Hacettepe Journal of Mathematics and Statistics
Volume 36 (1) (2007), 53 – 64

A TABU SEARCH ALGORITHM TO SOLVE A COURSE TIMETABLING PROBLEM

Çağdaş Hakan Aladağ* and Gülsüm Hocaoğlu*

Received 13 : 06 : 2005 : Accepted 26 : 02 : 2007

Abstract

University course timetabling problems must be solved by the administration every year, or even term, and they involve a large amount of human and material resources. In the literature, the problem formulation does not contain the constraint that there should be no conflict between lessons in the same section. In this paper it is shown how a course timetabling problem which also includes this constraint can be formulated, and a tabu search algorithm is proposed to solve this problem. To show the effectiveness of the proposed algorithm, it is applied to the timetabling problem of the Statistics Department of Hacettepe University using a computer program based on this algorithm. It is observed that the proposed algorithm produces very good timetables that contain no conflict between lessons in the same section.

Keywords: Course timetabling, Metaheuristics, Tabu search, University.

2000 AMS Classification: 62 P 99

1. Introduction

The problem of building a university timetable consists of assigning instructor-course-room combinations into specific time periods. The objective in a classical course timetabling problem is to reduce the number of conflicts, which occur when courses involve common students, common teachers or require the same classrooms. For large institution such as universities, the problem becomes more difficult since additional constraints have to be taken into account.

Solving a course timetabling problem is very difficult. The main difficulty is related to the size of the problem. It involves a large number of students, teachers, courses and rooms, linked in many ways by objectives and conditions, and therefore each solution procedure must take into account very large number of variables and constraints. Moreover, the structure of the timetables varies from university to university, due to differences in the education systems. Even within a university, there are major differences among

*Hacettepe University, Faculty of Science, Department of Statistics, Beytepe, Ankara, Turkey. E-mail: (Ç. H. Aladağ) aladag@hacettepe.edu.tr (G. Hocaoğlu) hocaoglu@hacettepe.edu.tr

departments depending on the particular ways in which teaching is organized. For these reasons, during the last few decades many contributions related to course timetabling have appeared and a huge variety of timetabling models have been described in operations research literature [2].

The solution techniques range from graph coloring to complex metaheuristic algorithms, including linear programming formulations and heuristics tailored to the specific problem at hand. The more efficient procedures which have appeared in recent years are based on metaheuristics. Dowsland [6] and Elmohamed [7] have used simulated annealing, Burke [4], Corne [5] and Paechter [11] have developed procedures based on variants of genetic algorithms and Hertz [9], [10] and Alvarez [3] have used tabu search techniques. Gueret [8] has developed a different approach, Constraint Logic Programming.

The next section presents the elements of the problem solved. Mathematical formulation of the problem appears in Section 3. The proposed tabu search algorithm is expressed in Section 4 and the used program is summarized in Section 5. Section 6 gives obtained results from application of the proposed algorithm. Finally, in last section, result of the application of the proposed algorithm is discussed.

2. Objectives and Constraints of the Problem

First of all it is need to explain some basic concepts in course timetabling. Let 'Probability' course has two section courses, two hours theoretical and two hours practical per week. Its sections are denoted by 'Probability 01' and 'Probability 02' and its lessons are denoted by 'Probability 01 Theory', 'Probability 01 Practice', 'Probability 02 Theory' and 'Probability 02 Practice'. Class is a set of courses which is taken by a group of students. Lesson which has one hour can be assigned to a single period.

In a course timetabling problem, generally, constraints are considered in two types. One of them is called hard constraints. Every acceptable timetable must satisfy these constraints. For our problem, these constraints are:

- (H1) Every lesson has to be assigned to a period or a number of periods, depending on the lesson's length.
- (H2) No teacher can give two simultaneous lessons.
- (H3) All lessons of the same section cannot be assigned to the same day.
- (H4) No room can be assigned to two simultaneous lessons.
- (H5) The room assigned to a given lesson must be of the required type.
- (H6) All the preassignments and forbidden periods for classes, teachers and rooms must be respected.
- (H7) Lessons of sections of the same class cannot conflict.

