

Tabu Search Approach to the Solution of the General Lectures Scheduling Problem

Do Xuan Duong and Pham Huy Dien

Institute of Mathematics, 18 Hoang Quoc Viet Road, 1037 Hanoi, Vietnam

Received September 22, 2002

Revised July 8, 2003

Abstract. The lectures scheduling problem of a large-scale size (suitable for realistic applications) is considered very hard and cannot be solved by exact deterministic methods. Ferland *et al.* have succeeded with a heuristic approach based on the Exchange Procedure and gained remarkable results. In this paper we apply Tabu Search technique for solving this problem and show that with the appropriate Tabu list length and search neighborhood size one can obtain the similar or even better results, for problems of larger scale.

1. Introduction and Mathematical Model

The lectures scheduling problem (Timetabling Problem) discussed in this paper deals with the case when lectures do not have the same length, and scheduling is done for each semester based on a repeated weekly timetable. Each lecture is scheduled at a time period in a week. The goal is to construct an optimal timetable, which means that there are fewest unsolvable scheduling conflicts and the preferences of lecturers are fulfilled most. Two lectures are said to be in conflict if they are held (at least partly) simultaneously and

- they are given by the same lecturer, or
- they require the same classroom, or
- at least one student is registered in both lectures.

In the general lectures scheduling problem, beside the constraint that each lecture has to take place at a time period of the week, the constraints for avoiding conflict situations between lectures play an important role, because if they are in conflict then only one of them can be scheduled at the given time, and the other one has to be moved.

Given n lectures and m time periods, let x_{ij} ($i = 1, \dots, n, j = 1, \dots, m$) be the decision variable, that is

$$x_{ij} = \begin{cases} 1, & \text{if lecture } i \text{ starts at time period } j \\ 0, & \text{otherwise} \end{cases}$$

We use the following notations:

J_{ijk} - the set of time periods for lecture k which are conflicting the assignment of lecture i to time period j ;

B - the number of classrooms types;

K_{bj} - the set of lectures which may take place at the time period $j, j = 1, \dots, m$, and require a classroom of type $b, b = 1, \dots, B$;

Q_{bj} - the number of classrooms of type b that can be used at the time period j .

Assignment constraints mean that each lecture has to take place at exactly one time period of the week. This can be described as follows

$$\sum_{j=1}^m x_{ij} = 1, \quad i = 1, \dots, n.$$

Additional side constraints for avoiding conflicts situations can be specified as follows

$$x_{ij} + x_{kl} \leq 1, \quad l \in J_{ijk}, \quad i < k \leq n, \quad 1 \leq i \leq n, \quad 1 \leq j \leq m,$$

and the additional side constraints ensuring that the utilization of classrooms does not exceed their availabilities can be specified as follows

$$\sum_{i \in K_{bj}} x_{ij} \leq Q_{bj}, \quad j = 1, \dots, m, \quad b = 1, \dots, B.$$

The goal is to determine an assignment for each lecture to a time period in order to minimize total assigning cost and fulfill additional side constraints.

Denote by $c1_{ij}$ the cost of assigning lecture i to the time period j . This cost must be specified in terms of availability and preferences of the lecturer responsible for lecture i . In this model, the preference of lecturers for each time period is ranging from 1 (highly preferred) to 4 (lecturer unavailable). For that, the cost structure is specified as follows

$$c1_{ij} = \begin{cases} k, & \text{if time period } j \text{ is ranked } k \text{ for lecture } i, \quad 1 \leq k \leq 3 \\ n.m + 1, & \text{if time period } j \text{ is ranked } 4 \text{ for lecture } i. \end{cases}$$

Then, the total cost of lectures assignment is

$$F(x) = \sum_{i=1}^n \sum_{j=1}^m c1_{ij} x_{ij},$$

and the mathematical model can be described as follows

$$(P1) \quad F(x) = \sum_{i=1}^n \sum_{j=1}^m c1_{ij} x_{ij} \rightarrow \min$$

subject to

- 1) $\sum_{j=1}^m x_{ij} = 1, i = 1, \dots, n; x_{ij} \in \{0, 1\}, i = 1, \dots, n, j = 1, \dots, m.$
- 2) $x_{ij} + x_{kl} \leq 1, l \in J_{ijk}, n \geq k > i, i = 1, \dots, n, j = 1, \dots, m.$
- 3) $\sum_{i \in K_{bj}} x_{ij} \leq Q_{bj}, j = 1, \dots, m, b = 1, \dots, B.$

