Computers Math. Applic. Vol. 21, No. 1, pp. 137–156, 1991
Printed in Great Britain

0097-4943/91 \$3.00 + 0.00
Pergamon Press plc

A PARTICULAR TIMETABLE PROBLEM: TERMINAL SCHEDULING

ANDRÁS SEBŐ
Computer and Automation Institute of the Hungarian Academy of Science
H - 1132, Budapest, Victor Hugo 18–22; Hungary

(Received May, 1990)

Abstract. A timetable has to be constucted for ℓ identical machines, m users and n disjoint hours under the following conditions:
a/ Each machine has its working hours, a subset of the n disjoint hours.
b/ To each user belongs a number, the number of hours he would like to work and also the subset of hours he is able to work on a machine.
c/ The users have other wishes: the set of hours is the ordered set of the hours of the week, and some of the users would like lon g intervals of time, some others may want their time to be uniformly distributed within the week, etc.
With the help of well known graph theoretic means, a good characterization for the feasibility of the basic problem defined by a/ and b/ and a quick algorithm to solve this problem can be given. We will also show how some other requirements arising in practice can be built in our model (e.g. those mentioned in c/). This model can be considered as the common formulation of a class of problems arising in practice.

0. INTRODUCTION

In the IBM 3031 computing center of the H.A.S. (Hungarian Academy of Sciences) the time requests of display – terminal users significantly surpass the total amount of available terminal time. The U.S. embargo restrictions allow at most 5 display terminals in the configuration, and each of these work 13 hours a day. The users belong to different institutions, and most of them come from a long way off, therefore they have are able to use the terminals. This paper is the result of the research that aimed at making a schedule for this computing center for weekly use.

The same model can be used to plan *consulting hours* for clients at offices, or *customers' timetables* at servicing enterprises, etc...

Generally, the problem is formulated as follows: Given a set T of identical terminals, a set U of users and a set H of disjoint hours $|T| = \ell, |U| = m, |H| = n$, a timetable is to be constructed. We first give an informal description of the basic constraints (without precisely defining what a timetable is), the formal description will follow after some simplications:

0/ Each user in each hour can work at most al 1 terminal, and each terminal in each hour can serve at most 1 user.

1/ A function $\Omega : T \rightarrow 2^H$ is given. Terminal $t \in T$ is functioning in hour $h \in H$ if and only if $h \in \Omega(t)$.

2/ A function $\Gamma : U \rightarrow 2^H$ is given. User $u \in U$ is ready to work in hour $h \in H$ if and only if $h \in \Gamma(u)$.

3/ A function $r : U \rightarrow \mathbf{N}$ (N) is the set of positive integers) is given. $r(u)$ $(u \in U)$ is the time *request* of user u, i.e. the number of hours user u has to work at a terminal.

The other conditions will be considered later.

This is not the most general model we must and can treat (e.g. we do not need all terminals to be equal, it is enough to have for each user a subset of terminals equally good for him). But

Typeset by $\mathcal{A}\mathcal{M}\mathcal{S}$-TEX this is the fundamental model that shows the main ideas. However, the generalizations will also be mentioned in Part 4.

The above formulation is to emphasize the connection to the classical "class – teacher timetable problem" (see e.g. [1] or [3]): if the teachers are identical, i.e. for each class the number of hours spent at school is given, but in the timetable the distribution of this number among the different teachers is arbitrary, then the user – class and terminal – teacher correspondences make clear that the two problems are the same. (Our problem is not purely a special case, as the classical problem did not allow such a request that a class has to spend "either 1 hour with teacher A or 1 hour with teacher B", which can happen if the teachers A and B are equal. Such "disjunctive" contraints will also be discussed in Part 4.) Note that the classical problem was proved in [3] to be NP – complete in general, and solvable in polynomial time in the special case, when all the teachers are always free. (The latter fact has already been mentioned in the earlier paper of de Werra in [8].) Our approach could also be used as a heuristic tool to approximate the NP – complete problem: in Part 1. we give a quick algorithm that solves the basic problem defined in 1./, 2./ and 3./ or in (0.1), (0.2), (0.3) below more formally, but the results of Part 3. and some remarks of Part 4. concerning the case when differences may occur berween the terminals can also help in using the results of Part 1. as a heuristic tool for the construction of general class – teacher timetables. However, the author has no computational experience in this direction yet.

The formulation of 1./ can be much simplified: the set T and the function Ω can be replaced by a single function $t : H \to \mathrm{N}$, where $t(h)$ $(h \in H)$ is the number of terminals functioning in hour h. $/t(h) = |\{x \in T : h \in \Omega(x)\}|/$ As the terminals are equal, t gives enough information about the availability of terminals, so T and Ω can be completely omitted from the model. We formalize now what we mean by timetable.

Definition 0.1

The *timetable – problem* defined by the sets U, H and the functions $\Gamma : U \to 2^H$, $r : U \to \mathrm{N}$, $t : H \to \mathrm{N}$ will be denoted by $P(U,H,\Gamma,r,t)$. A *timetable* for $P(U,H,\Gamma,r,t)$ is a function $\tau : U \times H \to \{0,1\}$ that satisfies (0.1), (0.2), (0.3) :

(0.1) $\tau(u,h) = 1 \Rightarrow h \in \Gamma(u)$ $(u \in U, h \in H)$

(0.2) $\sum\limits_{h \in H} \tau(u,h) \le r(u)$

(0.3) $\sum\limits_{u \in U} \tau(u,h) \le t(h)$

If (0.2) is satisfied with equality for all $u \in U$, then τ will be called a *complete timetable.* If a complete timetable exists, we shall say that $P(U,H,\Gamma,r,t)$ is a *feasible problem.*

Remark 0.2

If τ is a timetable, it is easy to see that the users can be placed to the terminals in T to satisfy 0/ and 1/. The *user – terminal timetable can even be constructed so that consecutive hours os a user are effectuated without a change of terminals.* The checking of these easy statements is left to the reader.

Definition 0.3

The *value* of the timetable problem is defined by $\max\limits_{\tau} \sum\limits_{h \in H} \sum\limits_{u \in U} \tau(u,h)$ where the maximum is taken over all timetables τ for $P(U,H,\Gamma,r,t)$. Let us fix $U, H; \Gamma, t$ and denote by $\nu(r)$ the value of the problem $P(U,H,\Gamma,r,t)$. A *maximal timetable* τ is one with $\sum\limits_{h \in H} \sum\limits_{u \in U} \tau(u,h) = \nu(r)$ $\nu(r) \le \sum\limits_{u \in U} r(u)$ is obvious from (0.2), and equality is satisfied if and only if $P(U,H,\Gamma,r,t)$ is feasible.

The paper consists of 4 parts. In part 1. we give *a necessary and sufficient condition (a "good characterization") for $P(U,H,\Gamma,r,t)$ to be feasible.* We also give a polynomial algorithm for the optimization of a weighted version of the problem. The results are direct applications of network flow theory.

In parts 2., 3., 4. the basic conditions (0.1), (0.2), (0.3) are completed by other important constraints. The aim of the paper is to construct timetables satisfying all these conditions.

In part 2. we investigate the case when the problem $P(U, H, \Gamma, r, t)$ is not feasible. Defining the notion of "fairness" we show how the requestfunction r can be *"fairly" modified so as to yield a feasible problem.* The value $\nu(r)$ of the problem should not be decreased by the modification. This is one reason why the notion of "fairness" must depend on the problem structure, and so it needs a mathematical background.

In part 3. we give a general method of modifying already existing timetables so that they satisfy other constraints. We use "minimal path" algorithms to design suitable cyclic permutations of the users. This method e.g. partly solves the problem of the users who require *several consecutive hours without break.* It may also help the lgorithms of this paper to provide good heuristics for more difficult timetable problems.

In part 4. we sketch the solution methods for some restrictions and generalizations of the problem, which arise in practice, or prove the "intractibility" in the sense used by Garey and Johnson in [10]. (E.g. the users may have preferences on some of the terminals or may want to give their requests for each day of the week, etc.)

Network flows are the main tools in this paper:

Alternating chains are used for various goals throughout the paper, and a subroutine that finds minimal paths between two points of a graph is used in part 3.

Network flow theory as heuristic tool for some types of timetable problems or a means to solve scheduling problems has been used by many authors (see e.g. [1], [2], [6]). Here we borrow some ideas of matching theory to give *exact solutions* to the type of timetable problems we investigate. The results of part 1. may be considered as modifications and generalizations of theorems in [2], but despite the similarity of the tools our model has no strong connection with [1] or [6]: in these, class – teacher relations play the main role, while we have user – *time* relations in the centre.

The APL codes of the algorithms are put together to form a package that solves weekly the user – terminal/time problem at the computing centre of the H.A.S. They have $0((n+m)^3)$ as worst case performance with a $0(n^2m)$ behaviour at running, without any risk of failure to produce a timetable, which seems to be rare among exact solutions to timetable problems arising in practice.

