SCHEDULING JOBS ON SEVERAL MACHINES WITH THE JOB SPLITTING PROPERTY

Paolo Serafini
Department of Mathematics and Computer Science
University of Udine, Italy

ABSTRACT: This scheduling model is derived from the real problem of scheduling looms in a textile industry. Jobs may be independently split over several specified machines and preemption is allowed. Deadlines are specified for each job and jobs are assumed to be available. It is shown that minimizing maximum weighted tardiness can be done in polynomial time. The case of uniform machines (as in the considered application) can be modeled as a network flow and minimization of maximum tardiness can be done in strongly polynomial time. The case of unrelated machines can be solved either by generalized flow techniques or by Linear Programming. Attention is also focused on the problem of finding so-called Unordered Lexico Optima, in order to schedule non-binding jobs as early as possible.

This paper deals with a scheduling problem that arises in the production of different types of fabric in a textile industry. The problem is characterized by the presence of several machines (the looms) on which several jobs have to be processed (the articles to be woven). Each job is associated to a definite subset of compatible machines on which it can be processed. A peculiar feature of the problem is that each job can be split arbitrarily and processed independently on these machines. Preemption is also allowed. Each job has a deadline and the objective is to minimize the maximum tardiness or the maximum weighted tardiness in case the jobs have been assigned different weights.

It turns out that this problem is solvable in polynomial time. In the general case of unrelated machines, i.e. machines with different speeds for different jobs, the problem can be solved by Linear Programming or by generalized network flow techniques. In the case of uniform machines, which has a practical relevance in the mentioned application, a network flow model can be developed with algorithms based on max flow computations. In general the proposed algorithms are weakly polynomial. However, the case of uniform machines and equal weights can be solved even in strongly polynomial time.

Minimizing the maximum tardiness may provide optimal solutions with non-binding jobs scheduled later than necessary. In order to schedule all jobs as early as possible we will address also the problem of finding so-called Unordered Lexico optimal solutions.

In our model there are no release dates, since all known jobs are assumed to be available for processing. This assumption is usually met in practice. Of course, it always happens in the real production process that new jobs, previously not known, enter the process before the current production is completed. Since preemption is allowed it is enough to recompute from scratch the schedule each time a new job is available. This situation corresponds to the one described as 'on-line' by Labetoulle et al. (1984).

Apparently this specific problem has not been investigated in the literature. There are papers (like Horn 1974, Labetoulle et al. 1984, Slowinski 1984, 1988, Blazewicz, Drabowski and Weglarz 1986 and Federgruen and Groenevelt 1986) dealing with parallel machines and independent jobs. However, the special possibility of independent job splitting is not present in these papers and this makes the problem different. For instance Horn (1974) allows job splitting without simultaneous processing on two machines of the same job. The same type of constraint is considered by Labetoulle et al. (1984) and Federgruen and Groenevelt (1986) where each job can be processed on at most one machine at a time and this possibility combined with preemption is like the job splitting considered by Horn.

Moreover the quoted papers make the assumption of uniform machines whereas we also address the general problem of unrelated machines. Clearly, the possibility of a polynomial-time algorithm also for the general case is due to the independent job splitting property.

As will be shown network flow techniques may be efficiently applied to the case of uniform machines. The idea of using these techniques to solve scheduling problems is not new. For instance Horn (1974), Labetoulle et al. (1984) and Federgruen and Groenevelt (1986) use them in a way which has some similarities with our approach. However, since the problem we address is different, the derivation of the results is simpler.

As mentioned at the outset this paper originated from the practical need to schedule looms in a textile factory. Although we prefer to focus on the properties of the abstract scheduling model we provide a brief description of the real production process in Section 1, so that the reader can attach a concrete meaning to the various features of the abstract model.

The rest of the paper is organized as follows: in Section 2 we describe and characterize the problem mathematically; then we investigate first the case of uniform machines in Sections 3 and 4. Section 3 is devoted to the design of algorithms for the maximum tardiness problem whereas Section 4 is devoted to the problem of finding Unordered Lexico optimal solutions. Then Section 5 deals with the general case of unrelated machines.

1. A BRIEF DESCRIPTION OF THE LOOM SCHEDULING PROBLEM

The scheduling problem presented in this paper originated in a few textile factories in Northern Italy. However, its features can be considered typical of most textile factories, since loom technology differs very little from plant to plant.

In general the scheduling manager has to face two problems. First he must assess meaningful delivery dates in making contracts with customers. Then he has to decide how to divide the production of all his articles among the compatible looms and when to schedule these partial productions in order to respect the delivery dates as much as possible. Both problems can be reduced to the same scheduling problem.

The technological constraints which must be taken into account will now be explained. In principle there is no restriction on the assignment of articles to looms since each article can be woven by any loom. However, each loom must be equipped with a warp chain and a particular article can be produced only by the looms equipped with a warp chain belonging to a definite set of warp chains.

The production of an article requires a definite number of beats per length unit (a beat corresponds to the insertion of the weft yarn). Each loom has a standard productivity which is measured in beats per minute and is computed statistically, taking into account the idle times and the weft breaks (see below).

The time required to produce a certain length of fabric on a certain loom is immediately derived from these data. From this type of computation it turns out that the looms are nearly uniform (see next section) so that it is more convenient to adopt the model described in Sections 3 and 4.

When the yarn of the warp chain is finished, it takes only a few hours to replace a chain of the same type, while it takes a few days to replace a chain of a different type, and during these times the loom is idle. So the latter operation is performed as rarely as possible (typically once in several months) and is regarded as an exceptional event which is not taken into consideration in normal current planning. Since we are interested in providing a model for current planning we do not address the problem of changing the chain type. In the situations we have seen the current plan is computed once in a month either manually or with the aid of a spreadsheet. This way it takes one full day to produce a new plan. In order to increase productivity and also to assist in deciding the delivery dates it would certainly pay to produce these plans more frequently with the aid of a faster computing support.

Switching from weaving one article to weaving another one on the same loom can be accomplished by simply moving a bar or a similar device. So there is virtually no set-up time.

The production of an article can be split over any set of compatible looms, as long as the length of a single piece of article produced on each loom is not shorter than 200 or 300 meters, which is a quantity typically much shorter than a usual order length (several thousand meters). This is a constraint of a disjunctive type (either no production or at least a certain quantity) and introducing it explicitly into the model calls for integer variables with all the related complications. We disregard this constraint and assume it is possible to preempt and split jobs without any limitation since we expect that solutions which do not violate the constraint are found in most cases, due to the large ratio between the quantity to be produced and the minimum admissible length. If they do violate, we may think of adjusting the schedule a posteriori, for instance by excluding a loom from the subset compatible with the article.

As far as the deadlines are concerned it is important to observe that deadlines are assigned in terms of weeks and in practice shorter time units are not considered. Therefore the idle times caused by chain replacements correspond to a small fraction of production time and can be simply taken into account in the standard productivity. This simplification is justified both by the fact that the error introduced this way is comparable to the data error and also by the computational tractability of the resulting mathematical model.

The main objective in the production process is to respect the deadlines as much as possible, also considering that not all articles have the same importance. Therefore it is quite realistic to minimize the maximum weighted tardiness. This allows the manager also to revise a solution, if necessary, by simply adjusting the weights. If the total weighted tardiness were chosen as an objective, this would result in an NP-hard model with no appreciable gain in the accuracy. Therefore we have opted for objectives of maximum tardiness type.

Another feature of the real production process is that each loom is subject to frequent and very short breakdowns due to the weft break. They last from a few seconds to a few minutes. Experienced workers are assigned to the immediate repair of the weft. Since these breaks are evenly distributed in time it is better to take into account these very short idle times in the computation of loom productivity.

As a matter of fact the real problem raised by the weft breaks is another. It is obviously necessary that all looms in production are attended by some workers. This means that not all looms can be used if there is not sufficient manpower and only particular subsets, according to their spatial distribution within the factory, may be attended and put into production.

