A Heuristic-Based Technique for University Resource Allocation Problems

Dr. Mohamed Tounsi

Computer Science Department, Prince Sultan University, P.O. Box 66833, Riyadh, 11586, KSA,
mtounsi@cis.psu.edu.sa

ABSTRACT —

Most academic institutions face the problem of scheduling both courses and examinations in every year. As the difficulty of the problem increases, due to a large number of students, courses, exams, rooms and invigilator constraints, an automated resource allocation system that can produce feasible and high quality timetables is often required. To provide a good university planning, a fast and efficient solver is required. Various techniques are proposed to solve the timetabling problem, since course planning is a combinatorial optimization problem, we apply a heuristic based approach to solve it. In this paper we propose a solver based on using an efficient heuristic for planning: tabu search. We show how all hard and soft constraints are taken into account to solve some real life benchmarks. We conclude the paper by presenting some issues to improve the proposed algorithm and discussing their the possibility for hybridizing with other powerful heuristics.

***Index Terms* — Local Search, Heuristic, Tabu Search, Resource allocation Problems.**

I. INTRODUCTION

Course planning is defined here as a process of assigning courses to timeslots and instructors, which satisfy all constraints. The basic challenge is to schedule courses over a limited timeslots so as to avoid conflicts and to satisfy a number of side constraints. It belongs to timetabling problems.

There are two main problems in timetabling [2]. The first one is related to the combinatorial nature of the problems, where is difficult to find an optimal solution because it is impossible to enumerate all nodes in such a large search space. The second one is related to the dynamic nature of the problems where variables and constraints are changing in accordance with the development of an organization. Various techniques have been proposed to solve timetabling problem, such as Linear programming [10], graph coloring [2], genetic algorithm, knowledge-based reasoning [4] etc.

This paper is divided in five sections:

First, we introduce the Tabu search method and list our requirements, formulate and analyze the model.

We the present Tabu design, including the design of neighborhood structure, move mechanism, initial solution generation, evaluation function, and termination criterion. The next section introduce the planning system, present experiment result and result analysis, finally we present a conclusion and present some future works

II. TABU SEARCH

Tabu Search is a general search procedure devised for finding a (hopefully) global minimum of a function f, defined on a feasible set X. For each solution s in X, we define a neighborhood N(s) which consists of all feasible solutions that can be obtained by applying to s a simple type of modification m.

The procedure starts from an initial feasible solution and tries to reach a global optimum of the problem by moving step by step. Whenever a feasible solution s has been reached, we generate a subset V* of N(s) and we move to the best solution s* in V*. If N(s) is not too large, it is possible to take V*=N(s). The use of the best move criterion in TS is based on the supposition that moves with higher evaluations have higher probability of leading to an optimal (or near optimal) solution or of leading to such a solution in a fewer number of steps. The notation $s^*=s \oplus m$ means that s* is obtained from s by applying modification m. In order to be able to escape from local minima, the move to s* is made even if s* is worse than s(i.e. $f(s^*)>f(s)$). This strategy may clearly induce cycling of the algorithm. In order to prevent this, we introduce a Tabu list T. This list contains the modifications which

were made in the last |T| steps of the algorithm. When constructing V*, we forbid the generation of solutions that are obtained from s by applying the reverse of the modification memorized in the Tabu list. This principle reduces the risk of cycling since it guarantees us not to return for a given number of iterations to a solution visited previously. Unfortunately, it may also forbid us to move to some solutions which have not been reached yet. Deciding that a modification m is , at a given step, Tabu or not may be too absolute. In order to have a higher degree of freedom in generating the subset V*, it should be possible to forget the Tabu status of a modification when it seems reasonable to do this. This is why we introduce for every 'tabu' for every possible value z of the objective function an aspiration level A(z): a solution s' in N(s) which would be ''Tabu' because of list T can nevertheless be taken into account if f(s')<A(f(s)). The function A is called the aspiration function. A simple example of application of this idea is obtained by setting $A(f(s))=f(s^o)$ where s^o is the best solution encountered so far. In this case, we accept a Tabu modification only if it leads to a neighbor solution better than s^o. This criterion can on no account introduce an additional possibility of cycling. For more details please refer [1].