Notation and terminology

In the paper we shall use graphs and digraphs as a tool. We mainly use the notation and terminology of [9]: The vertex set of a (di)graph G will be denoted $V(G)$, the edge set $E(G)$; The notation for the *degree* of $x \in V(G)$ will be $d_G(x); \Gamma_G(x)$ is the set of *neighbours* of x. $(d_G(x) = |\Gamma_G(x)|; \Gamma_G(x) = \{y \in V(G) : (x, y) \in E(G)\})$ and $\Gamma_G(x) = \bigcup_{x \in X} \Gamma_G(x)$ $(X \subset V(G))$; the sequence $(x_1, \cdot, x_k)$ is a *walk* if $(x_i, x_{i+1}) \in E(G)$ $(i = 1, \dots, k-1)$ the walk is *closed* if $x_1 = x_k$, otherwise *open*; If $x_1, \dots, x_k$ are distrinct, then the walk $(x_1, \dots, x_k)$ is called a *path*, and if $x_1, \dots, x_{k-1}$ are distrinct but $x_1 = x_k$, then it is called a *circuit* for graphs and a cycle for digraphs. If $e \in E(G)$, then $G \backslash e$ is the graph with $V(G \backslash e) = V(G)$, $E(G \backslash e) = E(G) \backslash e$.

1. GOOD CHARACTERIZATION AND ALGORITHM FOR THE BASIC PROBLEM

In this part of the paper we sketch an algorithm that either *constructs a complete timetable or proves* that it does not exists, and in the latter case too, builds up a maximal timetable. If a weight function $w(u, h)$ $(u \in U, h \in H)$ is given, the timetable with maximal weight, i.e. with $\sum_{h \in H} \sum_{u \in U} \tau(u, h)\ w(u, h)$ maximal, is also determined. The results also follow from the minimal cost maximal flow algorithm and are closely related to theorems on "f – factors" (see eg. in [9]) or Ore's theorem for directed graphs. The reader aquainted with the bases of flow theory or at least with the elements of the matching theory of bipartite graphs may skip most arguments in this part of the paper. As the notions and ideas are quite common, occasionally we shall be informal. We are going into details only because these details will be used and referred to in parts 2., 3. and 4., and where this is not the case, we just give an informal explanation.

The main results of this part are THEOREM 1.2, 1.8 and 1.12.

Definition 1.1

The *request* $r(u,Y)$ of user $u \in U$ in $Y \subset H$ is defined by $r(u,Y) = \max\{r(u) - |\Gamma(u)\backslash Y|, 0\}$ Obviously

$$r(u) = r(u,H) \forall u \in U.$$

In fact, it is easy to see that user u must have at least $r(u,Y)$ hours in Y on a feasible timetable but not necessarily more. Thus the only if part of the following tehorem is obvious.

Theorem 1.2

$P(U,H,\Gamma,r,t)$ is feasible if and only if for all $Y \subset H$

$$(\star) \qquad \sum_{u \in U} r(u,Y) \leq \sum_{h \in Y} t(h)$$

(If $r(u) = t(h) = 1$ $\forall u \in U$, $\forall h \in H$ this gives the "König – Hall theorem".)

Proof. The only if part is obvious as the left hand side is the sum of the requests and the right hand side is the total amount of hours available for work in Y. To prove the if part, let us consider *the bipartite graph G with* $V(G) = U \cup H, E(G) = \{(u,h) : h \in \Gamma(u)\}$ (As $\Gamma_G(u) = \Gamma(u)$ $\forall u \in U$, we shall write $\Gamma(u)$ for $\Gamma_G(u)$.) A timetable τ defines a subgraph T of G with $V(T) = V(G)$, $E(T) = \{(u,h) : \tau(u,h) = 1\}$, having the following properties:
(1.1) $E(T) \subset E(G)$ (equivalent to (0.1))
(1.2) $d_T(u) \leq r(u)$ (equivalent to (0.2))
(1.3) $d_T(h) \leq t(h)$ (equivalent to (0.3))
((0.1), (0.2) and (0.3) are references to definition 0.1 part 0.) Moreover τ is a complete timetable if and only if (1.2) is satisfied with equality for all $u \in U$. Algorithm I. below is an algorithm that *either*:
(i) constructs T with properties (1.1.); (1.2.) with equality and (1.3.), *or*:
(ii) constructs $H^* \subset H$ with property (1.4.):

$$\sum_{u \in U} r(u,H^*) > \sum_{h \in H^*} t(h) \text{ (see proposition 1.6. below)} \qquad (1.4.)$$

Now if (P,U,H,Γ,r,t) is not feasible, then (i) does not hold, thus (ii) holds, consequently $(\star)$ does not hold for $Y = H^*$. □

Definition 1.3

Let the edgas of the graph G be coloured with two colours, red and blue. A path (or walk) $(x_1, \dots, x_k)$ is called *alternating* if:

$$(x_i, x_{i+1}) \text{ is blue} \Rightarrow (x_{i+1}, x_{i+2}) \text{ is red}$$

$$(x_i, x_{i+1}) \text{ is red} \Rightarrow (x_{i+1}, x_{i+2}) \text{ is blue}$$

We call an alternating path *red – red* or *red – blue* or *blue – blue* or *blue – red* depending on the colour of its first and last edge. (E.g. if both the first and the last edges are red, then it is a red – red alternating path.) From now on the word *timetable* will be used sometimes for a subgraph T satisfying (1.1.), (1.2.), (1.3.).

The following algorithm is essentially the same as the so called "Hungarian method" (see [9].

Algorithm I

illustration on figure 1.a and 1.b

Input

U, H, Γ, r, t (see part 0.)

Output

$U^\star H^\star, \tau$

Assignment commands are denoted by " $\leftarrow$ ".

Step 0

Consider the graph G determined by U, H, Γ as defined in the proof of Theorem 1.2. Colour some edges of G in red so that the graph T consisting of the red edges should satisfy (1.1.), (1.2.), (1.3.). (T might be the empty graph, but the more edges are now coloured in red, the shorter running time we can expect.) Colour the rest of the edges in blue.

Remark

A user $u \in U$ (or hour $h \in H$) will be called *saturated* if $d_T(u) = r(u)$ $d_T(h) = t(h)$), otherwise *unsaturated*. The essence of this algorithm is to look for blue - blue alternating paths from unsaturated users to unsaturated hours: interchanging the colours of such an "increasing alternating path" (1.1.), (1.2.), (1.3.) are still satisfied (with $E(T) = \{$ red edges $\}$), and the number of red edges is increased by 1. If there is no such path, we shall deduce that (ii) holds. Increasing alternating paths will be found by a standard labelling process.

A *blue /red/ neighbour* of a point $x \in V(G)$ is a point $y \in V(G)$ with $(x, y) \in E(G)$ and (x, y) a blue /red/ edge. ($(x, y) \in E(G)$ is red if and only if $(x, y) \in E(T)$, otherwise blue.)

Step 1

$L_U \leftarrow \{u \in U : d_T(u) < r(u)\} = \{$ unsaturated users $\}$, and assign the label "0" to the points of L_U

Step 2

If $L_U = 0$

$$\text{STOP} : \tau(x, y) = \begin{cases} 1 & \text{if } (x, y) \in E(T); \\ 0 & \text{otherwise} \end{cases}$$
$$U^\star = 0; \; H^\star = 0$$

Step 3

Assign a label to all unlabelled points of II that have a blue neighbour in L_U. Let the label be equal to an arbitrary neighbour in L_U. Let L_H be the set of points we labelled in this step.

$$H^\star \leftarrow H^\star \cup L_H$$

Remark

A label can be formalized to be a function ℓ with $\ell(x) \in H \cup \{0\}$ if $x \in U$ and $\ell(x) \in U$ if $x \in H$. The set of "labelled" points in U and H is always $U^\star$ and $H^\star$ resp. L_U will always consist of the newly labelled points of U and L_H of those of H.

Step 4

If $L_H \neq 0 = L_H \cap \{h \in H : d_T(h) < t(h)\}$ GO TO Step 5.
If $L_H \cap \{h \in H : d_T(h) < t(h)\} \neq 0$ (i.e. an unsaturated hour is labelled) GO TO Step 6.
If

$$L_H = 0 \quad STOP : \tau(x, y) = \begin{cases} 1 & \text{if } (x, y) \in E(T) \\ 0 & \text{otherwise} \end{cases}$$

Step 5

Assign a label to all unlabelled points of U that have a red neighbour in L_H. Let the label be equal to one of the neighbours in L_H. Let L_U be the set of points we labelled in this step; $U^\star \leftarrow U^\star \cup L_U$ GO TO Step 2.

Step 6

Choose $h_0 \in H^\star \cap \{h \in H : d_T(h) < t(h)\}$. The label given to h_0 is some $u_1 \in U^\star$ which itself has a label $h_2 \in H^\star \cup \{0\}$ etc. A path P between h_0 and some $u^\star \in U^\star$ with label "0" i.e. with $d_T(u^\star) < r(u^\star)$ is thus determined. It follows from Step 3. and Step 5. that P is a ***bluec - blue alternating path***. Interchange the colour on the edges on P; Redefine T with $E(T) = \{$ red edges $\}$ (Or equivalently drop the red edges of P from, and take the blue edges of P to T); GO TO Step 1.