This constraint constitutes a severe challenge for a mathematical programming modelling. Not only does the added combinatorial feature make the problem much harder, but due to the many factors involved in the decision it seems very difficult to formalize all of them within a mathematical programming model. It

seems more reasonable to leave this decision to the sole expertise of the manager aided possibly by partial solutions offered by the model.

The algorithms designed for the uniform machine case have been used in a prototype decision support system which has been tested by the manager of one of the factories. The algorithmic core of the DSS worked very promptly, as expected, since it is based on max flow computations, and the schedules provided by the algorithms were approved by the manager. Then there was the need to visualize all data and design a customized interactive interface, but problems of this kind are out of the scope of this paper.

2. PROBLEM MODELLING

The following quantities constitute the data of the problem:

$$
\begin{aligned}
M &:= \text{set of machines,}\\
M^j &:= \text{set of machines on which job } j \text{ may be processed,}\\
J &:= \text{set of jobs,}\\
J^m &:= \text{set of jobs which can be processed on machine } m,\\
n &:= \sum_j |M^j| = \sum_m |J^m| \le |M| \cdot |J| = \text{ the 'size' of the problem,}\\
q_j &:= \text{quantity to be processed by job } j,\\
d_j &:= \text{deadline of job } j,\\
r_{jm} &:= \text{time required for machine } m \text{ to process a unit quantity of job } j,\\
\nu &:= \text{scheduling time unit.}
\end{aligned}
$$

According to the values of the parameters r_{jm} we may distinguish the following three cases:

1. The machines are identical, that is $r_{jm} =: r_j$, $\forall m \in M^j$;
2. The machines are uniform, that is
$$\frac{r_{jm}}{r_{jh}} = \frac{r_{km}}{r_{kh}} =: \rho_{mh} = \frac{1}{\rho_{hm}} \qquad \forall m, h \in M^j \cap M^k, \quad \forall j, k \in J;$$
3. The machines are unrelated.

These assumptions fall into the usual taxonomy of multi-machine scheduling (see Lawler 1983). Just note that for uniform machines the quantities r_{jm} can be easily extended to all pairs (j, m) even if $j \notin J^m$. (In fact in some cases they cannot be extended, but this is because the problem is split into independent problems with fewer jobs and machines).

We have already remarked that in the loom scheduling problem the machines can be considered uniform. Hence greater emphasis is given to this case in the paper. However, the general case is also discussed for the sake of completeness.

The particular type of process we are considering allows both for job preemption (i.e. any job can be interrupted at any time to process another job with no set-up time) and job splitting (i.e. any job can be split and processed independently on different machines).

In this scheduling model there are no release dates since all jobs are known and available for processing. This is not a serious limitation. It is simple to extend this model to allow new jobs to become available in later times. The possibility of preempting jobs allows the schedule to be recomputated from scratch.

Finally, the unknown quantities which characterize a problem solution are:

$$\begin{aligned} x_{jm} &:= \text{quantity of job } j \text{ assigned to machine } m, \\ C_{jm} &:= \text{completion time of quantity } x_{jm}, \\ C_j := \max_m C_{jm} &:= \text{ completion time of job } j. \end{aligned}$$

Let us note that C_{jm} is left undefined if $x_{jm} = 0$. In this case we may conventionally assign $C_{jm} := 0$. In order to evaluate a schedule we consider the following quantities:

$$\begin{aligned} T_j &:= \max\{0, C_j - d_j\} &&:= \text{ the tardiness of job } j, \\ T_* &:= \max_j T_j &&:= \text{ the maximum tardiness}, \\ T_*^w &:= \max_j w_j T_j &&:= \text{ the maximum weighted tardiness}, \end{aligned}$$

where the w_j's are positive weights assigned to the jobs according to their perceived importance. Let us also denote by Problem T_* and Problem T_*^w the problems of minimizing the respective quantities.

By schedules *compatible* with certain deadlines we mean schedules with completion times within the stated deadlines. So schedules compatible with d_j are schedules with tardiness equal to zero.

In general we may also consider monotonically non-decreasing cost functions with respect to the completion times for each job $f_j(C_j)$. We may also assume quite reasonably that they are lower-semicontinuous. Let us define the inverse functions as $f_k^{-1}(z_k) := \sup\{C : f_k(C) \leq z_k\}$. Note that the inverse functions are monotonically non decreasing and upper-semicontinuous.

We denote by T^f the problem of finding a schedule whose cost achieves a stated goal $z \in R^{|J|}$, i.e. a schedule with completion times C_j such that $f_j(C_j) \leq z_j$, $\forall j$; and by T_*^f the problem of minimizing $\max_j f_j(C_j)$. Note that finding solutions with values T_* or T_*^w not larger than a stated quantity is a particular case of Problem T^f.

Any minimal solution of Problem T_*, Problem T_*^w or Problem T_*^f is called *optimum*. However, we note that optima may not be satisfactory solutions. Among optima there usually exist schedules whose completion times C_j are dominated by the completion times C'_j of some other optimal schedule, in the sense that $C_j \geq C'_j$, with strict inequality for at least one j. Clearly we prefer non-dominated optima.

Still, non-dominated optima may not be satisfactory solutions either. Among them there are many solutions which favor one job too much with respect to the others and these solutions may not be acceptable. Our attitude is to give highest priority to the criterion of minimizing the maximum tardiness (or any other monotone function of it). Once this objective is reached we may then consider minimizing the maximum tardiness of those jobs whose completion times can be anticipated. This procedure is repeated recursively until all jobs have been assigned definite completion times.

The solution found this way is obviously optimal and non-dominated. Solutions of this type have been introduced in Game Theory under the name of Nucleolus of the Game by Maschler, Peleg and Shapley (1979) although the concept seems to be older (see Maschler 1992, p. 610, for a general presentation). They also appeared in the framework of Mathematical Programming under a different name, namely Unordered Lexico Optima (see Schrage 1991, p. 298). This is the term we shall use in this paper.

Formally these particular optima can be defined in the following way: given a vector $a \in R^n$ let $\theta(a) \in R^n$ be the vector obtained from a by permuting its entries so that the entries in $\theta(a)$ are arranged in non-increasing order (in case of equal entries break the tie in any fixed way). Then given two vectors $a, b \in R^n$ we say that a is *Unordered Lexico better* than b if $\theta(a)$ is lexicographically smaller than $\theta(b)$, i.e. there exists an integer k such that $\theta_i(a) = \theta_i(b)$ for $i < k$ and $\theta_k(a) < \theta_k(b)$. A vector $a \in A \subset R^n$ is an Unordered Lexico Optimum in A if there is no $b \in A$ which is Unordered Lexico better than a.

Then Unordered Lexico optimal schedules can be defined by considering the vectors T, or wT or $f(C)$ according to the chosen optimality criterion. This is a robust optimality concept for our problem taking into account its multi-objective characteristics. Of course the decision maker may be not satisfied with a particular Unordered Lexico optimal schedule either, but this is because he is giving the jobs different priorities. In this case we should remodel the problem with different weights .

As far as the scheduling process is concerned, we may note that there is no advantage in preempting a given job on a given machine, because all jobs are simultaneously available. So the quantity x_{jm} is always processed without interruption and we may speak of the processing sequence, or permutation, of jobs on each machine.

