On the Solution of 2-Machine Flow Shop Problems With a Common Due Date [1]

Jatinder N. D. Gupta
Department of Management, Ball State University,
Muncie, IN 47306, USA
email: jgupta@bsu.edu

Volker Lauff and Frank Werner
Otto-von-Guericke-Universität, Fakultät für Mathematik, PSF 4120,
39016 Magdeburg, Germany
email: volker.lauff@mathematik.uni-magdeburg.de and frank.werner@mathematik.uni-magdeburg.de

Keywords: Scheduling, flow shop, nonregular criteria, branch and bound, heuristics, earliness and tardiness penalties

1 Introduction

Traditionally, the lion's share of scheduling research has been devoted to the study of problems with *regular* criteria. A criterion is called regular, if the objective function to be minimized is non-decreasing with respect to the completion times of the jobs. Nevertheless, with the raise of just-in-time production methods, single and parallel machine scheduling problems in which a job i is penalized both for being early or late in comparison with the given due date d_i have received tremendous attention. These criteria are called *non-regular*. The objective is to minimize either the sum or the maximum of the job penalties.

One of the earliest papers on this topic was published by Sidney [9], who provided an efficient algorithm to minimize the maximum earliness or tardiness penalty on a single machine, in which jobs have a restricted variety of due dates. An improvement of this algorithm was given by Lakshminarayan et al. [7].

Another direction of research was opened by the work of Kanet [6]. He considered the problem of minimizing the sum of the unweighted earliness and tardiness penalties, in which all jobs have a common due date. Furthermore, it was assumed that this due date is *unrestrictive*, which means that the optimal objective function value does not decrease if one drops the condition of non-negative starting times. Under this assumption, Kanet gave a polynomial time algorithm for determining an optimal solution.

Unfortunately, there are only a few papers dealing with non-regular criteria for problems with multi-stage processing systems. In this paper, we present an exact algorithm for the two-machine flow shop problem with a common arbitrary due date for all jobs. The objective is to minimize an arbitrary function formed by the weighted sum of earliness and tardiness penalties. Additionally, we discuss some heuristics for this problem.

The paper is organized as follows. In Section 2, we give the formulation and some basic properties of the problem. Afterwards, we discuss some details of the exact algorithm in Section 3. Finally, we briefly comment on the developed heuristics.

2 Problem Formulation and Properties of Optimal Schedules

Let $N = \{1, \ldots, n\}$ be the set of jobs which have to be processed on two machines M_1 and M_2. On both machines, only one job may be processed at a time. Hence, the flow shop condition means that every job has to be processed completely first on M_1 and then on M_2. We assume that preemption of an operation is not allowed. The values p_{ij} representing the processing time of job i on machine M_j are given and integer. For problems with a regular criterion, integer problem data are sufficient to guarantee that the starting times of an optimal schedule are integer. This remains to be true for the non-regular case if the due date is sufficiently restrictive. For an unrestrictive due date, there are optimal integer starting times if there exists an optimal schedule having one job completing precisely at the due date, see [6].

Since we are interested in a general treatment of the problem, we have to cope with arbitrary due dates. We restrict ourselves to integer starting times of the jobs for simplicity. Nevertheless, the precision of the algorithm we present is selectable (see the part on the bisection search later).

The objective is to minimize the sum of (if not noted differently job-specific) functions depending on the absolute deviations of the completion time C_i of job i from the given common due date d, i.e. $F =$

[1] Supported by Deutsche Forschungsgemeinschaft (Project ScheMA)

$\sum_{i=1}^{n} f_i(C_i)$. It is assumed that

- $f_i(x)$ is non-increasing for $x \le d$
- $f_i(x)$ is non-decreasing for $x \ge d$.

Special cases are the weighted sum of the deviations of the completion times from the due date, i.e. $F = \sum_{i=1}^{n} w_i|C_i - d|$, and the total weighted squared deviation, i.e. $F = \sum_{i=1}^{n} w_i(C_i - d)^2$. These penalty functions are the same for all jobs if all w_i are equal.