Example 1.4

$U = \{A, B, C, D\}$ $H = \{1, 2, 3\}$
$r(A) = r(B) = r(D) = 1$; $r(C) = 2$
$t(1) = t(2) = 2$; $t(3) = 1$

= blue edge

= red edge

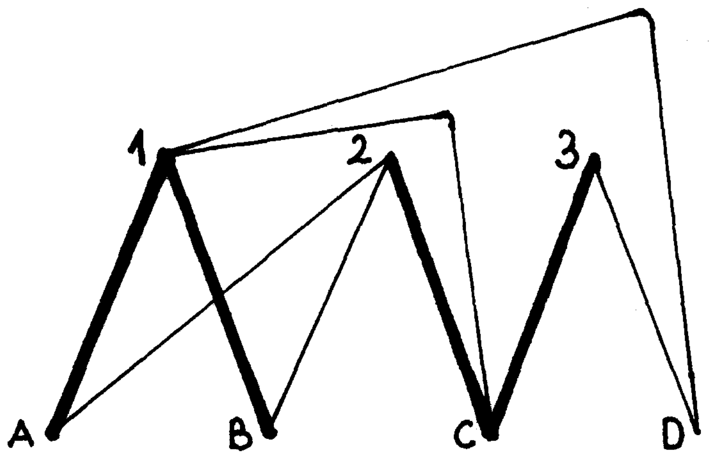

Figure 1.a.

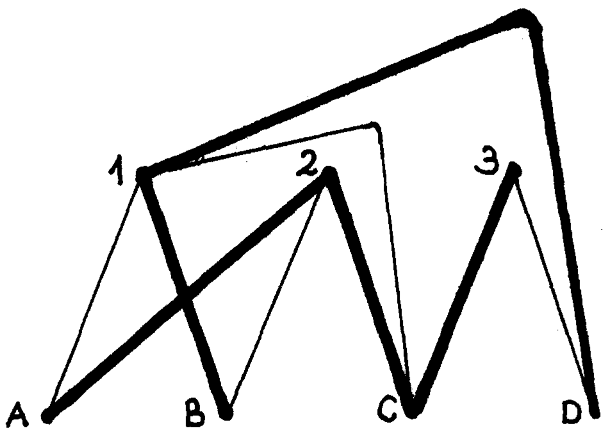

Figure 1.b.

Figure 1.a. shows an initial timetable (Step 0.) Step 1.: $L_U = U^\star = \{D\}$; $H^\star = 0$
$\ell(u)$ denotes the label of $u \in U$.
Step 2., 3.: $\ell(1) = l(3) = D; H^\star = L_H = \{1, 3\}$
Step 4., 5.: $\ell(A) = \ell(B) = 1; \ell(C) = 3$

$$L_U = \{A, B, C\};\ U^\star = \{A, B, C, D\}$$

Step 2., 3.: $\ell(2) = A$; $L_H = \{2\}$; $H^\star = \{1, 2, 3\}$
Step 4., 5.: $\ell(2) = A$; $\ell(A) = 1$; $\ell(1) = D$; $\ell(D) = 0$; $(D, 1, A, 2)$ is a blue - blue alt. path

Figure 1.b. shows the situation after interchanging the colours. A feasible timetable is constructed and the algorithm stops at Step 2. because of $L_U = 0$.

Remark 1.5

After each occurance of Step 4. either $U^\star$ gets larger or the number of red edges increases by 1 or the algorithm stops. Thus it stops after $c(n+m)\ |E(H)|$ number of steps. When the algorithm has stopped, $U^\star and H^\star$ have the following properties (see figure 1.):

$$\forall h \in H^\star:\ d_T(h) = t(h);\ \forall u \in U \backslash U^\star:\ d_T(u) = r(u) \tag{1.5.}$$

for if $h \in H$ with $d_T(h) < t(h)$ is labelled, then Step 6. follows Step 4., and after Step 6. the labelling begins again. $d_T(u) = r(u)$ follows from Step 1.

If $u \in U^\star, h \in H$ and $(u,h) \in E(G)$ is a blue edge ($(u,h) \in E(G)\backslash E(T)$), then by Step 3. $h \in H^\star$ i.e.

(1.6.) $u \in U^\star, h \in H\backslash H^\star \Rightarrow (u,h)$ is not a blue edge, and similarly from Step 5.

(1.7.) $u \in U\backslash U^\star, h \in H^\star \Rightarrow (u,h)$ is not a red edge. If $U^\star \neq 0$, then obviously (see Step 1.)

(1.8.) $\sum\limits_{u \in U^\star} d_T(u) < \sum\limits_{u \in U^\star} r(u)$

Proposition 1.6

Consider the sets $U^\star, H^\star$ after the algorithm has stopped. Then

(i) $U^\star = 0$ if and only if τ is a complete timetable.

(ii) $U^\star \neq 0$ if and only if (1.4.) holds.

PROOF. (1.1.), (1.2.), (1.3.) are satisfied all along algorithm I. When it stops, $U^\star = 0$ if and only if (1.2.) is satisfied with equality (see Step 1.), thus (i) is proved.

To prove (ii) first remark that (1.6.) and (1.7.) together imply

$$\sum_{h \in H^\star} d_T(h) = \sum_{u \in U^\star} (d_T(u) - |\Gamma_G(u)\backslash H^\star|) \tag{1.9.}$$

for (1.7.) implies that the number of red edges adjacent to $H^\star$ is equal to the number of red edges from $U^\star$ to $H^\star$ and (1.6.) implies that all the $|\Gamma_G(u)\backslash H^\star|$ edges from $u \in U^\star$ to $H\backslash H^\star$ are red, thus from U to $H^\star$ there are $d_T(u) - |\Gamma_G(u)\backslash H^\star|$ red edges.

Suppose now that $U^\star \neq 0$. Apply first (1.5.) and (1.9.), then (1.8.) and definition 1.1.:

$$\sum_{h \in H^\star} t(h) = \sum_{h \in H^\star} d_T(h) = \sum_{u \in U^\star} (d_T(u) - |\Gamma(u)\backslash H^\star|) <$$

$$\leq \sum_{u \in U^\star} (r(u) - |\Gamma(u)\backslash H^\star|) \leq \sum_{u \in U^\star} r(u, H^\star) \leq$$

$$\leq \sum_{u \in U} r(u, H^\star).$$

(The last two inequalities are in fact equalities: if $u \in U^\star$, then $r(u) - |\Gamma(u)\backslash H^\star| > 0$ thus $r(u,H^\star) = r(u) - |\Gamma(u)\backslash H^\star|$, and if $u \notin U^\star$, then $r(u) - |\Gamma(u)\backslash H^\star| \leq 0$, whence $r(u,H^\star) = 0$ [see definition 1.1.]).

"(1.4.) $\Rightarrow U^\star \neq 0$" is trivial as (1.4.) implies there is no complete timetable $\Rightarrow$ for all T that satisfies (1.1.), (1.2.), (1.3.) there exists $u \in U$ with $d_T(u) < r(u)$ thus $U^\star \neq 0$ at Step 1. at any time.

Remarks 1.7

1. Ore's theorem (see e.g. [7]) gives a good characterization for an arbitrary directed graph to have a subgraph with given lower and upper bounds for the in – and outdegrees in each point. It follows readily from Ore's theorem that $P(U,H,\Gamma,r,t)$ is feasible if and only if $\forall x \subset U$:

$$\sum_{u\in X} r(u) \le \sum_{h\in H} \min\{|X\cap\Gamma_G(h)|, t(h)\}$$

which is another necessary and sufficient condition. Theorem 1.2. and this statement can easily be proved from each other. Ore's theorem is a special case of the Ford – Fulkerson "max flow min cut" theorem (see [7]). As alternating chains are the special case of "flow improving chains" for $0-1$ capacities, algorithm I. has nothing surprising in it.

2. We shall, however, need the above formulation of the algorithm and mainly the pair of sets $(U^\star, H^\star)$. It has the heuristic meaning that the *users in $U^\star$ want to work too much in the hours of $H^\star$, and this is what prevents* the existence of a complete timetable. These sets will play an important role in the future.

3. In the proof of proposition 1.6. we have only used (1.5.) through (1.8.). *Thus if we have sets $U^\star, H^\star$ and a timetable T satisfying (1.5.) through (1.8.), then we know that T is maximal.*

If the condition of theorem 1.2. does not hold, then $\min_{Y\in H}\{\sum_{h\in Y} t(h) - \sum_{u\in U} r(u,Y)\}$ is negative. The inequality

$$\nu(r) \le \sum_{u\in U} r(u) + \min_{Y\subset H}\{\sum_{h\in Y} t(h) - \sum_{u\in U}(r(u,Y))\} \tag{1.10.}$$

is obvious. On the other hand, the graph T constructed by algorithm I. (after stopping) has

$$|E(T)| = \sum_{u\in U} r(u) + (\sum_{h\in H^\star} t(h) - \sum_{u\in U^\star} r(u,Y)) \tag{1.11.}$$

edges and satisfies (1.1.), (1.2.), (1.3.). (1.11.) can easily be proved using (1.5.) through (1.8.) with similar calculations as in the proof of (ii) in proposition (1.6.) (1.10.) and (1.11.) together prove the following theorem:

Theorem 1.8

$\nu(r) = \sum_{u\in U} r(u) + \min_{Y\subset H}\{\sum_{h\in Y} t(h) - \sum_{u\in U} r(u,Y)\}$. The timetable constructed by algorithm I. is maximal.