The constraints of type 1) ensure that each lecture is assigned at exactly one time period during a week. The constraints of type 2) aim at eliminating the conflict situation between two lectures. The constraints of type 3) ensure that the total number of lectures that take place the time period j does not exceed the number of usable classrooms at that time.

This is a $\{0, 1\}$ - integer programming problem. The most difficult thing in the above model is how to deal with the constraints of type 2) because with each lecture i at time period j there is a set of lectures that conflict with lecture i at that time. To overcome this difficulty, we use penalty technique to include those constraints into the objective function. The constraints of type 3) are also included in the similar way. With such a penalty technique, we can deal with the constraints in a flexible way, giving priority to one type of constraints or the other by changing their penalty coefficients.

The mathematical model can now be formulated as follows

$$\begin{aligned}
 \text{(P2)} \quad & \sum_{i=1}^n \sum_{j=1}^m \left[c1_{ij} x_{ij} + \frac{1}{2} \sum_{k=1}^n \sum_{l=1}^m c2_{ijkl} x_{ij} x_{kl} \right. \\
 & \left. + c3 \sum_{b=1}^B \text{Max} \left\{ 0, \sum_{i \in K_{bj}} x_{ij} - Q_{bj} \right\} \right] \rightarrow \min \\
 & \text{subject to} \\
 & \sum_{j=1}^m x_{ij} = 1, \quad i = 1, \dots, n; x_{ij} \in \{0, 1\}, i = 1, \dots, n, j = 1, \dots, m.
 \end{aligned}$$

Here $c3$ is the penalty coefficient related to the shortage of classrooms and

$$c2_{ijkl} = \begin{cases} M.t_{ijkl} & \text{when } l \in J_{ijk} \\ 0 & \text{when } l \notin J_{ijk} \end{cases},$$

where M is a certain constant and t_{ijkl} is the overlapping time between lecture i and lecture k when they are assigned to time periods j and l , respectively. The parameter $c3$ can be controlled by the user in relation with $c2_{ijkl}$, depending on what constraints should be given priority. Obviously, (P2) is a nonlinear integer programming problem.

2. Solution Technique

Currently there are a number of methods for solving integer programming problems, such as: the Gomory's cutting method, the Land-Doig branch and bound method,... and a number of software dealing with the integer programming problems have been designed. However, the lectures scheduling problem (P1) and/or

(P2) cannot be solved efficiently by the known techniques, because of the non-linearity and the number of constraints for avoiding conflicts between lectures is too large. For instance, scheduling of 10 lectures for only 5 successive time periods (in a day) generates the problem with about 500 constraints. Moreover, the number of these constraints grows very fast when the size of the problems (the number of lectures and time periods) increases.

In practice, some heuristic approaches, such as: the Tabu search, the Exchange Procedure, etc... proved to be more effective for solving the problem in question. Ferland *et al.* [1, 5] have succeeded with the Exchange Procedure technique and gained remarkable results for middle-sized problems (number of lectures is less than 500). As mentioned in [1], the large number of conflict situations between lectures is the main reason for high costs of assignment (penalty coefficient associated with this situation is relatively high). To overcome this, the authors propose to divide the problem into several subproblems of smaller scales. This essentially reduces the number of conflicts between lectures and makes the problem tractable easier. In fact, the lectures scheduling problem was solved at the level of faculties or groups. However, the optimality of timetables at the faculty level does not imply the optimality of the corresponding timetable at the university level. In this paper we apply Tabu search technique for solving the general lectures scheduling problem and show that with the appropriate Tabu list length and search neighborhood size we can obtain the similar or better results. It is essential that we can treat the problem of larger size and, hence, solve the lectures scheduling problem for a university in the whole.

