem discussed here from the problems of the other studies, some examples may be given as follows: in the problem which is studied by Abdennadher and Marte (2000), lecturers must have one hour break between courses and monday afternoon is reserved for professors; Schimmelpfeng and Helber (2006) described teaching groups for their problem and assign these groups to the lectures; MirHassani (2006) included an overload constraint for the lecturers in his study; the problem which Henz and Würtz (1996) studied has two unusual constraints which limits some courses to certain time slots and introduces unavailability times for some lecturers; Avella and Vasil'ev (2005) predefined some penalty values according to the courses which are scheduled at a given time; and Daskalaki et al. (2004) included a constraint in which a timetable should accommodate requests for sessions of consecutive teaching periods. In this regard, the results of the SA approach has been compared to those of the genetic algorithms (GA) and tabu search (TS) approaches which have been adapted from similar studies. The use of SA as a technique for discrete optimization dates back to the early 1980s. It was heralded with much enthusiasm as it appeared to be both simple to implement and widely applicable, and as a result of articles in popular scientific journals researchers from a wide variety of disciplines experimented with it in the solution of their own problems (Reeves, 1995). Simulated annealing has been used in various combinatorial optimization problems and has been particularly successful in circuit design problems (Kirkpatrick et al., 1983). Timetabling problems are sort of scheduling problems, since they can be defined as the scheduling of a set of activities requiring a given amount of resources, and involving groups of people over a finite time period (Avella and Vasil'ev, 2005). A SA example to the scheduling problem is the study of Anagnostopoulos et al. (2006). They considered the traveling tournament problem to abstract the salient features of Major League Baseball in the United States. They aimed at scheduling Major League Baseball such that total travel distance should be minimized and home/away game constraints should be satisfied as well. The study of Loukil et al. (2007) can be mentioned as another example on this topic. Their study deals with a production scheduling problem in a flexible job-shop with particular constraints: batch production; existence of two steps: production of several sub-products followed by the assembly of the final product; possible overlaps for the processing periods of two successive operations of the same job. They tried to schedule multi-objective production case by SA.

In this study, the timetabling problem is based on the rules which must be observed in Istanbul University Faculty of Business Administration (IUFBA). These rules are assembled into two categories. First category is that of obligatory rules which make the solution of the problem erroneous even if one of them is violated. The other category is that of the rules which are observed the more, the more satisfactory solution is obtained. Indeed, satisfying lecturer desires is the only rule for this category. These categories may be, respectively, called *hard constraints* which usually relate to operational limitations that can not be bypassed in the real world and *soft constraints* which are deemed desirable (Burke and Newall, 2004). Obligatory rules are as follows:

1. Each lecture must be assigned to only one room and one day. In other words, each lecture must be assigned to a single timeslot. Although actual lengths of lectures vary, each lecture must initially be assigned to one hour which will be the starting time of the lecture.

2. The lengths of the lectures and school hours must be taken into consideration while assigning the lectures. For example if the school hours are from 9 am to 5 pm and the length of the lecture is 3 hours, this lecture can not be assigned to 4 pm since it would have exceed the official school hours.

3. More than one lecture can not be assigned to a given room at the same time interval.

4. A lecturer can not have more than one lecture assigned in a given time interval.

5. To allow students to choose alternative lectures from the same department, some predefined lectures must not overlap. In addition to this, the lectures of the same class must not overlap as well.

6. At least half of the total school hours in each day must be filled with lectures.

The rest of the paper organized as follows: Section 2 presents the mathematical formulation of above problem; Section 3 identifies the main SA structure; Section 4 presents two other comparative methods which are GA and TS; Section 5 gives an application of developed algorithms; and section 6 gives a brief conclusion.

2. FORMULATION OF THE PROBLEM

The problem in this study can be stated as follows: The number of lectures, the number of lecturers and the number of different rooms are denoted by *J*, *I* and *L*, respectively. Lectures can be assigned to any school day. Each day consists of 8 hours. Thus, $D = 5$, $H = 8$, and they denote the number of days and hours of timetable respectively. Y_j denotes the length of lecture *j* ($j=1,...,J$), C_i denotes the seniority coefficient of lecturer *i* ($i=1,...,I$). The coefficients of veteran lecturers will be greater. Thus, the probability of satisfying their wishes will increase, and P_{idh} denotes the desire coefficient (a higher value indicating a higher preference) of lecturer *i* for day *d* ($d=1,...,D$) and hour *h* ($h=1,...,H$).

$$\max \sum_{j=1}^{J}\sum_{i=1}^{I}\sum_{d=1}^{D}\sum_{h=1}^{H}\sum_{l=1}^{L}\sum_{h^*=h}^{\min\{(Y_j+h),H\}} X_{ji} \times P_{idh} \times C_i \times S_{jldh} \tag{A}$$

$$\text{subject to} \sum_{l=1}^{L}\sum_{d=1}^{D}\sum_{h=1}^{H} S_{jldh} = 1, \qquad j = 1,...,J. \tag{1}$$

$$\beta_{jldh} \le H, \qquad j = 1,...,J,\ h = 1,...,H,\ l = 1,...,L,\ d = 1,...,D. \tag{2}$$