Theorem 1.2. follows easily from theorem 1.8., as its condition is equivalent to

$$\min_{Y\in H}\{\sum_{h\in Y} t(h) - \sum_{u\in U} r(u,Y)\} = 0 \text{ and in this case}$$

by theorem 1.8., we have $\nu(r) = \sum_{u\in U} r(u)$. In the following we shall often write $P(G,r,t)$ instead of $P(U,H,\Gamma,r,t)$ and we shall call T that satisfies (1.1.), (1.2.), (1.3.) a *timetable for* $P(G,r,t)$. (It is a *complete timetable* if (1.2.) is satisfied with equality.) This convention does not cause any problem, as $\{U,H,\Gamma\}$ and G as well as T and r uniquely determine each other.

In the proof of (ii) of proposition 1.6. and in the justifying argument for (1.11.) which play the main role in the proof of theorem 1.2. and 1.8. respectively, we referred to the structure of $G, T, U^\star, H^\star$ reflected in (1.5.) through (1.8.). *In fact we do not need to know that $U^\star$ and $H^\star$ emerge in algorithm I., the existence of $U^\star, H^\star$ with properties (1.5.) through (1.8.) are sufficient to prove the maximality of T* : this fact will be important to us to recognize maximal timetables:

Theorem 1.9

Let T be a timetable for $P(G,r,t)$. Colour the edges $e \in E(T)$ red and the edges $e \in E(G)\backslash E(T)$ blue. If there exist sets $U^\star \subset U$ and $H^\star \subset H$ with (1.5.) - (1.8.), then $P(U,H,\Gamma,r,t)$ is not feasible, moreover $|E(T)| = \nu(r)$, i.e. τ is a maximal timetable for $P(U,H,\Gamma,r,t)$.

Of course, the converse is also true: if τ is maximal, then either it is comlete, i.e. (1.2.) holds with equality, or it is not, and then there exist sets $U^\star, H^\star$ with (1.5.) - (1.8.). (Naturally (1.5.) - (1.8.) could be written in therms of τ and Γ instead of T and G).

We have already remarked that at the end of algorithm I. $U^\star = \{u \in U : r(u,H^\star) > 0\}$. In fact $U^\star$ *does not depend on algorithm I. We shall need the following characterization, wich shows that it only depends on the problem* $P(U,H,\Gamma,r,t)$.

Proposition 1.10

Consider $U^\star$ at the end of algorithm I. $U^\star = \{u \in U : \nu(r^{(u)}) = \nu(r)\}$ where

$$r^{(u)}(x) = \begin{cases} r(x) - 1 & \text{if } x = u \\ r(x) & \text{if } x \in U \ x \neq u. \end{cases}$$

Or equivalently $u \in U\backslash U^\star$ if and only if $\nu(r^{(u)}) = \nu(r) - 1$.

PROOF. If $u \in U\backslash U^\star$, then deleting a red edge adjacent to u, the red edges satisfy (1.1.), (1.2.), (1.3.) with $r^{(u)}$ instead of r, and $T\backslash e$ instead of T. As $U^\star, H^\star$ and T satisfy (1.5.) - (1.8.) (see remark 1.5.), it follows that $U^\star, H^\star$ satisfy (1.5.) - (1.8.) replacing T by $T\backslash e$ and r by $r^{(u)}$. Thus by theorem 1.9. $|E(T)\backslash e| = \nu(r^{(u)})(T\backslash e$ is maximal for $P(U,H,\Gamma,r^{(u)},t))$ i.e. $\nu(r^{(u)}) = \nu(r) - 1$.

If $u \in U^\star$, then u is a point labelled once in step 5. (or step 1.) If u is unsaturated (i.e. it was labelled in step 1.), then clearly $\nu(r^{(u)}) = \nu(r)$. If u is saturated, then it was labelled in step 5., thus there exists a blue - red alternating path P from some unsaturated point i to u. Interchanging the colour of the edges of P, the number of red edges does not change while (1.1.), (1.2.), (1.3.) are true with r replaced by $r^{(u)}$ i.e. a timetable for $P(U,H,\Gamma,r^{(u)},t)$ is constructed. Hence $\nu(r^{(u)}) = \nu(r)$.

In the implementation we also use the weighted version of algorithm *I.* and theorem 1.8. We summarize the results without proofs:

Definition 1.11

Let $w : U \times H \to \mathrm{N}\cup\{0\}$. $\sum_{h\in H}\sum_{u\in U} \tau(u,h)w(u,h)$ is the *value* of the timetable τ. If $\tau^\star$ is a timetable with

$$\sum_{h\in H}\sum_{u\in U} \tau^\star(u,h)\ w(u,h) = \max_\tau \sum_{h\in H}\sum_{u\in U} \tau(u,h)\ w(u,h)$$

(where τ runs over all timetables), then we shall call it *optimal timetable* (for w). The integer $\omega(w) = \sum_{h\in H}\sum_{u\in U} \tau^\star(u,h)w(u,h)$ is the *optimum value*; If $w(u,h) > 0 \Longleftrightarrow h \in \Gamma(u)$, we say that w is compatible (with Γ or with G). Obviously a maximal timetable is optimal for the weight function

$$w_0(u,h) = \begin{cases} 1 & \text{if } h \in \Gamma(u) \\ 0 & \text{if } h \in H\backslash\Gamma(u) \end{cases}$$

Theorem 1.12

$\omega(w) = \min\{\sum_{u\in U} \xi(u)r(u) + \sum_{h\in H} \eta(h)t(h) + \sum_{u\in U}\sum_{h\in H} \max(0, w(u,h) - \xi(u) - \eta(h))\}$ where the minimum on the right hand runs over all functions $\xi : U \to \mathrm{N}\cup\{0\}$ and $\eta : H \to \mathrm{N}\cup\{0\}$. We omit the proof as it will not be needed later. This is a "duality theorem", and the experienced reader can easily prove it using the weighted version of the max - flow - min - cut algorithm (see [7]). In the implementation we generalized the Egerváry - Kuhn primal - dual algorithm (see [7]), and the generalized algorithm has the same complexity. For the case $A = r(u) = t(h)$ $(\forall u \in U,\ \forall h \in H)$

theorem 1.12. gives the Egerváry – Kuhn theorem (see [7]). Theorem 1.12. can also be given interesting "economic" interpretations.

In part 4. we shall consider the question of the choice of w in life. For obvious reasons we require that an optimal timetable should be maximal as well (i.e. optimal for both w and w_0), so w must have a special structure. It cannot be left to the users to define w, and one must be very careful to determine it in a reasonable way.

2. FAIR REDUCTION OF THE REQUESTS

In this part we investigate the case when algorithm I. stops with proving that there exists no complete timetable. We give a mathematical definition of the notion of "fairness", a definition we find to be a natural description of our ethic sense. We then give an algorithm that constructs a fair timetable showing also the converse, i.e. that all fair timetables are given by our algorithm. First we need some definitions and lemmas.

Let us suppose that for each $u \in U$ a monoton nondecreasing function $p_u : \{0, 1, \ldots, r_u\} \to \mathrm{N}$ is given. The function value $p_u(d)(0 \leq d \leq r_u\ d$ is integer) has the heuristic meaning of measuring how user u is "favourized" if the receives d hours in the timetable. Fairness will mean some sort of "uniformity" or "balance" in the distribution of time, and we need the functions $p_u(u \in U)$ to give a measure: a strict uniformity would prevent flexibility with regard to the needs, merits etc. We suppose that all these differences are comprised in p_u and the fairest solution would be one with

$$p_i(d_i) = p_j(d_j \forall i, j \in U(d_i = \sum_{h \in H} r(i,h)$$

is the time received by user i). Of course, generally this is not possible. Moreover the set $\{\underline{d} = (d_1, \ldots, d_n) : \underline{d}$ is the degree sequence of a timetable $\}$ depends on the structure of the function Γ or equivalently on the graph G. Thus the notion of "fairness" must also depend on G.

We first define an ordering on $U \times N$.

Definition 2.1

Let us write $(u_1, d_1) < (u_2, d_2)$ $(u_i \in U,\ d_i \in \mathrm{N}\ 0 \leq d_i \leq r_i\ i = 1, 2)$ and read user u_1 with d_1 hours is *less favoured* than user u_2 with d_2 hours if both of the following conditions hold:

$$1/\ d_1 < r_1$$
$$2/\ p_{u_1}(d_1 + 1) - p_{u_2}(d_2 - 1) < p_{u_2}(d_2) - p_{u_1}(d_1)$$

Proposition 2.2

a/ If $(u_1, d_1) < (u_2, d_2)$, then $p_{u_1}(d_1) < p_{u_2}(d_2)$

b/ $<$ is a partial ordering, i.e. an irreflexive, transitive antisymmetric relation.

PROOF. Both a/ and b/ are immediate using 1/ and 2/ and the monotonity of the functions $p_{u_i}(i = 1, 2)$. □

Example 2.3

Let A and B be two users, $r_A = r_B = 3$ and the functions p_A and p_B as on figure 2.