Let $\pi := \{\pi^m\}_{m\in M}$ denote permutations of jobs on the machines (in general the permutations are not equal on each machine). Thus π^m is a particular permutation of the set J^m such that the job j takes the π^m_j-th place. Let $\hat{\pi}^m := (\pi^m)^{-1}$ be the inverse permutation, i.e. $\hat{\pi}^m_{\pi^m_j} = j$, so that $\hat{\pi}^m_h$ is the job taking the h-th place. In order to have a shorthand notation for the job which follows or immediately precedes another job in a certain permutation on a particular machine, if (k, m) is a pair denoting job k on machine m, let

$$(k^+, m) = (\hat{\pi}^m_{\pi^m_k+1}, m) \quad \text{and} \quad (k^-, m) = (\hat{\pi}^m_{\pi^m_k-1}, m)$$

be the pairs denoting respectively the jobs immediately following and preceding job k on machine m. Let

$$J^m_k(\pi) := \left\{j \in J^m : \pi^m_j \le \pi^m_k\right\} = \left\{j \in J^m : j = \hat{\pi}^m_h,\, h \le k\right\},$$

i.e. the set consisting of the jobs preceding job k on machine m plus job k itself. For any given permutation π the following relationships hold, if $x_{km} > 0$:

$$\sum_{j\in J^m_k(\pi)} r_{jm}x_{jm} = C_{km}(\pi)\,, \qquad \forall k \in J^m, \quad \forall m \in M, \tag{1}$$

where the dependence of the completion time C_{km} on π has been emphasized. The following constraints must be observed in order to produce the stated quantities for the jobs:

$$\begin{aligned} \sum_{m\in M^j} x_{jm} &= q_j\,, \qquad \forall j \in J \\ x_{jm} &\ge 0\,, \qquad \forall j, m. \end{aligned} \tag{2}$$

For arbitrary deadlines $\bar{d}_j$, $j \in J$, we may write from (1)

$$\sum_{j\in J^m_k(\pi)} r_{jm}x_{jm} \le \bar{d}_k \qquad \forall k \in J^m, \quad \forall m \in M, \tag{3}$$

constraining a schedule (under a given sequence π) to be completed within the deadlines $\bar{d}_j$. From (3) it is immediate to derive the following constraints concerning Problems T_*, T^w_* and T^f respectively:

$$\sum_{j\in J^m_k(\pi)} r_{jm}x_{jm} \le d_k + T_* \qquad \forall k \in J^m, \quad \forall m \in M, \tag{4}$$

constraining a schedule (under a given sequence π) to have maximum tardiness value not larger than T_*,

$$\sum_{j\in J^m_k(\pi)} r_{jm}x_{jm} \le d_k + \frac{T^w_*}{w_k} \qquad \forall k \in J^m, \quad \forall m \in M, \tag{5}$$

constraining a schedule (under a given sequence π) to have maximum weighted tardiness value not larger than T_*^w, and

$$\sum_{j \in J_k^m(\pi)} r_{jm} x_{jm} \leq f_k^{-1}(z_k) \qquad \forall k \in J^m, \quad \forall m \in M, \tag{6}$$

constraining a schedule (under a given sequence π) to meet the goals z_j, or if $z_j = T_*^f$, $\forall j$, to have maximum cost not larger than T_*^f.

A crucial point consists in the choice of π. For the kind of problems we are dealing with the choice of π is based on the following result:

Lemma 1: *Given arbitrary deadlines $\bar{d}_j$, for each assignment x_{jm} there is a permutation π^m on machine m, given by non-decreasing deadlines $\bar{d}_j$, which minimizes* $\max_{j \in J^m} \left\{C_{jm} - \bar{d}_j\right\}$.

Proof: On each machine m we deal with a One Machine Problem with processing times x_{jm}, $j \in J^m$. It is a well known result (see Lawler 1983) that tardiness minimization is accomplished by scheduling the tasks in order of non-decreasing deadlines. ∎

Then we immediately derive that:

Theorem 2: *Given arbitrary deadlines $\bar{d}_j$, there exists a schedule compatible with them, if and only if the constraints (2) and (3) are feasible with respect to permutations π^m given by non-decreasing deadlines $\bar{d}_j$.*

Proof: The thesis derives in a straightforward way from the lemma. The only detail which has to be cleared concerns the fact that the inequality $\sum_{j \in J_k^m(\pi)} r_{jm} x_{jm} \leq \bar{d}_k$ should be removed from (3) if $x_{km} = 0$. However, due to the hypothesis on the permutation this constraint is redundant in this case, since it is not more binding than the one given by the job $h = k^-$ immediately preceding job k on machine j. For if $x_{km} = 0$

$$\sum_{j \in J_k^m(\pi)} r_{jm} x_{jm} = \sum_{j \in J_h^m(\pi)} r_{jm} x_{jm} \leq \bar{d}_h \leq \bar{d}_k.$$

The argument can be applied recursively if $x_{hm} = 0$ as well. So we may consider the whole set of inequalities (3) without worrying about the actual values x_{km}. ∎

Then the following corollaries follow easily.

Corollary 3: *There exists a schedule with maximum tardiness value not larger than T_* if and only if the constraints (2) and (4) are feasible with respect to a permutation π given by non-decreasing deadlines.* ∎

Corollary 4: *There exists a schedule with maximum weighted tardiness value not larger than T_*^w if and only if the constraints (2) and (5) are feasible with respect to a permutation π given by non-decreasing values $(d_k + T_*^w / w_k)$.* ∎

Corollary 5: *There exists a schedule meeting the goals z_k if and only if the constraints (2) and (6) are feasible with respect to a permutation π given by non-decreasing values $f_k^{-1}(z_k)$.* ∎

In the case of uniform machines the constraints (2) and (3) can be rewritten in a form which allows a network flow modelization. By definition of uniform machines we have

$$C_{km}(\pi) = \sum_{j \in J_k^m(\pi)} r_{jm} x_{jm} = \sum_{j \in J_k^m(\pi)} r_{j1} \rho_{m1} x_{jm}$$

and by defining $\xi_{jm} := r_{j1} x_{jm}$ we have

$$C_{km}(\pi) = \sum_{j \in J_k^m(\pi)} \rho_{m1} \xi_{jm} = \rho_{m1} \sum_{j \in J_k^m(\pi)} \xi_{jm} = \frac{1}{\rho_{1m}} \sum_{j \in J_k^m(\pi)} \xi_{jm},$$

whence

$$\sum_{j \in J_k^m(\pi)} \xi_{jm} = \rho_{1m} C_{km}(\pi) \qquad \forall k \in J^m, \quad \forall m \in M, \tag{7}$$

and the inequalities (3) can be rewritten as

$$\sum_{j \in J_k^m(\pi)} \xi_{jm} \leq \rho_{1m} \bar{d}_k \qquad \forall k \in J^m, \quad \forall m \in M. \tag{8}$$

From (2) we may write

$$\begin{aligned} \sum_{m \in M^j} \xi_{jm} &= r_{j1} q_j \qquad \forall j \in J, \\ \xi_{jm} &\geq 0 \qquad\quad \forall j \in J^m, \quad \forall m \in M, \end{aligned} \tag{9}$$

so that the constraints (8) and (9) which replace the constraints (3) and (2) for the case of uniform machines exhibit only sums of variables.

3. TARDINESS MINIMIZATION FOR UNIFORM MACHINES

Although it is possible to solve both Problem T_* and T_*^w by Linear Programming as we will show in Section 5, it is more convenient in the case of uniform machines to exploit the particular structure embodied in the constraints (8) and (9) and to build a network flow model for which faster algorithms may be designed with more compact data structures.

For each permutation π we build the following network $G(\pi) = (N, E(\pi))$: there are $|J|$ source nodes, denoted $s(j)$, $j \in J$, one sink node t and nodes $n(j,m)$, $\forall j \in J^m$, $\forall m \in M$ (or equivalently $\forall m \in M^j$, $\forall j \in J$).
There is a set of arcs, called *vertical arcs*, independent of the permutation π, given by

$$(s(j), n(j,m)),\ \forall m \in M^j,\ \forall j \in J,$$

and a set of arcs, called *horizontal arcs*, dependent on π given by

$$(n(k,m)\,,\, n(k^+,m)), \quad k = \hat{\pi}_h^m, \quad h := 1, \ldots, |J^m|, \quad \forall m \in M,$$

where we have conventionally denoted $n(k^+,m) := t$, if $k = \hat{\pi}_{|J^m|}^m$, $\forall m$. The horizontal arcs form $|M|$ chains of arcs. Note that the number of both nodes and arcs is $O(n)$.

