Expert Systems with Applications 36 (2009) 12349–12356

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

# The effect of neighborhood structures on tabu search algorithm in solving course timetabling problem

Cagdas Hakan Aladag [a,*], Gulsum Hocaoglu [a], Murat Alper Basaran [b]

[a] Department of Statistics, Hacettepe University, Ankara 06800, Turkey
[b] Department of Mathematics, Nigde University, Nigde 51000, Turkey

ARTICLE INFO

*Keywords:*
ANOVA
Combinatorial optimization
Course timetabling problem
Neighborhood structure
Tabu search
Type of move

ABSTRACT

The course timetabling problem must be solved by the departments of Universities at the beginning of every semester. It is a though problem which requires department to use humans and computers in order to find a proper course timetable. One of the most mentioned difficult nature of the problem is context dependent which changes even from departments to departments. Different heuristic approaches have been proposed in order to solve this kind of problem in the literature. One of the efficient solution methods for this problem is tabu search. Different neighborhood structures based on different types of move have been defined in studies using tabu search. In this paper, the effects of moves called simple and swap on the operation of tabu search are examined based on defined neighborhood structures. Also, two new neighborhood structures are proposed by using the moves called simple and swap. The fall semester of course timetabling problem of the Department of Statistics at Hacettepe University is solved utilizing four neighborhood structures and the comparison of the results obtained from these structures is given.

© 2009 Elsevier Ltd. All rights reserved.

## 1. Introduction

Assigning instructor-course-room combinations into specific time periods is a course timetabling problem which occurs in the Universities. 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 (Aladag & Hocaoglu, 2007a). For large institution such as Universities, the more constraints are added to the problem, the more difficult the solution of the problem is obtained.

Solving a course timetabling problem is a very difficult task because of the size of the problem and the nature of the changing structure of the problem. The solution techniques range from graph coloring to complex metaheuristic algorithms, including linear programming formulations and heuristics tailored to the specific problem at hand. MirHassani (2006) has developed an integer programming approach, Al-Yakoob and Sherali (2006, 2007) have used mixed-integer programming model, Boland, Hughes, Merlot, and Stuckey (2008) has used integer linear programming. Gueret, Jussien, Boizumault, and Prins (1995) has developed a different approach, constraint logic programming. The more efficient procedures which have appeared in recent years are based on metaheuristics. Dowsland (1990) and Elmohamed et al. (1997) have used simulated annealing, Burke, Newall, and Weare (1996), Corne and Ross (1996) and Paechter et al. (1996) have developed procedures based on variants of genetic algorithms and Alvarez, Martin, and Tarmarit (1996) and Hertz (1991, 1992) and have used tabu search techniques. Chiarandini, Birattari, Socha, and Rossi-Doria (2006) has introduced a hybrid metaheuristic algorithm which combines various construction heuristics, tabu search, variable neighborhood descent and simulated annealing. Burke, Petrovic, and Qu (2006) has used a case-based heuristic selection approach. Head and Shaban (2007) have developed a method based on heuristic functions. Pongcharoen, Promtet, Yenradee, and Hicks (2008) has described the stochastic optimization timetabling tool which includes genetic algorithms, simulated annealing and random search methods. Beligiannis, Moschopoulos, Kaperonis, and Likothanassis (2008) has designed an adaptive algorithm based on evolutionary computation techniques. Causmaecker (2009) has developed a decomposed metaheuristic approach.

One of the most efficient algorithms for the solution of the problem is tabu search algorithm. Tabu search method has proved its efficiency in the solution of the combinatorial optimization problems (1997). A tabu search algorithm consists of the usage of advanced strategies and common components such as tabu list, various memories, neighborhood structures, and so on (Aladag, 2004). One of the most important factors which affect the efficiency of the algorithm is a defined neighborhood structure pertained to the nature of the problem (Aladag & Hocaoglu, 2007b). In this paper, we examine the four different neighborhood structures based on types of move such as simple, swap. Two of the four neighborhood structures used in this study were used by Aladag and Hocaoglu (2007a) and Alvarez et al. (2002). The two neighborhood

* Corresponding author. Tel.: +90 312 2977900.
*E-mail address:* chaladag@gmail.com (C.H. Aladag).

0957-4174/$ - see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2009.04.051

structures differ in terms of the used moves which are simple and swap. Other used neighborhood structures proposed in this paper compose of combining the simple and swap moves. By doing so, we aimed at constituting a diversification effect in the used tabu search algorithm. Based on the usage of four neighborhood structures, the fall semester of course time tabling problem of the Department of Statistics at Hacettepe University is solved utilizing a tabu search algorithm introduced by Aladag and Hocaoglu (2007a). According to obtained results, multiple comparisons among all neighborhood structures are statistically conducted.

Section 2 includes the definition and the formulation of the solved problem. In Section 3, the flow chart of the used tabu search algorithm and the definitions of the used neighborhood structures based on simple and swap moves are given. The proposed neighborhood structures are introduced in Section 4. In the implementation section, the defined problem is solved based on four neighborhood structures and the results are given. The last section concludes the study.

## 2. The solved course timetabling problem

The definition of the solved problem given below was introduced by Aladag and Hocaoglu (2007a).

### 2.1. Objectives and constraints of the problem

First of all, it is necessary to explain some basic concepts in course timetabling problem. Let 'Probability' course has two section courses, two hours theoretical and 2 h 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 1 h 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 pre-assignments and forbidden periods for classes, teachers and rooms must be respected.
(H7) Lessons of sections of the same class cannot conflict.