User B with 1 hour is not less favoured than user A with 2 hours, but $(B, 2) < (A, 3)$ because

$$1 = p_B(3) - p_A(2) < p_A(3) - p_B(2) = 2.$$

That is *even if* $p_A(d_A) < p_B(d_B)$ holds, we do not say that user A is less favoured, unless *the difference between them decreases if we take 1 hour from B and give it to A, or the difference does not change sign. In other words,* $(A, d_A) < (B, d_B)$ *if and only if* $p_A(d_A) < p_B(d_B)$, *and there is some other way to distribute* $d_A + d_B$ *hours between A and B more fairly, i.e. with smaller* $|p_A(d_A) - p_B(d_B)|$.) *In the example it is fairer if B receives 3 hours and A receives 2 hours than conversely.*

Definition 2.4

For given G, r, t we say that $d : U \to \mathbb{N}$ is a distribution if there exists a timetable T for $P(G, r, t)$ with $d(u) = d_T(u)$. The distribution is fair if
1/ T is maximal
2/ $(u_1, d(u_1)) < (u_2, d(u_2))$ implies that d' with

$$d'(u) = \begin{cases} d(u)+1 & \text{if } u = u_1 \\ d(u)-1 & \text{if } u = u_2 \\ d(u) & \text{otherwise} \end{cases}$$

is not a distribution.

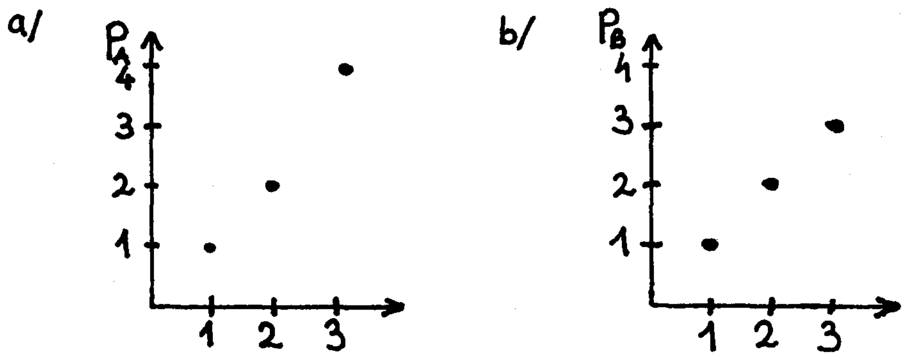

Figure 2.

Remark 2.5

Condition 1/ might appear exaggerated, but in light of proposition 2.6. below, there is no reason to give up maximality. Condition 2/ means that in a fair distribution there should not be users who could give 1 hour to less favoured users.

Proposition 2.6

If T_1 is an arbitrary timetable, then there exists a maximal timetable T_2 with $d_{T_2}(u) \geq d_{T_1}(u) \forall u \in U$.

PROOF. Colour the edges of T_1 red, and starting from this, use algorithm I. to obtain a maximal timetable. Denote this latter one T_2. Obviously, $d_{T_2}(u) \geq d_{T_1}(u) \forall u \in U$ because changing the colours along alternating paths does not decrease the degree of any point. □

We describe now the reduction algorithm which is based on proposition 1.10., and then show that the fair distributions are exactly those determined by this algorithm.

Algorithm II

```
Input: $G, r, t$
Output: $d : U \to \mathbb{N}$ fair distribution.
Step 0.: Call algorithm I. The result is a maximal timetable $T$, and $U^\star \subset U$.
Step 1.: If $U^\star = 0$,
         then GO TO Step 5.
         else GO TO Step 2.
Step 2.: Choose $u \in U^\star$ with

         $p_u(d_T(u)) + p_u(d_T(u) - 1) = \max_{i \in U^\star} p_i(d_T(i) + p_i(d_T(i) - 1)$

         GO TO Step 3.
Step 3.:
        a/ If $r(u) > d_T(u)$, GO TO Step 4.
        b/ If $r(u) = d_T(u)$, then $u \in U^\star$ implies the existence of a blue – red alternating
           path from some point with $r(u) > d_T(u)$. Interchange the colours of this path.
           GO O Step 4.
Step 4.: $r(u) \leftarrow r(u) - 1$ and modify the labelling according to the modofication of the request
        function and possibly T. (Determine $U^\star$. If $r(u) > d_T(u)$ still holds, then $U^\star$ is not
        changed.)
           GO TO Step 1.
Step 5.: $d(u) \leftarrow r(u)$ $(\forall u \in U)$
           STOP.
```

Remark

The quantity we take the maximum of in Step 2. might appear strange. However, even heuristically it is natural that both $p_u(d_T(u))$ and $p_u(d_T(u)-1)$ must influence who we take one hour from, as $p_u(d_T(u))$ is the actual favour of user u and $p_u(d_T(u) - 1)$ is the favour he will have if one hour is taken from him.

Theorem 2.7

d determined by algorithm II. is a fair distribution. Conversely, if d is a fair distribution, then d results of algorithm II. with some choice of $u \in U$ in Step 2.

PROOF. First we prove that d is a fair distribution. "1/" of definition 2.4. is obvious as we never decreased $|E(T)|$ and proposition 1.10. gives a precise justification why we did not have to. (It says that taking one hour from $u \in U^\star$ does not decrease ν.) Now suppose $(A,\ d(A)) < (B,\ d(B))$, and let us prove "2/" of definition 2.4. $(A,\ d(A)) < (B,\ d(B))$ means (see definition 2.1.):
(2.1.) $d(A) < r(A)$
(2.2.) $p_A(d(A)+1) - p_B(d(B)-1) < p_A(d(A)) - p_B(d(B))$.

From (2.1.) follows that at least in one occurance of Step 2. A is the chosen user $u \in U^\star$. Let $d^\star$ be the degree function d_T just before the last such occurance of A. We have:
(2.3.) $p_A(d^\star(A)) + p_A(d^\star(A)-1) \geq p_x(d^\star(x)) + p_x(d^\star(x)-1)$
for all $x \in U^\star$, because A is the chosen user in Step 2., and
(2.4.) $d^\star(A) = d(A)+1$
because it is the last occurance of A in Step 2. Substitute (2.4.) in (2.3.), and substitute $p_x(d(x))$ for $p_x(d^\star(x))$ using $d^\star(x) \geq d(x)$ and the monotony of p_x. After rearranging:
(2.5.) $p_A(d(A)+1) - p_x(d(x)-1)) \geq p_x(d(x)) - p_A(d(A))$
for all $u \in U^\star$. Comparing (2.5.) with (2.2.) we get the result that $B \notin U^\star$ in the step we are considering. It follows that $B \notin U^\star$ for all the rest of the algorithm, and this implies $d(B) = d^\star(B)$. But then, erasing a red edge at B, and putting $d'(B) \leftarrow d^\star(B) - 1, d'(x) = d^\star(x)$ otherwise, (1.5.) through (1.8.) are satisfied for the modified timetable and request function, which, applying Theorem 1.9., means the maximality of the timetable defined by the red edges. Thus $\nu(d') = \nu(d) - 1$. Let

$$d'' = \begin{cases} d(u)+1, & \text{if } u = A \\ d(u)-1, & \text{if } u = B.\ \text{As} d'' \leq d', \\ d(u), & \text{otherwise} \end{cases}$$

we have just proved $\nu(d'') \leq \nu(d') \leq \nu(d) - 1$, and comparing this with
$\sum\limits_{u \in U} d''(u) = \sum\limits_{u \in U} d(u) = \nu(d)$
we have that d'' is not a distribution. We have proved that 2/ of definition 2.4. is also satisfied, thus d is a fair distribution, and the first part of the theorem is proved.

Now let d be an arbitrary fair distribution. We sketch the proof of the fact that d can occur as the output for algorithm II. Let algorithm II. run with the modification that having several choices in Step 2. we choose $u \in U^\star$ with $d_T(u) > d(u)$. If this is always possible, then d is the output of the algorithm. Suppose indirectly that at some stage this is not possible, i.e. $U^\star \neq 0$ and the sets

$$M_1 := \{x \in U^\star : b_x(d_T(x)) + b_x(d_T(x) - 1) = \max_{u \in U^\star} b_u(d_T(u)) + b_u(d_T(u) - 1)\}$$

and $M_2 := \{x \in U^\star : d_T(u) > d(u)\}$ are disjoint. (d_T denotes the degrees in the considered step.) $M_2 \neq 0$ because $M_2 = 0$ would mean $d_T(u) = d(u)$ if $u \in U^\star$, and this would imply $d_T = d$ (use proposition 1.10. and the fact that d is a distribution to prove this), in contradiction with $U^\star \neq 0$. $M_1 \neq 0$ is obvious, and $d_T(u) = d(u)$ if $u \in M_1$. Let $A \in M_2$ and $B \in M_1$ be arbitrary.
As $A \in M_2$:
(2.6.) $r(A) \geq d_T(A) > d(A)$ and as $A \notin M_1$, and
(2.7.) $p_B(d(B)) + p_B(d(B) - 1) = p_B(d_T(B) - p_B(d_T(B) - 1) > p_A(d_T(A)) + p_A(d_T(A) - 1) \geq p_A(d(A) + 1) + p_A(d(A))$.
(2.6.) and (2.7.) imply $(A, d(A)) < (B, d(B))$ although we can choose $u = B$ in the following step 2. From this easily follows that d' with

$$d'(x) = \begin{cases} d(x) + 1 & \text{if } x = A \\ d(x - 1) & \text{if } x = B \text{ is also a distribution} \\ d(x) & \text{otherwise} \end{cases}$$

which is a contradiction with the supposition that d is a fair distribution. (It contradicts 2/ of definition 2.4.) The theorem is proved.