$$\beta_{jldh} \times S_{j^*ldh^*} \le \beta_{j^*ldh^*}, \quad j = 1,...,J,\ h = 1,...,H,\ l = 1,...,L,\ d = 1,...,D,\ h^* = h,...,H,\ j^* = 1,...,J,\ j^* \ne j. \tag{3}$$

$$\beta_{jldh} \times S_{j^*l^*dh^*} \times X_{j^*i} \times X_{ji} \le \beta_{j^*ldh^*} \times X_{j^*i}, \quad j = 1,...,J,\ h = 1,...,H,\ l = 1,...,L,\ d = 1,...,D,\ h^* = h,...,H,\ j^* = 1,...,J,\ j^* \ne j. \tag{4}$$

$$\beta_{jldh} \times S_{j^*ldh^*} \times Z_{j^*j} \le \beta_{j^*ldh^*} \times Z_{j^*j}, \quad j = 1,...,J,\ h = 1,...,H,\ l = 1,...,L,\ d = 1,...,D,\ h^* = h,...,H,\ j^* = 1,...,J,\ j^* \ne j. \tag{5}$$

$$\beta_{jldh} = (h + Y_j) \times S_{jldh}, \quad j = 1,...,J,\ h = 1,...,H,\ l = 1,...,L,\ d = 1,...,D. \tag{6}$$

$$\alpha_{dh^*} = S_{jldh}, \quad j = 1,\dots,J,\ h = 1,\dots,H,\ l = 1,\dots,L,\ d = 1,\dots,D,\ h^* = h, \min\{(Y_j + h), H\}. \tag{7a}$$

$$\sum_{h=1}^{H} \alpha_{dh} \geq \left\lceil \frac{H}{2} \right\rceil, \quad d = 1,\dots,D. \tag{7b}$$

$$S_{jldh}, \alpha_{dh} \in \{0,1\}. \tag{8}$$

where $X_{ji} = 1$ if i lectures j, otherwise $X_{ji} = 0$, $\alpha_{dh} = 1$ if a lecture is assigned to any room on day d and at hour h, otherwise $\alpha_{dh} = 0$, $Z_{j^*j} = 1$ if lecture j^* and lecture j can not overlap, otherwise $Z_{j^*j} = 0$, and $S_{jldh} = 1$ if lecture j is assigned to room l on day d and at hour h, otherwise $S_{jldh} = 0$.

It can obviously be seen that such a mathematical model is not easily implemental. Indeed, it may require a computer with very high level of specifications and software with very high level of performance. However, this model is helpful to explain the problem.

As previously mentioned, the problem has two categories of rules. The first one is dialing with satisfying the lecturer desires, which is represented by (A) in the model. Second category is discussed as follows:

1. Obligatory rule 1 is imposed by (1),
2. Obligatory rule 2 is imposed by (2),
3. Obligatory rule 3 is imposed by (3),
4. Obligatory rule 4 is imposed by (4),
5. Obligatory rule 5 is imposed by (5),
6. Obligatory rule 6 is imposed by (7a) and (7b),

in this model.

3. THE MAIN SA STRUCTURE

As mentioned before, some of the lectures can not overlap. In order to avoid overlapping of these lectures, assigning each lecture to a unique day-hour pair may be considered, if there is enough available space in the timetable. However, the time period is limited to one week (5 days × 8 hours a day) while the number of lectures is 133 for the test problem, which makes such an assignment impossible.

Inspired by the partitioning approach of Abdullah et al (2007), this study employs some subsets (vectors) in solution progress: maximum L lectures may overlap in a given time interval as long as all the constraints are satisfied. Suppose that there are

K vectors, denoted by b_k (k=0,..., K-1), and each of them has maximum L elements denoted by $b_{k,l}$ (l=0,…,L-1). So, the index l represents room number. Lectures will be assigned to each $b_{k,l}$. Of course there should be no restriction for the lectures to be located into a given b_k. A variable $hour_k$ is used for limiting the maximum hour of a lecture in b_k. For example, a 3-hour lecture can not be located into b_k where $hour_k$ = 2. Fig. 1 shows a sample b_k vector where L = 15. Since $b_{k,0}$ = 40, lecture 40 is assigned to b_k, $b_{k,2}$ and $b_{k,4}$ are empty as to the Figure 1.

In the case of K = 15, table 1 shows a sample timetable matrix to where all of the b vectors are assigned and is denoted by $s_{d,h}$ in this study (d=0,…,D-1; h=0,…,H-1). If b_k is not assigned to $s_{d,h}$ then $s_{d,h}$ = -1. However, a preceding element (b_k) of $s_{d,h}$ may overflow through current element (b_k^*) dependent on $hour_k$. For example, if $s_{0,0}$ is 13 and $hour_{13}$ is 3 then b_{13} will overflow through $s_{0,2}$ as seen in table 1. However, there will not be an overlapping in that case, since $s_{0,2}$ is empty ($s_{0,2}$=-1).

SA algorithm which is developed for this study swaps the cells in s and it swaps lectures among b_k to maximize objective function explained in (A).