Using the 3-field classification introduced by Graham et al. [2], the problem under consideration may be denoted by $F2|d_i = d|\sum f_i(C_i)$.

Since we do not allow preemption, a schedule S may be given by the starting times s_{ij} of all operations (i,j), i.e. $S = (s_{ij})$, or alternatively by the matrix $C = (c_{ij})$ of completion times, where $c_{ij} = s_{ij} + p_{ij}$.

It is well-known (see Conway et al.[1]) that for flow shop problems with up to $m = 3$ machines and a regular criterion there always exists an optimal permutation schedule. For a given sequence $\pi = (\pi_1, \ldots, \pi_n)$ of a set of jobs N, the related best schedule S can be calculated by the following procedure:

R1: Set $s_{\pi_1 1} = 0$ and $s_{\pi_1 2} = p_{\pi_1 1}$.

R2: For $i = 2, \ldots, n$ set $s_{\pi_i 1} = c_{\pi_{i-1} 1}$ and $s_{\pi_i 2} = \max\{c_{\pi_{i-1} 2}, c_{\pi_i 1}\}$.

Next, we give some properties which can be used to restrict the set of candidates for optimal schedules for the problem $F2|d_i = d|\sum f_i(C_i)$.

Property 1 *There exists an optimal permutation schedule for the problem.*

Notice that Property 1 also holds for problem $F3|d_i = d|\sum f_i(C_i)$.

Property 2 *There exists an optimal schedule starting the first job at time 0 and having no idle times on M_1.*

Property 3 *There exists an optimal schedule, in which the jobs on M_2 which are started before or at d have no idle times in-between.*

3 Description of the Exact Algorithm

In this section, we describe some main components of our enumerative algorithm. Assume a given partition $E\dot{\cup}T$ of N, where E contains the jobs which are completed before or at the due date and T is the set of jobs being completed after the due date. Notice that in general not every partition is feasible, since the optimal makespan value $C^*_{\max}(E)$ for processing the jobs of set E must fulfill the inequality $C^*_{\max}(E) \le d$. The sequences of the jobs in E and T are denoted by $\epsilon(E)$ and $\tau(T)$, respectively. Using Properties 1–3, we are able to calculate a best schedule for a fixed pair of sequences $\epsilon(E)$ and $\tau(T)$, if additionally the starting time $s = s_{\tau_1 2}$ on M_2 of the first tardy job $\tau_1(T)$ is given.

If, more generally, only the partition $E\dot{\cup}T$ and the first tardy job $\tau_1(T)$ are given, the starting time s may be chosen out of an interval $[I_S, d]$ with

$$I_S = \max\left\{ d - p_{\tau_1 2} + 1, \sum_{j \in E} p_{j1} + p_{\tau_1 1}, C^*_{\max}(E) \right\} \quad (1)$$

The first term guarantees that job τ_1 is not completed before d (because τ_1 has to be tardy), the second is due to Property 2. $C^*_{\max}(E)$ may be determined in $O(n \log n)$ time by Johnson's algorithm [5]. (Indeed, this effort is needed only once for the determination of the Johnson sequence for the whole job set N — a makespan optimal sequence for a subset can be derived in $O(n)$ time.)

By $\tau^*(T, \tau_1, s)$ we denote a (not necessarily unique) *optimal* sequence of the jobs completed after the due date d for a given starting time s of the first tardy job τ_1 on M_2. We emphasize that the optimal sequence ϵ^* of the jobs of the set E depends only on the *starting time* of the first job of T on M_2, since this gives the maximal allowed makespan of sequence ϵ^*, too — it is not influenced by the choice of a sequence for T, except for the condition of non-negative starting times, which requires

$$s - p_{\tau_1 1} - \sum_{j=1}^{|E|} p_{\epsilon_j^* 1} \ge 0 \quad (2)$$

according to equation (1). Hence, we denote this sequence by $\epsilon^*(E, s)$, s being the starting time of the first job of T, skipping the job τ_1 itself in this notation. The calculation of these sequences and the related schedule is described in detail in Subsections 3.2 and 3.3.