Remarks

1/ In the implementation we use the function $p_u(r) = \ell_u + r(u \in U, r \in \mathrm{N})$ where $\ell_u \in \mathrm{N}$ is a number depending on the parameters of user U (i.e. on how reliable he is in using his terminal time, how much CPU time he usually uses, etc.) The smaller ℓ_u is the better it is for user u.

For this special case theorem 2./. can be strengthened: It is true that there is essentially only one fair distribution. The following statements are true:

a/ There are numbers $d_u(u \in U)$ such that for all fair distribution d either $d(u) = d_u$ or $d(u) = d_u + 1$.

b/ There exists a partition $\{P_i : 1 \leq i \leq p\}$ of the users and a set of numbers $k_i : 1 \leq i \leq p$ so that $\ell_u + d_u$ is constant in each P_i, and in every fair distribution d :

$$|\{u \in P_i : d(u) = d_u + 1\}| = k_i \ (i = 1, \dots, p)$$

This means that only one hour may depend on luck and in every fair distribution the same number of "lucky" users are chosen from sets of equally favoured users. Differences of favours measured by $\ell_u + d_u$ are due to the problem structure and cannot be helped unless we take some hours from favoured users and give in to nobody, which is obviously not reasonable.

In this special case, the fair distributions give the lexicographical minimum (out of all distributions), of the sequence $d(u_1), d(u_2), \dots, d(u_m)$ with $d(u_1) \geq d(u_2) \geq \dots \geq d(u_m)\star$

2. In the general case both the set of distributions thatcan arise as the output of the algorithm and the set of fair distributions are equal to the local lexicographical minimums of the series

$$p_{u_1}(d(u_1)), \dots, p_{u_m}(d(u_m)) \ (p_{u_1}(d_{u_1}) \geq \dots \geq p_{u_m}(d_{u_m})$$

on the set of all distributions. Unfortunately, in general there are several local optima, but in the case $p_u(r) = \ell_u + r$, there is only one.

3. Algorithm II. can be replaced by a modification of algorithm I. that searches for improving alternating paths from users with minimal $p_u(d_T(u)) + p_u(d_T(u) + 1)^\star$. This remark makes possible in many cases to speed up the algorithm. (This is a remark of P. Kas').

$\star$ The remarks on lexicographical ordering and that on the alternative for algorithm II. (see 1. and 3.) are due to Péter Kas from the network flow group of the Computer and Automation Institute whom I am very grateful for his contributions.

4. Let us consider the functions $f : I_n \to \mathbb{R}$ where $I_n = \{x = (x_1, \dots, x_n) : \sum_1^n x_i = 1\}$, and suppose f is convex, symmetric and has its minimum in the point $(\frac{1}{n}, \dots, \frac{1}{n})$. Let

$$D = \left\{ \left(\frac{t_1}{\sum\limits_{u=1}^{m} t_u}, \dots, \frac{t_m}{\sum\limits_{u=1}^{m} t_u} \right) : t_u = p_u(d(u)) \text{and} \, d \, \text{and is a distribution} \right\}$$

Then the fair distributions minimize f on D in the case $p_u(r)\ell_u + r$ (and are local maximums in the general case.) Taking the special case $p_u(r) = r$ and $f(x_1, \dots, x_n) = -H(x_1, \dots, x_n)$ where H is the Shannon entropy or $f(x_1, \dots, x_n) = \sum\limits_{i \neq j}(x_i - x_j)^2$ we can interpret our result as the maximization of the entropy function or the minimization of the latter function on the set of distributions. This is another way of saying that the fair distributions distribute the time "the most equally" or in the "most balanced" way among users.

3. MODIFICATION OF TIMETABLES

In this part we suggest a method to modify existing timetables in order to satisfy additional constraints. In the implementation we use this method to produce several consecutive hours for users who need it.

If we have a set of pairs $(u_i, h_i)(i = 1, 2, \dots, s)$ so that $h_i \neq h_j$ $(i \neq j), (u_i, h_i) \in T$ $(i = 1, \dots, s)$ then we could look for permutations π of the set $\{1, \dots, s\}$ with $(u_{\pi(i)}, h_i) \in \Gamma(i = 1, \dots, s)$. Then $T' = (T \setminus \{(u_i, h_i) : i = 1, \dots, s\}) \cup \{(u_{\pi(i)}, h_i) : i = 1, \dots, s\}$ is also a timetable, and we are looking for T' with "better properties" than T. In the following heuristic arguments such a timetable T' will be called a permutation of the timetable T. The main idea is the following: we search among permutations of a timetable a more advantegeous timetable, but as general permutations would be too time consuming to consider, we restrict ourselves to cyclic permutations. Then we use cyclic permutations one after the other. Since all permutations can be written as the product of cyclic permutations, we restrict generality only by expecting from each cyclic permutation to improve the timetable, i.e. those "good" permutations which are not the product of "good" cyclic permutations are not conssidered.

We shall define a digraph on H that reflects the possible and the desired changes. Then we shall look for a cycle in this digraph. The cycle defines a cyclic permutation along which we effectuate the changes. Then we redefine the digraph, and begin the procedure again. The algorithm will end up when there are no more cycles in the defined digraph.

Such an approach in not without problems: on figure 3. e_1 represents a "desired" change and e_2 a "possible" change.

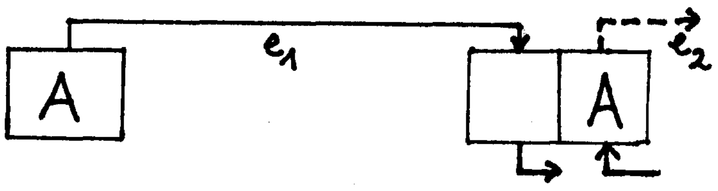

Figure 3.

If both e_1 and e_2 are contained in a permutation, and e_1 has the goal of making 2 consecutive hours for user A, then this goal is not reached. To avoid this kind of difficulty we shall require that to each user there should belong only one change in each permutation. We shall also require from the permuted timetable all the good properties of the original one. (e.g. optimality with respect to the given weight function.)

To describe a method that fulfills these requirements we shall need a more precise formulation of the problem.

Suppose a timetable T for the problem $P(G,r,t)$ and a weight function $w : U \times H \rightarrow \mathbf{N} \cup \{0\}$ are given (w is compatible with G).

Definition 3.1

We say that a digraph Ω belongs to user $u \in U$ (according to T) if $V(\Omega) \subset \Gamma(u) e = (h_1,h_2) \in E(\Omega)$

$$h_1 \in \Gamma_T(u)(\subset \Gamma_G(u)), h_2 \in \Gamma_G(u) \backslash \Gamma_T(u) \qquad (3.1.)$$

$$w(u,h_1) \leq w(u,h_2) \qquad (3.2)$$

((3.1.) means that according to T u works in the starting point of each $e \in E(\Omega)$ but does not work in the endpoint, and (3.2.) means that a change along e cannot decrease the objective value of u.)

To each user u we shall associate two digraphs that belong to him: a "wish – graph" and a "dotted graph". A wish – are (h_1,h_2) will represent a "required" change, i.e. one that improves the situation of u if "nothing else is changed for u". On the other hand, a dotted edge (h_1,h_2) belonging to u will mean that it is indifferent for u whether he works in hour h_1 or h_2 :

Definition 3.2

We say that *the pair of digraphs* W_u, D_u are a *wish graph and a dotted graph* belonging to $u \in U$ if:

a/ W_u and D_u belong to u according to T

b/ if $(a_1,b_1),(a_2,b_2) \in E(W_u) \cup E(D_u)$ and $w(u,a_1) = w(u,a_2)$, then
$(a_1,b_2),(a_2,b_1) \in E(W_u) \cup E(D_u)$

Moreover, if $(a_1,b_1) \in E(W_u)$ holds too, the either $(a_1,b_2) \in E(W_u)$ or $(a_2,b_1) \in E(W_u)$

The first part of b/ means that $W_u \cup D_u$ is the union of a complete directed bipartite graph and of isolated points.

The requirements of definition 3.2. are satisfied or "almost" satisfied in the applications that have the heuristic meaning we intended to describe: e.g. the second part of b/ requires that in case $(a_1,b_1),(a_2,b_2) \in E(_u) \cup E(D_u)$ either b_2 should be better than a_1 or b_1 should be better than a_2 for u, but this is quite natural because $(a_1,b_1) \in E(W_u)$ means either that a_1 is not good enough for u or b_1 is very good for u. On the other hand, a/ and b/ are very important conditions of the usability of our method:

Let for each $u \in U$ the pair of wish and dotted graphs W_u, D_u belonging to u be given, and denote $W = \bigcup_{u \in U} W_u$, $D = \bigcup_{u \in U} D_u$, $e \in E(W)$ will be called *wish – edge* and $e \in E(D)$ *dotted edge*. If $C = (x_1, \ldots, x_s)$ is a cycle and $C' = (x_i, x_{i+1}, \ldots, x_j)$ is also a cycle, then C' will be called a subcycle of C. Let $\Delta_u = W_u \cup D_u$, $\Delta = \bigcup_{u \in U} \Delta_u$. We state now the main result of this part of the paper:

Theorem 3.3

If T is optimal for $P(G,r,t)$ and w, then all cycles that contain at least one wish – edge have a subcycle that also has at least one wish – edge and at most one edge in each $\Delta_u (u \in U)$.

