STUDIA UNIV BABEŞ-BOLYAI, MATHEMATICA, XXXVIII, 3, 1993

A MATHEMATICAL MODEL TO SOLVE THE TIMETABLE PROBLEM USING PROLOG

Dragoş POP*

Dedicated to Professor Emil Muntean on his 60th anniversary

Received January 12, 1994
AMS subject classification 68Q40

REZUMAT. - Un model matematic pentru rezolvarea problemei orarului utilizînd limbajul Prolog. Lucrarea prezintă un model matematic general al problemei orarului precum şi utilizarea acestuia de către un algoritm de rezolvare a problemei Modelul matematic propus asigură descrierea unor restricţii extrem de diverse ce caracterizează soluţiile fiabile Algoritmul propus asigură găsirea soluţiei optime din punct de vedere al mai multor criterii, construind numai soluţiile fiabile susceptibile de a fi soluţii optime

1. Introduction. Timetable problems are by their fundamental nature resource allocation problems, whose solutions represent activity plans Every activity (also called 'meet') needs certain available resources (persons, development places, time, etc) and different conditions for development, depending on the activity itself or other activities (avoiding certain times, activities sequences and certain parallel activity pairs)

These problems depend on the educational systems and they can be mutually very different Some practical requirements cannot be easily caught in mathematical formulas Therefore it is very difficult to model and solve these problems For this reason a model should be universal and very flexible

Timetable problems are known to be **NP** - complete, only some reduced problems are

* *"Babeş-Bolyai" University, Faculty of Mathematics and Computer Science, 3400 Cluj-Napoca, Romania*

polynomial In our situation, such a special case requires the permanent availability of lecturers and rooms

In this paper we are concentrating on timetables for university faculty and propose a PROLOG rule system which can be run on a microcomputer The mathematical model is extended such that a priori fixed assignements, faculty connections and other miscellaneous constraints are supported

2. Problem formulation Let's assume given the set of faculty teaching staff, the set of student groups (called classes), the set of available rooms and their sizes and the set of activities that have to be scheduled in a certain week

Furthermore, a maximal allowed number of time periods (called hours) is assigned to each day The lunch breaks the day into two daily ammounts of consecutive hours (called daily quantums) with a given extent An activity should not be interrupted by this break

Every professor and class should have a set of unavailable hours due to a priori scheduled activities or other reasons

For the feasibility, the timetable have to satisfy certain requirements The set of all the requirements of the problem is partitioned into three groups, corresponding to the various degrees of strictness ([1])

- hard requirements, which must always be satisfied
- medium requirements, which should be satisfied although they can be relaxed in some cases
- soft requirements, which should be satisfied if the other requirements allow this

The hard requirements should lead to physically feasible timetable and they are expressed as constraints In our case, conflicts due to activities taking place simultaneously but involving classes or professors in common, have to be avoided Other constraints arise from the fact that the activities should not be interrupted and they should not be scheduled in an inadequate room or at hours which are unavailable for one of its participants Also, all the activities have to be scheduled during the time span of one week

The soft requirements deal with preferences and they are modelled as objectives Our objectives are that the timetables for classes and professors are compact, with no time windows, with as many as possible morning courses and other pedagogical recommendations

Every medium requirement can appear either as a constraint or as an objective, depending on the nature or the interpretation of the problem As a medium requirement, two courses of the same topic (called equivalent courses) should not be scheduled in the same day

The objectives are the actual parameters for a given quality function which reflects the preference for one timetable solution over the other It is often difficult to quantify the desirability aspect of timetabling However, a weighted sum of the objectives is a satisfactory, flexible and simple solution and for these reasons we chose it

On these conditions, the problem to solve is to find the feasible optimal solutions for the timetable If there results several optimal solutions, then a decision maker will choose the preferred solution