We get the following contributions of the sequences $\epsilon^*(E,s)$ and $\tau^*(T, \tau_1, s)$ to the total penalty $F(\pi^*) = F(\epsilon^*) + F(\tau^*)$:

$$F(\epsilon^*(E,s)) := \sum_{j=1}^{|E|} f_j \left(s - \sum_{k=j}^{|E|} p_{\epsilon_k^* 2} \right) \quad (3)$$

$$F(\tau^*(T, \tau_1, s)) := \sum_{j=1}^{|T|} f_j(c_{\tau_j 2}) \quad (4)$$

$\sum_{i=1}^{n} f_i(C_i)$. It is assumed that

- $f_i(x)$ is non-increasing for $x \leq d$
- $f_i(x)$ is non-decreasing for $x \geq d$.

Special cases are the weighted sum of the deviations of the completion times from the due date, i.e. $F = \sum_{i=1}^{n} w_i|C_i - d|$, and the total weighted squared deviation, i.e. $F = \sum_{i=1}^{n} w_i(C_i - d)^2$. These penalty functions are the same for all jobs if all w_i are equal.

Using the 3-field classification introduced by Graham et al. [2], the problem under consideration may be denoted by $F2|d_i = d|\sum f_i(C_i)$.

Since we do not allow preemption, a schedule S may be given by the starting times s_{ij} of all operations (i,j), i.e. $S = (s_{ij})$, or alternatively by the matrix $C = (c_{ij})$ of completion times, where $c_{ij} = s_{ij} + p_{ij}$.

It is well-known (see Conway et al.[1]) that for flow shop problems with up to $m = 3$ machines and a regular criterion there always exists an optimal permutation schedule. For a given sequence $\pi = (\pi_1, \ldots, \pi_n)$ of a set of jobs N, the related best schedule S can be calculated by the following procedure:

R1: Set $s_{\pi_1 1} = 0$ and $s_{\pi_1 2} = p_{\pi_1 1}$.

R2: For $i = 2, \ldots, n$ set $s_{\pi_i 1} = c_{\pi_{i-1} 1}$ and $s_{\pi_i 2} = \max\{c_{\pi_{i-1} 2}, c_{\pi_i 1}\}$.

Next, we give some properties which can be used to restrict the set of candidates for optimal schedules for the problem $F2|d_i = d|\sum f_i(C_i)$.

Property 1 *There exists an optimal permutation schedule for the problem.*

Notice that Property 1 also holds for problem $F3|d_i = d|\sum f_i(C_i)$.

Property 2 *There exists an optimal schedule starting the first job at time 0 and having no idle times on M_1.*

Property 3 *There exists an optimal schedule, in which the jobs on M_2 which are started before or at d have no idle times in-between.*

3 Description of the Exact Algorithm

In this section, we describe some main components of our enumerative algorithm. Assume a given partition $E \dot{\cup} T$ of N, where E contains the jobs which are completed before or at the due date and T is the set of jobs being completed after the due date. Notice that in general not every partition is feasible, since the optimal makespan value $C^*_{\max}(E)$ for processing the jobs of set E must fulfill the inequality $C^*_{\max}(E) \leq d$. The sequences of the jobs in E and T are denoted by $\epsilon(E)$ and $\tau(T)$, respectively. Using Properties 1–3, we are able to calculate a best schedule for a fixed pair of sequences $\epsilon(E)$ and $\tau(T)$, if additionally the starting time $s = s_{\tau_1 2}$ on M_2 of the first tardy job $\tau_1(T)$ is given.