Tabu search allows the search to move away from the local minima in order to search more extensively in the feasible domain. A short term Tabu list is used as a safeguard against cycling. The length of the Tabu list (TL) is a regulable parameter dependent on specific properties and the size of problem. This list (TL) includes the most recent modifications used. When the Tabu list is full, the oldest element will be eliminated if a new one is brought in. By this way, the algorithm allows the chance to come back to the points, which are formerly forbidden. With the current solution x , a neighborhood $N(x)$ of x is generated for searching procedure. Only points of the set $N(x) \setminus TL$ are subject to search. If a searched point $a \in N(x) \setminus TL$ is found better than x , then the current solution is moved to point a , and x is included to the Tabu list together with all the "related" points. In the other case, the point a is included in a list of searched points around x , which is denoted by $N^*(x)$. Since the size of the neighborhood $N(x)$ increases rapidly with the problem size, it may become unreasonable to scan the whole neighborhood to identify the best neighboring solution. In implementing Tabu search method, we often restrain the search to a subset of $N(x)$, by limiting the number of points of the set $N^*(x)$. When the number of elements of $N^*(x)$ reaches the limit value NV , then x is included to the Tabu list and the following tasks are taking place:

- The best point of $N^*(x)$ is chosen as the current solution;
- The *best current point* parameter is updated by x if it is not better than x .

The algorithm stops if one of the following situations happens:

- The total computing time reaches a limit value, or
- The number of iterations reaches a certain maximal value, or
- The number of successive iterations with no improvement reaches a certain given value.

The *best current point* is taken to be the final solution.

3. Algorithm

Let

$$X = \left\{ x : \sum_{j=1}^m x_{ij} = 1, 1 \leq i \leq n, x_{ij} \in \{0, 1\}, 1 \leq i \leq n, 1 \leq j \leq m \right\}$$

and let x be a *current solution*. Denote by $R(x, i)$ the time period when lecture i is taking place and by $z = x \oplus (i, j)$ the solution z that is generated from x by reassigning lecture i from time period $R(x, i)$ to the time period j (all other lectures remain unchanged).

We define the neighborhood of the current solution x , i.e. $N(x)$, as follows

$$N(x) = \left\{ z \in X : \exists (i, j) \text{ such that } z = x \oplus (i, j), 1 \leq i \leq n, 1 \leq j \leq m, j \neq R(x, i) \right\}.$$

Two points x and x' are said to be *relative at* (i, j) if $R(x, i) = R(x', i) = j$.

The set of all relative points at (i, j) is denoted by $[i, j]$ and is called the *relative family* at (i, j) . Therefore, $x \in [i, j]$ if $R(x, i) = j$.

We define a *Tabu list* (TL) to be a set of relative families and the *associated Tabu set* (TLS) to be the set of all points that belong to one of the relative families in the Tabu list. Thus,

$$TLS = \{ x : x \in [i, j], [i, j] \in TL \}.$$

The algorithm proceeds as follows.

Step 0. Let x be an initial solution of the problem (P2).

Specify:

MCT - the limit value for total computing time,

MI - the limit value for total of iterations,

$MSIN$ - the maximal number of successive iterations with no improvement,

NV - the limit value for number of elements of $N^*(x)$,

LTL - the length of the Tabu list.

Set:

CCT - the variable counting the total computing time,

CI - the variable counting the total of iterations,

$CSIN$ - the variable counting the number of successive iterations with no improvement,

$CNTL$ - the variable counting the number of elements in the Tabu list,

BCP - the best current point.

Let $CSIN = 0$, $CI = 0$, $CCT = 0$, $CNTL = 0$,
 $N^*(x) = \emptyset$, $TL = \emptyset$, $BCP = x$.

Step 1. Generate an element $z \in N(x) \setminus TLS$ by choosing at random a pair (i, j) such that $[i, j] \notin TL$, and setting $z = x \oplus (i, j)$.