3. The mathematical model In this section it will be presented a mathematical model for the timetable problem based on the situation in our University, but the model is flexible

enough to allow application to many different situations

We consider the following sets

G - the set of classes

D - the set of teaching staff

P=GUD - the set of participants in teaching activities

S - the set of available rooms

M - the set of teaching activities (courses, seminars, practical training, etc)

H - the set of available hours per week For example, if we have 10 available hours per day, we consider that Monday is represented by the {1 10} hours, Tuesday is represented by the {11 20} hours and so on This convention can be changed in order to satisfy certain interests

and the constants

hdpɛN* - the number of hours per day

mhɛN* - the number of hours of the first daily quantum

(morning hours)

To solve in a smart way the problems of the lunch pauses and the end of a day, we add an hour between the two daily quantums and an hour after each day Then we will mark all classes and professors as unavailable at these fictitious hours Therefore, we avoid the interruption of the activity

Now we can define the time window as the free time period between two occupied time periods, which does not contain fictitious hours

For an easy handle, the rooms and the activities are coded with positive integer numbers Throughout this paper, the activity lists are ordered ascendingly on the codes and the room lists are ordered ascendingly on the room capacity

The following functions give us information about these sets

u $G \to \mathbf{N}^*$ $u(g)$ = the number of students in the group g

v $P \to P(H)$ $v(p)$ = the set of hours when p is available

c $M \to P(S)$ $c(s)$ = the set of suitable rooms for the activity m

p $M \to P(P)$ $p(m)$ = the set of participants in the activity m

r $M \to \mathbf{N}^*$ $r(m)$ = the extent of the activity m

From this information we can determine the function

a $M \to P(H)$ $a(m)$ = the set of hours when the activity m could begin

This function can be calculated using the following formula

$a(m) = \{y \in V \mid y, \ldots, y+r(m)-1 \in V\}$ where

$V = \bigcap v(x)$ represents the hours when all the $x \in p(m)$ participants at the activity m are free

In these conditions we consider a timetable solution as a function $t: M \to H \times S$, $t=(t_1,t_2)$ with the following properties

i) $\{t_1(m), \ldots, t_1(m)+r(m)-1\} \subseteq a(m) \quad \forall m \in M$

ii) $t_2(m) \in c(m) \quad \forall m \in M$

iii) $m \neq n$ and $\{t_1(m), \ldots, t_1(m)+r(m)-1\} \cap \{t_1(n), \ldots, t_1(n)+r(n)-1\} \neq \emptyset \Rightarrow p(m) \cap p(n) = \emptyset$ and $t_2(m) \neq t_2(n) \quad \forall m,n \in M$

(The simultaneous activities must have different participants and must be

scheduled in different rooms)

iv) $m \neq n$ and $p(m)=p(n) \Rightarrow [t_1(m)/hdp] \neq [t_1(n)/hdp]$

(The equivalent activities should not be scheduled in the same day)

Due to the relations between the activities, their different lengths and the problems concerning the rooms which may appear in the general case, the set of activities which could be scheduled at a certain hour depends on the activities scheduled before Therefore we are forced to construct every possible solution, hour by hour, in an heuristic way and then to evaluate its quality Note that there exist papers on the optimal timetable construction in a deterministic way using Operations Research, but they solve the problem only in particular cases ([4])

We understand now that, in the general case, since the timetable problem is **NP**-complete the number of solutions we have to construct is extremely high and it has an exponential growth in the number of participants ([2])

As an interesting particular situation, if all the activities have the same lengths and there are no problems with the rooms, we can consider the graph with the vertex set $V = M \cup H$ and with edges among every different hours, among the activities which have common participants and among the activities and their unsuitable hours In this case, the timetable construction problem is equivalent to the graph colouring problem, with $|H|$ colours

Also, we are very interested in the reduction of the number of constructed solutions There are two posibilities

The first is almost obvious If we evaluate the solution quality during the construction of the solution, we can check at certain moments (at the end of a day, for example) if we are

able to reach a quality which has already been obtained If we don't, it is useless to continue the construction of that solution