Table 1. A Sample *s* Matrix

	Day 1(d=0)	**Day 2(d=1)**	**Day 3(d=2)**	**Day 4(d=3)**	**Day 5(d=4)**
Hour 1(h=0)	*13*	*2*	*8*	*4*	*3*
Hour 2(h=1)	-1	-1	-1	-1	-1
Hour 3(h=2)	-1	*11*	-1	*5*	*10*
Hour 4(h=3)	*1*	-1	*6*	-1	-1
Hour 5(h=4)	-1	-1	-1	-1	-1
Hour 6(h=5)	*9*	*7*	-1	*14*	*12*
Hour 7(h=6)	-1	-1	*0*	-1	-1
Hour 8(h=7)	-1	-1	-1	-1	-1

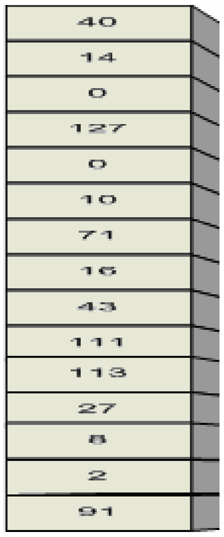

Figure 1. A Sample b Vector

3.1. Defining the Number of *b* Vectors

Let *X* be the number of different lecture lengths. Thus, each different length, the number of lectures with this length and the number of b_k where $hour_k$ equals this length are denoted by λ_x, δ_x and μ_x (x=1,…,X) respectively. For example, if there are 3 lectures and their lengths are 2 hours, 2 hours and 3 hours successively, then the number of different lengths (X) will be 2 (λ_1=2 hours and λ_2=3 hours), and δ_1 will be 2 and δ_1 will be 1.

It is assumed that the proportion between the number of lectures with a specific length and the number of b_k where $hour_k$ equals that length must be direct for the test problem. If it is supposed that X is 2 and μ^*_1, μ^*_2 represent the number of *b* vectors for only one day, following equations will be obtained:

$$\lambda_1 \mu^*_1 + \lambda_2 \mu^*_2 \leq H \tag{9}$$

$$\frac{\mu_1^*}{\mu_2^*} = \frac{\delta_1 / J}{\delta_2 / J} \Rightarrow \mu_2^* = \frac{\delta_2 \mu_1^* / J}{\delta_1 / J} \tag{10}$$

Eq.(11) is obtained with the help of Eq. (9) and Eq. (10).

$$\mu_1^* \le \frac{\delta_1}{J} \times \frac{H}{\left(\frac{\lambda_1\delta_1}{J} + \frac{\lambda_2\delta_2}{J}\right)} \Rightarrow \mu_x^* = \text{round}\left\{\frac{\delta_x}{J} \times \frac{H}{\sum_{a=1}^{X} \frac{\lambda_a\delta_a}{J}}\right\} \tag{11}$$

$$\text{where } \sum_{x=1}^{X} \lambda_x \mu_x^* \le H \tag{12}$$

In order to ensure Eq. (12), μ^*_x values might have to be adjusted by reducing one each. As explained above, μ^*_x is calculated for one day in Eq. (11). Hence, μ^*_x must be multiplied by D to obtain final μ_x value for whole timetable. The number K of b vectors will be $\sum_{x=1}^{X} \mu_x$ after all.

For the test problem $X = 2$, $\lambda = \{2, 3\}$, $\delta = \{37, 96\}$, $J = 133$, $D = 5$, and $H = 8$ in this study. Thus, K will be 15 and set of *hour* will be {2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3}, according to calculation above.

3.2. Filling the *b* Vectors with Lectures

After defining K and $hour_k$, next step will be filling each b_k with lectures. Each b_k is filled with randomly selected lectures initially. Therefore, that would most probably cause errors which are violations of the obligatory rules 4 and 5.

In this study, a fitness function is used for measuring the compatibility between lecture j and vector b_k. This functions' value is a rational number between 0 and 1. As the fitness functions' value approaches to the unity the compatibility between j and b_k increases while it decreases as the fitness function' value approaches to 0. The fitness function is defined as:

$$f_1(j,k,o) = \begin{cases} 0 & \text{if } Y_j > hour_k \\ \dfrac{g_{j,k} - \theta_k + \varepsilon}{L - \theta_k - q_{j,k} - q_{o,k} + \varepsilon} & \text{if } Y_j \le hour_k \end{cases} \tag{13}$$

where θ_k is the number of empty elements in b_k, ε is a very small number to avoid division by zero error, o is a lecture, $q_{j,k}$=1 if j exists in b_k, otherwise 0, $q_{o,k}$ =1 if o exists in b_k, otherwise 0 and $g_{j,k}$ is the number of compatible elements with j in vector b_k, it includes empty elements but excludes j and o. In other words, if b_k contains lecture j currently or lecture o is in b_k and it is compatible with j, $g_{j,k}$ will not enclose them. The "compatibility of two lectures" is yielded when these lectures can overlap without violating the rules 4 and 5.

```
For $k = 0$ to $K - 1$
    For $l = 0$ to $L - 1$
      If $b_{k,l} \neq 0$ then
        Search the $b_{k^*}$ such that $f_l(b_{k,l}, k^*, -1) - f_l(b_{k,l}, k, -1)$ is
            maximum and $f_l(b_{k,l}, k^*,-1) > f_l(b_{k,l}, k,-1)$
        If $k^*$ is found, search lecture $l^*$ in $b_k^*$ such that $f_l(b_{k^*,l^*}, k, b_{k,l})$ is
maximum
            and $f_l(b_{k^*,l^*}, k, b_{k,l}) \geq f_l(b_{k^*,l^*}, k^*, -1)$
        If $l^*$ is found, swap $b_{k,l}$ and $b_{k^*,l^*}$
      End if
    End For
End For
```

Figure 2. An Algorithm for Error Reducing

Until all of the b vectors have no error (violations of rule 4 and 5), an algorithm which is called *error reducer algorithm* is used for assembling compatible lectures into the proper b_k. Figure 2 shows the sketched algorithm.