We would like to note that Alvarez et al. [2] considered (H7) as a soft constraint. Instead of this, the constraint is considered as a hard constraint so it is guaranteed that the proposed tabu search algorithm will yield solutions which have no conflicts in the lessons of sections.

On the other hand, there are some conditions that are considered helpful but not essential in a good timetable. The more these conditions are satisfied, the better the timetable will be. They are called soft constraints and therefore they will have a weight in the objective function. For our problem these constraints are:

- (S1) Class timetables should be as compact as possible, eliminating idle times for students.
- (S2) In class timetables, students should not have more than a specified number of lessons hours.

(S3) In class timetables, if possible students should not have a day with a single lesson.
(S4) Teachers' non-desired periods should be avoided.
(S5) For rooms, the objective is adjusting their capacity to the number of students assigned to them.

3. Problem formulation

We need to define the following elements:

A	Set of courses
$Y = \{Y_{11}, Y_{12}, \dots, Y_{jk}, \dots\}$	Set of section lessons
Y_{jk}	Set of section k of course j
L	Set of lessons
C	Set of classes
$C_s = \{C_{11}, C_{12}, \dots, C_{ij}, \dots\}$	Set of classes with section
C_{ij}	Set of section j of class i
T	Set of teachers
LT_k	Set of lessons of teacher T_k
P	Set of periods
P_l	Set of periods of day l
D	Set of days
d_i	Duration of lesson i
R	Set of rooms
R_r	Set of rooms of type r
LR_r	Lessons requiring rooms of type r
TR	Different types of rooms
m_{rt}	Number of rooms of type r
F	Set of preassigned lessons
p_i	Preassigned period of lesson i
U_i	Set of forbidden periods of lesson i

We define the variables:

$$x_{ita} = \begin{cases} 1, & \text{if lesson } i \text{ starts at period } t \text{ in room } a \\ 0, & \text{otherwise} \end{cases}$$

In the objection function, for the constraints (S1), (S2) and (S3) which are for the students, the functions $f_{s1}(x)$, $f_{s2}(x)$ and $f_{s3}(x)$; for the constraint (S4) which is for the teachers, the function $f_t(x)$; and for the constraint (S5) which is for the rooms, the function $f_r(x)$ are defined. Each objective appears with its corresponding weight w: w_{s1}, w_{s2} and w_{s3} corresponding to the students, w_t corresponding to the teachers, and w_r corresponding to the rooms. In the computer program used, the weights can be determined by the user. In this way, a user can decide how much importance each constraint has.

The problem is [1]:

$$\min f(x) = w_{s1} f_{s1}(x) + w_{s2} f_{s2}(x) + w_{s3} f_{s3}(x) + w_t f_t(x) + w_r f_r(x), \tag{3.1}$$

subject to

$$\sum_{a\in R}\sum_{t\in P} x_{ita} = 1,\ \forall i \in L, \tag{3.2}$$

$$\sum_{a\in R}\sum_{i\in LT_k}\sum_{\tau=t-d_i+1}^{t} x_{i\tau a} \leq 1,\ \forall t \in P,\ \forall k \in T, \tag{3.3}$$

$$\sum_{a\in R}\sum_{t\in P_l}\sum_{i\in Y_{jk}} x_{ita} \leq 1,\ \forall Y_{jk} \in Y,\ \forall l \in D, \tag{3.4}$$

$$\sum_{i\in L}\sum_{\tau=t-d_i+1}^{t} x_{i\tau a} \leq 1, \forall a \in R,\ \forall t \in P, \tag{3.5}$$

$$\sum_{a\in R_r}\sum_{i\in LR_r}\sum_{\tau=t-d_i+1}^{t} x_{i\tau a} \leq m_{rt},\ \forall t \in P,\ \forall r \in TR, \tag{3.6}$$

$$\sum_{a\in R} x_{ip_i a} = 1,\ \forall i \in F, \tag{3.7}$$

$$\sum_{a\in R}\sum_{t\in U_i}\sum_{\tau=t-d_i+1}^{t} x_{i\tau a} = 0,\ \forall i \in L, \tag{3.8}$$

$$\sum_{a\in R}\sum_{i\in C_j}\sum_{\tau=t-d_i+1}^{t} x_{i\tau a} \leq 1,\ \forall C_{ij} \in C_s,\ \forall t \in P. \tag{3.9}$$

The constraints (2), (3), (4), (5), (6) and (9) are a mathematical expression of the hard constrains (H1), (H2), (H3), (H4), (H5) and (H7), respectively. Constraints (7) and (8) are an expression of constraint (H6).