If, more generally, only the partition $E \dot{\cup} T$ and the first tardy job $\tau_1(T)$ are given, the starting time s may be chosen out of an interval $[I_S, d]$ with

$$I_S = \max\left\{ d - p_{\tau_1 2} + 1, \sum_{j \in E} p_{j 1} + p_{\tau_1 1}, C^*_{\max}(E) \right\} \tag{1}$$

The first term guarantees that job τ_1 is not completed before d (because τ_1 has to be tardy), the second is due to Property 2. $C^*_{\max}(E)$ may be determined in $O(n \log n)$ time by Johnson's algorithm [5]. (Indeed, this effort is needed only once for the determination of the Johnson sequence for the whole job set N — a makespan optimal sequence for a subset can be derived in $O(n)$ time.)

By $\tau^*(T, \tau_1, s)$ we denote a (not necessarily unique) *optimal* sequence of the jobs completed after the due date d for a given starting time s of the first tardy job τ_1 on M_2. We emphasize that the optimal sequence ϵ^* of the jobs of the set E depends only on the *starting time* of the first job of T on M_2, since this gives the maximal allowed makespan of sequence ϵ^*, too — it is not influenced by the choice of a sequence for T, except for the condition of non-negative starting times, which requires

$$s - p_{\tau_1 1} - \sum_{j=1}^{|E|} p_{\epsilon^*_j 1} \geq 0 \tag{2}$$

according to equation (1). Hence, we denote this sequence by $\epsilon^*(E, s)$, s being the starting time of the first job of T, skipping the job τ_1 itself in this notation. The calculation of these sequences and the related schedule is described in detail in Subsections 3.2 and 3.3.

We get the following contributions of the sequences $\epsilon^*(E, s)$ and $\tau^*(T, \tau_1, s)$ to the total penalty $F(\pi^*) = F(\epsilon^*) + F(\tau^*)$:

$$F(\epsilon^*(E, s)) := \sum_{j=1}^{|E|} f_j \left(s - \sum_{k=j}^{|E|} p_{\epsilon^*_k 2} \right) \tag{3}$$

$$F(\tau^*(T, \tau_1, s)) := \sum_{j=1}^{|T|} f_j(c_{\tau_j 2}) \tag{4}$$

Notice that formally on the left side of the Equations (3) and (4) the arguments of ϵ^* and τ^* should be given explicitly as arguments of F, too. In order to keep the formulae simple, the introduced shorthand notations in (3) and (4) will be used throughout the remainder of the paper.

3.1 Enumeration Procedure

As outlined in the preceding section, the algorithm we suggest has to enumerate all partitions $N = E\dot{\cup}T$. We call the enumeration of the possible sets E the *main tree* (illustrated in Figure 1).

We perform a depth-first search algorithm in the main tree. In every node we consider successively every job in T as the first job of sequence $\tau(T)$, obtaining different nodes $(E, \tau_1(T))$. This is illustrated in Figure 2, where the left node is denoted by ({2,3},1). According to Equation (1), we calculate the left boundary I_S of the possible starting times $s = s_{\tau_1(T)\,2}$.

Obviously, $F(\epsilon^*(E,s))$ is a non-increasing function (dfu(s)) and $F(\tau^*(T,\tau_1,s))$ is a non-decreasing function (ifu(s)) with respect to the starting time s (see Equations (3) and (4)). But if a function F decomposes into a non-increasing function dfu and non-decreasing function ifu, one may easily find its minimum value for integer arguments of a given interval by a bisection search. We call this procedure MINSEARCH.

The procedure is illustrated in Figure 3 for the first two steps. We start with the calculation of a lower bound for the best objective function value with respect to the whole interval $I1 = [I_S, d]$ (of the possible starting times $s_{\tau_1 2}$):

$$\begin{aligned} LB_{I1} &= LB_{I1}(\text{ifu}) + LB_{I1}(\text{dfu}) \\ &= \text{ifu}(I_S) + \text{dfu}(d) \end{aligned}$$

We then calculate a lower bound for the optimal objective function value with a starting time $s_{\tau_1 2}$ in the intervals $I2 = [I_S, \frac{I_S+d}{2}]$ and $I3 = [\frac{I_S+d}{2}, d]$.