Two rules can be defined in order to interrupt the whole TS process. The first is to stop as soon as nimax iterations have been performed without decreasing the value of the best solution obtained. We may also interrupt the procedure when the value of the current solution is close enough to the minimum value f_{min} of f, which is known in certain problems [5]. This second rule avoids us executing nimax additional iterations after having reached an optimal solution or a solution judged sufficiently good. A general description of the TS procedure is given in Figure 1.

III. COURSE PLANNING REQUIREMENT

- Less than 40% classes before 12:00 noon or from 12:00 ~ 5:00 pm, and
- More than 20% classes after 3:00 pm.
- Schedule no more that one undergraduate course and one graduate course in each allowable time slot, so that a student can take any two courses offered by the department.
- Any two courses taught by the same instructor cannot be scheduled in the same time slot.
- Schedule all the courses taught by any instructor on the same days (e.g., MW, TR).
- Satisfy all faculty preferences (e.g., no classes before 9:00 am, or after 5:00 pm).
- Schedule all junior (senior) undergraduate classes on the same days; such that students don't need to come to school on everyday if choose so.
- Schedule all classes using university standard time slots.
- Whenever possible, schedule all FEEDS classes to start before 8:00 am, at 12:00 noon, or after 3:00 pm.

a. Course planning case formulation

There are 4 course classifications:
Undergraduate elementary courses, Junior and senior courses, Graduate courses and Partial Graduate courses.

C(k) ={course index | i∈ course class k}

There are 4 course types:
Two Hour Courses (one section), Two Hour Courses (Two section), Three Hour Courses (Two section) and Evening Courses. Each type has its own standard time slots community, which is specified by the university. For example, if a course belongs to course type 4, then this course can only be set into one of the 12 time slots.
So these 12 time slots constitute a community of course type 4. Once a course type is fixed, then the time slots choices are fixed.
Which type a course belongs to is preset before planning?

We define

T(i)={timeslot index j |j∈timeslot ∈ coursetype}

A. The relationship between the professors and courses is preset. We define

P(r) ={course i| i is taught by professor r}

B. Let

$$x_{ij} = \begin{cases} 1, & course\ i\ is\ set\ to\ time\ slot\ j \\ 0, & otherwise \end{cases}$$

Note that for course i, if $coursetype(i) = l$, then $j \in T(l)$, we don't need to set j to all the time slots.

C. For different course type time slots, there is overlap among timeslots.

$O(j) = \{time\ \ slot\ \ index\ \ j' | j'\ \ overlap\ \ with\ \ j\}$

D. Some faculty has an own time constraints, (e.g., no classes before 9:00 am, or after 5:00 pm). We define a constraint as:

$F(r) = \{timeslot index\ j | j\ \ is\ \ not\ \ prefered\ by\ prefosson\}$

E. Whenever possible, schedule all FEEDS classes in specified time period (e.g. start before 8:00 am, at 12:00 noon, or after 3:00 pm.). This implies a constraint:

$D = \{course\ \ index\ \ i | i\ \ is\ \ FEED\ \ course\}$

$FTIME = \{timeslot\ j\ |\ j\ \ within\ FEED\ course time prefered\}$

i. Constraints:

We can divide the constraints into strong constraints and weak constraints. Then our schedule must satisfy strong constraints and try to satisfy weak constraints as possible as we can.

Strong constraints:

1) The same classification courses can not be arranged at the same time.

$$\sum_{i \in C(k)} x_{ij} \le 1, \forall k, j \qquad \text{(Eq. 1)}$$

2 The course taught by the same instructor cannot be arranged at the same time (Notice overlap timeslots).

$$\sum_{j+O(j)} \sum_{i \in P(r)} x_{ij} = 1, \forall r, j \qquad \text{(Eq. 2)}$$

Weak constraints:

1) Schedule all the courses taught by any instructor on the same days (e.g., MW, TR).