3.3. The Energy Function

In this paper, the energy function is derived from the objective function (A). In a similar manner, the b vectors and the s matrix are used for calculating the energy function which will be maximized by the SA solver algorithm. Since b vectors are located in s matrix, each of them is assigned to a unique d, h pair. Thus, the calculation of the energy function of b_k for a given d, h pair will be as:

$$f_2(k,d,h) = \sum_{l=0}^{L-1} \sum_{h^*=h}^{h+hour_k} P_{X_{b_{k,l}},d,h} \times C_{X_{b_{k,l}}} \tag{14}$$

where X_j (X_0=0) is a number which represents the lecturer who lectures j, $P_{i,d,h}$ ($P_{0,d,h}$=0) and C_i (C_0= 0) have already been explained for (A) and they are used in the same purpose here. If $Y_j < hour_k$ and j is located in b_k then sliding j within $hour_k$ will be allowed to satisfy preferences of lecturer of j. In this study, the energy function is described as:

$$f(s)=\left(\sum_{d=0}^{D-1}\sum_{h=0}^{H-1}\begin{cases}0 & \text{if } s_{d,h}=-1\\ f_2(s_{d,h},d,h)-f_3(s,d,h)-f_4(s,d,h) & \text{if } s_{d,h}\geq 0\end{cases}\right)- f_5-\sum_{d=0}^{D-1}f_6(s,d) \tag{15}$$

$$f_5=\gamma\times M \tag{16}$$

$$f_3(s,d,h)=\max\{h+hour_{s_{d,h}}-H,0\}\times M \tag{17}$$

$$f_4(s,d,h)=\sum_{h^*=h+1}^{h+hour_{s_{d,h}}}\begin{cases}0 & \text{if } s_{d,h^*}=-1\\ (h+hour_{s_{d,h}}-h^*)\times M & \text{otherwise}\end{cases} \tag{18}$$

$$f_6(s,d)=\max\left\{\left\lceil\frac{H}{2}\right\rceil-\zeta,0\right\}\times M$$

$$\text{where } \zeta=\left\lceil\frac{H+\varepsilon}{\sum_{h=0}^{H-1}\begin{cases}hour_{s_{d,h}} & \text{if } s_{d,h}\neq -1\wedge\theta_{s_{d,h}}<L\\ 0 & \text{otherwise}\end{cases}+\varepsilon}\right\rceil \tag{19}$$

$$M=J\times\max\{Y_1,...,Y_j\}\times\max\{C_1,...,C_I\}\times\max\{P_{1,0,0},...,P_{I,D,H}\}+1 \tag{20}$$

where θ_k is number of empty elements in b_k and γ is total error count as already mentioned. That is, γ is the total count of rule 4 and 5 violations in all b vectors. M is a huge number to penalize violations of the obligatory rules. It is calculated as seen in Eq. (20) for this study. This penalization approach is similarly used in some integer programming formulations of the university course timetabling problems (Schimmelpfeng and Helber, 2006; Avella and Vasil'ev, 2005; MirHassani, 2006; Daskalaki et al., 2004). Most commonly, the objective functions of the models contain penalizations of the violations with respect to some weights.

While Eq. (16) is penalizing the violations of the rules 4 and 5; Eq. (17), Eq. (18) and Eq. (19) are penalizing violations of the rules 2, 3 and 6 respectively.

3.4. The Cooling Schedule and Acceptance Probability Function

Classical SA which is also called Boltzman annealing is taken as basis in this study. Its acceptance probability function is $P(T_r)$. This function controls accepting new solution s^{new} and b^{new} with energy function value $f(s^{new})$ while current solution is s^{old} and b^{old} with energy function value $f(s^{old})$ and the current temperature is T_r.

The rate at which the temperature parameter is reduced is vital to the success of any annealing process. This is governed by the number of repetitions at each temperature and the rate at which the temperature is reduced. The theory suggests that the system should be allowed to move very close to its stationary distribution at the current temperature before temperature reduction, and that the temperature should converge gradually to a value of zero. It also suggests that, in order to achieve this, a number of iterations exponential in problem size will be necessary at each temperature (Reeves, 1995).

Dhawan suggested a cooling schedule (Yeh and Fu, 2007) which has performed the best for SA algorithm in this study. It is defined as

$$T_r = \frac{T_0}{\ln(r+1)}, \qquad r = 1, 2, ..., \infty \tag{21}$$

where T_0 is the initial temperature and it has been set to 25 empirically for this study. The stopping condition has been met when $T_r = 3.5$.

