STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **XLVI**, Number 2, 2001

FLOW IN NETWORK MODELING TIME TABLING AND SCHEDULING PROBLEM

TEODOR TOADERE

Abstract. In this paper a new model for solving the Time Tabling and Scheduling problem by determining the maximal flow in a specified transportation network is proposed. This transportation network is also related to an assignment problem where n-uples must be determined instead of pairs, as with the classical assignment problem.

The network is based on a multipartite graph, whose set of vertices is formed by the source, the sink and subsets with elements to be placed in schedule. The subsets contain: the disciplines, the professors, the class of learners (students, pupils, ...), the classrooms and time duration of the activities.

1. Ford-Fulkerson algorithm

We suppose that the Ford-Fulkerson algorithm is well known but we will describe it in Pseudocode. For this purpose we will define the "ecarts" graph attached to a transportation network from the source to the sink, and to a compatible flow.

As it is known, the flow ϕ in the network $G = (X, U)$, is called compatible flow if $0 \leq \phi_u \leq c_u$, $\forall u \in U$.

Definition 1.1. Let $G = (X, U)$ be a graph. To each compatible flow $\phi = (\phi_1, \ldots, \phi_m)^t$ the "ecarts" graph $G(\phi) = (X, U(\phi))$ is attached. The set of edges (arrows) is built in the following mode: $\forall u = (i, j) \in G$, $u^+ = (i, j) \in U(\phi)$ if $\phi_u < c_u$ and $u^- = (j, i) \in U(\phi)$ if $\phi_u > 0$. The capacities for edges are $c_u - \phi_u$ for u^+ and ϕ_u for u^-.

The capacities for edges from $G(\phi)$ are called residuals capacities. The capacity (i.e. $\min \sum_{e \in W} c(e)$) for each path w from $G(\phi)$ is called the path residual capacity.

The following result based on Ford-Fulkerson algorithm is well known. The result may help us to find out a maximal flow in a transportation network.

Theorem 1.1. [1] Let $\bar{\phi}$ be a flow from the source s to sink t, in a transportation network. Let $G(\phi)$ be, also, the attached "ecarts" graph. Then, the flow $\bar{\phi}$ is maximal if and only if there is no path from source s to sink t, in $G(\phi)$.

Algorithm 1.1. Ford-Fulkerson algorithm. [2]:

2000 *Mathematics Subject Classification.* 05C38.

1998 *CR Categories and Descriptors.* G.2.2 [**Mathematics of Computing**] : Discrete Mathematics – *Graph Theory*; Path and Circuit Problems.

(a) Initializations;
k:=0;
Suppose ϕ^0 a compatible flow, e.g. $\phi^0 = (0, 0, \ldots, 0)$.
(b) Halt criterium:
If $\exists \mu^k$ a path from s to t in $G(\phi^k)$
Then goto (c)
Else stop; ϕ^k is optimal flow.
(c) Let ε be the residual capacity for μ^k;

$$\phi_u^{k+1} = \begin{cases} \phi_u + \varepsilon, & u^+ \in \mu^k \text{ or } u = 0 \\ \phi_u - \varepsilon, & u^- \in \mu^k, \text{ for } u \in U \\ \phi_u, & \text{otherwise} \end{cases}$$

k:=$k+1$;
goto (b).

For the existence of μ^k, a path in the "ecarts" graph $G(\phi^k)$ and for its determination (and also for determination of the value of ε) we may use two procedures [2]:

p_1) From $G(\phi^k)$ a tree is built such that the root is s and the other vertices are the vertices from $G(\phi^k)$. We may use a procedure to mark the vertices and then we may find out the residual capacity for a path form s.
p_2) Without building the "ecarts" graph $G(\phi^k)$:
Each $j \in X$ has a label (e_1, e_2, e_3) such that: $e_1 \in X \cup \{0\}$; $e_2 \in \{+, -\}$; $e_3 \in \mathbb{R}_+$, with the following meaning:

- if $e_2 = +$ and $e_1 = i$, then $\exists u = (i, j) \in G$ and $\exists \mu \in G(\phi^k)$, path from s to j and the last edge is $u^+ = (i, j)$, its capacity is e_3.
- if $e_2 = -$ and $e_1 = i$, then $\exists u = (j, i) \in G$ and $\exists \mu \in G(\phi^k)$, path from s to j and the last edge is $u^- = (i, j)$, its capacity is e_3.