Corollary 3.4

The cycle that has the minimal number of edges among the cycles that have at least one wish edge has at most one edge in each Δ_u.

The proof of theorem 3.3. is based on lemma 3.5. below. Let us remark that to a walk $P(h_1, \ldots, h_s)$ in the digraph Δ there corresponds a *red – blue alternating walk* $(h_1, u_1, h_2, u_2, h_{u-1}, u_{s-1}, h_s)$ *where* $(h_i, h_{i+1}) \in \Delta u_i$. This follows from the fact that $\Delta = \bigcup_{u \in U} \Delta_u$ and thus $\exists u_i \in U : (h_i, h_{i+1}) \in \Delta_{u_i}$. On the other hand, " Δu_i belongs to u_i " means that $h_i \in \Gamma_T(u_i), h_{i+1} \in \Gamma_G(u_i) \backslash \Gamma_T(u_i)$ whence the walk is red – blue (see definition 3.2.).

Lemma 3.5

If T is an optimal timetable and $(h_1, u_1, h_2, u_2, \dots, u_{s-1}, h_s)$ is a closed red – blue alternating walk that satisfies

$$\begin{aligned} &w(h_i, u_i) \le w(u_i, h_{i+1}), \text{then} && (3.3.)\\ &u_i = u_j \quad \text{implies}\\ &w(h_i, u_i) = w(h_j, u_j) = w(u_i, h_{i+1}) = w(u_j, h_{j+1})\\ &(\text{ see figure 4.}) \end{aligned}$$

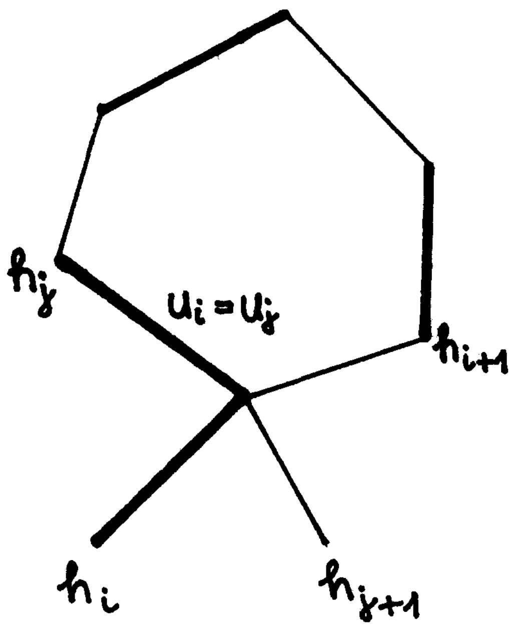

Figure 4.

PROOF. As T is optimal, (3.3.) is always satisfied with equality. Suppose indirectly that e. g. $w(h_i, u_i) = w(u_i, h_{i+1}) = w_1 < w_2 = w(h_j, u_j) = w(u_j, h_{j+1})$. We may suppose $i < j$ without restricting generality. Then $u_i = u_j$ implies that

$$P' = (u_i, h_{i+1}, u_{i+1}, h_{i+1}, \dots, h_j, u_j)$$

is also a closed alternating walk.Interchanging the colours of P' we increase the weight of the timetable, contradicting the optimality of T.

Proof of theorem 3.3

Let $C = (h_1, \dots, h_s)$ be a cycle in Δ that contains a wish – edge. If $\exists u \in U$ and $i < j$ such that

$$(h_i, h_{i+1}), (h_j, h_{j+1}) \in \Delta_u, \text{we prove that C} \qquad (3.4.)$$

has a proper subcycle in Δ that has a wish edge, and from this the statement follows.

We know that $(h_t, h_{t+1}) \in \Delta_(u_t)$ for some $u \in U$ $(t = 1, 2, \dots, s)$ and $(h_s, h_1) \in \Delta_{u_s}$ for some $u_s \in U$. Then $(h_1, u_1, h_2, u_2, \dots, h_s)$ is a closed alternating walk, and by our assumption (3.4.) we have $u_i = u_j = u$. We have also $w(h_t, u_t) \le w(u_t, h_{t+1})$ as Δ_t belongs to u_t (see (3.2.) in definition 3.1.). Thus the conditions of lemma 3.5. hold. By lemma 3.5. $w(u, h_i) = w(u, h_j)$. But then, *as* W_u *and* D_u *are a wish – graph and dotted graph, and* $\Delta_u = W_u \cup D_u$, we have by (3.4) and definition 3.2. that both $(h_i, h_{j+1}) \in E(\Delta_u)$ and $h_j, h_{j+1}) \in E(\Delta_u)$ hold. $C_1 = (h_1, \dots, h_i, h_{j+1}, \dots, h_s)$ and $C_2 = (h_{i+1}, \dots, h_j, h_{i+1})$ are subcycles of C and all edges of C, except C_1 or C_2. Thus if both $(h_i, h_{i=1})$ and (h_j, h_{j+1}) are dotted, then the wish - edge(s) of C is (are) contained by either C_1 or C_2 and the statement is proved. On the other hand, (h_i, h_{i+1}) and (h_j, h_{j+1}) satisfy (3.4.), so if at least one of them is a wish - edge, then by part b/ of definition 3.2. we know that at least one of (h_i, h_{j+1}) and $h_j, h_{i+1})$ is a wish - edge too. The former is contained by C_1 and the latter by C_2. thus either C_1 or C_2 (or both) contains a wish edge in any case.

Corollary 3.4. also provides an algorithm to construct suitable timetables: the minimal cycle that has a wish - edge is easy to construct with the help of any algorithm that is capable of finding the minimal path between two points of a graph (see e.g. [7.] or [8.]). We suggest an improving algorithm based on this fact:

Algorithm III

```
Step 0: Construct a timetable $T$, optimal for
        $P(G, r, t)$ and $w$.
        GO TO Step 1.
Step 1: Define the wish - graph $W_u$ and the dotted graph $D_u$ for each user $u \in U$ so that
        definition 3.2. should be satisfied. Determine $W = \bigcup_{u \in U} W_u, D = \bigcup_{u \in U} D_u$ and $\Delta = W \cup D$.
        GO TO Step 2.
Step 2: Look for a minimal cycle in $\Delta$ that has an edge in $W$.
        GO TO Step 3.
Step 3: If a cycle $C$ is found, GO TO Step 4.
        If there is no such cycle, STOP.
Step 4: Execute the changes corresponding to the arrows of $C$. Let $T$ be the new timetable.
        ($T$ arises by interchanging the colours on the corresponding alternating circle, see Corol-
lary
        3.4. and lemma 3.5.)
        GO TO Step 1.
```

Since T is an optimal timetable, and W_u, D_u belong to $u \in U$ (see definition 3.1.), in step 4. the new timetable is also optimal: it follows from (3.2.) and the optimality of T that actually $w(u_i, h_i) = w(h_i, u_{i+1})$ (see Lemma 3.5.). So at each occurance of Step 1. we have to decide again and again which are the arrows that express wishes, and which are those that are indifferent (are not inconvenient) for the users. This is the most time consuming step in the implementation (although its complexity is also bounded by $0((n+m)^3)$).

The algorithm stops when *there is no more improving cycle.* Althoungh we cannot tell anything stronger (e.g. design a quantity that would be optimized by this algorithm), we hope our approach will be justified in the next part of the paper: we have reason to think that even the simplest exact optimization requirements would probably require a much greater order of complexity.

4. MISCELLANEOUS

In this part we briefly summarize the analysis of different modifications and generalizations of the problem that arose in practice, and finally we tell a few words on the computational experience.

Complexity analisys has played a great role in the solution of our problem: here polynomiality and NP completeness have a very actual, practical meaning. An NP completeness proof means that we have reason to think that the methods we use are no more applicable for the considered modification or generalization. Thus, complexity analysis made possible to avoid losing time for intractable generalizations, and enabled us to concentrate on compromises between tractability and the number and quality of the fulfilled tasks.

We list now the various additional constraints we may want to require from our timetable:

1. Minimization of the number of one – hour work sessions or the minimizations or the minimization of the number of interruptions

These problems (i.e. the corresponding "decision problems") are NP – complete: several well – known NP – complete scheduling problems are easily reduced to simple special cases of these problems (E.g. in [10] SEQUENCING ON ONE PROCESSOR can be reduced to the special case $|T| = 1$.) So we content ourselves with the method described in part 3., which aims at reaching a state that cannot be improved by cyclic permutations. After several months of computational experience in real life use we can conclude that this approach completely satisfies the practical needs.

2. *Is it possible to expand the results to the more general model when to each $u \in U$ a system of sets $H_u = \{H_{(u,1)}, \dots, H(u, s_u)\}$ $(H_{u,j} \subset H)$ and a vector $r_u = (r_{(u,1)}, \dots, R_{u,s_u})$ is given that means that the request of u for $H_{(u,i)}$ is $r_{(u,i)}$?* (E.g. the question is wether requests of the following type are permitted: "I want 9 hours, 2 of which must be in the first part of the week, 4 hours while disc B is on, and the last 2 hours in the afternoon.) The SET PACKING problem can easily be reduced even to the special case $|U| = 2$.* (SET PACKING is the integer programme $Ax \leq 1,\ x \in \{0,1\}^n$, min $\sum_{i=1}^{n} x_i, A$ is a 0 - 1 matrix of size mxn. It is NP-complete, see [10].)