With the second method we obtain an extremely important reduction of the number of constructed solutions, with a very low risk of loosing high quality solutions Let's suppose we are constructing the hour h of the timetable and at this hour we can schedule activities from the A set Actually, due to restrictions, we can schedule only subsets of A Let E_h be the set of all these subsets, $E_h \subseteq P(A)$ The idea is to construct only the timetable solutions which have scheduled at the hour h the maximal elements of $(E_h, \subseteq)$

For example, let $\{m_1, \ldots, m_k\}$ be a maximal element of E_h (the activities $m_1, \ldots, m_k$ may be scheduled to begin together at the hour h) A full heuristic algorithm will try to construct solutions with all the following activity sets

$$\emptyset, \{m_1\}, \ldots, \{m_k\}, \{m_1, m_2\}, \ldots, \{m_1, \ldots, m_k\}$$

scheduled to begin at the hour h This means 2^k alternatives

But scheduling only a subset of $\{m_1, \ldots, m_k\}$ implies that some professors and classes will have empty places in their timetable and this fact will decrease the quality of the solution Therefore we can try only with the maximal configuration and the risk to loose this way an optimal solution, is very low

This method also decreases the number of remaining activities and thus it reduces the complexity of the following stages

4. Using the PROLOG programming techniques. The structure of the program

We restrict the presentation and explanation of the program to those elements which

are indispensable for a global comprehension of its ideas and an insight of its structure

The information about faculty teachers, activities, rooms and classes are stored in an internal database

The first module is an initialisation module which creates the information database using the consult predicate to read a given text file containing the input data Then it creates a work database which contains the constraints deduced from the information database These constraints concern the activity pairs that could not be scheduled simultaneously due to theirs common participants, the hours that are inadequate for a certain activity due to the activity length and the suitable rooms for every activity

The target of the main module is the heuristic search of feasible timetable considering all the restrictions, including the one which deals with the size of the rooms and the auxiliary support The best solutions achieved up to the present are stored in a distinct database and their quality is compared with the quality of the latest constructed solution If their quality is equal then the new solution is attached to the database Else, if the quality of the new solution is better, then the solutions stored in the database are removed and the new solution is attached to the database Therefore, at the end of the construction process, we have only the best solutions If the result database contains several solutions, we can apply other constraints concerning pedagogical requirements such as rational distribution of courses and effort during the week or other preferences

The last module is a tool which allows an interactive refinement of the solution The user can choose the preferred solution, if there are several optimal solutions from the given criteria

point of view, and then he may introduce the a priori assignments This module also assists the user to change some assignments preserving the validity of the solution

At the user's choice, this module calls the output module which sends the results to the screen, printer or a given file Among the results there is a new database which represents the updated input database with the new situation of the participants ocupation This database can be useful for unexpected situations which may occur during the semester, and to connect faculties This is an important fact because in general, the professors from a certain faculty assure the majority of courses in the University, which are related to the faculty realm

From all the facts mentioned until now, it results that the algorithm that I propose is semi-heuristic and that it is based on the backtracking mechanism Since the timetable construction problem is suitable for the descriptive programming and the backtracking mechanism is an internal mechanism of the PROLOG language, it becomes clear that the programming effort was considerably reduced this way ([3])

We will present now the predicate for the timetable hours construction which ensures the optimization described above and the predicate for the timetable construction, in a PROLOG-like pseudocode language

constructhour(h,m,A,S,[y|L]) if

$\exists y \in A$, $\exists s_h \in S$ a suitable room for the activity y which is

free between the hours h and h+r(m),

!,

Select $y \in A$, y>m, for which exist suitable rooms where it should be scheduled to begin at the hour h and

let r_y be the most suitable from them,

A1 = A \ l(y),

S1 = S \ {r_y},

constructhour(h,y,A1,S1,L)

constructhour(_,_,_,_,[])

where mεM

A is the ascendingly ordered list of nonscheduled activities

l M -> P(M), l(m) represents the set of activities which have common participants with m,

$$l(m) = \{n \varepsilon M \mid p(m) \cap p(n) \neq \varnothing\}$$

ttconstruct(_,[],_,[],_,[]) if l

ttconstruct(h,A,R,Eq,Ocr,[L|Tt]) if h ≤ the number of hours per week,

Construct Act - the list of activities from A which might be scheduled to begin at the hour h,