The underlying design concept is that at a high temperature, it is quite probable to find a solution distant from the real global minimum having only a fewer energy function values than the current value. Normally, this kind of new solution will always be accepted. To reduce these undesirable moves, the acceptance probability function is defined as (Yeh and Fu, 2007)

$$P(T_r) = \frac{1}{1 + e^{\Delta f / T_r}} \tag{22}$$

where $\Delta f = f(s^{new}) - f(s^{old})$, since the energy function searches global maximum in this study, Δf is defined as $\Delta f = f(s^{old}) - f(s^{new})$.

4. SA AND SOME OTHER HEURISTICS FOR TIMETABLING PROBLEM

Since the problem is institution specific, there is not a commonly used solver tool for the university-course timetabling problem. In addition to this, the IUFBA test data is such huge that the model described in Eq. (A) – Eq. (8) can not be solved by any known software (e.g. Lingo 8.0). Thus, it is not possible to compare the results of this study to those of a previously proposed method or to those of reliable software.

In addition to SA, the most commonly used metaheuristic algorithms for this sort of problems are GA and TS. In order to compare the results of this study to those of the both, a similar structural design has been adapted into each of them.

Note that all the parameters mentioned in this study have been determined empirically for each one of the benchmark algorithms. That is, each algorithm has been tested individually for different parameter values. The parameter values which ensure the best solution are accepted and given in this paper. Thus, analyses are made with respect to those parameter values in section 5 and section 6.

4.1. The Solver SA Algorithm

It has been shown that all the obligatory rules are observed by penalizing the violations except for rule 1. Since the lectures are assigned to only one $b_{k,l}$ initially, there can be no repetition of any given lecture. Thus, whenever the energy function reaches a value that is greater or equal to zero, all of the obligatory rules would be observed, if the punitive number M is defined as seen in Eq. (20). Fig. 3 shows the solver SA algorithm.

In this algorithm, three additional terms which are *ratio*, *centers* and *shaking* take place. These terms are discussed successively as follows:

ratio: This is a variable which can have rational values between 0 and 1. Initially it has the value 1, but it is reduced by cooling schedule progressively.

centers: There are three center variables: *centerforb*, *centerford*, and *centerforh*. These all determine the selections for swapping operations. Initial center values are randomly set as follows:

$$\mathit{centerforb} = \text{round}\{\text{random}(0, 1) \times (K - 1)\} \tag{23}$$

$$\mathit{centerford} = \text{round}\{\text{random}(0, 1) \times (D - 1)\} \tag{24}$$

$$\mathit{centerforh} = \text{round}\{\text{random}(0, 1) \times (H - 1)\} \tag{25}$$

```
Create initial $s$ and $b$
Create initial centers
Set $ratio = 1$
Set $best\_s = s$
Set $best\_b = b$
Set $error$ = count of total errors for $b$
Set $besterror = error$
Set $energy = f(s)$
Set $bestenergy = energy$
while $T_r > 3.5$
  for $counter = 0$ to $1500$
      Set $bswapped$ = false
      Set $lectureswapped$ = false
      if random(0, 1) < 0.000033 then
        $energy = bestenergy$
        $error=besterror$
        $b = best\_b$
        $s = best\_s$
        $centerforb = bestcenterforb$
        $centerforh = bestcenterforh$
        $centerford = bestcenterford$
      Else if
        random (0, 1) < 0.000033 then
           shake centers
      End if
      if $besterror > 0$ and
        random(0,1) < 0.0001 then
          Call error reducer algorithm for $b$
      Else if random(0,1) < 0.71 then
        Select $d_1^*, h_1^*, d_2^*, h_2^*$ randomly
          depending on centers
        Set $bswapped$ = true
      End if
      if $bswapped$ = False or
        random(0, 1) < 0.71 then
        select $k_1^*, k_2^*$ randomly depending
        on $cenerforb$
        Set $lectureswapped$ = true
      End if
      if $bswapped$ = true then
        swap $S_{d_1^*,h_1^*}$ and $S_{d_2^*,h_2^*}$
      End if
      if $lectureswapped$ = true then
        select $l_1^*, l_2^*$ randomly
        swap $b_{k_1^*,l_1^*}$ and $b_{k_2^*,l_2^*}$
      End if
      $error$ = count of total errors for $b$
      $\Delta f = energy - f(s)$
      if $\Delta f < 0$ or
        random(0,1) < $P(T_r)$ then
        $energy = energy - \Delta f$
      Else
        if $bswapped$ = true then
          swap $S_{d_1^*,h_1^*}$ and $S_{d_2^*,h_2^*}$
        End if
        if $lectureswapped$ = true then
          swap $b_{k_1^*,l_1^*}$ and $b_{k_2^*,l_2^*}$
        End if
      End if
      if $bestenergy$ < energy then
        if $lectureswapped$ = true then
          $centerforb$ = round{($k_1^* + k_2^*$) / 2}
          $bestcenterforb = centerforb$
        End if
        if $bswapped$ = true then
          $centerforh$ = round{($h_1^* + h_2^*$) / 2)}
          $centerford$ = round{($d_1^* + d_2^*$) / 2)}
          $bestcenterforh = centerforh$
          $bestcenterford = centerford$
        End if
        $bestenergy$ = energy
        $besterror = error$
        $best\_s = s$
        $best\_b = b$
      End if
    End for
    if $bestenergy < 0$ then Set r = 0
    $r = r + 1$
    $T_r = T_0 / ln(r + 1)$
    $ratio = min\{2 / ln(r + 1), 1\}$
End while
```