The flow must be non-negative on all arcs. Each node $s(j)$ sends a flow equal to $r_{j1} q_j$, which is split into quantities ξ_{jm} on the vertical arcs $(s(j), n(j,m))$, thus taking care of the constraints (9). On each horizontal arc $(n(k,m)\,,\, n(k^+,m))$ the flow amounts to $\sum_{j \in J_k^m(\pi)} \xi_{jm} =: \zeta_{km}$. So by suitably upper bounding the flow

ζ_{km} on the horizontal arcs with a capacity c_{km}, we may take care of the inequalities (8). Finally all flow goes into the sink t.

In Figure 1 a small example is provided with four machines labeled A, B, C and D and six jobs, labeled 1 through 6. The four horizontal chains correspond to the sequences of jobs on the machines and to the horizontal arcs. The four sets J^m are $J^A := \{1,4,5,6\}$, $J^B := \{2,3,4\}$, $J^C := \{1,2,4,6\}$, $J^D := \{3,4,5,6\}$. The deadlines are $d_1 := 50$, $d_2 := 70$, $d_3 := 100$, $d_4 := 150$, $d_5 := 200$, $d_6 := 300$, and the quantities to be processed are $q_1 := 100$, $q_2 := 100$, $q_3 := 50$, $q_4 := 100$, $q_5 := 500$, $q_6 := 210$; furthermore $r_{jm} := 1$, $\forall j, m$. The permutation on each machine is given in order of increasing deadlines. In the sequel we shall use this example to illustrate the procedures.

For given capacities c_{km} a simple way to find a feasible flow on $G(\pi)$ is to add a supersource s and arcs $(s, s(j))$, $j \in J$, with capacities $r_{j1}q_j$. Let $\hat{G}(\pi)$ be the enlarged network. Then if the max flow $s \to t$ on $\hat{G}(\pi)$ has value $F := \sum_{j\in J} r_{j1}q_j$ this flow is also feasible for $G(\pi)$. Otherwise there is no feasible flow. Let us denote by $MF(n)$ the polynomial time complexity bound for a max flow algorithm applied to $\hat{G}(\pi)$.

Therefore in order to solve Problem T^f we have simply to set $c_{km} := \rho_{1m} f^{-1}(z_k)$ with permutation π given by non-decreasing values $f^{-1}(z_k)$ and solve a max flow problem on $\hat{G}(\pi)$. So we have:

Theorem 6: *Problem T^f can be solved in time $O(MF(n))$ if the machines are uniform.* ∎

Now we are going to derive a general scheme based on binary search in order to solve Problems T_*, T_*^w and T_*^f for the uniform machines case. This scheme leads to weakly polynomial algorithms. However, Problem T_* can also be solved in strongly polynomial time by a different method as will be shown later. Let us first deal with Problem T_*.

If we fix π as the permutation with non-decreasing deadlines and set the capacities as $c_{km} := \rho_{1m}(d_k+T)$, the feasible flow ξ (if any) corresponds to assignments $x_{jm} := \xi_{jm}/r_{j1}$ for which the maximum tardiness value is not larger than T. This fact suggests we can find the minimum maximum tardiness through a series of max flow computations in a binary search fashion. The search is carried out in the interval $[0, U]$ with U the trivial upper bound $U := \max_m(\sum_j r_{jm}q_j) - \min_j d_j$ and it is stopped within an approximation ν. So we have:

Theorem 7: *Problem T_* can be solved by binary search with approximation ν in time $O(MF(n)\ \log(U/\nu))$ if the machines are uniform.* ∎

The solution obtained through the binary search is approximated, but it is possible to refine it to optimality. Although there is quite often no practical need to have a solution more refined than the actual scheduling time unit ν, we do need exact optimality for algorithmic purposes in order to find Unordered Lexico Optima as will be clear later.

Let $\tilde{T}_*$ be the optimal tardiness. Let Q be any source-sink cut in $\hat{G}$. For a given value of T the capacity $c(Q)$ of the cut Q can be expressed as $c(Q) = \alpha(Q) + \beta(Q)T$ with $\alpha(Q) \geq 0$ and $\beta(Q) \geq 0$ values depending only on the cut Q. So the function $F(T)$ expressing the max flow value with respect to T, i.e.

$$F(T) := \min_Q \alpha(Q) + \beta(Q)T, \tag{10}$$

is an increasing piecewise linear concave function for $T \leq \tilde{T}_*$ and constant with value F for $T \geq \tilde{T}_*$. Therefore for values $T < \tilde{T}_*$ sufficiently close to $\tilde{T}_*$ there is a minimal cut $\tilde{Q}$ which is also minimal for $\tilde{T}_*$. Hence the following equality must hold

$$\alpha(\tilde{Q}) + \beta(\tilde{Q})\tilde{T}_* = F$$

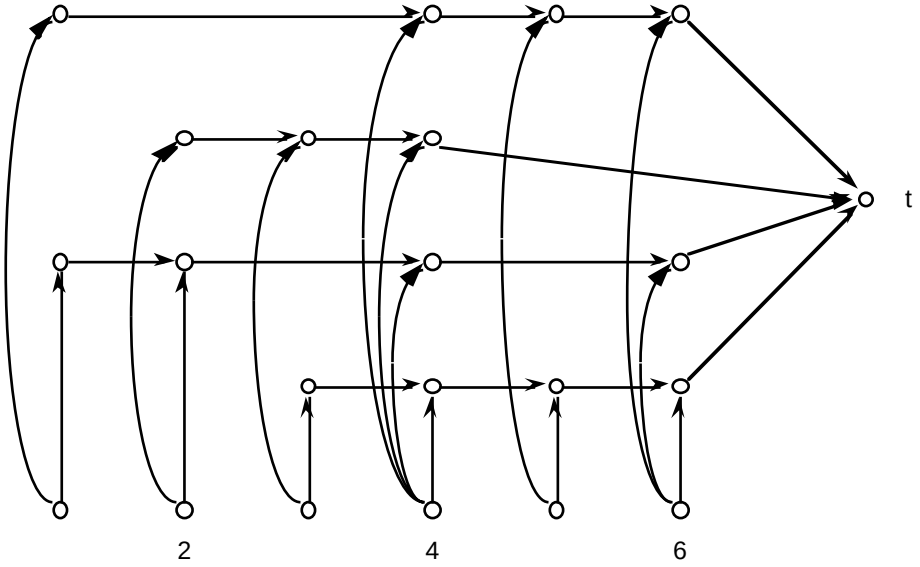

Figure 1

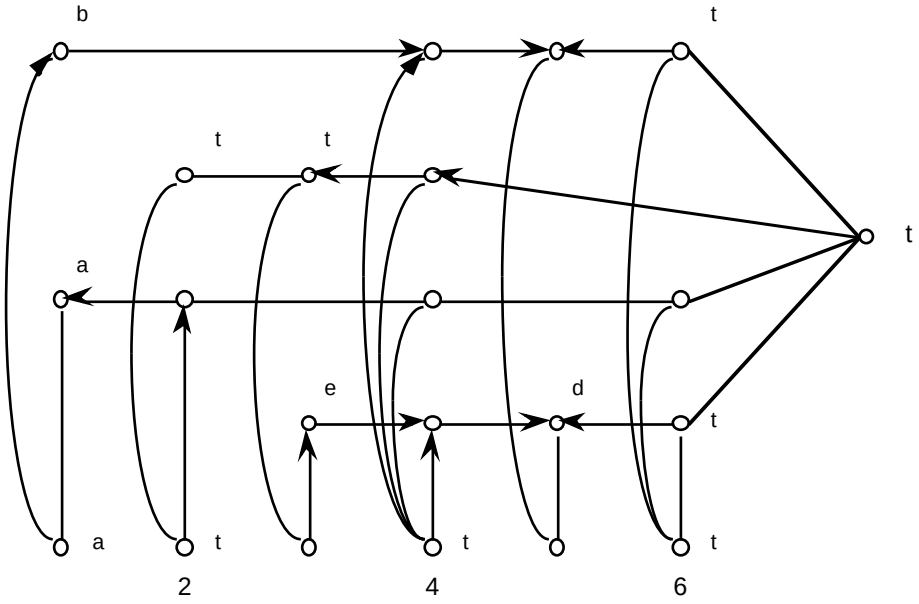

Figure 2

from which

$$\tilde{T}_* := \frac{F - \alpha(\tilde{Q})}{\beta(\tilde{Q})}. \tag{11}$$

If the binary search is executed with a sufficiently small value for ν then the cut output by the max flow routine for the left extreme of the final search interval is precisely $\tilde{Q}$. In this case a simple computation gives $\alpha(\tilde{Q})$ and $\beta(\tilde{Q})$ and $\tilde{T}_*$ is immediately computed from (11).