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 weights 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 lessons hours more than a specified quantity.
(S3) In class timetables, students should not have a day with a single lesson as possible.
(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.

### 2.2. Problem formulation

First of all, following symbol should be denoted:

| | |
|---|---|
| $A$ | Set of courses |
| $Y=\{Y_{11},Y_{12},\ldots,Y_{jk},\ldots\}$ | Set of section lessons |
| $Y_{jk}$ | Set of section $k$ of course $j$ |
| $L$ | Set of lessons |
| $C$ | Set of classes |
| $C_s=\{C_{11},C_{12},\ldots,C_{ij},\ldots\}$ | Set of classes with section |
| $C_{ij}$ | Set of section $j$ of class $i$ |
| $\mathcal{T}$ | Set of teachers |
| $LT_k$ | Set of lessons of teacher $\mathcal{T}_k$ |
| $P$ | 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 pre-assigned lessons |
| $p_i$ | Pre-assigned period of lesson $i$ |
| $U_i$ | Set of forbidden periods of lesson $i$ |

The variables are defined as follows:

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

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

The problem is:

$$\text{Min}\quad f(x)=w_{s1}f_{s1}(x)+w_{s2}f_{s2}(x)+w_{s3}f_{s3}(x)+w_tf_t(x)+w_rf_r(x) \tag{1}$$

$$\text{subject to}\quad \forall i\in L,\quad \sum_{a\in R}\sum_{t\in P}x_{ita}=1 \tag{2}$$

$$\forall t\in P,\quad \forall k\in\mathcal{T},\quad \sum_{a\in R}\sum_{i\in LT_k}\sum_{\tau=t-d_i+1}^{t}x_{i\tau a}\leqslant 1 \tag{3}$$

$$\forall Y_{jk}\in Y,\quad \forall l\in D,\quad \sum_{a\in R}\sum_{t\in P_l}\sum_{i\in Y_{jk}}x_a\leqslant 1 \tag{4}$$

$$\forall a\in R,\quad \forall t\in P,\quad \sum_{i\in L}\sum_{\tau=t-d_i+1}^{t}x_{i\tau a}\leqslant 1 \tag{5}$$

$$\forall t\in P,\quad \forall r\in TR,\quad \sum_{a\in R_r}\sum_{i\in LR_r}\sum_{\tau=t-d_i+1}^{t}x_{i\tau a}\leqslant m_{rt} \tag{6}$$

$$\forall i\in F,\quad \sum_{a\in R}x_{ip_ia}=1 \tag{7}$$

$$\forall i\in L,\quad \sum_{a\in R}\sum_{t\in U_i}\sum_{\tau=t-d_i+1}^{t}x_{i\tau a}=0 \tag{8}$$

$$\forall C_{ij}\in C_s,\quad \forall t\in P,\quad \sum_{a\in R}\sum_{i\in C_j}\sum_{\tau=t-d_i+1}^{t}x_{i\tau a}\leqslant 1 \tag{9}$$

Constraints (2)–(6) and (9) are mathematical expression of hard constrains (H1), (H2), (H3), (H4), (H5) and (H7), respectively. Constraints (7) and (8) are expression of constraint (H6).

A sample problem is introduced to show the dimension of the timetabling problem. For example, the number of days is 5 and there are 8 periods in each day, and therefore there are totally 40 periods. There are 4 classes, 2 sections for courses, 36 section courses, 57 lessons, 27 teachers, 6 rooms and 2 types of rooms. 10 pre-assigned lessons and forbidden periods for 10 lessons are assumed in advance. 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 1977 constraints totally and total number of variable is 13680.

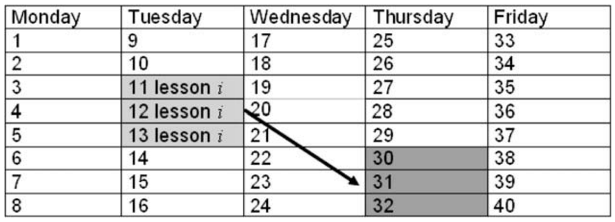
## 3. Used tabu search algorithm

Tabu search algorithm which was introduced by Glover is a well known heuristic method (Glover & Laguna, 1997). In this study, the tabu search algorithm introduced by Aladag and Hocaoglu (2007a) is used. The algorithm is given below.

### 3.1. Elements of the proposed tabu search algorithm

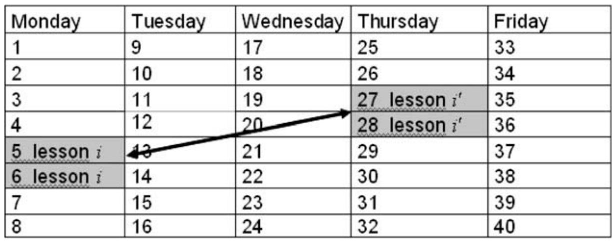
Elements of the tabu search algorithm in connection to timetabling problem are defined in here.