Figure 3. The Solver SA Algorithm for The Problem

In this study, whenever a random selection occurs for b_k or $s_{d,h}$; the domain of the selection would be focused on these center values as follows:

ℓ= round {random (-*ratio* × 0.5 × (*K*-1), *ratio* × 0.5 × (*K*-1))} + *centerforb* (26)

∂= round {random (-*ratio* × 0.5 × (*D*-1), *ratio* × 0.5 × (*D*-1))} + *centerford* (27)

Ω= round {random (-*ratio* × 0.5 × (*H*-1), *ratio* × 0.5 × (*H*-1))} + *centerforh* (28)

$$\mathrm{k}_{\mathrm{n}}^{*} = \begin{cases} \ell \bmod (\mathrm{K}-1) & \text{if } \ell \geq 0 \\ \ell + \mathrm{K} & \text{if } \ell < 0 \end{cases} \tag{29}$$

$$\mathrm{d}_{\mathrm{n}}^{*} = \begin{cases} \partial \bmod (\mathrm{D}-1) & \text{if } \partial \geq 0 \\ \partial + \mathrm{D} & \text{if } \partial < 0 \end{cases} \tag{30}$$

$$\mathrm{h}_{\mathrm{n}}^{*} = \begin{cases} \Omega \bmod (\mathrm{H}-1) & \text{if } \Omega \geq 0 \\ \Omega + \mathrm{H} & \text{if } \Omega < 0 \end{cases} \tag{31}$$

shaking: This is an operation to randomly change the places of centers. The probability of shaking is 0.000033.

There are some probabilities in the algorithm. All of these are determined empirically. The swapping probability of both *b* and *s* at the same time is 0.71 × 0.71 ≈ 0.5 as seen in Figure 3.

4.2. Genetic Algorithms

GA is a population-based evolutionary heuristic, where every possible solution is represented by a specific encoding, often called individual (Colorni et al., 1998). First developments in GA field took place nearly 40 years ago. However, most early applications were in the realm of artificial intelligence – game playing and pattern recognition for instance. Some of the early researches focused on function optimization. Recently, many GA studies have taken place in Operational Research area. GAs were developed initially by Holland and his associates in the 1960s and 1970s. Goldberg gives an interesting survey of some of the practical work carried out in this era. The name genetic algorithm originates from the analogy between the representation of complex structure by means of a vector of components, and the idea, familiar to biologists, of the genetic structure of a chromosome (Reeves, 1995).

With respect to the structural design of this study, a GA is formed where population size is 30 and the possibility of mutation operation is set to 0.005. Each individual consists of two groups of chromosomes which represent *b* vectors and *s* matrix denoted by *sb* (the number of genes in *sb* is *K* × *L*) and *ss* (the number of genes in *ss* is *D* × *H*) respectively. This separated design ensures the number of lectures not to directly affect the chromosome lengths. That is because lectures are factors which increase the search space dramatically (Beligiannis et al., in press).

In many cases it has been observed that increasing the number of crossover points has improved the performance of GA (Reeves, 1995). Thus, the two-point crossover operator in which two parts of the first parent are copied and the rest between is taken in the same order as in the second parent is employed. Following example shows how the crossover operator works:

$sb_1 = \{2, 0, 0, \mathbf{0}, 0, 1, 3, 0, \mathbf{5}, 0, 0, 0, 0, 4, 0\}$ (crossover points = {4, 9})

$sb_2 = \{\underline{0}, \underline{0}, \underline{2}, \underline{\mathbf{1}}, \underline{0}, \underline{0}, \underline{0}, \underline{0}, \underline{\mathbf{4}}, \underline{5}, \underline{3}, \underline{0}, \underline{0}, \underline{0}, \underline{0}\}$

$ss_1 = \{\mathbf{-1}, -1, -1, 1, \mathbf{-1}, -1, 0, 2, -1\}$ (crossover points = {1, 5},

$ss_2 = \{\underline{\mathbf{-1}}, \underline{1}, \underline{-1}, \underline{-1}, \underline{\mathbf{2}}, \underline{-1}, \underline{0}, \underline{-1}, \underline{-1}\}$

$sb^{offspring} = \{2, 0, 0, \underline{1}, \underline{0}, \underline{0}, \underline{0}, \underline{5}, \underline{3}, 0, 0, 0, 0, 4, 0\}$

$ss^{offspring} = \{\underline{-1}, \underline{1}, \underline{-1}, \underline{-1}, \underline{-1}, -1, 0, 2, -1\}$

where $K = 3$, $L = 5$, $J = 5$, $D = 3$ and $H = 3$.