If $F(z) < F(x)$ then let $x' = z$, and update the best current point (BCP) by x' if $F(x') < F(BCP)$. Set $MSIN = 0$ and move to Step 3. Otherwise, let $N^*(x) = N^*(x) \cup \{z\}$.

Repeat Step 1, until $|N^*(x)| = NV$. Move to Step 2.

Step 2. Determine $x' = \underset{z \in N^*(x)}{\text{Arg min}} \{F(z)\}$. Set $MSIN = MSIN + 1$.

Step 3. Set $x = x'$ and update the Tabu list by including $[i, R(x, i)]$ into TL .

If $CNTL > LTL$ then eliminate the oldest element in Tabu list.

Set $N^*(x) = \emptyset$.

Step 4. Set $CI = CI + 1$.

If $CI < MI$ and $CCT < MCT$ and $CSIN < MSIN$ then repeat step 1. Otherwise, stop the procedure.

4. Numerical Results and Comparison

For practical implementation, the data input is to be built from the Course Information (CI) sheets and the Preferences Table (PT) for the course times. A CI sheet normally consists of the name course, the lasting time, the classroom type needed for the course, and the name of the lecturer who handles the course. The PT presents the preferability of the lecturer (of a given course) for every time period (in a week). From the CI sheets, the courses are encoded (enumerated) in the range $1, 2, \dots, n$, and then the set J_{ijk} may be defined. In fact, in order to solve (P2), there is no need for computing J_{ijk} but it is necessary to compute $c2_{ijkl}$, instead. This is carried out by counting the overlapping time between the lectures i and k if they are assigned to the time periods j and l , respectively. The constant $c1_{ij}$ of assigning lecture i to the time period j is defined from the Preferences Table for the lecture i .

4.1. Numerical Results for a Case of Practical Data

Up to now, the timetabling procedure in Vietnam hardly ever takes care of preferences of teachers handling the courses, and the scheduling problems are solved somehow without optimality criterion. Taking account of the teachers' preferences, the existing Timetables in most universities may have very high cost. Our aim now is to apply the methods described in the previous parts to get a better (cheaper) schedule in comparison with the existing ones.

Here we consider the lectures scheduling problem with real data of the Vietnam Commercial University (VCU), where the number of lectures is $n = 300$, each of which has the same length and is to be associated with a time period of 2 teaching hours. Thus, every workday has 4 time periods and the number of time periods in a week is 20. Furthermore, the schedule in VCU is generated separately for morning sessions and afternoon sessions, so that the number of

time periods to be scheduled for one week is, in fact, only $m = 10$. The table below shows the results of Exchange Procedure (EP) and Tabu Search (TS) techniques for the lectures scheduling problem (for the fifth semester of K35 at VCU), where the teachers preferences are given at random and the starting Timetable is the existing one (4 numerical tests are carried out for 4 different random teachers preferences data, where possible values are: 0 (good), 1 (normal) and 5 (impossible)).

Test N°	Ini.Val.of Obj.Func.	Method used	Fin.Val of Obj.Func.	Falling rate	Comp.Time (sec.)
1	9903722	EP	2701250	72.7249%	1
		TS	1800706	81.8174%	12
2	9903864	EP	3601750	63.6333%	1
		TS	1800992	81.8152%	11
3	9903709	EP	2701260	72.7247%	1
		TS	1800735	81.8175%	11
4	9903757	EP	2701312	72.7243%	1
		TS	1800771	81.8172%	11

Here, the Tabu Search method used the Tabu list of the length 1000 and the search neighborhood of the size 1000.

The results show that the EP method is faster, but the result of the TS method is better. Further comparison results will be given in Subsec. 4.3. Clearly, for scheduling problems, the most important factor of efficiency is the reduction of the cost. Normally, the computation is carried out once a year, and the cost of the whole day computing is not comparable with one percent of the total scheduling cost.

The problem taken above is rather simple due to the following facts: the number of lectures is not very large (only 300), the number of time periods (in one week) is very small (only 10), and the most essential fact is that all lectures are of the same length. For demonstrating the ability of the algorithm in solving problems of more general setting, we have carried out the some numerical tests for larger numbers of lectures with different lengths.