Construct AvR - the list of rooms from R which are available at the hour h,

constructhour(h,0,Act,Avr,L),

Create the A1, R1, Eq1, Ocr1 lists as the A, R, Eq and Ocr updated lists,

h1 = h+1,

ttconstruct(h1,A1,R1 Eq1,Ocr1,Tt)

where, in addition to the above described lists

R is the list of available rooms at the hour h

Ocr is the list of occupied rooms at the hour h

(RUOcr is the constant set of all rooms)

Eq is the activity list which should not be scheduled in the present day due to the constraint concerning the equivalent activities

Tt is the constructed solution

Note that these lines must be understood with the PROLOG conventions, i e if an assertion fails, the program will automatically try to find other solutions for the previous assertions in the same clause or if this is impossible, with the following clauses of the predicate

This predicate respects the above considerations on the maximality At a certain moment, there is selected $F=\{m_1, \ , m_i\}$, $F \varepsilon E_h$, where $m_1 < m_2 < \ < m_i$

If $\exists x \varepsilon A \backslash F$, $x>m_i$ such that $FU\{x\} \varepsilon E_h$, then x will be attached to F and the searching process continues Otherwise there are two possible situations

1 If $\nexists x \varepsilon A \backslash F$ such that $FU\{x\} \varepsilon E_h$, then F is a maximal element of $(E_h, \subseteq)$ and the activities from F will be scheduled at the hour h In this case the first clause of the predicate fails and it will be used the second clause

2 If $\exists x \varepsilon A \backslash F$ such that $FU\{x\} \varepsilon E_h$ but $x<m_i$, then F is not a maximal element of E_h and due to the ascending order selection of the activities, $FU\{x\}$ was selected in a previous stage The predicate fails due to the cut backtracking (!) predicate call

5. Concluding remarks and possible extensions Now we can notice the tremendous advantage of considering a relation oriented approach using logical programming techniques The PROLOG programming language is a very fast and flexible prototyping tool Additional

restrictions and situations can be easily included as new predicates or by extension of the existing attribute list.

If this database is too large to fit in the memory, we can use the PROLOG facilities to work with external databases stored on disk

I mention here that I used Borland's Turbo Prolog 2 0. Certainly a PROLOG compiler does not generate very fast programs but Turbo Prolog allows interfacing PROLOG programs with modules written in C or even assembler language for the speed critical parts of the program. Also, knowing that menus and user interface could be programmed more efficient with traditional programming languages, we can use C libraries to do this

I also suggest two useful extensions for this model

- the introduction of the predecessor-succesor relation between the activities as new constraints.
- the introduction of dynamic topics, when we dispose only of the total amount of hours for a certain subject matter, so that we can choose their activity grouping

REFERENCES

1 Eiselt H A , Laporte G , Combinatorial optimization problems with soft and hard requirements, Journal of Operations Research Society 38, 1987

2 Even S , Ilai A , Shamir A , On the complexity of timetable and multicommodity flow problems, SIAM Journal of Computation 5, 1976

3 Hertz A , de Werra D , The Tabu search methaheuristic how we used it, Ann. Math. Artificial Intelligence 1, 1990

4 Mihoc G , Balas E , The problem of optimal timetables, Revue Roumaine de Mathematiques Pures et Appliquees 10, 1965

5 Schmidt G , Strohlein T , Timetable Construction - An annotated bibliography, Computer Journal 23, 1980