Let N(r) denotes the number of weekdays professor *r* has classes, $N(r) \le 5$.

Let F_1 denote the summation of total number of all the professors. $F_1 = \sum_{allr} N(r)$ (Eq. 3)

2) Whenever possible, schedule all FEEDS classes to start before 8:00 am, at 12:00 noon, or after 3:00 pm. We define

$$C'_{ij} = \begin{cases} 1, & if \quad i \in FEED, j \notin FTIME \\ 0, & otherwise \end{cases}$$

Let F_2 denote the total number of feed courses which are not satisfied with FEED course time requirement.

$$F_2 = \sum_{alli} \sum_{allj} C'_{ij}\ x_{ij} \qquad \text{(Eq. 4)}$$

3) Less than 40% classes before 12:00 noon or from 12:00 ~ 5:00 pm, and ≥ 20% classes after 3:00 pm.

Let *t1* denote the time period before 12:00 noon

Let *t2* denote the time period from 12:00 ~ 5:00 pm.

Let *t3* denote the time period after 3:00 pm.

Then $N(t1)$ denotes the total number of courses in *t1* period.

$N(t2)$ Denotes the total number of courses in *t2* period.

$N(t3)$ Denotes the total number of courses in *t3* period.

Let F_3 denote the satisfiability for this constraint.

$F_3 = Max(0,(N(t1)/n - 40\%)) + Max(0,(N(t2)/n - 40\%))$

$+ Max(0, 20\% -$

4) Satisfy all faculty preferences (e.g., no classes before 9:00 am, or after 5:00 pm).

Define

$$C''_{ij} = \begin{cases} 1, & if \quad i \in p(r), j \notin F(r) \\ 0, & otherwise \end{cases}$$

Let F_4 denote the total number of courses, which do not satisfy professor preference.

Then $F_4 = \sum_{alli} \sum_{allj} C''_{ij}\ x_{ij}$

For all these weak constraints, we can evaluate their satisfied degree. F_1, F_2, F_3, F_4 are used to evaluate these constraints. Then our objective is to minimize these values as possible as we can.

IV. THE TABU SEARCH ALGORITHM

1) Evaluation Function

The objective is to minimize the cost of violating the soft constraints.

Min $F = w_1F_1 + w_2F_2 + w_3F_3 + w_4F_4$

F_1, F_2, F_3, F_4 Have been defined above. W_1, W_2, W_3, W_4 Are the weights for each constraint. We determine the weight for each constraint based on the importance of the constraint.

2) Move

We define two moving mechanisms: *Transfer* and *Interchange*.

Transfer: $x_{ij} -> x_{ik}$ change course from one time slot to another time slot.

Precondition:

$x_{ij} = 1, x_{ik} = 0, j, k \in T(l), \quad \sim \forall i', i, i' \in C(k)$

If we transfer course i from timeslot j to timeslot k,j,k should belong to the same course type time slots, and in time slot k, there is no other courses with the same classification with i).

Effect: $x_{ij} = 0, x_{ik} = 1$.

Interchange: $x_{ij} \leftrightarrow x_{i'j'}$ interchange two courses time slot.

Precondition:

$x_{ij} = 1, x_{i'j'} = 1, j, j' \in T(l), \quad i, i' \in C(k)$

Interchange course i and i' timeslots, if i and i' are in the same classification, their timeslots j and j' belong to the same course type.

Effect: $x_{ij'} = 1, x_{ij} = 0, x_{i'j} = 1, x_{i'j'} = 0$

After one move, the new solution should satisfy the above two strong constraints. So before performing a move, first check its precondition, then check its feasibility of constraint (b), after a move, update information based on effect. (Constraint (a) will be satisfied automatically, due to the precondition definition)

3) Neighborhood structure

There are 4 time slot communities since there are 4 course types, T(1), T(2), T(3), T(4). Because the course type is fixed, the set of neighborhood can be reached from a single move in a single community, which means that we only search the timeslots with the same course type instead of all the timeslots.