(a) *The solution x*: Each solution $x$ is a vector whose elements are variables of $x_{ita}$ that is previously defined. The number of elements taking the value of 1 is equal to the number of lessons. The rest of the elements are equal to 0.
(b) *The initial solution*: The proposed tabu search algorithm begins with an initial solution. This initial solution is generated randomly.
(c) *The solution space X*: It is set of solutions satisfying constraints (2)–(9).
(d) *The objective function f(x)*: The objective function, in which individual objective appears with its corresponding weight, is shown at (1).
(e) *The neighborhood N(x)*: Two alternative neighborhoods are created by the following moves:The 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 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 two lessons $i$ and $i'$.
The simple move and the swap move are illustrated in Figs. 1 and 2, respectively.
(f) *Candidate list strategy*: For a given solution $x$, it is computationally too expensive to explore its whole neighborhood $N(x)$. 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 herein is restricted, because the chosen next solution $x'$ has to be a feasible solution satisfying hard constraints. Thus, for solution $x$, instead of examining of all neighborhood $N(x)$, a candidate list, consist of neighbors which are equal to the number of lessons is examined.



| 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 |

**Fig. 1.** The simple 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 |

**Fig. 2.** The swap move.

(g) *The tabu list*: The solutions are not kept in tabu list completely. *Attributive memory* is used for tabu list, e-attributes of accepted move is stored. Changed lesson $i$, period $(from - t)_i$ at which lesson $i$ started before the change, and room $(from - a)_i$ in which lesson $i$ started before the change are kept in tabu list. Therefore, when determining tabu status of a move, changed lesson $i'$, period $(to - t)_{i'}$ to which lesson $i'$ is assigned, and new room $(to - a)_{i'}$ are considered. For example, for a given move made by changing lesson $i'$, if $i' = i, (to - t)_{i'} = (from - t)_i$, and $(to - a)_{i'} = (from - a)_i$ this move classified as tabu. For duration of tabu, it is not allowed to assign a changed lesson resulting from a move to a period at which and a room in which the lesson have started before this move.For the tabu list, "first in first out" (FIFO) rule is used as a data structure.
(h) *The aspiration criterion*: As an aspiration criterion, *global aspiration by objective* is used. If an explored move produces a solution $x'$ with the objective function value lower than objective function value of the best solution obtained so far, the move is made in spite of its tabu status.
(i) *Selection of the best neighbor*: It is mentioned that the number of trial moves is the same as the number of lessons. The current solution's neighbors generated by these moves are ordered according to objective function values of the neighbors. 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 satisfies the aspiration criterion, it is accepted, but if 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 next current solution is the one produced by this move.
(k) *Intensification strategy*: After new starting solution is generated, solution with the best objective function value obtained in this new region is saved by using *explicit memory*. If a solution better than the saved one, which is the regional best solution, cannot be obtained during the number of iterations determined by the user, the algorithm will return to the saved solution. Then, same operations are repeated by setting the saved solution as a new starting solution. Therefore, it is provided that the search focuses on neighbors of good solutions. The number of how many returns will be taken place is another parameter of the tabu search algorithm.
(l) *Diversification strategy*: For a diversification strategy, *restarting strategy* is used. In intensification strategy, the number of iterations that algorithm uses for focusing on a region and how many times the search will return to regional best solution are determined. These are parameters of tabu search algorithm for intensification strategy. After the pre-deter-

mined number of iterations for focusing is finished, the algorithm will start to explore another region if a solution better than the regional best solution is not found. A new initial solution is generated and the algorithm begins to run again from this point. Thus, it is expected that the algorithm examines different regions in the solution space.

(m) *Stopping criterions*: When the objective function reaches zero value, the algorithm is stopped. When the pre-determined maximum iteration bound or the specified number of iterations is reached, the algorithm will be terminated if a solution better than the best one found so far cannot be found.

### 3.2. Parameters of the proposed tabu search algorithm

Five parameters led to the algorithm can be determined by the user. These parameters are:

(1) *Iteration bound*: It is a maximum number of iterations that the search can continue for.
(2) *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.