4.2. Numerical Results of Tabu Search Method for Problems of Random Data

The implementation is carried for problem (P2), where the number of time periods is 50 (in one week), the number of lectures is in the range from 300 to 1000, and lectures are of different lengths (in the range from 1 to 4 teaching hours). With each set of lectures, the test is implemented for 4 different collections of *parameters* (*length* of Tabu list, *size* of neighborhood and the *number* of iterations), on the Pentium III 750MHz CPU.

In the tables below, NL stands for number of lectures, IV - the initial value of the objective function, TL - the Tabu list length, NS - the neighborhood size,

FV - the final value of the objective function, FR - the falling rate, NI - the number of iterations, CT - the computing time (in second).

Table 1 gives the results for problems of middle sizes, where the number of lectures is in the range from 300 to 500.

Table 1

NL	IV	TL	NS	FV	FR	NI	CT
300	986731	1000	2000	144020	84.71%	2000	19
300	986731	2000	3000	138015	85.35%	3000	146
300	986731	2500	4000	129505	86.25%	4000	343
300	986731	4500	6000	121040	87.15%	6000	1659
400	1737767	1000	2000	424416	76.30%	2000	8
400	1737767	2000	3000	366515	79.53%	3000	113
400	1737767	2500	4000	331781	81.47%	4000	274
400	1737767	4500	6000	330035	81.57%	6000	1515
500	2870004	1000	2000	1057931	62.16%	2000	8
500	2870004	2000	3000	710394	74.59%	3000	74
500	2870004	2500	4000	703334	74.84%	4000	227
500	2870004	4500	6000	672270	75.95%	6000	1348

Table 2 shows the numerical results of Tabu Search Method for the problems of larger sizes, where the number of lectures is in the range from 600 to 1000.

Table 2

NL	IV	TL	NS	FV	FR	CT
600	3821207	300	1000	990208	74.086%	12000
600	3999762	400	1000	1050429	73.737%	12000
600	4107102	400	1000	1022226	75.110%	12000
600	3821207	300	1000	985177	74.128%	24000
800	7134969	500	1000	2464496	65.451%	12000
800	7188352	400	1000	2515549	65.000%	12000
800	7186582	300	1000	2452112	65.879%	12000
800	7134969	500	1000	2391540	66.481%	24000
1000	11046325	150	300	4491078	59.343%	12000
1000	10851674	1000	2000	4374209	59.690%	12000
1000	10669750	500	1000	4284257	59.846%	12000
1000	10669750	500	1000	4231167	60.344%	24000

4.3. Comparison of the Exchange Procedure and Tabu Search Methods

As mentioned earlier, the Exchange Procedure has been implemented for problems of middle sizes only. The comparison is thus available for problems with the number of lectures in the range from 300 to 500.

The computation shows that, at the beginning, the results of the Exchanged Procedure (EP) method look quite similar to that of the Tabu Search (TS) method. However, after a certain time, the result of EP method remains unchanged while the TS method essentially improves the solution when the computing time increases. This is illustrated on the following examples, which show the results of the Exchanged Procedure Method and the Tabu Search Method implemented with LTL=1000, NV=1000, MI= 10⁶, MSIN=5000.

Test 1. For the case of 300 lectures, 50 time periods (the data is taken at random and the initial value of objective function is 1016767).

COMPUTING TIME in sec.	RESULTS OF THE EXCH. PROCED. METHOD		RESULTS OF THE TABU SEARCH METHOD	
	Value of OBJ. FUNC.	Falling Rate	Value of OBJ. FUNC.	Falling Rate
60	144028	85.83%	114483	88.74%
120	144028	85.83%	87958	91.34%
300	144028	85.83%	87958	91.34%
600	144028	85.83%	76057	92.51%
900	144028	85.83%	76057	92.51%
1200	144028	85.83%	76057	92.51%
1800	144028	85.83%	76051	92.51%
2400	144028	85.83%	76051	92.51%
3000	144028	85.83%	76051	92.51%
3600	144028	85.83%	76051	92.51%

Test 2. For the case of 400 lectures, 50 time periods (the data is taken at random and the initial value of objective function is 1730654).