We search course assignments based on course sequence. During one iteration, we are going to change every course's timeslot, which the number of solutions during one iteration equal to the number of courses. But there are multiple possible moves for one course, which equal to the number of total time slots of that course type. Our design is not to search the whole neighbor but randomly pick up a time slot in the same course type. If it is occupied by other course in the same classification, then performing *interchange*, otherwise performing *transfer*. So the neighborhood size is reduced to the number of courses.

4) Initial solution

The courses and time slots are indexed sequentially, so just put the first course into first feasible time slot, then the second … at the same time, should check strong constraints feasibility.

Using this method to get initial solution, there is possibility that some courses cannot get feasible timeslots. But the feasible solution is much more important than other constraints. In this sense, we can issue strong punishment for infeasible solutions

Define n as the number of courses without feasible timeslots. During an iteration, if n>=0, we set new F'=n*10*F.

So from above, we can see that to reduce the number of courses without feasible timeslots will significantly reduce the penalty, which will definitely be accept during neighborhood search iteration.

5) Termination ***criteria***

Define two ways to terminate searching:

a) Reaching the maximum repeating time
b) The time stuck at the same solution exceed the limit.

6) Tabu ***list***

Tabu list composes of two attributes of a move, one is the timeslot ID, another is classification ID.

About the Tabu tenure, the length should be shorter than the number of timeslots of any course type. Otherwise it will happen that during next iteration, all possible moves are in Tabu list.

IV.I. SEARCH PROCEDURE

The Local Search Procedure:

Local-search(T)

1 Course index i = 1 ;

2 Randomly pick up a time slot index j from *T(l)*, for i belongs to course type l, $j \in T(l)$;

3 Check Tabu list, if this move is in Tabu list, return to 2.

4 Precondition check, if j is occupied by the course in the same classification, go to step 5 perform *interchange* move, otherwise go to 6 perform *transfer* move.

5 Perform *interchange* move, check whether this move is feasible for constraint (b), if not go to step 2, else go to step 7;

6 Perform *transfer* move, check whether this move is feasible for constraint (b), if not go to step 2, else go to step 7;

7 Calculate its evaluation value F, if the number of courses without feasible solution *n* is larger than zero, F'=F*10**n*.

8 i++; if i > Number-of-Courses then go to 9,else go to 2;

9 Return best move with the least penalty value F.

The Main Procedure:

Tabu-search(T)

1 Generate initial solution T, LocalT = T, BestT = T;

2 Repeat time rT =1; Stuck_time sT = 1;

3 T = Local-search(T);

4 Update tabu list and other information;

5 LocalT =T;

6 If BestT>T, then BestT =T;

7 RT++; if T remains the same, then sT++;

8 If rT> max_flip or sT > stuckLimit, go to 9,else 3;

9 Return BestT.

Figure 1: The Tabu Search Algorithm

V. EXPERIMENT RESULT

The course planning system is composed of three parts: the database and knowledge base module, it contains timeslots, courses, instructors' information, including 5 arrays: course, course classification, course type, instructor, and timeslot. The Data management module, this module is coded in C language, in charge of managing data from database. It retrieves data from database, and outputs result into database and the schedule generation module, this module is coded in C language, it performs Tabu search to generate a good planning.

Experiment data

We used benchmarks from the Timetabling Research Group and ASAP research Group; these data are real life problems. We have 29 courses and 20 instructors, including 20 FEED classes. We build the professors' preference.

Table 1

The course distribution

Courses number	29	Time slots number for type 1	63
Instructors number	20	Time slots number for type 2	20
Classification	4	Time slots number for type 3	12
Course type	4	Time slots number for type 4	12
Feed course number	11		

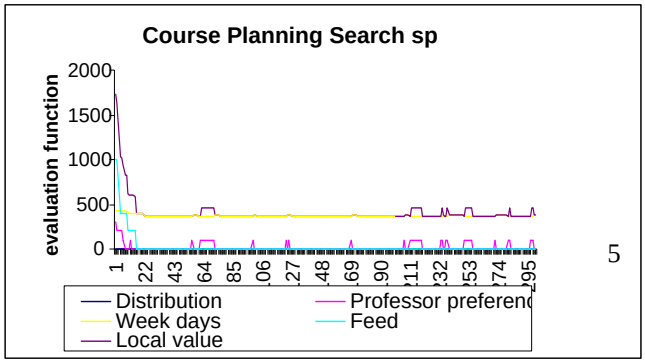
Table 2:

The Professor preferences

Professor ID	Class not before	Class not after
1	10:00	18:00
3	9:00	
4	9:00	16:00
18	10:00	

Numerical analysis

First, since the smallest timeslot number is 12, so I choose 8 as Tabu tenure length, and maximum iteration length is 400. To keep each weak constraint penalty balanced, we set weights W_i in a way so as to



keep $F_i (i = 1,2,3,4)$ in the same digital size. $W_1 = 10$ (for professors' week days), $W_2 = 200$ (for FEED courses), $W_3 = 2000$ (for course distribution), $W_4 = 100$ (for professor preference).

Fig.1
Course Planning Search Space

After numerous experiments, I found all results had no much difference and the best solution was pretty close. Each running time was around 4 seconds, which was very fast.

Compared with the initial solution, the final solution has 11 courses that change their timeslots. The final solution satisfied all the professors' preference. For example, in the initial solution, there are two courses of professor 1 scheduled before 10:00 am. After iterations, these two courses are changed to the timeslots after 10:00 am.

Distribution constraint is satisfied at the initial solution, and never be violated during iteration. There are five feed courses out of required time period at the initial solution, but finally all have been adjusted to the right timeslots. This algorithm also tries to schedule professor to work at the fewer workdays, such as professor 3.

We can also see the improvement from the below "Course planning search space" figure. The "evaluation" axis represents all F_i and local value F during each iteration. This figure indicates the performance trend of solutions. From the figure, we can see that during the first 24 iterations, All F value decrease quickly, and then enter a plateau. Though Tabu list can prevent a cycling, but we can still see peaks. So there is still cycling in Tabu search.

V. CONCLUSION

We have developed a meta heuristics technique approach for solving course planning problem, and propose many computational study.

Course planning is intractable and seen as the most difficult problem, since it is linked with the scheduling problems. If it is done manually, it is a time consuming and tedious job. This paper shows that it is possible to find good solutions in a short time using a meta-heuristics technique as Tabu search.

Because the sample problem size is not big and easy to solve, this paper does not consider any aspiration function and the diversification of Tabu search. But we can diversify our search by controlling the weights W_i, through weights; we can direct the search direction.

REFERENCES

[1] F.Glover, M.Laguna. *Tabu Search*, Kluwer Academic Publisher, 1997.

[2] Michael Pinedo, Xiuli Chao. *Operations Scheduling with Applications in Manufacturing and Service.* Irwin McGraw hall. 1999.

[3] Daniel Costa. Theory and Methodology: A Tabu Search algorithm for computing an operational timetable. European Journal of Operational Research, 76(1994), 98-110.

[4] SB Deris, S Omatu, H Ohta, PABD Samat, University timetabling by constraint-based reasoning: A case study. Journal of the Operation Research Society (1997) 48,1178-1190.

[5] Edward L. Mooney, Ronald L. Rardin, W.J. Parmenter. Large-scale classroom scheduling. IIE Transactions (1996), 28, 369-378.

[6] T Birbas, S Daskalaki, E Housos. Timetabling for Greek high schools. Journal of the Operation Research Society (1997) 48,1191-1200.

[7] Buyang Cao, Fred Glover. Tabu Search and Ejection Chains----Application to a Node Weighted Version of the Cardinality-Constrained TSP. Management Science. Vol 43, No 7, July 1997. 908-921.

[8] C. Privault. Solving a Real World Assignment Problem with a Metaheuristic. Journal of Heuristics, 4, 383-398(1998).

[8] Michael W. Carter, A lagrangian relaxation approach to the classroom assignment problem. Infor vol. 27, no 2, May 1989

[9] Karl Gosselin, Michel Truchon. Allocation of Classrooms by Linear Programming. Journal of the Operational Research