4.3. Tabu Search

TS is a higher-level metaheuristic procedure for solving discrete and continuous optimization problems. TS has its antecedents in methods designed to cross boundaries of feasibility or local optimality normally treated as barriers, and systematically to impose and release constraints to permit exploration of otherwise forbidden regions. The modern form of tabu search derives from Glover. Seminal ideas of the method are also developed by Hansen (Reeves, 1995). Zhao and Zeng (2008) studied a metaheuristic method which is combination of TS, SA and greedy search algorithm for optimizing transit networks, including route network design, vehicle headway, and timetable assignment. Causmaecker et al. (2008) studied university course timetabling. They developed a TS based metaheuristic and their work consists of a multistage approach to solve a non-weekly recurring real world timetabling problem with overlapping time slots. In order to generate a TS algorithm for the university course timetabling problem, this study uses a similar hyperheuristic of Burke at al. (2003) which can be outlined by the following pseudocode:

```
Do
    1- Select heuristic, $e$, with highest rank and apply it
    1- h1: Swap randomly selected elements between two random $b$ vectors and
            Swap elements of two random timeslots of $s$
    1- h2: Swap randomly selected elements between two random $b$ vectors
    1- h3: Swap elements of two random timeslots of $s$
    2-1 if $\Delta f > 0$ then $r_e = r_e + 1$
    2-2 else $r_e = r_e - 1$ and include $e$ in TABULIST
Until stopping condition is met
```

Where $\Delta f = f(s^{new}) - f(s^{\mathrm{old}})$ and r_e is the rank of heuristic e. This approach describes three movements which are denoted by h1, h2, and h3. Thus, this algorithm uses a sort of short term memory, which bars recent bad movements listed in TABULIST.

5. Applications and Discussions

The proposed algorithm has been tested with the 2006 – 2007 academic year, first term course timetabling data of IUFBA. The desire coefficient matrix consists of integer numbers between 0 and 5. There are three distinct seniority coefficient values which are 25, 5 and 1. These values correspond to the professors, to the associate professors and to the other academic staff respectively. The number of lecturers is 72.

Algorithm is tested on a Pentium M 2.13 computer. The average time of solving procedure is 1.74 minutes. Figure 4 and Figure 5 show the progression of the algorithm. As to the Figure 4, algorithm rapidly (in 37 iterations) finds positive solutions which mean feasible ones with the help of *error reducer* sub algorithm. However, it finds the most satisfactory solution relatively slowly (see Figure 5).

Table 2 gives a comparative summary of the results. Only the lecturers who make wishes are involved in those statistics, others are not involved since it doesn't matter for them where their lectures are assigned to. For each group of lecturers the mean desire coefficient and standard deviation of desire coefficients are given in Table 2. GA approach finds infeasible solutions. When compared to SA, it is seen that the TS approach finds not only less satisfactory solutions, but also more variant solutions, that is, mean desire coefficient does not represent satisfaction of all the lecturers in the group correctly. If the professors are considered, it will be seen that the standard deviation / mean is equal to 0.41 for SA approach, while it is 1.09 for TS approach.

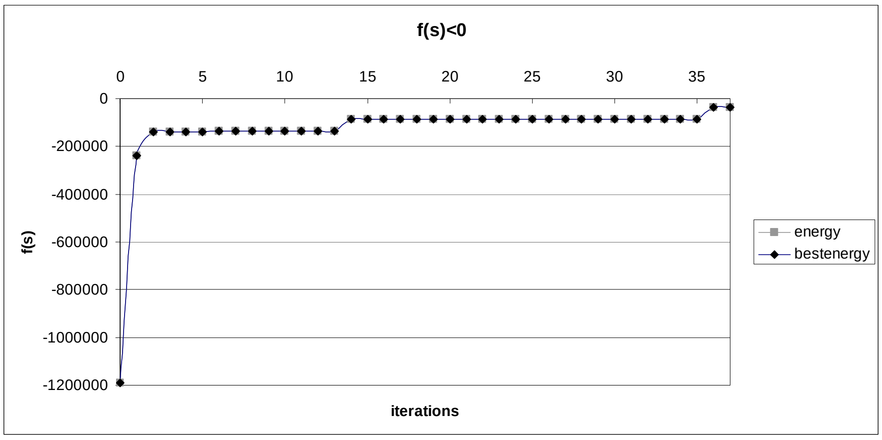

Figure 4. Progression of the Algorithm for First 37 Iterations

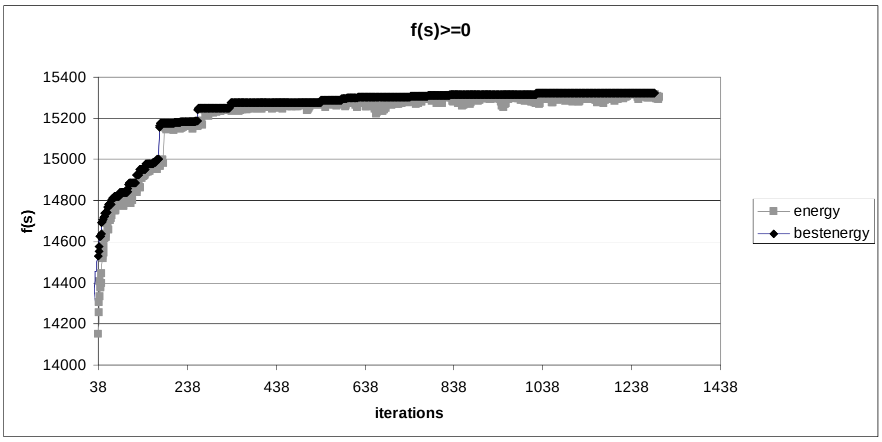

Figure 5. Progression of the Algorithm After 37th Iteration

Table 2. Desire Satisfaction Table of Lecturers

Seniority coefficient	**SA Approach**		**GA Approach**[***]		**TS Approach**	
	μ[*]	**σ**[**]	**μ**	**σ**	**μ**	**σ**
25(for professors)	4.117647	1.699447	1,003268	1,832142	1,98366	2,172379
5(for associate professors)	3.655172	1.767052	1,034483	1,586207	1,36782	1,727277
1(for the other academic staff)	2	2.020726	0,986111	1,665219	1,87847	2,166063

