An Enumerative Algorithm for Two-Machine Flow Shop Problems with Earliness and Tardiness Penalties

Jatinder N. D. Gupta,
University of Alabama in Huntsville,
Huntsville, AL 35899, USA

Volker Lauff and Frank Werner*
Otto-von-Guericke-Universität, Fakultät für Mathematik,
PSF 4120, 39016 Magdeburg, Germany

June 30, 2004

Abstract

We consider a two-machine flow shop problem with a common due date where the objective is to minimize the sum of functions which penalize early as well as tardy completion of jobs. Since the problem is NP-hard in the strong sense, we investigate some general properties of optimal schedules for the problem, we develop lower and upper bounds, derive dominance criteria, and propose an enumerative algorithm for finding an optimal schedule. The performance of the proposed algorithm together with the influence of the individual components is thoroughly discussed.

Keywords: scheduling, flow shop, non-regular criterion, enumerative algorithm, branch and bound, common due date

MSC classification: 90B35, 90C57, 68M20

1 Introduction

Recent trends in *Just-In-Time Production* and *Time-Based Management* require explicit consideration of job due date related objectives in solving scheduling problems where the due date of job i is its contractually agreed time of delivery d_i. While considering these processing philosophies, completing a job earlier or later than its due date is undesirable. For instance, costs of *earliness* (i.e.,

*Corresponding author, email: frank.werner@mathematik.uni-magdeburg.de completing a job *before* its due date) could result from deterioration or the need for storage and insurance. Costs of *tardiness* (i.e., completing a job *after* its due date) are more familiar and may include customer dissatisfaction, a loss of sale, or contract penalties. Thus, a penalty function f_i is assigned to each job i and it is desired to find a schedule that optimizes a well-defined criterion of optimality related to the job penalty function f_i.

In scheduling theory, an optimality criterion is called *regular* if the objective function to be minimized is non-decreasing with respect to the completion times of the jobs. Solving scheduling problems in which a job i is penalized for being completed early *as well as* tardy requires the development of analytical techniques where the optimality criteria are non-regular.

The majority of papers deal with single-stage processing systems, i.e. either one-machine problems or problems on parallel machines. Kanet [14] gave a polynomial algorithm for minimizing the total deviation about a common due date (i.e. $d_i = d$ for all jobs i) on one machine. The validity of the algorithm is limited to sufficiently high due dates. Such a due date is called unrestrictive (for a precise definition of restrictivity, see [17]). The problem above with a restrictive due date is NP-hard in the ordinary sense [10]. This also holds for the problem of minimizing the weighted deviation from an unrestrictive due date if job-specific weights w_i are given [9]. A comprehensive survey of the literature published until 1990 is provided by Baker and Scudder [2].

Very few studies deal with the problem of minimizing total earliness and tardiness penalties of multi-stage scheduling problems. Achuthan et al. [1] describe methods of minimizing the maximum earliness or tardiness penalty for a flow shop. Unlike to other articles, the authors define the earliness of a job as a monotonous function of the difference of the *desired and the actual starting time of the operation on the first machine.* Brandimarte and Maiocco [3] evaluate various neighborhood structures for solving a job shop problem where it is desired to minimize the weighted sum of earliness and tardiness in the case of job-dependent due dates d_i. Sotskov et al. [21] consider insertion heuristics for the job shop problem with setup times and quadratic earliness and tardiness penalty functions. Sarper was the first to extend the criterion of the sum of absolute deviations from a common due date to the flow shop environment with two machines [19]. He cites the results of Majaj [18], where a mixed integer mathematical model is studied. Majaj was able to solve six-job samples on a mainframe computer. During the submission of this paper, Gowrishankar et al. [7] presented a branch and bound algorithm for the m-machine flow shop problem with minimizing the sum of the squared deviations from a given due date. However, they consider only one special case of a given due date, namely that the due date is equal to the best value of mean completion time of the jobs. In view of this, as for regular problems they restrict to schedules, where the first operation starts at time zero. The algorithm has been tested on problems with up to 10 jobs and is able to find the best permutation schedule for a problem with 10 jobs and 5 machines in about 8 minutes. A heuristic insertion algorithm followed by an adjacent pairwise interchange procedure (again all schedules start at time zero) yields average percentage deviations from the optimal value of about 10 % for problems with 10 jobs. To our knowledge, no other results are available for solving multi-stage scheduling problems involving both earliness and tardiness penalties.