$$\begin{aligned} LB_{I2} &= LB_{I2}(\text{ifu}) + LB_{I2}(\text{dfu}) \\ &= \text{ifu}(I_S) + \text{dfu}\left(\frac{I_S+d}{2}\right) \\ LB_{I3} &= LB_{I3}(\text{ifu}) + LB_{I3}(\text{dfu}) \\ &= \text{ifu}\left(\frac{I_S+d}{2}\right) + \text{dfu}\,(d) \end{aligned}$$

The determination of each function value of ifu and dfu, respectively, requires the application of a branch and bound algorithm. Some details of this procedure are described in Subsections 3.2 and 3.3. From now on, we assume that the penalty function is job-independent (i.e. for all i: $f_i = f$).

3.2 Calculation of Sequence $\tau^*(T, \tau_1, s)$

The sequence $\tau^*(T,\tau_1,s)$ is calculated by a branch and bound procedure similar to problem $F2||\sum f(C_i)$ with a regular criterion. First, we determine the starting times for the first job to be sequenced:

$$s_{\tau_2 1} = p_{\tau_1 1} + \sum_{j\in E} p_{j1} \quad (5)$$

$$s_{\tau_2 2} = \max\{c_{\tau_1 2}, c_{\tau_2 1}\} \quad (6)$$

The Equation (5) is due to Property 2. We enumerate all possible sequences, where the appended jobs are scheduled according to R2. Consider a node $\tau^k = (\tau_1, \ldots, \tau_k)$ in which k jobs were already sequenced. Denote with $\gamma_1, \ldots, \gamma_{t-k}$ and $\delta_1, \ldots, \delta_{t-k}$ ($t = |T|$) the sequences of the remaining jobs of T which are not yet sequenced ordered by non-decreasing processing times on M_1 and M_2, respectively. Applying the SPT rule, we get a lower bound for the total penalty of the remaining jobs by the following equations:

$$LB1 = \sum_{j=1}^{k} f(c_{\tau_j 2}) + \sum_{j=1}^{t-k} \phi\left(c_{\tau_k 1} + \sum_{l=1}^{j} p_{\gamma_l 1} + p_{\delta_1 2}\right) \quad (7)$$

$$LB2 = \sum_{j=1}^{k} f(c_{\tau_j 2}) + \sum_{j=1}^{t-k} f\left(\max\{c_{\tau_k 2}, c_{\tau_k 1} + p_{\gamma_1 1}\} + \sum_{l=1}^{j} p_{\delta_l 2}\right) \quad (8)$$

with

$$\phi(x) = \begin{cases} 0 & x < d \\ f(x) & x \geq d \end{cases} \quad (9)$$

We use ϕ in order to respect the starting time s. Finally, we obtain

$$LBT = \max\{LB1, LB2\} \quad (10)$$

Notice that the bounds given above are a generalization of the bounds given by Ignall and Schrage [4] for the regular problem. Most other approaches, e.g. the Lagrangian relaxation method given by van de Velde [10], do not apply in the more general case we are dealing with.

3.3 Calculation of the Sequence $\epsilon^*(E, s)$

For the determination of $\epsilon^*(E,s)$, we sequence the jobs of E sucessively, beginning with the latest job processed, by a branch and bound algorithm again. The starting times for the branch and bound algorithm are the completion times of the last early job on both machines and given by $s - p_{\tau_1 1}$ and s on M_1 and M_2, respectively. The procedure has some modifications compared with the above for the determination of τ^*.