For our problem, the number of days is 5 and there are 8 periods in each day, so there are 40 periods in all. There are 4 classes, 2 sections for courses, 36 section courses, 57 lessons, 27 teachers, 6 rooms and 2 types of room. In addition, we assume that there are no preassignments or forbidden periods for any lesson. In this case, there are 57 constraints for (2), 1080 for (3), 180 for (4), 240 for (5), 80 for (6), 10 for (7), 10 for (8) and 320 for (9). Thus we have a total of 1977 constraints, and the total number of variable is 13680. Since the number of variable is bigger than the number of constraints, this formulation cannot be solved by exact methods.

4. Solution method

A tabu search algorithm is proposed to solve the problem defined in section 3. In this section,the elements, parameters and operation of this algorithm are expressed.

4.1. Elements of the proposed tabu search algorithm. The elements of the tabu search algorithm in connection with the timetabling problem are defined below.

a) *The solution x.*

In each solution x, a number of variables equal to the number of lessons have the value 1 and the others have the value zero.

b) *The initial solution.*

The proposed tabu search algorithm starts from an initial solution. This initial solution is generated randomly.

c) *The solution space X*

This is the set of solutions satisfying constraints (2)–(9).

d) *The objective function* $f(x)$.

The objective function, in which each individual objective appears with its corresponding weight, is as shown in (1).

e) *The neighborhood* $N(x)$.

We have defined two alternative neighborhoods created by the following moves:

A **simple move**, in which a solution $x' \in X$ is a neighbor of solution $x \in X$ if it can be obtained from x by changing the assignment of one lesson i from one period t to another period t'.

The **swap move** or interchange of lessons, in which a solution $x' \in X$ is a neighbor of solution $x \in X$ if it can be obtained by interchanging the periods assigned to the two lessons i and i'.

The simple move and the swap move are illustrated in Figure 1 and Figure 2, respectively.

Figure 1. A Simple Move

Monday	Tuesday	Wednesday	Thursday	Friday
1	9	17	25	33
2	10	18	26	34
3	11 lesson i	19	27	35
4	12 lesson i	20	28	36
5	13 lesson i	21	29	37
6	14	22	30	38
7	15	23	31	39
8	16	24	32	40

Figure 2. A Swap Move

Monday	Tuesday	Wednesday	Thursday	Friday
1	9	17	25	33
2	10	18	26	34
3	11	19	27 lesson i'	35
4	12	20	28 lesson i'	36
5 lesson i	13	21	29	37
6 lesson i	14	22	30	38
7	15	23	31	39
8	16	24	32	40

For our problem, when the proposed algorithm is implemented, it is observed that the simple move produces better results than the swap move. Therefore, in our algorithm, we have used the simple move to produce neighborhoods.

f) *Candidate list strategy.*

For a given solution x, it is computationally too expensive to explore its whole neighborhood Nx. Therefore, each lesson is moved randomly and the best one is chosen. For example, in a simple move, a period to which a lesson is assigned is chosen randomly among all empty periods. However, randomness here is restricted, because the chosen next solution x' has to be a feasible solution satisfying the hard constraints. Thus, for solution x, instead of examining of all neighborhoods Nx, a candidate list consisting of neighbors which are as many as the number of lessons is examined.

g) *The tabu list.*

Complete solutions are not kept in the tabu list. *Attributive memory* is used for the tabu list, and the the e-attributes of an accepted move are stored. Changed lesson i, period $(\text{from} - t)_i$ at which the lesson i started before the change, and the room $(\text{from} - a)_i$ in which lesson i started before the change are kept in the tabu list. Therefore, when determining the tabu status of a move, the changed lesson i', period $(\text{to} - t)_{i'}$ to which the lesson i' is assigned, and the new room $(\text{to} - a)_{i'}$ are considered. For example, for a given move made by changing lesson i', if $i' = i$, $(\text{to} - t)_{i'} = (\text{from} - t)_i$, and $(\text{to} - a)_{i'} = (\text{f**rom} - a)_i$, this move is classified as tabu.