The source s is labeled with $(0, +, maxint)$. The main iteration of this "labeled" procedure consists in:

repeat
k:=0;
for each labeled $i \in X$ do
if $\exists j \in X$ (not labeled) such that $u = (i, j) \in U$ and $\phi_u < c_u$,
then one assigns to j the label $(i, +, min\{e_3(i), c_u - \phi_u\})$; k:=1;
endif;
if $\exists j \in X$ (not labeled) such that $u = (j, i) \in U$ and $\phi_u > 0$,
then one assigns to j the label $(i, -, min\{e_3(i), \phi_u\})$; k:=1;
endif;
endfor;
until (t is labeled) or (k=0);

If t is labeled at the end of this algorithm then we find the path $\mu^k = (d_p, d_{p-1}, \ldots, d_1)$, from $s = d_p$ to $t = d_1$, using the values of e_1. One starts from $e_1(t)$, so:

$p := 1$;
d_1:= t;
while $d_p \neq s$ do
    p:= $p + 1$;
    d_p:=$e_1(d_{p-1})$;
endwhile;

The final value of ε is equal to $e_3(t)$.
The other final condition, $k = 0$, shows us that there is no path in $G(\phi^k)$.

2. Flow in Network Modeling Time Tabling and Scheduling Problem

In few words the Time Tabling and Scheduling Problem is:

Data:
- the set of disciplines;
- the set of professors;
- the set of classes of learners (students, pupils, ...);
- the set of classrooms (rooms);
- the days and the time periods for activities planning;
- the dependences between the all five previous sets.

Requirements: A schedule than every element from every set must be well found:
- all of disciplens must be planned;
- every professor can realize the weekly hours number;
- all of classes (students, pupils, ...) must be in schedule;
- all of rooms must be optimally used.

A schedule can be interpreted as a string of information (or as a database). From it may be extracted:

- the schedule of all classes (students, pupils, ...);
- the schedule of all professors;
- the schedule of all rooms.

A record (a position in this schedule) must contain: the date, the time, the discipline, the room, the class and the professor.

Suppose that we can encode the time periods of a week. For example: suppose that all activities (courses, seminars, laboratories, ...) last two hours (120 minutes). One can count the pair hours between 8 and 20 from Monday to Friday. One count 5 days x 6 period = 30 values (the attribute **H**our). On the other hand the class of students, pupils, ... can be encoded (the attribute **C**lass). A similar codification can be done for each discipline (the attribute **D**iscipline). From each pair Class-Discipline one identifies the professor.

This activity of making the schedule is modeling like a maximal flow problem in a special graph. Than, the elements of the previous sets will be considered as vertices. Will be five types of vertices:

- Disciplines;
- Professors;

- Class (students, pupils, ...);
- Rooms;
- Hours.

A record (a position in the schedule) can be interpreted as a path in this special graph and contains vertices from every type.

The first proposed model attaches to the Time Tabling and Scheduling Problem a multipartite graph. This graph has five partitions:

- the first partition contains the disciplines;
- the second partition contains the professors;
- the third partition contains the classes (students, pupils,);
- the fourth partition contains the classrooms;
- the final partition contains the hours.

In this case for each professor, each discipline, each class, each room there is a corresponding vertex in that graph. For each hour (or two hours = 120 minutes) we have a vertex in graph in the hours partition as, for example:

- Monday: the time 10-12 is labeled with 2;
- Wednesday: the time 14-16 is labeled with 16, and so on.

There will be 30 vertices (5 days x 6 periods of time / each day). If we would like to solve the Time Tabling and Scheduling Problem for a school of pre-university level one can consider 60 vertices for 60 hours (5 days x 12 periods of time / each day).

The vertices with compatibilities with some activities will be connected. For example, every discipline is taught by a subset of the set of professors, every professor do some activities only with certain classes of students, every room must be available at certain hours weekly.

In addition to the five partitions of vertices, we have two vertices: s for source and t for sink. An example is shown in Figure 1.

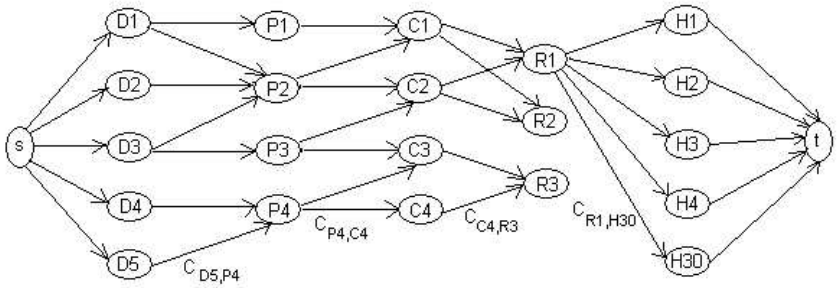

FIGURE 1. A multipartite graph example

As we see, every discipline is connected to a subset of professors-vertices. For example: discipline D_1 can be taught only by professors P_1 and P_2; the capacity value between vertices D_5 and P_4 is $c_{D_5P_4}$.