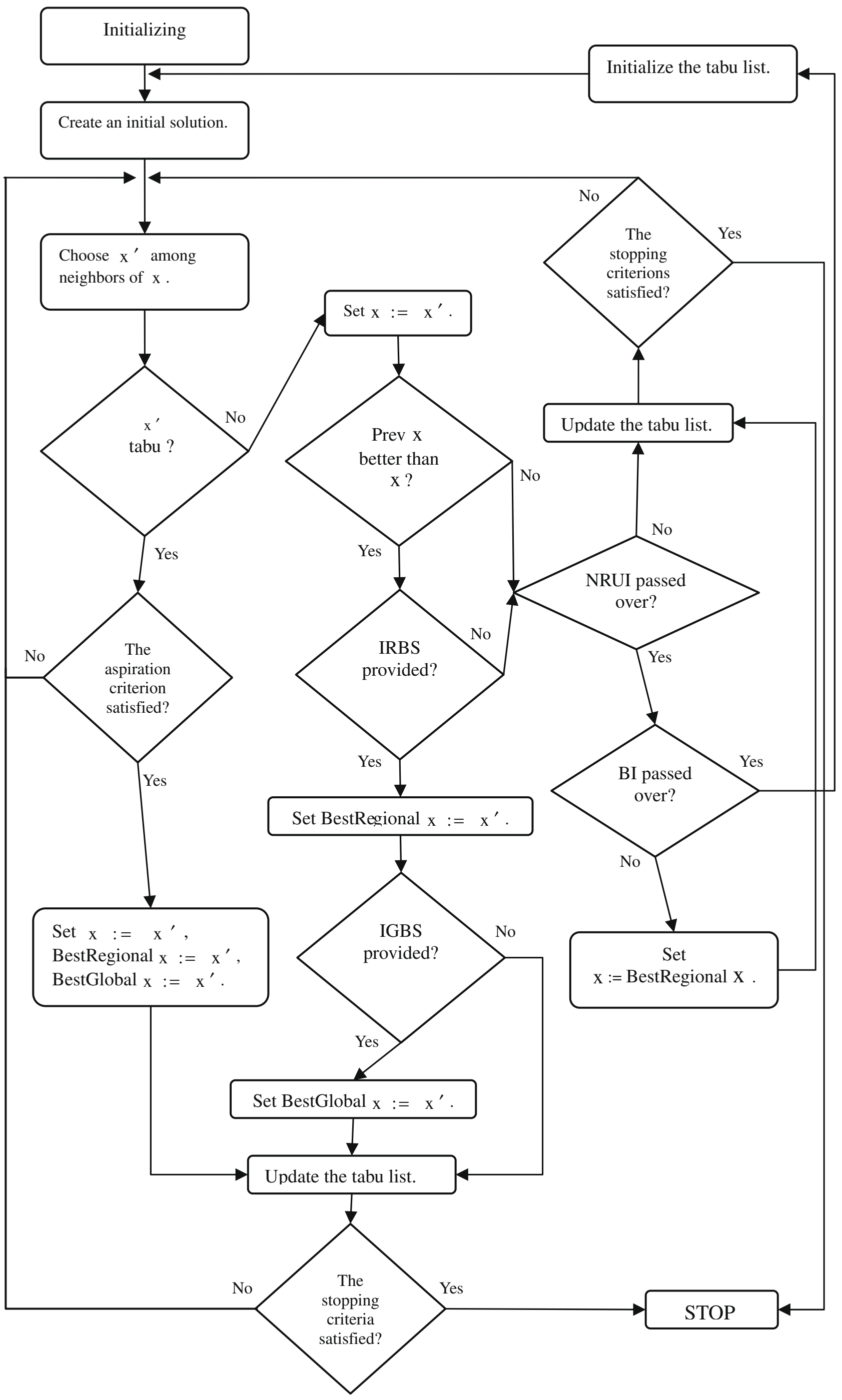

Initializing
Initialize the tabu list.
Create an initial solution.
No
Yes
The stopping criterions satisfied?
Choose x′ among neighbors of x.
Set x := x′.
x′ tabu ?
No
Prev x better than x ?
No
Update the tabu list.
Yes
Yes
NRUI passed over?
No
No
The aspiration criterion satisfied?
IRBS provided?
No
Yes
Yes
Yes
BI passed over?
Set BestRegional x := x′.
No
Set x := x′, BestRegional x := x′, BestGlobal x := x′.
IGBS provided?
No
Set x := BestRegional x.
Yes
Set BestGlobal x := x′.
Update the tabu list.
No
The stopping criteria satisfied?
Yes
STOP

**Fig. 3.** Flow chart of the used tabu search algorithm.

(3) *Number of regional unimproved iterations* (NRUI): If a solution better than the best one found regionally cannot be obtained for the number of regional unimproved iterations, the search will restart from the regional best solution again.
(4) *The boundary of intensification* (BI): It represents the number of returning of the search to the best regional solution.
(5) *The length of tabu list*: It is represents that the number of iterations which the tabu status of a move continues.

### 3.3. Flow chart of the used tabu search algorithm

Let $x, x', BestRegional\ x$ and $BestGlobalx$ denote the current solution, any neighbor solution obtained from $x$, the best solution found regionally and the best solution found so far, respectively. Let *Prevx* denote the previous current solution. Finally, let IRBS and IGBS denote improvement in the regional best solution and in the global best solution, respectively. To explain how the used tabu search algorithm operates, its flow chart is shown in Fig. 3 (Aladag, 2004).

## 4. The proposed neighborhood structures

Two neighborhood structures are introduced in Aladag and Hocaoglu (2007a) using simple and swap moves which are used by Alvarez, Crespo, and Tamarit (2002). In the previous sections, how these neighborhood structures work is explained in detail. The only difference between these two structures is the type of used move. Aladag and Hocaoglu (2007a) preferred the simple move to swap move based on the results of their proposed algorithm, while Alvarez et al. (2002) did not prefer the simple move according to the problem they solved. In this paper, we proposed two new neighborhood structures which combine both these two moves.

The simple move enables the algorithm to reach better solutions dependent upon the solved problem in Aladag and Hocaoglu (2007a). In the simple move, only just one lesson can be changed in a course timetabling. In the swap move, however, two lessons can be interchanged. Therefore, the swap move has more diversification effect than has the simple move. The aim of the proposed neighborhood structures is to combine the advantages of two moves.

In the first proposed neighborhood structure, after a starting solution is generated, the algorithm begins with this generated solution by using simple move. If a solution ($x$) which is better than the best regional solution (*brs*) is not found at the end of the predetermined NRUI, the algorithm returns to the best regional solution's neighbor generated by the swap move instead of returning to the regional best solution. In this case, the value of the objective function found using swap move should be equal to or less than the value of the best regional solution. While a new solution is being generated from the regional best solution, the selection of the best neighbor procedure given in Section 3.1 is used. Thus, the diversification effect is constituted since the algorithm is not allowed to return to the same best regional solution.