For the tabu list a "first in first out" (FIFO) data structure is used.

h) *The aspiration criterion.*

As an aspiration criterion, *global aspiration by objective* is used. If an explored move produces a solution x' with an objective function value lower than the objective function value of the best solution obtained so far, the move is made in spite of its tabu status.

i) *Selection of moves.*

It has been mentioned that the number of trial moves is the same as the number of lessons. These moves are ordered by the value of the objective function of solutions produced by them. Then the move with the best objective function value is chosen, and it is accepted, if it is not tabu. If it is tabu and satisfying the aspiration criterion it is accepted, but it does not satisfy the aspiration criterion, another move with the best objective function value is examined. Thus, the best acceptable move is chosen and the new current solution is the solution produced by this move.

k) *Intensification strategy.*

After a new starting solution, a solution with the best objective function value obtained in this new region is saved by using *explicit memory.* If a better solution than the saved one is not be obtained after a fixed number of iterations determined by the user, this saved solution will be examined again. Therefore, it is ensured that the search focuses on neighbors of good solutions.

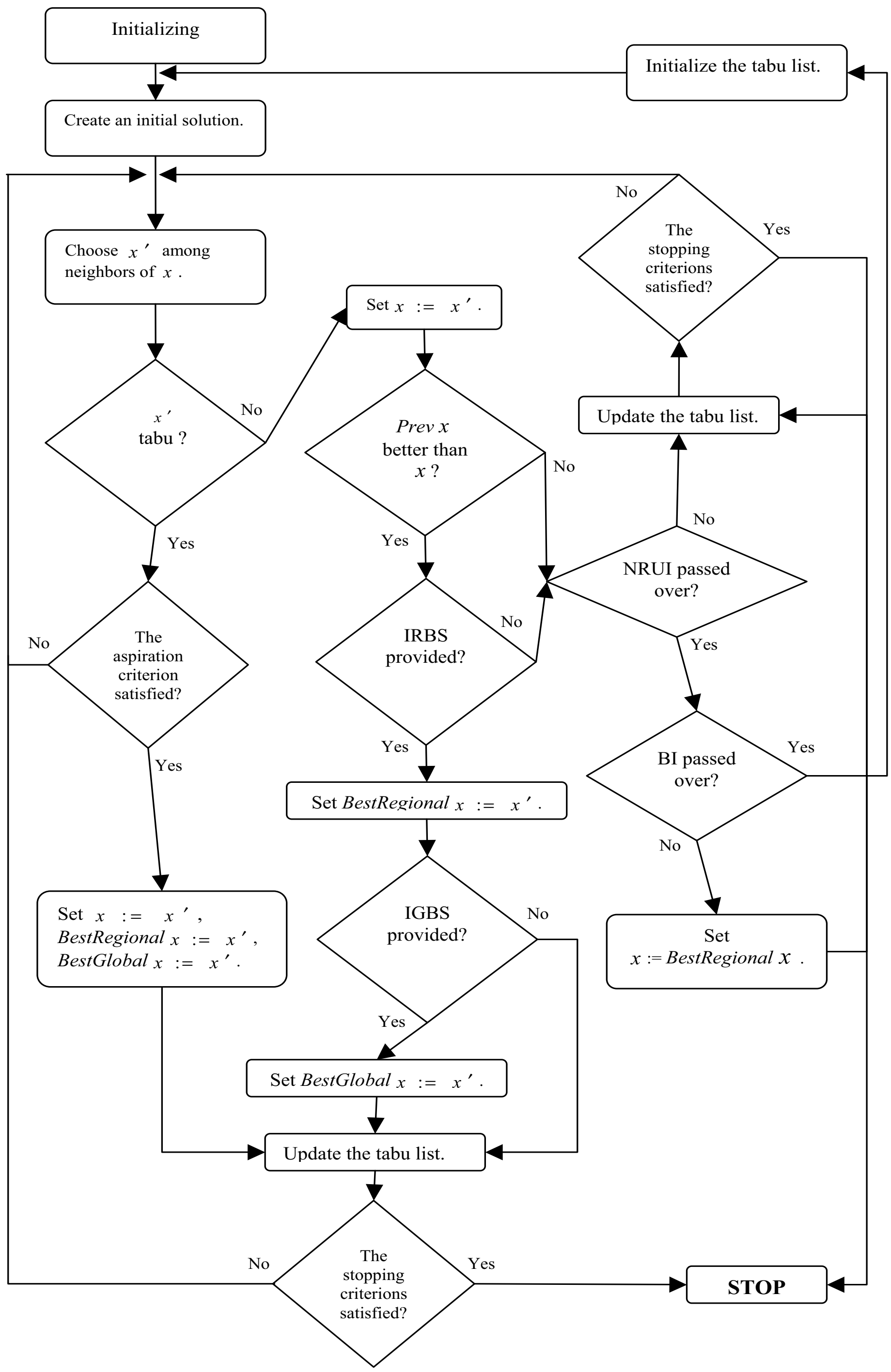
l) *Diversification strategy.*

As a diversification strategy, the *restarting strategy* is used. In an intensification strategy, the number of iterations made while the search is focusing on the region, in other word, how the number of times the search returns to the regional best solution is determined. For this number of iterations, if a solution better than the regional best solution is not found, the search will start to explore another region. A new initial solution is produced, and from this point, the algorithm begins to run again. Thus, it is ensured that the search examines different regions in the solution space.

m) *Stopping criteria.*

When a solution with a zero objective function value is obtained, the search is stopped. When the maximum iteration bound determined by the user is reached, or for the number of iterations specified by the user, if a solution better than the previous best one cannot be found, the search is terminated.

Figure 3. Flow chart of the proposed tabu search algorithm.



4.2. Parameters of the proposed tabu search algorithm. Five parameters lead the search can be determined by a user. These parameters are:

Iteration bound: This is the maximum a number of iterations that the search can continue for.

Number of global unimproved iterations: If a solution which is better than the best one found so far cannot be obtained for the number of global unimproved iterations, the search will be stopped.

Number of regional unimproved iterations (NRUI): If a solution better than the best one found regionally cannot be obtained for the number of global unimproved iterations, the search will restart from the regional best solution.

The boundary of intensification (BI): This represents the number of times the search can return to the regional best solution.

The length of tabu list: This represents the number of iterations for which the tabu status of a move continues.

4.3. Flow chart of the proposed tabu search algorithm. Let x, x', $\mathit{BestRegional}\,x$ and $\mathit{BestGlobal}\,x$ represent the current solution, a neighboring solution obtained from x, the best solution found regionally and the best solution found so far, respectively. Let $\mathit{Prev}\,x$ represent the previous current solution. Finally, let IRBS and IGBS represent the improvements in the regional best solution and in the global best solution, respectively. To explain how the proposed tabu search algorithm operates, its flow chart is shown in Figure 3 on the previous page.

5. The computer program used

In Figure 4, the main window of the program implementing the proposed tabu search algorithm is shown. A user can add or remove a course and change information of any course from here.

When a user clicks the 'Atama' (Assign) button in the main window, the window in Figure 5 appears. In this window, all the lessons are shown, and the user can define constraints about any lesson by clicking the 'Öncelik Belirle' (Constraints) button.

When a user clicks the 'Seçenekler' (Properties) button in the main window, the window in Figure 6 appears. There, a user may set the objectives' weights according to their importance and the tabu algorithm parameters from among a wide range of possibilities.

Finally, after clicking the 'Atama' (Assign) button, the'Çözüm' (solution) button in the main window will be enabled, and then a course timetable will be obtained by clicking 'Çözüm' (Solution) button. An example of program output for the problem was defined in the end of Section 3 is shown in Table 1. For instance, in the example, the lesson 'Kalite Denetimi' (Quality Control) starts at periods 6, on Friday, in room 2.

The computer program was coded using Delphi 7.0. Full information about the program can be found in Aladağ [1].