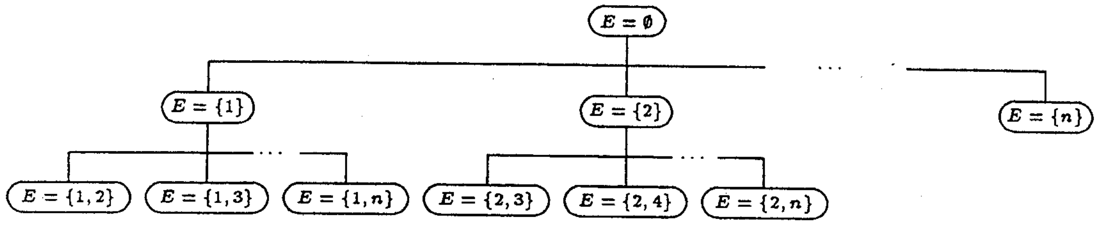

Figure 1: Main tree

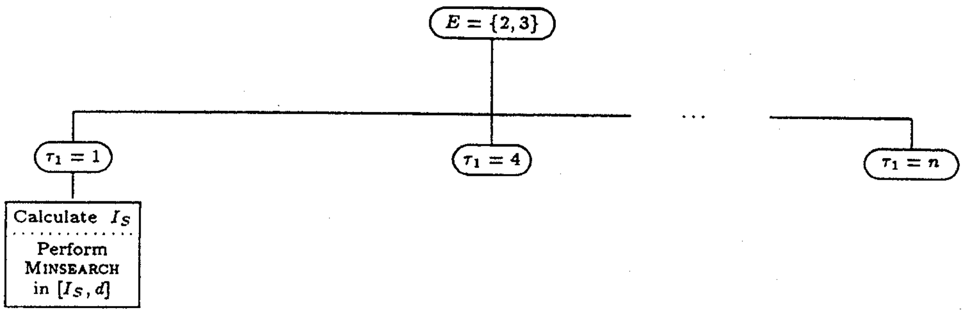

Figure 2: Subtree

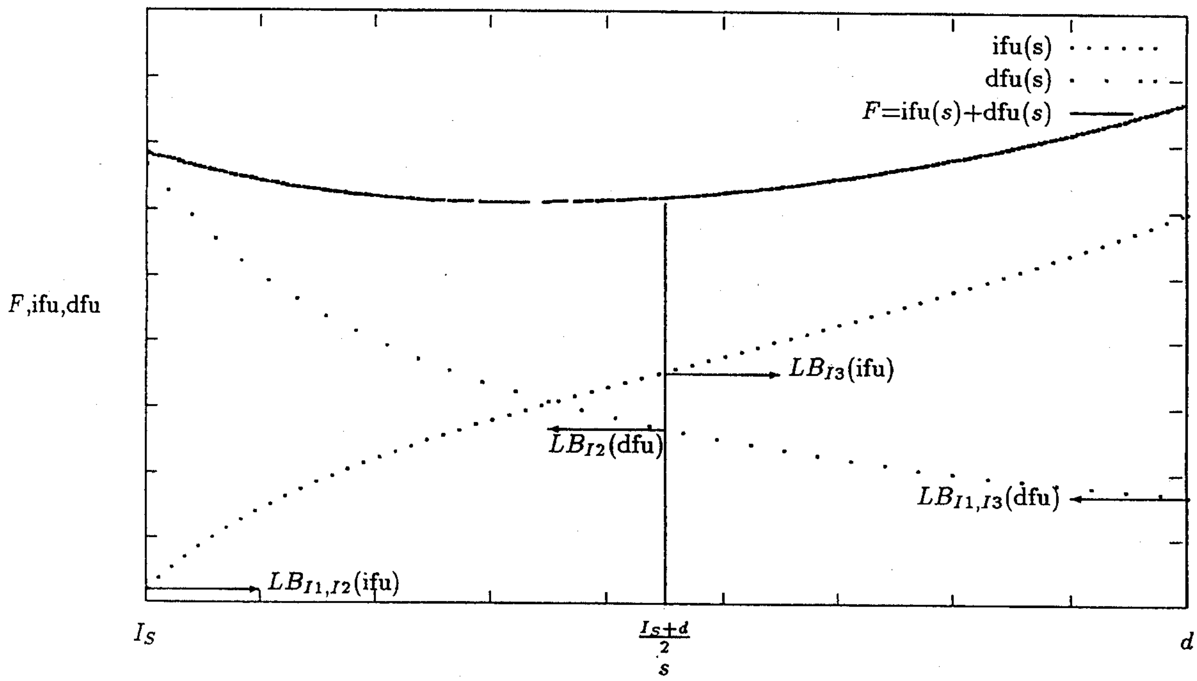

Figure 3: Illustration of the calculation of lower bounds (double indices mean that the values are equal for the two indices).