However, we must check whether the obtained cut is indeed $\tilde{Q}$. If not the value $\tilde{T}_*$ given by (11) is less than the optimal one. In order to check the optimality of $\tilde{T}_*$ it is enough to run the max flow routine again with this value. If the max flow f is equal to F this is a proof of the optimality of $\tilde{T}_*$. If $f < F$ then ν was chosen not small enough and we have to proceed in the binary search until the stated optimality condition is satisfied. These further computations still have polynomial time complexity as shown in the following theorem where L is the maximum size of any problem input number:

Theorem 8: *Problem T_* can be solved exactly by binary search in time $O(MF(n)\ (L+\log n))$ if the machines are uniform.*

Proof: Let us note that the breakpoints of $F(T)$ correspond exactly to the T-coordinates, in the space $R \times R^n$ of the variables T and ξ_{jm}, of some vertices of the polyhedron defined by

$$\begin{array}{lll}
\displaystyle\sum_{m \in M^j} \xi_{jm} \leq r_{j1} q_j & \forall j \in J, & \\
\displaystyle\sum_{j \in J_k^m(\pi)} \xi_{jm} - \rho_{1m} T \leq \rho_{1m} d_k & \forall k \in J^m & \forall m \in M, \\
\xi_{jm} \geq 0 & \forall j \in J^m & \forall m \in M.
\end{array} \tag{12}$$

Hence the T-coordinate of any vertex is obtained by solving a linear system of $(n+1)$ equations from (12). All entries in this submatrix are either 0 or 1 except for the T-column. By Cramer's rule the value of T is given by the ratio of two determinants, and both determinants can be computed by expanding along the T-column. So they are given by sums of problem input numbers times the determinants of 0-1 $n \times n$ submatrices. Since these submatrices are totally unimodular, as is not difficult to see, the size (i.e. the number of its digits) of T is $O(L + \log n)$. Therefore also the difference between two distinct breakpoints T' and T'' is bounded by the same quantity, i.e. $-\log |T' - T''| = O(L + \log n)$ which combined with the previous results proves the thesis. ∎

Let us remark that we could just start the binary search without any approximation value ν and instead perform at every step the optimality check previously described. However, this would increase the number of max flow problems to be solved. From a practical point of view it is better to take a value for ν which experience has shown to be good and only then to check optimality at every step.

An alternative approach to find $\tilde{T}_*$ is given by the following recursion which exploits the property that $F(T)$ is piecewise linear and concave: starting with the value $T := 0$, compute the capacities c_{km}, then the max flow f and the minimal cut Q; if $\beta(Q) > 0$ reset $T := (F - \alpha(Q))/\beta(Q)$; if $f = F$ or $\beta(Q) = 0$ stop, else repeat the loop with the new T value. We call this procedure *monotone search.*

The number of iterations of the monotone search depends on the number of breakpoints of $F(T)$. We are going to show that there are at most $|M|$ breakpoints. First we need two lemmas. The first lemma is a known fact, already stated in Hu (1970). A very short proof can also be found in Picard and Queyranne (1980). Here we provide an alternative proof. Given a cut Q let $S(Q)$ denote the subset of nodes induced by the cut Q and containing the source.

Lemma 9: *If Q' and Q'' are two optimal cuts in a Max Flow Problem, then the cuts induced by the node subsets $S(Q') \cap S(Q'')$ and $S(Q') \cup S(Q'')$ are also optimal.*

Proof: Let $Q^{\cap}$ and $Q^{\cup}$ be the cuts corresponding to the node subsets $S(Q') \cap S(Q'')$ and $S(Q') \cup S(Q'')$ respectively. The capacity of a cut viewed as a set function over subsets of nodes is submodular, i.e. $c(Q^{\cup}) + c(Q^{\cap}) \leq c(Q') + c(Q'')$. Since by minimality we have $c(Q') = c(Q'') \leq c(Q^{\cap})$ and similarly for $c(Q^{\cup})$, by comparing the inequalities we get $c(Q') = c(Q^{\cup}) = c(Q^{\cap})$. ∎

Lemma 10: *At most one saturated horizontal arc for each horizontal chain is contained in an optimal cut.*

Proof: Let us suppose there are two saturated horizontal arcs from the same chain in an optimal cutset Q, denoted as $(n(j,m), n(j^+,m))$ and $(n(k,m), n(k^+,m))$ with job j preceding job k. Then $n(j,m) \in S(Q)$, $n(j^+,m) \notin S(Q)$ and similarly for $n(k,m)$. Obviously $n(j^+,m) \neq n(k,m)$. Then the path $n(k,m) \rightarrow n(j^+,m)$ along the chain has at least one arc and at least one arch of this path must traverse the cut in the backward direction. Then it must have zero flow. But the arc $(n(j,m), n(j^+,m))$ is saturated and this implies a positive flow on all arcs of the chain following $(n(j,m), n(j^+,m))$. ∎

Theorem 11: *Problem T_* can be solved exactly by monotone search in time $O(|M|\ MF(n))$ if the machines are uniform.*

Proof: Let us first note that we may freely restrict the set of cuts in (10) to those cuts which are optimal for some value T. Hence from Lemma 10 we may write $\beta(Q) = \sum_{m \in M'} \rho_{1m}$ for some subset M' of machines depending on the cut Q, which depends in turn on the value of T for which it is optimal. Next we want to show that there are at most $|M|$ subsets M' since they are ordered by inclusion.

Let $\hat{T}$ be a breakpoint of the function $F(T)$. This means that there exist two cuts Q^1 and Q^2 such that for a sufficiently small $\varepsilon > 0$

$$\begin{aligned} \alpha(Q^1) + \beta(Q^1)(\hat{T} - \varepsilon) &= \min_Q \alpha(Q) + \beta(Q)(\hat{T} - \varepsilon), \\ \alpha(Q^2) + \beta(Q^2)(\hat{T} + \varepsilon) &= \min_Q \alpha(Q) + \beta(Q)(\hat{T} + \varepsilon) \end{aligned} \tag{13}$$

and

$$\alpha(Q^1) + \beta(Q^1)\hat{T} = \alpha(Q^2) + \beta(Q^2)\hat{T} = \min_Q \alpha(Q) + \beta(Q)\hat{T},$$

with $\beta(Q^i) = \sum_{m \in M^i} \rho_{1m}$, and $M^1 \neq M^2$ otherwise $\hat{T}$ is not a breakpoint. From Lemma 9 also the cut $\hat{Q}$ such that $S(\hat{Q}) = S(Q^1) \cap S(Q^2)$ is optimal for $T = \hat{T}$, so that

$$\alpha(Q^2) + \beta(Q^2)\hat{T} = \alpha(\hat{Q}) + \beta(\hat{Q})\hat{T}. \tag{14}$$

Clearly $\beta(\hat{Q}) = \sum_{m \in \hat{M}} \rho_{1m}$ with $\hat{M} \subset M^1 \cap M^2$. Hence $\hat{M} \subset M^2$. The inclusion cannot be strict otherwise $\beta(\hat{Q}) < \beta(Q^2)$ and this together with (14) contradicts (13). Therefore $\hat{M} = M^2$, whence $M^2 \subset M^1$ with strict inclusion. Therefore the breakpoints correspond to subsets ordered by strict inclusion and so there can be at most M of them. ∎