Let *itr* denotes by the iteration numbers and $f(x)$ be the objective function. Pseudo code of the first proposed neighborhood structure is given below.

```
If ($itr$ = NRUI) and (such a $x$ satisfying $f(x < f(brs)$, is still not found) then
    Use the swap move to generate neighbor of $brs$ $x'$ such that $f(x') \leqslant f(brs)$
    The next solution = $x'$
```

In the second proposed neighborhood structure, a different way of combining the simple and swap moves is designed. Both the simple and swap moves are used to generate the neighbors of the current solution, instead of using just one of them. The best solutions, which are not tabu, generated by these moves are compared with each other in terms of objective function values. After the comparison, the solution which has smaller objective function value is assigned as the next solution. In other words, the smaller objective function determines which move will be used to generate the solution which will be the current solution for the next iteration. If the objective function values generated by these moves are equal, the solution generated by the simple move is preferred since this move shown that better solutions are obtained for the solved problem (2007a). In the process of selecting the best neighbor, the procedure given in Section 3.1 is used.

Let $f(x)$ be the objective function, $x_S$ and $x_{SW}$ be the best neighbors of the current solution generated by the simple and swap move, respectively. Pseudo code of the second proposed neighborhood structure is given below.

```
If $f(x_S) < f(x_{SW})$ then
The next solution = $x_S$
If $f(x_S) = f(x_{SW})$ then
The next solution = $x_S$
If $f(x_S) > f(x_{SW})$ then
The next solution = $x_{SW}$
```

New modules which include the proposed neighborhood structures are attached to the program used in Aladag & Hocaoglu (2007a). The modules are coded in Delphi programming language.

## 5. Implementation

Simple_N and Swap_N stand for neighborhood structures where simple and swap moves are used, respectively. These structures were used by Aladag & Hocaoglu (2007a). We called the first and second proposed neighborhood structures as Mixed_1 and Mixed_2, respectively. In order to gauge the performance of all neighborhood structures, timetables are generated randomly by a method which is used to generate the random starting solutions. Fifty timetable solutions for each employed methods are produced and each corresponding objective function value is also kept.

**Table 1**
Descriptive statistics.

| | *N* | Mean | Std. Deviation | Std. Error | Minimum | Maximum |
|---|---|---|---|---|---|---|
| Mixed_1 | 50 | 0.7250 | 0.56975 | 0.08058 | 0.00 | 2.00 |
| Mixed_2 | 50 | 19.3450 | 3.86206 | 0.54618 | 11.25 | 26.25 |
| Sirnple_N | 50 | 0.7900 | 0.57888 | 0.08187 | 0.00 | 2.00 |
| Swap_N | 50 | 3.8300 | 2.59535 | 0.36704 | 2.50 | 13.00 |
| Random | 50 | 43.4400 | 5.56193 | 0.78658 | 33.50 | 58.25 |
| Total | 250 | 14.6260 | 16.29893 | 1.03084 | 0.00 | 58.25 |

Number of global unimproved iterations, NRUI, BI, and the length of tabu list are taken as 200, 8, 2, and 12, respectively. This situation is same as the one in Aladag & Hocaoglu (2007a). Iteration bound is taken as 320. The weights in the objective function $w_{s1}, w_{s2}, w_{s3}, w_t$, and $w_r$ are taken as 3, 1, 1, 2, and 2, respectively.

The descriptive statistics of the 5 employed methods for 50 runs are given in Table 1 below. All statistical analyses are done using SPSS 15 version.

In order to show the difference statistically among all employed methods, ANOVA test is conducted for the mean values of the objective function. First, test of homogeneity is conducted and the result is given in Table 2.

**Table 2**
Test of homogeneity.

| Levene statistic | df1 | df2 | Sig. |
|---|---|---|---|
| 46.537 | 4 | 245 | 0.000 |

Based on Levene statistic, the variances for all methods are not assumed to be equal at the 0.05 significance level. Therefore, Dunett T3 Post Hoc test is used for multiple comparisons. Table 3 denotes the results of this test.

According to Table 3, Mixed_1 and Simple_N structures are not different at the 0.05 significance level. Mixed_2, Swap_N, and Random methods are different from all other methods. From Table 1, the mean values of the objective function for Mixed_1, Mixed_2, Simple_N, Swap_N, and Random are 0.725, 19.345, 0.790, 8.830, and 43.440, respectively. It is observed that Mixed_1 and Simple_N are statistically same methods and produce the best results. The second best method is Swap_N, the third one Mixed_2. The worst case is the results generated by Random method. In other words, No matter which structure is employed in the Tabu Algorithm, it is certain that a better solution than a solution generated randomly can be obtained.

In order to show the operations of neighborhood structures visually, the graphs below are given. The horizontal axis represents the number of iteration and the vertical axis represents the values