3.4 Computational Results

Dominance criteria may be found in paper [3]. The algorithm was coded in C++ and run on different types of problems. The processing times were integers randomly chosen from the interval $[1, p_{max}]$, where we selected $p_{max} \in \{5, 20, 50\}$, and the common due date was chosen as

$$d \in \{0, 0.1T, 0.25T, 0.5T, 0.75T, 1T\},$$

where T is the higher machine load of both machines. We tested linear and quadratic earliness tardiness penalties. The program was in most cases able to solve problems up to 20 jobs satisfactorily on a Sun Ultra Sparc 10 workstation.

We observed that an increase of the due date from $d = 0$ (in this case we have a regular optimization criterion) up to $d = 0.5T$ hardly increased the computational times. The most difficult problems appeared for $d = 0.75T$. Although the algorithm applies a bisection search procedure (MINSEARCH) in the interval of possible starting times (see Equation 1), where the length of the interval increases with the job processing times, we have observed that the problems with a higher value of p_{max} do not require higher computation times than the problems with $p_{max} = 5$.

4 Heuristics

Three types of heuristics were developed and compared. The first one was a beam search algorithm derived from the exact algorithm described above.

Secondly, we developed a heuristic which replaced the determination of the optimal sequences ϵ^* and τ^* for a given partition $N = T \dot{\cup} E$ of the set of jobs and given starting time $s_{\tau_1 2}$ by different local search heuristics. The partition itself and the starting time $s_{\tau_1 2}$ were also realized by random decisions.

For reasons of comparison, we implemented a heuristic which was not as sophisticated as the second one. Here, permutations of the whole job set N were considered at each iteration. For a certain permutation, the jobs on M_2 were pushed away from 0 in several steps of random length. Some details and computational results of the algorithms are given in [8].

References

[1] R. W. Conway, W. L. Maxwell, and L.W. Miller. *Theory of Scheduling.* Addison-Wesley, 1967.

[2] R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. H. G. Rinnoy Kan. Optimization and approximation in deterministic sequencing and scheduling. *Annals of Disc. Appl. Math.*, 5:287–326, 1979.

[3] J.N.D. Gupta, V. Lauff, and F. Werner. A branch and bound algorithm for two-machine flow shop problems with earliness and tardiness penalties. *Working Paper, Otto-von-Guericke-Universität*, 1999.

[4] E. Ignall and L. Schrage. Application of the branch and bound technique to some flow-shop scheduling problems. *Oper. Res.*, 13(3):400–412, 1965.

[5] S. M. Johnson. Optimal two and three stage production schedules with set up times included. *Naval Res. Log. Quart.*, 1(1):61–68, 1954.

[6] J. Kanet. Minimizing the average deviation of job completion times about a common due date. *Naval Res. Logist. Quart.*, 28:643–651, 1981.

[7] S. Lakshminarayan, R. Lakshmanan, R. Papineau, and R. Rochette. Optimal single-machine scheduling with earliness and tardiness penalties. *Oper. Res.*, 26:1079–1082, 1978.

[8] V. Lauff and F. Werner. Heuristics for a two-machine flow shop with a common due date. *Working Paper, Otto-von-Guericke-Universität*, 1999.

[9] J. Sidney. Optimal single-machine scheduling with earliness and tardiness penalties. *Oper. Res.*, 25:62–69, 1977.

[10] S. L. van de Velde. Minimizing the sum of the job completion times in the two-machine flow shop by lagrangean relaxation. *Ann. Oper. Res.*, 26:257–268, 1990.