This paper considers a two-machine flow shop environment with common due date where the objective is to minimize an arbitrary function of the weighted sum of earliness and tardiness penalties. It is organized as follows. Section 2 describes the problem and presents its basic structural properties that serve as a starting point for the solution strategy and enumerative algorithm presented in Section 3. Several components of the proposed enumerative algorithm are described in Section 4. A brief comparison with known results for regular 2-machine problems is given in Section 5. The effectiveness and efficiency of the proposed dominance criteria and the proposed enumerative algorithm are evaluated in Section 6 where computational results for problems with up to 20 jobs are reported. In addition, the algorithm is illustrated on a small numerical example. Finally, Section 7 concludes the paper with some possible directions for future research.

2 Problem Description and Basic Properties

To formulate our problem, let $N = \{1, \ldots, n\}$ be the set of jobs to be processed on two machines M_1 and M_2 with a given common due date d. Each job has to be processed completely first on M_1 and then on M_2. The processing time required to process each job $i \in N$ on each machine M_j is represented by $p_{i\,j}$, $j = 1, 2$. Only one job may be processed at any given time on each machine. Preemption of an operation is not allowed. For each job $i \in N$, let C_i be its completion time. The objective is to find a processing schedule of these n jobs on machines M_1 and M_2 such that the total cost function given by $F = \sum_{i=1}^{n} f_i(C_i)$ is minimal. It is assumed that $f_i(d) = 0$ and

$$\left.\begin{array}{l}\text{M1: } f_i(x) \text{ is non-increasing for } x \le d \\ \text{M2: } f_i(x) \text{ is non-decreasing for } x \ge d,\end{array}\right. \qquad \text{(M)}$$

for $i = 1, \ldots, n$. Note that this assumption includes regular criteria as a special case, since by setting $f_i(x) = 0$ for $x \le d$ every arbitrary regular criterion minimizing total penalty may be modeled. Special cases are the weighted sum of the absolute 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. $F = \sum_{i=1}^{n} w_i (C_i - d)^2$.

Using the standard three-field classification scheme described by Graham et al. [6], the problem under consideration may be denoted by $F2|d_i = d| \sum f_i(C_i)$. For $d = 0$ and $f_i(C_i) = C_i$, one gets the total flow time criterion, which is known to be NP-hard in the strong sense [5]. Hence, in general, the problem investigated in this paper is strongly NP-hard.

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

$C = (c_{i\,j})$ of completion times of all operations (i, j), where $c_{i\,j} = s_{i\,j} + p_{i\,j}$.

For problem $F2||\sum C_i$, it is known that the set of all permutation schedules contains an optimal schedule. The following lemma extends this property to the $F2|d_i = d|\sum f_i(C_i)$ problem considered in this paper.

Lemma 1 *For the $F2|d_i = d|\sum f_i(C_i)$ problem, the set of all permutation schedules contains an optimal schedule.*

Proof: Assume that an optimal permutation schedule does not exist. For each machine M_j, let ρ^j be the job sequence where ρ^j_i represents the job in the i-th position on machine M_j. Select an optimal schedule S with $\rho^1_i = \rho^2_i$ for $i = 1, \dots, k$ and $\rho^1_{k+1} \neq \rho^2_{k+1}$, where $k < n$ is maximal. Let $\rho^1_{k+l} = \rho^2_{k+1}$ (see the Gantt-chart in Figure 1). Obviously, changing the job sequence on M_1 into

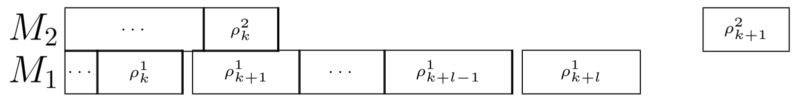

Figure 1: Gantt-chart in Lemma 1

$$(\rho^1_1, \dots, \rho^1_k, \rho^1_{k+l}, \rho^1_{k+1}, \dots, \rho^1_{k+l-1}, \rho^1_{k+l+1}, \dots, \rho^1_n)$$

is possible without violating the feasibility or altering the optimal objective function value of the resulting schedule S', since one may guarantee $c'_{\rho^1_{k+l-1}\,1} \leq c_{\rho^1_{k+l}\,1}$, i.e. the completion time of each operation (i, j) in S' is not greater than the completion time of this operation in S. This contradicts the assumption that S is a schedule with maximal k (defined as above). ■