**Table 3**
Dunett T3 Post Hoc.

| | | Mean difference | | | 95% Confidence interval | |
|---|---|---|---|---|---|---|
| (1) Method | (J) Method | (I–J) | Std. Error | Sig. | Lower bound | Upper bound |
| Mixed_1 | Mixed_2 | −18.62000* | 0.55209 | 0.000 | −20.2310 | −17.0090 |
| | Simple_N | −0.06500 | 0.11487 | 1.000 | −0.3936 | 0.2636 |
| | Swap_N | −8.10500* | 0.37578 | 0.000 | −9.1994 | −7.0106 |
| | Random | −42.71500* | 0.79069 | 0.000 | −45.0244 | −40.4056 |
| Mixed_2 | MixedJ | 18.62000* | 0.55209 | 0.000 | 17.0090 | 20.2310 |
| | Simple_N | 18.55500* | 0.55228 | 0.000 | 16.9435 | 20.1665 |
| | Swap_M | 10.51500* | 0.65805 | 0.000 | 8.6267 | 12.4033 |
| | Random | −24.09500* | 0.95761 | 0.000 | −26.8417 | −21.3483 |
| Simple_N | MixedJ | 0.06500 | 0.11487 | 1.000 | −0.2636 | 0.3936 |
| | Mixed_2 | −18.55500* | 0.55228 | 0.000 | −20.1665 | −16.9435 |
| | Swap_M | −8.04000* | 0.37606 | 0.000 | −9.1350 | −6.9450 |
| | Random | −42.65000* | 0.79082 | 0.000 | −44.9597 | −40.3403 |
| Swap_N | MixedJ | 8.10500* | 0.37578 | 0.000 | 7.0106 | 9.1994 |
| | Mixed_2 | −10.51500* | 0.65805 | 0.000 | −12.4033 | −8.6267 |
| | Simple_N | 8.04000* | 0.37606 | 0.000 | 6.9450 | 9.1350 |
| | Random | −34.61000* | 0.86800 | 0.000 | −37.1153 | −32.1047 |
| Random | MixedJ | 42.71500* | 0.79069 | 0.000 | 40.4056 | 45.0244 |
| | Mixed_2 | 24.09500* | 0.95761 | 0.000 | 21.3483 | 26.8417 |
| | Simple_N | 42.65000* | 0.79082 | 0.000 | 40.3403 | 44.9597 |
| | Swap_M | 34.61000* | 0.86800 | 0.000 | 32.1047 | 37.1153 |

* The mean difference is significant at the 0.05 level.