COMPUTING TIME in sec.	RESULTS OF THE EXCH. PROCED. METHOD		RESULTS OF THE TABU SEARCH METHOD	
	Value of OBJ. FUNC.	Falling Rate	Value of OBJ. FUNC.	Falling Rate
60	381839	77.93%	318770	81.58%
120	381839	77.93%	287755	83.37%
300	381839	77.93%	272851	84.23%
600	381839	77.93%	247355	85.707%

COMPUTING TIME in sec.	RESULTS OF THE EXCH. PROCED. METHOD		RESULTS OF THE TABU SEARCH METHOD	
	Value of OBJ. FUNC.	Falling Rate	Value of OBJ. FUNC.	Falling Rate
900	381839	77.93%	247355	85.707%
1200	381839	77.93%	246190	85.77%
1800	381839	77.93%	228293	86.808%
2400	381839	77.93%	228293	86.808%
3000	381839	77.93%	228293	86.808%
3600	381839	77.93%	228293	86.808%

Test 3. For the case of 500 lectures, 50 time periods (the data is taken at random and the initial value of objective function is 2726620).

COMPUTING TIME in sec.	RESULTS OF THE EXCH. PROCED. METHOD		RESULTS OF THE TABU SEARCH METHOD	
	Value of OBJ. FUNC.	Falling Rate	Value of OBJ. FUNC.	Falling Rate
60	756692	72.24%	725838	73.37%
120	756692	72.24%	669321	75.45%
300	756692	72.24%	657888	75.87%
600	756692	72.24%	625814	77.047%
900	756692	72.24%	625814	77.047%
1200	756692	72.24%	625814	77.047%
1800	756692	72.24%	625814	77.047%
2400	756692	72.24%	597242	78.095%
3000	756692	72.24%	597242	78.095%
3600	756692	72.24%	589752	78.37%

5. Conclusions and Acknowledgement

The results above show that the Tabu Search method is suitable for solving the lectures scheduling problems of practical size. For problems of middle-size, it may produce results of quality similar to the ones of the Exchange Procedure method and, when the computing time increases, it can essentially improve the solution while the latter cannot. Further, more importantly, the Tabu Search Method can be implemented for problems of large size and, therefore, it can solve the lectures scheduling problem for a university in the whole and needs no trick of decomposition that may lead to a "partially good" solution only.

The authors are grateful to Prof. Ferland, Dr. Pham Canh Duong for helpful suggestions and wish to express deep gratitude to Pham Ngoc Hung and Nguyen Quang Minh for handling various numerical tests of the algorithms.

References

1. Jean Aubin and Jacques A. Ferland, A large scale timetabling problem, *Comput. Ops Res.* **16** (1989) 67–77.
2. J. A. Ferland, A. Hertz, and A. Lavoie, Object-Oriented methodology for Solving assignment type problems with neighborhood search techniques, *Operations Research* **44** March-April, 1996.
3. J. A. Ferland, *Generalized Assignment-Type Problems A Powerful Modeling Scheme*. Publication No.1069, Département d'informatique et de recherche opérationnelle Université de Montréal, 1997.
4. J. A. Ferland, Soumia Ichoua, Alain Lavoie, and Eric Gagne, *Scheduling Medical Students Internships Using Tabu Search Methods with Intensification and Diversification*, Puplication No.1068. Département d'informatique et de recherche opérationnelle Université de Montréal, 1998.
5. J. A. Ferland and Serge Roy, Timetabling Problem for University As Assignment of Activities To Resources, *Comput. Ops. Res.* **12** (1985) 207–218.
6. J. A. Ferland, Ilham Berada, Imene Nabli, B Ahiod, Philippe Michelon, and Viviane Gascon, *Generalized Assignment Type Goal Programming Problem*, Application to Nurse Scheduling, Puplication No.1112, Mars 1998.
7. S. Lavoie, *Probleme de confection d'horaire de cours au niveau d'un GECEP*, M. Sc. report, Université de Montréal, Canada, 1985.