Techniques based on binary search can be applied to Problem T_*^w as well. The main difference is that the permutation on each machine is not necessarily given a priori by non-decreasing deadlines. Due to the presence of weights the optimal permutation depends on the optimal assignments x_{jm}, which in turn depend on the optimal permutation. In order to overcome this difficulty we may fix several different target values of maximum weighted tardiness, thereby implicitly fixing deadlines which must not be violated by any job. Clearly the best way to find a solution compatible with these deadlines is to sort these values and to schedule the jobs in this order, according to Theorem 2. Then we may use again a binary search technique to find an approximated solution. Note that the permutations may change in each iteration of the binary search, which requires rebuilding the network at each step. However, this extra work is dominated by the max flow computation. A trivial upper bound on the optimal value of weighted tardiness can be given by $\bar{U} := \max_j w_j \cdot U$. So we have :

Theorem 12: *Problem T_*^w can be solved by binary search with approximation ν in time $O(MF(n)\ \log(\bar{U}/\nu))$ if the machines are uniform.* ∎

In order to refine the solution to optimality we may use the previous ideas. There is one important difference. The function $F(T)$ is not necessarily concave since in this case

$$F(T) = \max_{\pi} F_\pi(T) \tag{15}$$

with $F_\pi(T)$ defined as in (10) with respect to the permutation π. In this case we also have to consider breakpoints given by those values of T for which the ordering of jobs changes. We call these breakpoints *permutation breakpoints* to distinguish them from the previous ones which we call *cutset breakpoints*. At permutation breakpoints which do not coincide with cutset breakpoints the slope of $F(T)$ increases as it is apparent from (15) and illustrated in the following example: there are jobs 1 and 2 and machines a, b and c. Job 1 is processed on machines a and b, and job 2 is processed on machines a and c. Furthermore $d_1 := 40$, $q_1 := 200$, $w_1 := 1$, $d_2 := 60$, $q_2 := 100$, $w_2 := 2$ and $r_{jm} := 1$. At the value $T = 40$ the function $F(T)$ has a permutation breakpoint due to the fact that job 2 exchanges priority with job 1. It is easy to see that $F(T) = 160 + 2T$ for $0 \le T \le 40$, $F(T) = 140 + 2.5T$ for $40 \le T \le 64$ and $F = 300$ for $T \ge 64$. The value $T = 64$ is clearly optimal.

However, within intervals given by adjacent permutation breakpoints, $F(T)$ is concave with possible cutset breakpoints. Since the permutation breakpoints can be computed in advance in time $O(|J|^2)$ as

$$\hat{T}_{hk} := \frac{w_h w_k}{w_h - w_k}(d_h - d_k) \qquad \forall h, k \in J : w_h \neq w_k, \tag{16}$$

it is easy to take care of them.

Since the size of the distance between two successive permutation breakpoints is $O(L)$ as evident from (16), arguments similar to the ones used in Theorem 8 show that

Theorem13: *Problem T_*^w can be solved exactly by binary search in time $O(MF(n)\ (L+\log n))$ if the machines are uniform.* ∎

The arguments used in Theorem 11 cannot be applied to Problem T_*^w. In this case $\beta(Q)$ is given by $\sum_{m \in M'} \rho_{1m}/w_k$ where the job index k depends, for each machine m, on the particular cut Q. This added feature makes it difficult to give a polynomial bound to the number of cutset breakpoints within two adjacent permutation breakpoints.

Theorem 12 extends immediately to Problem T_*^f. We may suppose that the functions f_j are coded so that an upper bound $\hat{U}$ on the optimal value has polynomial size with respect to the size of the encoding of the functions f_j. So we may write:

Theorem 14: *Problem T_*^f can be solved by binary search with approximation ν in time $O(MF(n)\ \log(\hat{U}/\nu))$ if the machines are uniform.* ∎

4. FINDING UNORDERED LEXICO OPTIMA FOR UNIFORM MACHINES

We first consider Problem T_*. Let us suppose Problem T_* has been solved with optimal value $\hat{T} = \max_j(\hat{C}_j - d_j)$ and with corresponding capacities $\hat{c}_{km}$. As remarked in Section 2 this solution has the best possible tardiness value but it may be not satisfactory because some jobs which are not binding at the optimal value may have any completion time $\hat{C}_j \leq d_j + \hat{T}$, without affecting the value of the objective function.

For instance let us consider an optimal solution for the previous example with the following values for x_{jm}:

	1	2	3	4	5	6
A	0	-	-	0	250	70
B	-	100	50	50	-	-
C	100	0	-	50	-	70
D	-	-	0	0	250	70

and completion times $C_1 = 100$, $C_2 = 100$, $C_3 = 150$, $C_4 = 200$, $C_5 = 250$ and $C_6 = 320$, tardiness values $T_1 = 50$, $T_2 = 30$, $T_3 = 50$, $T_4 = 50$, $T_5 = 50$, $T_6 = 20$ and maximum tardiness $\hat{T} = 50$. This solution is clearly optimal as can be seen by inspection of how job 5 is scheduled on machines A and D. However, it is not apparent whether also jobs 1, 3 and 4, whose tardiness value is equal to the maximum tardiness, have the smallest possible completion times.

Therefore we now address the problem of identifying the binding jobs. Clearly jobs for which $\hat{C}_j < d_j + \hat{T}$ are not binding. However, there may be jobs with $\hat{C}_j = d_j + \hat{T}$ and yet not binding. We note that due to the particular way of saturating the arcs of some max flow algorithms this is not a rare possibility. It is also clear that a job is binding if and only if at least one of its horizontal arcs is saturated and in a minimal cut. The problem is that there may be several minimal cuts and identifying all minimal cuts is computationally expensive in general. However, we may use a result of Picard and Queyranne (1980), which allows arcs belonging to minimal cuts to be identified quickly without necessarily enumerating all minimal cuts.

Given a max flow solution, we build the residual graph and compute its strongly connected components, by also contracting to t the set of all vertices which can reach t (in our case there are no vertices reachable from s since all arcs $(s, s(j))$ are saturated). A saturated arc has its endpoints in two different strongly connected components if and only if it belongs to some minimal cut (Corollary 6 in Picard and Queyranne 1980). So it is only matter of finding the strongly connected components of the residual graph, which can be done in $O(n)$ time (the network has $O(n)$ arcs).

After the binding jobs have been identified an algorithm minimizing the maximum tardiness is restarted with the capacities related to the binding jobs held fixed to the last found value. The procedure is repeated recursively until all jobs are binding. So we may conclude:

Theorem 15: *If the machines are uniform, Unordered Lexico optimal solutions for Problem T_* can be found either by monotone search in time $O(n\,MF(n))$ or by binary search in time $O(|J|\,MF(n)\,(L + \log(n)))$.*

Proof: First note that the overall procedure alternates between the process of finding an optimal solution and the one of identifying binding jobs. Obviously there are at most $|J|$ repetitions and the overall cost of finding

optimal solutions is $O(|J|\,|M|\,MF(n)) = O(n\,MF(n))$ by monotone search and $O(|J|\,MF(n)\,(L + \log(n)))$ by binary search. The overall computation of finding the binding jobs costs $O(n|J|)$. Clearly the first computation dominates the second one. ∎

Considering the example again it may be seen from Figure 2 (the strongly connected components are identified by a lower-case letter near a node) that jobs 1 and 5 are binding, so that the tardiness mimimization can be restarted by varying the capacities only for jobs 2,3,4 and 6. This gives an optimal value $T_* = 40$ with binding jobs 2 and 3. Then we get $T_* = 25$ with job 4 binding. Finally job 6 can be completed even before the deadline. The final Unordered Lexico optimal solution has the following x_{jm} values:

	1	2	3	4	5	6
A	0	-	-	0	250	45
B	-	90	50	35	-	-
C	100	10	-	65	-	120
D	-	-	0	0	250	45

with completion times $C_1 = 100$, $C_2 = 110$, $C_3 = 140$, $C_4 = 175$, $C_5 = 250$ and $C_6 = 295$, tardiness values $T_1 = 50$, $T_2 = 40$, $T_3 = 40$, $T_4 = 25$, $T_5 = 50$, $T_6 = 0$ and maximum tardiness obviously $T_* = 50$.