6. CONCLUSION

According to the results of application given in table 2, the GA algorithm has not reached a feasible solution, in other words, rules 4 and 5 have been violated. The *error reducer* sub-algorithm could not be adapted into the GA approach. However, it has produced much lesser satisfactory solution than SA approach. That sub-algorithm could be embedded into the TS approach and it has reached a non-violated solution which is still less satisfactory than that of the SA approach.

The mean desire coefficient of assigned lectures is 4.12 for professors in IUFBA. In other words, professor desires are satisfied with a proportion of 0.82 ($4.12 / 5 \approx 0.82$). This proportion is 0.73 for associate professors and it is 0.4 for the other academic staff. Thus, we can say that the SA based approach for the timetabling problem of IUFBA produces significant results.

[***] *Solution is errorenous since it includes violations of rules 4 and 5*
[*] *Mean desire coefficient*
[**] *Standard deviation of Desire coefficients*

7. REFERENCES

Abdennadher, S., and Marte, M., (2000), "University Course Timetabling Using Constraint Handling Rules", Applied Artificial Intelligence, 14, 311-325.

Abdullah, S., Ahmadi, S., Burke, E. K., and Dror, M., (2007), "Investigating Ahuja-Orlin's Large Neighborhood Search Approach for Examination Timetabling", OR Spectrum, 29, 351-372.

Anagnostopoulos, A., Michel, L., Van Hentenryck, P., and Vergados, Y., (2006), "A Simulated Annealing Approach to the Traveling Tournament Problem", Journal of Scheduling, 9, 177-193.

Avella, P., and Vasil'ev, I., (2005), "A Computational Study of Cutting Plane Algorithm for University Course Timetabling", Journal of Scheduling, 8, 497-514.

Beligiannis, G. N., Moschopoulos, C. N., Kaperonis, G. P., and Likothannassis, S. D., (Article in Press), "Applying Evolutionary Computation to the School Timetabling Problem: The Greek Case", Computers & Operations Research.

Burke, E. K., McCollum, B., Meisels, A., Petrovic, S., and Qu, R., (2007), "A Graph-Based Hyper-Heuristic for Educational Timetabling Problems", European Journal of Operational Research, 176, 177-192.

Burke, E. K., and Newall, J. P., (2004), "Solving Examination Timetabling Problems Through Adaptation of Heuristic Ordering", Annals of Operations Research, 129, 107-134.

Burke, E. K., Kendall, G., and Soubeiga, E., (2003), "A Tabu-Search Hyperheuristic for Timetabling and Rostering", Journal of Heuristics, 9, 451- 470.

Causmaecker, D. P., Demeester, P., and Berghe, G. V., (2008), "A Decomposed Metaheuristic Approach for A Real-World University Timetabling Problem", European Journal of Operational Research.

Colorni, A., Dorigo, M., and Maniezzo, V., (1998), "Metaheuristics for High School Timetabling", Computational Optimization and Applications, 9, 275-298.

Daskalaki, S., Birbas, T., and Housos, E., (2004), "An Integer Programming Formulation for A Case Study in University Timetabling", European Journal of Operatinal Research, 153, 117-135.

Head, C., and Shaban, S., (2007), "A Heuristic Approach to Simultaneous Course/Student Timetabling", Computers and Operations Research, 34, 919-933.

Henz, M., and Würtz, J., (1996), “Constraint-Based Timetabling- A Case Study”, Applied Artificial Intelligence, 10, 439-453.

Kirkpatrick, S., Gelatt, Jr. C. D., and Vecchi, M. P., (1983), “Optimization by Simulated Annealing”, Science, 220, 671-680.

Loukil, T., Teghem, J., and Fortemps, P., (2007), “A Multi-Objective Production Scheduling Case Study Solved by Simulated Annealing”, European Journal of Operational Research, 179, 709-722.

MirHassani, S. A., (2006), “A Computational Approach to Enhancing Course Timetabling with Integer Programming”, Applied Mathematics and Computation, 175, 814-822.

Reeves, C. R., (edt.), (1995), Modern Heuristic Techniques for Combinatorial Problems, McGraw-Hill.

Schimmelpfeng, K., and Helber, S., (2006), “Application of A Real-World University-Course Timetabling Model Solved by Integer Programming”, OR Spectrum, 1-21.

Yeh, J., and Fu, J. C., (2007), “Parallel Adaptive Simulated Annealing for Computer-Aided Measurement in Functional MRI Analysis”, Expert Systems with Applications, 33, 706-715.

Zhao, F., and Zeng, X., (2008), “Optimization of Transit Route Network, Vehicle Headways and Timetables for Large-Scale Transit Networks”, European Journal of Operational Research, 186, 841-855.