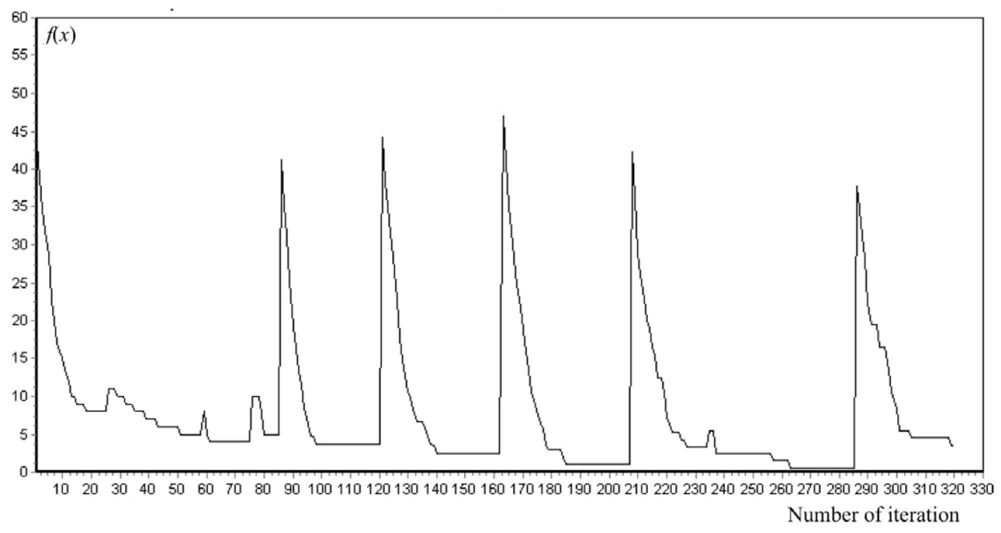


**Fig. 4.** An example for Mixed_1.

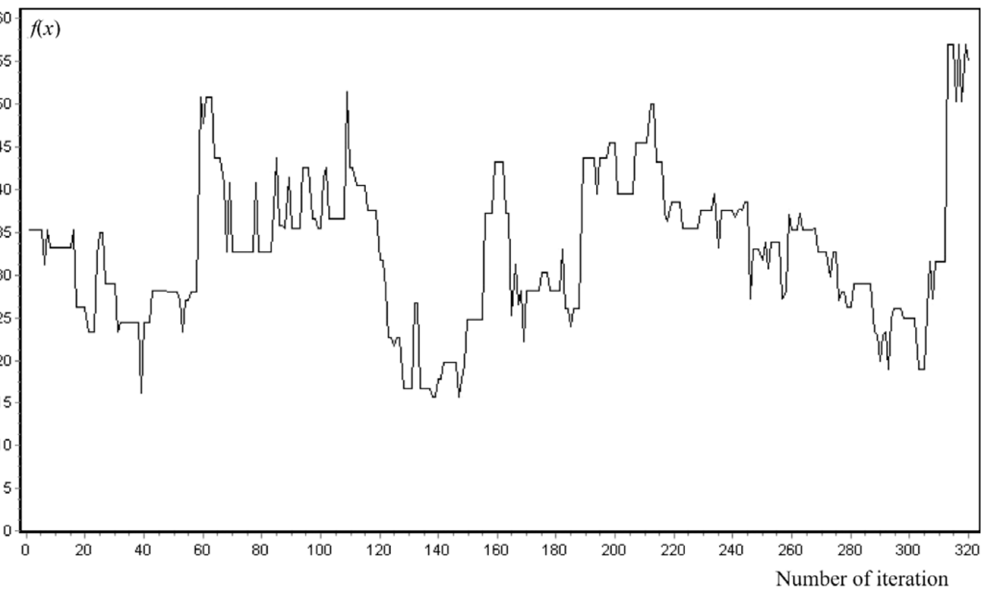


**Fig. 5.** An example for Mixed_2.

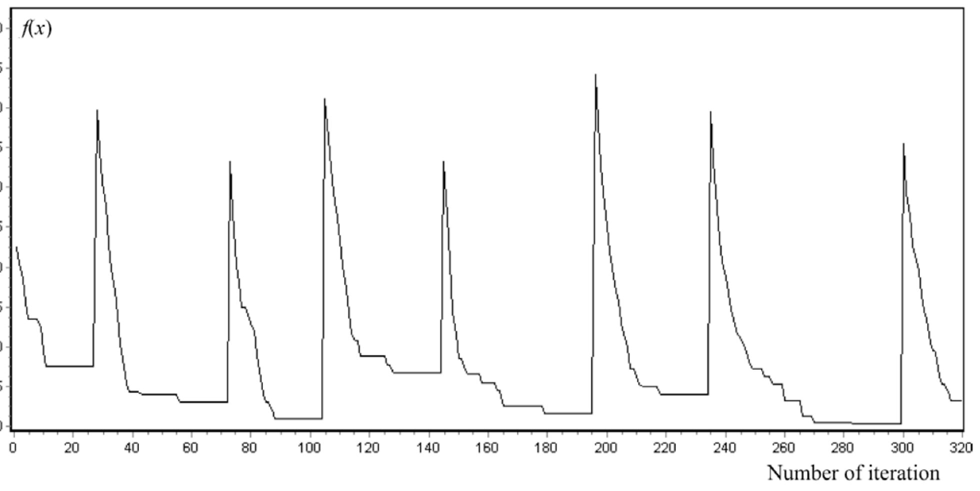


**Fig. 6.** An example for Simple_N.

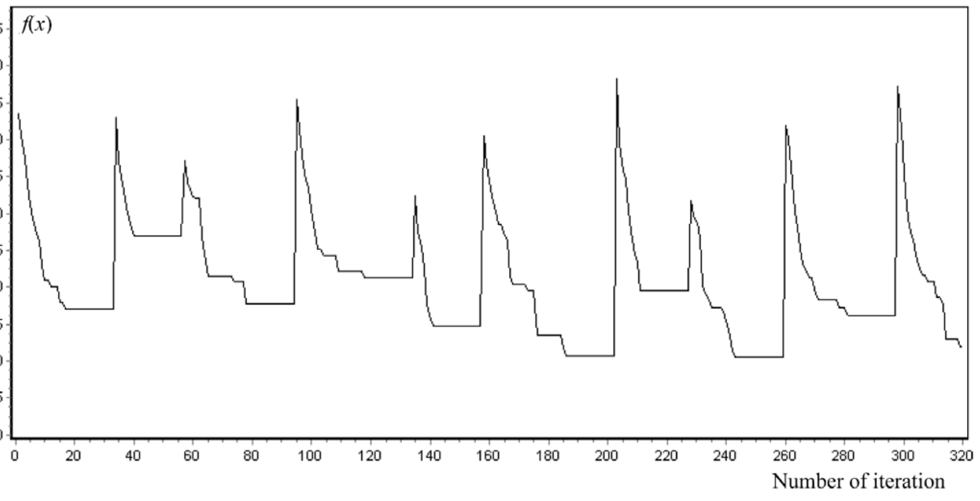


**Fig. 7.** An example for Swap_N.

of the objective function ($f(x)$) in the graphs. Figs. 4–7 show the operations of the tabu search algorithm when Mixed_1, Mixed_2, Simple_N, and Swap_N are used, respectively. Each figure shows the graph of just one sample of the 50 runs. When the graphs are examined, the obtained findings at the end of statistical analysis can be also seen visually.

## 6. Conclusion

Two neighborhood structures based on the simple and the swap moves were used in Aladag & Hocaoglu (2007a) & Alvarez et al. (2002). These moves affect the Tabu Algorithm in different aspects. The simple move causes more intensification effect since it alters just one lesson. On the other hand, the swap move causes more diversification effect since it interchanges two lessons. In this paper, we proposed two new neighborhood structures based on these moves since we aim at combining the different characters of these moves. The course timetabling problem of the Department of Statistics of Hacettepe University is solved by using these all structures. Also, random solutions are generated for the problem. Then, the results are statistically examined. The best timetables are obtained using Mixed_1, which is one of the proposed structures, or Simple_N. As a result of statistical analysis, it is observed that there is no statistical difference between Mixed_1 and Simple_N structures in terms of the values of the objective function. The second and the third best structures are Swap_N and Mixed_2, respectively. All of the employed neighborhood structures produce better solutions than those generated randomly. The effects of the neighborhood structures are also examined visually and similar results are observed from the graphs.

In a research done by Aladag & Hocaoglu (2007a), Simple_N was preferred to Swap_N since it produces better timetables. Mixed_1 which is proposed in this paper can be used as a good alternative for Simple_N. Alvarez et al. (2002) claimed that the swap move produces better timetables than does the simple move for their problem and Tabu Algorithm. The observed difference between Aladag & Hocaoglu (2007a) & Alvarez et al. (2002) comes from the defined problems and improved algorithms. Therefore, where the simple move or the swap move lacks of producing good timetables, Mixed_1 may be a good alternative neighborhood structure since this structure benefits from both moves at the same time. This result was obtained both statistically and visually.