The procedure extends in a straightforward way to Problem T_*^w. The only additional feature we must pay attention concerns the case in which the optimal value T_*^w is also a permutation breakpoint. In this case the search for Unordered Lexico optimal solutions must be performed with the permutation immediately to the left of T_*^w.

As far as Problem T_*^f is concerned, the identification of the binding jobs cannot be carried out along the previous lines if the functions $f_j(C_j)$ are not one-to-one. In this case, since f_j^{-1} is upper-semicontinuous, an infinitesimal decrease of the cost at a discontinuity point turns into a finite decrease of the completion time. We do not pursue this problem here.

5. TARDINESS MINIMIZATION FOR UNRELATED MACHINES

For unrelated machines there seems to be no way of expressing the constraints (2) and (3) as network flow constraints. We propose two alternative approaches: the first one is based on generalized network flow techniques and the second one on linear programming.

We only sketch the first approach. In generalized networks the flow entering an arc is allowed to leave the same arc multiplied by a gain factor. In our case the parameters r_{jm} are gain factors to be attached to the vertical arcs $(s(j), n(j,m))$. Physically they represent conversion factors from job quantity to time. On all other arcs the gains are equal to one.

The techniques developed by Goldberg, Plotkin and Tardos (1991) can be applied to our case. In order to fit exactly into the framework of Goldberg, Plotkin and Tardos (1991) we reverse the orientation of all arcs (so that the gains of the vertical arcs $(n(j,m), s(j))$ are changed to $1/r_{jm}$) and add one arc from the source to the sink with a very large gain. Since the source is the only node allowed to violate flow conservation, this large gain has the effect of almost canceling the flow coming out of the source (in the new reversed orientation), so that the excess at the source is almost entirely given by the flow entering into the source, and it is precisely this the quantity we wish to maximize, as in the case of uniform machines.

Then we are looking for a generalized circulation which maximizes the excess at the source. This problem, called the Generalized Circulation Problem, can be used for the binary search in the same way the

Max Flow Problem was used in the uniform machine case. Just note that our network is already restricted (in the terminology of Goldberg, Plotkin and Tardos 1991).

There are weakly polynomial algorithms which solve the Generalized Circulation Problem. The exact tardiness value can be obtained by rounding the solution obtained from the binary search once a sufficient degree of approximation is reached, since the problem is linear. Note however that the approximation required in this case is much stronger than in the uniform case, for which total unimodularity of the constraint matrix can be exploited.

Then generalized flow techniques can also be used to find the binding jobs in a way similar to the uniform case. In this case if there exists a generalized augmenting path to a node $s(j)$ then the job j is not binding. We recall that, when applied to our special network, a generalized augmenting path is either a residual directed path $t \rightarrow s(j)$ or a residual directed cycle with gain larger than one plus a residual directed path from a node in the cycle to $s(j)$. For instance the gain of the cycle $n(i,a) \rightarrow s(i) \rightarrow n(i,b) \rightarrow n(j,b) \rightarrow s(j) \rightarrow n(j,c) \rightarrow n(k,c) \rightarrow s(k) \rightarrow n(k,a) \rightarrow n(i,a)$ (only cycles of this type can be flow augmenting) can be expressed as

$$\frac{r_{ib}}{r_{ia}} \frac{r_{j,c}}{r_{j,b}} \frac{r_{k,a}}{r_{k,c}}$$

and it is clear that this expression is always equal to one for uniform machines. In the general case it is possible to exploit the lack of uniformity of the machines in order to get more job quantity within the same time limits if the above expression is larger than one. We recall that discovering a generalized augmenting path reduces to finding a negative length cycle with arc lengths computed by the negative of the logarithms of the gains. Hence the overall computation of finding Unordered Lexico optimal solutions is polynomial.

As a second approach we suggest using Linear Programming techniques. The disadvantage of using LP techniques is that it is not possible to exploit the efficient data structures typical of networks. The constraint matrix is rather sparse. On the other hand the LP formulation of the problem allows for direct minimization of the tardiness without resorting to the trick of binary search.

We assume without loss of generality that the optimal tardiness is strictly positive. It is a simple matter to check whether it is zero. In this case the techniques described in the sequel can be applied to minimize the maximum lateness (i.e. $\max_j(C_j - d_j)$).

In general Problem T_* can be solved exactly via the following Linear Programming problem with π given by non-decreasing deadlines:

$$\begin{array}{lll} \min & T & \\ \displaystyle\sum_{j \in J_k^m(\pi)} & r_{jm} x_{jm} \leq d_k + T & \forall k \in J^m, \quad \forall m \in M, \\ \displaystyle\sum_{m \in M^j} & x_{jm} = q_j & \forall j \in J, \\ & x_{jm} \geq 0 & \forall j, m. \end{array} \tag{17}$$

The situation is more complicated for Problem T_*^w because the optimal permutation π is not known in advance. The approach we adopt for Problem T_*^w in the case of unrelated machines is also based on iterating guessed solutions. The difference is that we do not guess tardiness values, rather we guess permutations. By doing so we need only a strongly polynomial number of iterations (however, since only weakly polynomial algorithms are known to date for Linear Programming, the overall algorithm remains weakly polynomial).

Given a target value T of weighted tardiness we associate to it the permutation $P(T)$ given by non-decreasing values $d_k + T/w_k$. For some values of T the permutation is not uniquely defined because two or more values may coincide. These are the permutation breakpoints previously introduced. If T is a permutation breakpoint the tie is solved by ordering for increasing values w_k. No other tie is possible unless

both w_k and d_k coincide and in this case the order between these jobs is irrelevant. The tie can be broken in a fixed arbitrary way. Alternatively $P(T)$ for a breakpoint T can be defined as $P(T-\varepsilon)$ where $\varepsilon > 0$ is chosen so that the interval $(T(\pi)-\varepsilon, T(\pi))$ does not contain breakpoints.

There are at most $O(|J|^2)$ breakpoints and they define a set of adjacent intervals (more exactly the number of breakpoints is given by the sum over the jobs of the number of steps one job has to move upwards in passing from its place in the permutation given by increasing deadlines to the new place in the permutation given by decreasing weights). Let $\Pi := \{P(T)\}_T$. Π can be naturally ordered as the corresponding intervals, so that we may write $\pi > \pi'$ for any two permutations $\pi, \pi' \in \Pi$. Clearly $|\Pi| = O(|J|^2)$.

Conversely given a permutation π we may define a tardiness value $T(\pi)$ as follows

$$\begin{array}{lll} T(\pi) := \min & T & \\ \displaystyle\sum_{j \in J_k^m(\pi)} & r_{jm}x_{jm} \leq d_k + \dfrac{T}{w_k} & \forall k \in J^m, \quad \forall m \in M, \\ \displaystyle\sum_{m \in M^j} & x_{jm} = q_j & \forall j \in J, \\ & x_{jm} \geq 0 & \forall j, m. \end{array} \tag{18}$$

Let $X(\pi, T)$ be the set of feasible x_{jm} in (18) for a given permutation π and tardiness T. Note that if $X(\pi, T) \neq \emptyset$ then $X(\pi, T') \neq \emptyset$ for all $T' \geq T$. Now we may define the following composite map

$$\pi \mapsto \Psi(\pi) := P(T(\pi))$$

which has the following properties.