Every professor can teach a subset of classes of learners. For example: Professor P_4 may teach only classes C_3 and C_4; the capacity value between vertices P_4 and C_4 is $c_{P_4C_4}$.

Every class of learners may carry its activities in certain rooms. For example: C_1 carries its activities only in R_1 and R_2. The capacity value between vertices C_4 and R_3 is $c_{C_4R_3}$.

Finally, every classroom will be connected to the available hours. For example: R_1 is related to $H_1, \ldots, H_{30}$; the capacity value between vertices R_1 and H_{30} is $c_{R_1H_{30}}$.

The source s is connected to every discipline-vertex and the sink t is connected to every hour-vertex.

Denotation:

$G = (V, E)$	–	the network (the graph attached to the problem);
V	–	the set of vertices;
E	–	the set of edges;
$R = (G, s, t, c)$	–	the transportation network;
s	–	the source;
t	–	the sink;
c	–	$c : E \to \mathbb{R}_+$, $c(i)$ = the capacity of edge i, $\forall i \in E$.

The capacities will be built such that:

- if a professor P must teach c hours for a discipline D then the edge between D and P has the capacity equal to c ($c_{D,P} = c$);
- if a professor P must do c activity hours with a class C then the edge between P and C has the capacity equal to c ($c_{P,C} = c$);
- for every edges between a class C and a classrooms R, the capacity is at least $\sum_{e \in \Gamma^{-1}(C)} c(e)$; the maximal may be either 30 or 60;
- the capacities attached to the edges between classrooms-vertices and hours-vertices will be set to 1 or 2;
- every edge form source s to discipline D has the capacity $c_{s,D}$ equal to the number of weekly-hours;
- the edges between every hour H and the sink vertex t have the capacity $c_{H,t} = 1$.

An example with one discipline (D_1), two professors (P_1, P_2), two classes (C_1, C_2), two classrooms (R_1, R_2) and three hours (H_1, H_2, H_3) is depicted in Figure 2.

Every path from s to t may be a position in schedule because the path locates the discipline, the professor, the class, the classroom and the time.

Now, for the schedule determination we solve the problem with Ford-Fulkerson algorithm, by determining the maximal flow in the network $R = (G, s, t, c)$. The maximal flow between s and t is found out by successive selection of certain paths in

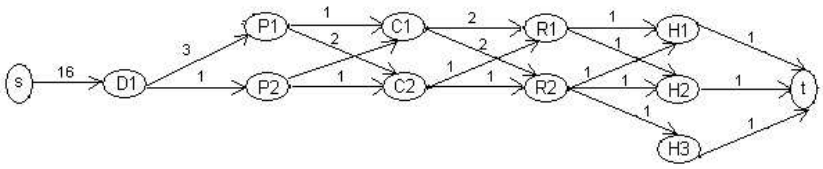

FIGURE 2. A specific example

the ecarts graph. Every selected path from s to t is a position in the schedule. For every edge of these paths, the capacity equals to the capacity minus the residual capacity.

When obtaining the maximal flow in the schedule, all the schedule positions have been set. Now the schedule is realized.

3. Conclusions and Improvements

By using the method presented above a schedule variation is determined, which depends on the selection order of the path in the ecarts graph.

If an optimization of the produced schedule is needed, we suggest that to the transportation network defined above some extra edges, called inhibitory edges, should be added. These edges define the schedule restrictions, e.g. if a certain professor is not available at specific hours, an inhibitory edge will be added between those professor-vertex and hour-vertex. As another example, if some classes, some activities may not take place, an inhibitory edge will be added between the corresponding vertices. When using the Ford-Fulkerson algorithm, the paths with inhibitory edges will not be selected. Then, the network will be represented by using the two types of edges, e.g. two adjacent matrices or a matrix with values equal the edges capacity, or zero, if no edge exists, or -1 for an inhibitory edge. Each solver may decide on the data representation of the network defined by the model described for the scheduling problem.

References

[1] Ford, L.R. Jr., *Network-flow theory*, The Rand Corporation, 1956, pp. 293
[2] Toadere, T., *Grafe: Teorie, algoritmi şi aplicaţii*, Editura Albastră, Cluj, 2001

Department of Computer Science, "Babeş-Bolyai" University, 1 M. Kogălniceanu St., RO-3400 Cluj-Napoca, Romania
E-mail address: toadere@cs.ubbcluj.ro