Although combining both the swap and the simple moves in a way which is done in Mixed_2 seems to be a good approach, Mixed_2 neighborhood structure is not able to produce good timetables. This result was obtained both statistically and visually. The intensification ability of the tabu search algorithm diminishes since the next solution is obtained based on different move types at the each iteration. In other words, when Mixed_2 neighborhood structure is used, the algorithm cannot overcome local minimum trap since the direction of the searching can change at the each iteration due to using different moves which are the simple and the swap. Thus, the combining effect provided by Mixed_2 does not produce timetables as well as Mixed_1 does. This is an important result for the future studies. When composing new neighborhood structures by combining the different move types, the results mentioned herein should be taken into account.

To sum up, the four neighborhood structures based on two different move types are statistically compared to observe the effects on tabu search algorithm. Two new neighborhood structures are proposed in order to exploit the different features of the simple and the swap moves. We have statistically shown that the proposed neighborhood structure called Mixed_1 reaches this aim. Therefore, Mixed_1 can be used in future studies for solving various timetabling problems and improving various tabu search algorithms.

## References

Aladag, C. H. (2004). Solving a course timetabling problem by using tabu search algorithm. Master's thesis, Hacettepe University, Graduate School.

Aladag, C. H., & Hocaoglu, G. (2007a). A tabu search algorithm to solve course timetabling problem. *Hacettepe Journal of Mathematics and Statistics*, *36*(1), 53–64.

Aladag, C. H., Hocaoglu, G. (2007b). The effect of neighborhood structure and of move types in the problem of course timetabling with the tabu search algorithm. In *Proceedings of the fifth statistics conference* (pp. 14–19).

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

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

Al-Yakoob, S., & Sherali, H. (2006). Mathematical programming models and algorithms for a class-faculty assignment problem. *European Journal of Operational Research, 173*(2), 488–507.

Al-Yakoob, S. M., & Sherali, H. D. (2007). A mixed-integer programming approach to a class timetabling problem: A case study with gender policies and traffic considerations. *European Journal of Operational Research, 180*(3), 1028–1044.

Beligiannis, G. N., Moschopoulos, C. N., Kaperonis, G. P., & Likothanassis, S. D. (2008). Applying evolutionary computation to the school timetabling problem: The Greek case. *Computers and Operations Research, 35*(4), 1265–1280.

Boland, N., Hughes, B. D., Merlot, L. T. G., & Stuckey, P. J. (2008). New integer linear programming approaches for course timetabling. *Computers and Operations Research, 35*, 2209–2233.

Burke, E., Newall, J. P., & Weare, R. F. (1996). A memetic algorithm for university exam timetabling. *Practice and theory of automated timetabling. Lecture notes in computer science*. Berlin: Springer (pp. 241–250, Vol. 1153).

Burke, E. K., Petrovic, S., & Qu, R. (2006). Case based heuristic selection for timetabling problems. *Journal of Scheduling, 9*(2), 115–132.

Causmaecker, D. P., Demeester, P., & Berghe, G. V. (2009). A decomposed metaheuristic approach for a real-world university timetabling problem. *European Journal of Operational Research, 195*(1), 307–318.

Chiarandini, M., Birattari, M., Socha, K., & Rossi-Doria, O. (2006). An effective hybrid algorithm for university course timetabling. *Journal of Scheduling, 9*(5), 403–432.

Corne, D., & Ross, P. (1996). Peckish initialization strategies for evolutionary timetabling. *Practice and theory of automated timetabling. Lecture notes in computer science* (Vol. 1153). Berlin: Springer (pp. 227–240).

Dowsland, K. A. (1990). A timetabling problem in which clashes are inevitable. *The Journal of Operational Research Society, 41*, 907–918.

Elmohamed, M. A. S., Coddington, P., & Fox, G. (1997). A comparison of annealing techniques for academic course scheduling. *Practice and theory of automated timetabling II. Lecture notes in computer science* (Vol. 1408). Berlin: Springer (pp. 92–112).

Glover, F., & Laguna, M. (1997). *Tabu search*. Dordrecht: Kluwer Academic Publishers.

Gueret, C., Jussien, N., Boizumault, P., & Prins, C. (1995). Building university timetables using constraint logic programming. *Practice and theory of automated timetabling. Lecture notes in computer science* (Vol. 1153). Berlin: Springer (pp. 130–145).

Head, C., & Shaban, S. (2007). A heuristic approach to simultaneous course/student timetabling. *Computers and Operations Research, 34*(4), 919–933.

Hertz, A. (1992). Finding a feasible course schedule using tabu search. *Discrete Applied Mathematics, 35*, 255–270.

Hertz, A. (1991). Tabu search for large scale timetabling problems. *European Journal of Operational Research, 54*, 39–47.

MirHassani, S. A. (2006). A computational approach to enhancing course timetabling with integer programming. *Applied Mathematics and Computation, 175*(1), 814–822.

Paechter, B., Cumming, A., Norman, M. G., & Luchian, H. (1996). Extensions to a memetic timetabling system. *Practice and theory of automated timetabling. Lecture notes in computer science* (Vol. 1153). Berlin: Springer (pp. 251–265).

Pongcharoen, P., Promtet, W., Yenradee, P., & Hicks, C. (2008). Stochastic optimisation timetabling tool for university course scheduling. *International Journal of Production Economics, 112*, 903–918.