Theorem 16: $\Psi(\pi) \in \Pi$, $\forall \pi$. ∎

Theorem 17: *Let $\pi \in \Pi$. Then π is the optimal permutation and $T(\pi)$ is the optimal maximum weighted tardiness if and only if $\Psi(\pi) = \pi$.*

Proof: Suppose $\pi = \Psi(\pi)$. By definition of Ψ we have $P(T(\pi) - \eta) = \pi$ for $0 \leq \eta \leq \varepsilon_1$ and for a sufficiently small $\varepsilon_1 > 0$. Suppose there exists π' such that $T(\pi') = T(\pi) - \varepsilon_2$, $\varepsilon_2 > 0$. Let $\varepsilon := \min\{\varepsilon_1, \varepsilon_2\}$. Hence $X(\pi', T(\pi) - \varepsilon) \neq \emptyset$ whereas $X(\pi, T(\pi) - \varepsilon) = \emptyset$. In other words, for the same target value $T(\pi) - \varepsilon$ there exist a compatible schedule for π' and there exists none for π. However, for this target value π should be the best permutation to find a compatible schedule according to Theorem 2. From the contradiction the sufficiency follows. The necessity is trivial if we only note that in Π the optimal permutation is unique (it is here that the assumption of strictly positive tardiness plays a crucial role). ∎

Theorem 18: $T(\Psi(\pi)) \leq T(\pi)$.

Proof: If $T(\pi)$ is not a permutation breakpoint then, since $X(\pi, T(\pi))$ is not empty and $P(T(\pi))$ is a better permutation than π for the target value $T(\pi)$ then $X(\Psi(\pi), T(\pi))$ is not empty as well and therefore $T(\Psi(\pi)) \leq T(\pi)$. If it is a breakpoint, then either $X(\Psi(\pi), T(\pi) - \varepsilon) \neq \emptyset$ and the thesis follows or $X(\Psi(\pi), T(\pi) - \varepsilon) = \emptyset$ for all $\varepsilon > 0$ but $X(\Psi(\pi), T(\pi)) \neq \emptyset$ because $X(\Psi(\pi), T(\pi)) = X(\pi, T(\pi))$ due to the fact that $T(\pi)$ is a breakpoint and the thesis follows with equality. ∎

Theorem 19: *Let $\pi, \pi^* \in \Pi$ with π^* the optimal permutation. If $\Psi(\pi) > \pi$ then $\pi^* > \pi$ and if $\Psi(\pi) < \pi$ then $\pi^* < \pi$.*

Proof: Clearly $\pi^* \neq \pi$ for both statements otherwise there would be a contradiction between the hypotheses and the results of Theorem 17. So suppose $\pi^* < \pi$. This means $T' \geq T(\pi^*)$ for any T' such that $P(T') = \pi$. Since $X(\pi^*, T(\pi^*)) \neq \emptyset$ by optimality we also have $X(\pi^*, T') \neq \emptyset$. By Theorem 2, $X(\pi, T') \neq \emptyset$ whence $T(\pi) \leq T'$ and $\Psi(\pi) \leq \pi$. By contradiction the first statement is proved. The second statement is a simple consequence of Theorem 18, from which $\pi^* \leq \Psi(\pi)$ always holds. ∎

Two alternative algorithms may be derived from these results. The first one is very simple and performs a monotone search of the optimal permutation: starting with an arbitrary permutation π (not necessarily in Π), iterate $\pi := \Psi(\pi)$ until $\pi = \Psi(\pi)$. The optimality of the final permutation is guaranteed by Theorem 17, the finiteness of the method is guaranteed by the monotone condition expressed by Theorem 18 and the strongly polynomial number of iterations by Theorem 16. Note that there is no need to compute the permutation breakpoints in advance. If we denote with $LP(n)$ the computational complexity of solving the LP problem (18) and assume that this computation dominates the sorting of the values $d_k + T/w_k$, then the optimal maximum weighted tardiness can be found in time $O(|J|^2\ LP(n))$.

As a second algorithm we may perform a binary search over the permutations in Π by using the results of Theorem 19. Note that the binary search can be speeded up by resetting the right-hand-side permutation as $\Psi(\pi)$ instead of π when the optimal permutation is on the left of the intermediate guess π. In order to start the binary search we need to compute and to sort the permutation breakpoints, which takes $O(|J|^2\ \log|J|)$ time. Then we only need $O(\log|J|\ LP(n))$ time to compute the optimal solution. So we have proven:

Theorem 20: *Problem T^w_* can be solved exactly in time $O(|J|^2\ \log|J| +\ LP(n)\ \log|J|)$ for unrelated machines.* ∎

The algorithm based on binary search is theoretically better than the monotone search. However, it may be argued that the upper bound of $|J|^2$ iterations is rarely met in practice and only a few iterations are needed. In addition, since the monotone search need not compute the breakpoints it may be worth considering.

We also note that Problem T^f_* can be solved with the same reasoning as Problem T^w_* if the functions f_j are well behaved, in the sense that it is possible to bound the number of breakpoints. If this is not possible it is better to approach the problem through a binary search on guessed goals.

The problem of finding Unordered Lexico Optima is somewhat simplified if we use the LP approach. Since the LP problem solves the tardiness problem directly, the optimal dual variables of this problem are directly linked to the binding jobs. It is clear that jobs whose corresponding dual variables are strictly positive are binding and so we may fix the tardiness values for these jobs and solve a new LP problem. It is also possible to show that the new LP problem has more strictly positive optimal dual variables than the previous one, so that we can repeat the procedure recursively. For Problem T^w_* we also have to take care of the permutations.

6. CONCLUSIONS

By abstracting from the features of the real process of several looms weaving different types of fabric a particular scheduling problem has been defined and thoroughly investigated. The main difference between this and other multi machine scheduling models is that it allows jobs to be split over several machines with independent processing. Due to this possibility it has been possible to derive polynomial algorithms.

In particular the problem with uniform machines, which closely corresponds to the real application, can be solved via the Max Flow problem. Due to the efficient data structures and time bounds these algorithms can be easily implemented in a system to support the decisions of the scheduling manager.

REFERENCES

Blazewicz, J., M. Drabowski and J. Weglarz. 1986. Scheduling Multiprocessor Tasks to Minimize Schedule Length. *IEEE Trans. on Computers C-35* **5**, 389-393.

Federgruen, A. and H. Groenevelt 1986. Preemptive Scheduling of Uniform Machines by Ordinary Network Flow Techniques. *Management Science* **32**, 341-349.

Goldberg, A.V., S.A. Plotkin and E. Tardos 1991. Combinatorial Algorithms for the Generalized Circulation Problem. *Mathematics of Operations Research* **16**, 351-381.

Horn, W.A. 1974. Some Simple Scheduling Algorithms. *Naval Res. Logist. Quart.* **21**, 177-185.

Hu, T.C. 1970. *Integer Programming and Network Flows.* Addison-Wesley, Reading, Ma.

Labetoulle, J., E.L. Lawler, J.K. Lenstra and A.H.G. Rinnooy Kan 1984. Preemptive Scheduling of Uniform Machines Subject to Release Dates. In *Progress in Combinatorial Optimization*, W.R. Pulleyblank (ed.). Academic Press, New York, 245-261.

Lawler E. 1983. Recent results in the Theory of Machine Scheduling. In *Mathematical Programming: The State of the Art*, A. Bachem, M. Grötschel and B. Korte (eds.). Springer, Berlin, 202-234.

Maschler, M. 1992. The Bargaining Set, Kernel, and Nucleolus. In *Handbook of Game Theory, with Economic Applications, Vol. I*, R.J. Aumann, and S. Hart (eds). North Holland, Amsterdam, 591-667.

Maschler, M., B. Peleg and L.S. Shapley 1979. Geometric Properties of the Kernel, Nucleoulus, and Related Solution Concepts. *Mathematics of Operations Research* **4**, 303-338.

Picard, J.C. and M. Queyranne 1980. On the Structure of all Minimum Cuts in a Network and Applications. *Mathematical Programming Study* **13**, 8-16.

Schrage, L. 1991. *LINDO: An Optimization Modeling System, Fourth Edition.* The Scientific Press, San Francisco, Cal..

Slowinski, R. 1984 . Preemptive scheduling of independent jobs on parallel machines subject to financial constraints. *European J.of Operational Research* **15**, 366-373.

Slowinski, R. 1988 . Production scheduling on parallel machines subject to staircase demands. *Eng. Costs and Prod. Economics* **14**, 11-17.