Figure 4. The Main window of the program

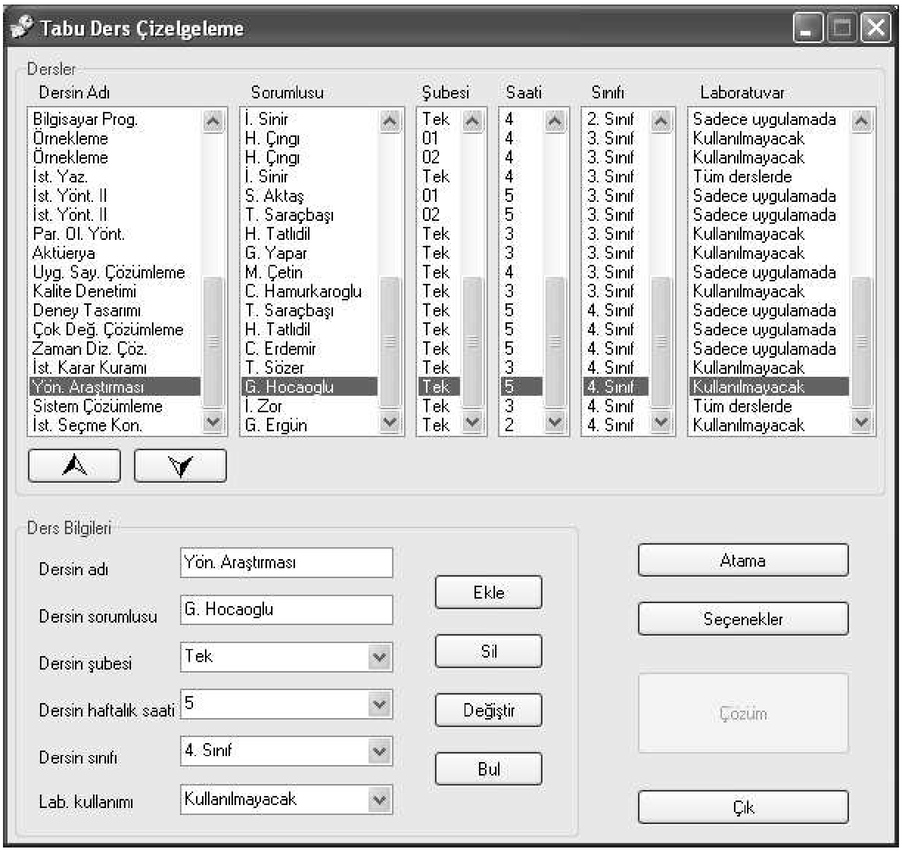


Figure 5. The Atama (Assign) window

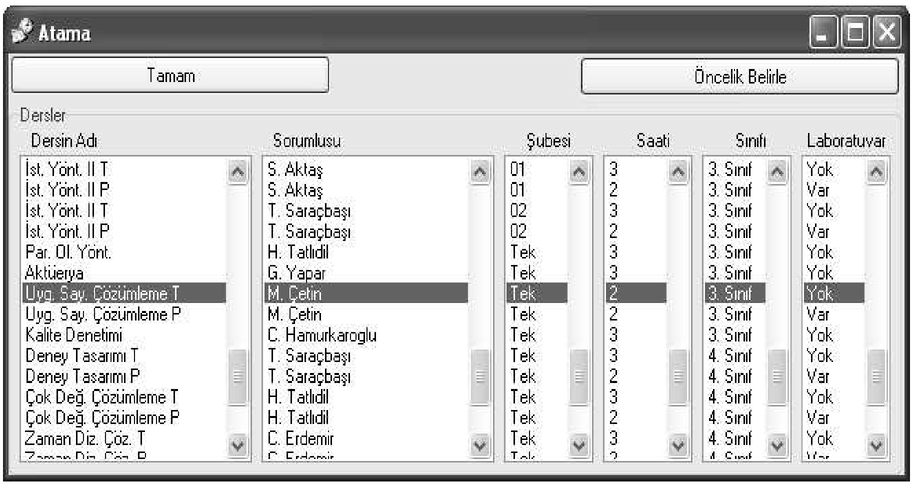


Figure 6. The Seçenekler (Properties) window

6. Application of the proposed tabu search algorithm

To show the effectiveness of the proposed algorithm, it was run 50 times for the problem defined at the end of section 3. The results are shown in Table 2, where 'Obj' represents the value of the objection function. The values of the weights and algorithm parameters are chosen as in Figure 6.