In practice, however, the special case

$$\forall u \in U : H_{u,i} \cap H_{u,j} = 0 \qquad (i \neq j) \tag{4.1}$$

is sufficient to be considered and this can be solved with our methods in polynomial time. It makes possible e.g. detailed requests for each day of the week or for mornings and afternoons, and each user may have his own division of the week to disjoint parts.

Let us remark that if we let a new user $u \in U$ $1 \leq i \leq s_u$ correspond to each pair (u, i), then * If $A = (a_{ij}mxn$ is a 0 - 1 matrix, denote $A_i = \{1 \leq j \leq n : a_{ij} = 1\}$ and let $U = \{u, v\}, H = \{1, 2, \dots, n\}, H_{u,i} = A_i, r_{u,i} = |A_i| - 1$ $(i = 1, \dots, m)$ and $H_v = H, r_v = c(c \in \mathbf{R}$ is a given number.) For these data a complete timetable exists if and only if there exists a solution x with $Ax = 1, \sum_{i=1}^{n} x_i = c$. the generalized problem is reduced to the original one. Of course, this transformation increases considerably the size of the problem. That is why we do not use this transformation, but we have modified the labelling technique instead, in such a way that it can hold such refined requests as well, without considerable increase in computing time.

3. *Some users may want to work in groups,* i.e. the users of the group do not want to have two terminals at a time (e.g. they cannot logon at the same time or prefer to have a terminal for much time instaed of many terminals for less time). If a user can participate in several groups, the problem is NP – complete (reduction is again from SET PACKING), but if each user belongs to at most one group, flow methods can be applied to this case case, and even to the generalization when the requirement is that each group must have at most ℓ terminals at a time. From a practical point of view, *disjoint groups* are completely satisfying.

4. *How to solve the problem when there is a difference between the terminals?* (This problem had to be solved, because there is a second terminal room with different terminals: teletypes and 2 kinds of displays.) If each user has a time request for each terminal, the problem is NP – complete. (It is the same as TIMETABLE DESIGN in [10].) But if each user has only one global time request and a subset of terminals equally convenient for him, then an extended network flow model solves the problem with the same order of complexity. We do not use, however, this approach, because it could cause injustice in the distribution of terminals. We use user – terminal weights instead, for each $h \in H$ and solve a weighted user – terminal marriage problem for each $h \in H$. The weights are carefully defined, and we also take into consideration that users do not want to change terminals if they work for several hours.

5. *Disks:* In the IBM 3031 computing centre of the H.A.S. the quantity of disks and disk – drives is an even tighter bottleneck than the quantity of terminals, they are also affected by the embargo restrictions. Most users store their programmes on disks that are on line only when one

of their owners has a terminal. There are two disk – drives and many disks, and each disk gives storage room for many users. As there are five terminals, the coordination of the disks with the terminal time of their owners may cause problems. In each time peroid several users using the same disk should work. Therefore a disk – time timetable has to be constructed.

If the user – time timetable τ is ready, the disktime problem is the same as the user – time one: if disk d is used by users $U_d \subset U$, then define

$$w(d,h) = |\{u \in U_d : \tau(u,h) = 1\}| \tag{4.2.}$$

and

$$r(d) = \left| \bigcup_{u \in (U)} \{h : \tau(u,h) = 1\} \right| \tag{4.3.}$$

Disks play the role of users and disk – drives the role of terminals, and the weights $w(d,h)$ are chosen so that an optimal timetable δ for this problem gives the exact maximum of the total number of coordinated triples, i.e. of triples (u,d,h) with

$$\tau(u,h) = 1,\ \delta(d,h) = 1,\ h \in U_d.$$

However, it is not reasonable to construct a user – time timetable first: this timetable will not try to put to the same time periods users who need the same disk. On the other hand, the disk – time timetable can not be made before the user – time timetable, since $w(d,h)$ and $r(d)$ cannot be defined without τ.

We use the compromise of a first stochastic user – time timetable, then a disk – time timetable with

$$w(d,h) = M(|\{u \in U_d : \tau(u,h) = 1\}|) \tag{4.4.}$$

$$r(d) = M(|\bigcup_{u \in U_d} \{h : \tau(u,h) = 1\}|) \tag{4.5.}$$

where $\tau(u,h)$ is a random variable, and M denotes the mean value. The random variable τ is defined by supposing that for user $u \in U$ all subsets of $\Gamma(u)$ having $r(u)$ elements are equally probable (or the probability is proportional with the weight of the sets). From this supposition the mean values (4.4.) and (4.5.) can be calculated by elementary means, and a first disk timetable δ is constructed with these data. Then we modify the weight function so that $w(u,h)$ has to be increased if there exists $d \in D$ with $u \in U_d$ and $\delta(d,h) = 1$ (but still we have to be careful, see 6. below). Then we construct the optimal timetable τ for $P(U,H,\Gamma,r,t)$ and $w.\tau$ redefines the data of the disks with (4.2.) and (4.3.), so at the end of all, we construct the definitive disk – timetable with these new data.

6. *Definition of the weight function* : In the implementation the users can design there sorts of weights for each hour: 1/ "I am not free in hour h ", 2/ "I am free in hour h ", 3/ "Hour h is very convenient for me". The weights themselves cannot be confined to the users as it would cause "inflation" . We define the weight function so that the optimal solution must be maximal, too, i.e. optimal for the weight function

$$w_0(u,h) = \begin{cases} 1 & \text{if } w(u,h) > 0 \ \ (\text{i.e. } (u,h) \in E(G)) \\ 0 & \text{if } w(u,h) = 0 \ \ (\text{i.e. } (u,h) \notin E(G)) \end{cases}$$

as well. We do this because we think that the number of hours the users work at terminals should not depend on the weight function: it would not be reasonable if a user who has great weights received more time.

It is easy to see that if we want the users to have s different weights, then if the weights are chosen from the set

$$\{s|U|.|H|,\ s|U|.|H|+1, \dots ,\ s|U|.|H|+s-1\},$$

our conditions are satisfied: an optimal solution is optimal for w_0 too. Here $|U|.|H|$ can be replaced by $max\{|U|,|H|\}$. (This remark is proved by analyzing the algorithm, and is useful in reducing the running time of the primal – dual method that solves the weighted version of the problem (see chapter 1.))

IMPLEMENTATION AND COMPUTATIONAL EXPERIENCE:

An implemantation of the results was coded *in APL language* under CMS operating system on the IBM 3031 computer, and for the IBM 3031 computing centre of the H. A. S. László Király, from the Software Department of the institute, prepared user interfaces which make comfortable for users to tell the necessary information and read the timetable.

The output of the whole system is a user – terminal/time timetable and a user/time timetable for a week's use.

The input of the algorithm is provided by L. Király's programme, which – besides the data given by the users – takes into consideration the data concerning the computing centre (the working hours of the computer, the number of working terminals, etc) and some data of users provided by a monitor programme that enables the calculation of the " ℓ_u " – s (see part 2.).

The transformation of the timetable to make several consecutive hours for users who ask for it (see part 3.) takes some minutes of CPU – time, all the rest of the algorithm runs for less than one minute. The behaviour of the algorithm in practice seems to be linear in the number of users and quadratic in time.

Last but not least, I wuold like to thank my collegues in the discrete programming group of the Department of Applited Mathematics, and P.Kas fot their useful remarks. I am mostly indebted to László Király from the user support group of the IBM 3031 who helped this work with his questions and ideas made possible the realization of the results.

REFERENCES

1. D. de Werra, Construction of School Timetables by Flow Methods, *Infor – Canad. J. Operational Res. and Information Processing* 9, 12–22 (1971).
2. I.W.A. Horn, Some Simple Scheduling Algorithms, *Naval Res. Logist. Quart.* 21, 177–185 (1974).
3. S. Even, A. Itai, A. Shamir, On the Complexity of Timetable and Multicommodity Flow Problems, *SIAM J. Comput.* 5 No. 4., 691–703 (1976).
4. R.L. Graham, E.L. Lawler, J.K. Lenstra, A.H.G. Rinooy Kan, Optimization and Approximation in Deterministic Sequencing and Scheduling, *A Survey, Annals of Discrete Math. 5.*, 287–326 (1979).
5. J.G. Schmidt, T. Ströhlein, Timetable constructionan annotataed bibliography, *The Computer Journal* 23 No. 4., 307–316.
6. J. M. Mulvey, A Classroom /Time Assignment Model, *European J. of Operational Research* 9, 64–70 (1982).
7. L.R. Ford Jr., O.R. Fulkerson, Flows in Networks, *Princeton University Press* (1962).
8. E.L. Lawler, Combinatorial Optimization: Networks and Matroids, *Holt Rinehart and Winston* (1976).
9. L. Lovász Combinatorial Problems and Exercises, *Akadémiai Kiadó* (1979).
10. Garey, Johnson, Computers and Intractability, *W. H. Freeman and Co.* (1979).