It is observed from Table 2 that the proposed algorithm yields successful results. The average value for these results is 1.43. For comparison, 50 solutions were generated randomly and their average value of the objection function was found to be 50.75. It is clearly seen that the proposed tabu search algorithm produced very good timetables. Moreover, all of these timetables certainly do not contain any conflict.

7. Conclusion

Alvarez et al. [2] gave a formulation of a timetabling problem, but in this formation conflicts in lessons of section was regarded as a soft constraint. That is, this given formulation does not contain this constraint. In this paper, a formulation of a timetabling problem containing this constraint is presented in (9). Then, to solve the problem posed, a tabu search algorithm is proposed. To show the effectiveness of the proposed algorithm, it is applied to a timetabling problem of the Statistics Department of Hacettepe University, using a computer program described in Section 5. It is observed from the results in Table 2 that the proposed tabu search algorithm produces very good time tables which do not contain conflicts. It is known that although there are many types of timetabling problem, all of them include similar components. Therefore, since the proposed tabu search algorithm is effective in solving the present problem, it can be used to solve timetabling problem of other Departments or any timetabling problem in Universities.

Table 1. An example of program output

No	Dersin Adı	Dersin Sorumlusu	Saati	Pazartesi	Salı	Çarşamba	Perşembe	Cuma
1	TKD	TKD01	2		3 - 4 (5)			
2	TKD	TKD01	2				3 - 4 (5)	
3	AİT	AİT01	2	2 - 3 (4)				
4	AİT	AİT01	2					3 - 4 (1)
5	Olasılık I	C. İnal	4		5 - 6 (5)	5 - 6 (1)		
6	Olasılık I	G. Ergün	4	6 - 7 (3)		2 - 3 (5)		
7	Mat. I	M. Türkyılmaz	5				6 - 7 (1)	3 - 4 - 5 (3)
8	Mat. I	M. Diker	5	1 - 2 - 3 (3)	7 - 8 (1)			
9	Doğ. Cebir I	T. Sözer	3				3 - 4 - 5 (1)	
10	Doğ. Cebir I	T. Sözer	3			6 - 7 - 8 (5)		
11	Bilg. Prog. Giriş	I. Zor	5			3 - 4 (Lab.)		6 - 7 - 8 (5)
12	Bilg. Prog. Giriş	I. Sinir	5		1 - 2 - 3 (3)		5 - 6 (Lab.)	
13	İleri Mat. I	H. Eş	5	4 - 5 (1)	4 - 5 - 6 (3)			
14	İleri Mat. I	R. Ertürk	5	1 - 2 (2)			2 - 3 - 4 (3)	
15	Mat. İst.	S. Günay	4			4 - 5 (5)		3 - 4 (2)
16	Mat. İst.	S. Günay	4	4 - 5 (4)	3 - 4 (2)			
17	Yön. Ar. Giriş	G. Hocaoğlu	5			6 - 7 (2)		5 - 6 - 7 (1)
18	Yön. Ar. Giriş	M. Sucu	5		1 - 2 (5)		6 - 7 - 8 (2)	
19	Bilgisayar Prog.	I. Sinir	4		7 - 8 (Lab.)			1 - 2 (1)
20	Örnekleme	H. Çıngı	4		4 - 5 (1)	3 - 4 (3)		
21	Örnekleme	H. Çıngı	4		6 - 7 (2)		1 - 2 (1)	
22	İst. Yaz.	İ. Sinir	4			1 - 2 (Lab.)		4 - 5 (Lab.)
23	İst. Yönt. II	S. Aktaş	5			5 - 6 - 7 (4)	1 - 2 (Lab.)	
24	İst. Yönt. II	T. Saraçbaşı	5			5 - 6 - 7 (3)		1 - 2 (Lab.)
25	Par. Ol. Yönt.	H. Tatlıdil	3		1 - 2 - 3 (1)			
26	Aktüerya	G. Yapar	3	6 - 7 - 8 (5)				
27	Uyg. Say. Çöz.	M. Çetin	4	4 - 5 (Lab.)			3 - 4 (4)	
28	Kalite Denetimi	C. Hamurkaroglu	3					6 - 7 - 8 (2)
29	Deney Tasarımı	T. Saraçbaşı	5		3 - 4 (Lab.)		5 - 6 - 7 (3)	
30	Çok Değ. Çöz.	H. Tatlıdil	5			2 - 3 - 4 (2)	3 - 4 (Lab.)	
31	Zaman Diz. Çöz.	C. Erdemir	5		5 - 6 (Lab.)			3 - 4 - 5 (5)
32	İst. Karar Kuramı	T. Sözer	3	1 - 2 - 3 (1)				
33	Yön. Araştırması	G. Hocaoglu	5	4 - 5 (5)				6 - 7 - 8 (3)
34	Sistem Çöz.	I. Zor	3			5 - 6 - 7 (Lab.)		
35	İst. Seçme Kon.	G. Ergün	2		7 - 8 (3)			

Table 2. Results of the proposed tabu search algorithm

No.	Obj	No.	Obj	No.	Obj	No.	Obj	No.	Obj
1	1.00	11	1.75	21	1.25	31	1.00	41	1.75
2	1.25	12	2.00	22	1.25	32	2.00	42	1.50
3	1.00	13	1.50	23	1.00	33	1.50	43	1.50
4	1.50	14	1.50	24	1.50	34	1.00	44	1.75
5	1.25	15	1.25	25	1.25	35	1.50	45	1.25
6	1.50	16	1.75	26	1.25	36	1.25	46	1.00
7	1.00	17	1.00	27	1.75	37	2.25	47	1.75
8	2.25	18	1.25	28	1.50	38	1.00	48	2.25
9	1.50	19	2.25	29	1.00	39	1.50	49	1.75
10	1.00	20	1.50	30	1.25	40	1.00	50	1.00

References

[1] Aladağ, Ç. H. *Solving a course timetabling problem by using tabu search algorithm*, (MSc Thesis (in Turkish), Department of Statistics, Hacettepe University, Ankara, 2004).

[2] Alvarez, R., Crespo, E. and Tamarit, J. M. *Design and implementation of a course scheduling system using tabu search*, European Journal of Operational Research **137**, 512–523, 2002.

[3] Alvarez, R., Martin, G. and Tarmarit, J. M. *Constructing good solutions for the Spanish school timetabling*, European Journal of Operational Research **47**, 1203–1205, 1996.

[4] Burke, E., Newall, J. P. and Weare, R. F. *A memetic algorithm for university exam timetabling*, in: Practice and Theory of Automated Timetabling, Lecture Notes in Computer Science **1153** (Springer, Berlin, 1996), 241–250.

[5] Corne, D. and Ross, P. *Peckish initialization strategies for evolutionary timetabling*, in: Practice and Theory of Automated Timetabling, Lecture Notes in Computer Science **1153** (Springer, Berlin, 1996), 227–240.

[6] Dowsland, K. A. *A timetabling problem in which clashes are inevitable*, Journal of Operational Research **41**, 907–918, 1990.

[7] Elmohamed, M. A. S., Coddington, P. and Fox, G. *A comparison of annealing techniques for academic course scheduling*, in: Practice and Theory of Automated Timetabling II, Lecture Notes in Computer Science **1408** (Springer, Berlin, 1997), 92–112.

[8] Gueret, C., Jussien, N., Boizumault, P. and Prins, C. *Building university timetables using constraint logic programming*, in: Practice and Theory of Automated Timetabling, Lecture Notes in Computer Science **1153** (Springer, Berlin, 1995), 130–145.

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

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

[11] Paechter, B., Cumming, A., Norman, M. G. and Luchian, H. *Extensions to a memetic timetabling system*, in: Practice and Theory of Automated Timetabling, Lecture Notes in Computer Science **1153** (Springer, Berlin, 1996), 251–265.