In view of Lemma 1, from now on, we restrict ourselves to the consideration of permutation schedules. Consider a permutation schedule $S = (s_{i\,j})$. If there is an idle time t on M_1 between the processing of two jobs k and l, we may change $s_{l\,1}$ into $s'_{l\,1} = s_{l\,1} - t$ altering neither the feasibility nor the objective function value of the schedule. Hence, we state the following property of the $F2|d_i = d|\sum f_i(C_i)$ problem:

Property 1 *There exists an optimal schedule in which the first job starts at time 0 and there is no inserted idle time on M_1.*

Furthermore, the following property can easily be proven by contradiction.

Property 2 *There exists an optimal schedule such that the jobs that are started on machine M_2 before or at d have no idle times in-between.*

Combining Properties 1 and 2 shows that, except for the unavoidable inserted idle times between tardy jobs on machine M_2, there exists an optimal schedule for the $F2|d_i = d|\sum f_i(C_i)$ problem that has no intentionally inserted idle time between jobs on any of the two machines. This result is similar to the one reported by Kanet [14] for the $1|d_i = d|\sum |C_i - d|$ problem. However, other properties of an optimal schedule for the one-machine problem may not hold for $F2|d_i = d|\sum f_i(C_i)$. One example is the V-shape property which means that early jobs and tardy jobs are sequenced in non-increasing and non-decreasing order of processing times, respectively. Furthermore, one job completes precisely at the due date for Kanet's problem. This is also wrong for the problems considered in this paper.

For the unrestrictive common due date, the starting times of operations may be non-integers even if the processing times and due dates are integers. If the two-machine flow shop problem is unrestrictive, the problem reduces to an unrestrictive one-machine problem involving machine M_2 only. However, if the one-machine problem with the processing times on M_2 is unrestrictive, this property does not necessarily hold for the two-machine flow shop problem.

For ease of presentation, we restrict ourselves to integer starting times of the jobs. Nevertheless, the proposed algorithm is general enough to solve the problems for restrictive and unrestrictive common due dates with a given accuracy (see the part on the bisection search later).

3 Description of the Enumerative Algorithm

In this section, we describe the main components of the proposed enumerative algorithm. Based on a partition of the job set N into two sets E and T with $N = E \dot{\cup} T$, where

- E is the set of early jobs, i.e. the set of jobs that are completed before the due date: $E = \{i \in N | C_i < d\}$,
- T is the set of tardy jobs, i.e. the set of jobs that are completed at or after the due date: $T = \{i \in N | C_i \geq d\}$ (the rather technical reason for putting a job completed precisely at the due date into E is explained in the next subsection),

the algorithm enumerates possible sets E of early jobs. In order to represent job sequences of sets N, E and T, we use the letters π, ϵ and τ, respectively, as follows:

$$\pi(N) = (\epsilon(E), \tau(T)) = (\epsilon_1(E), \ldots, \epsilon_{|E|}(E), \tau_1(T), \ldots, \tau_{|T|}(T)) \ .$$

Based on the properties given in Section 2, for some fixed set E we have to find both the best starting time s of the first tardy job τ_1 on M_2 and the best sequences of the sets E and T. To determine s, we have to perform a search over an interval the length of which depends on the first tardy job. Therefore, for a fixed set E, we have to consider several values of s and τ_1 and for each fixed combination, the best sequences of the jobs in E and T have to be determined by separate branch and bound algorithms (some components of them are given in Section 4).

The rest of this section is organized as follows. A characterization of a node of the main tree (enumeration of the possible sets of early jobs) of the algorithm is discussed in Subsection 3.1. The enumeration procedure together with the generation of the subnodes for a fixed set E is explained in Subsection 3.2. Some comments on the use of upper bounds (Subsection 3.3) and dominance criteria (Subsection 3.4) for investigating nodes and subnodes of the main tree are given in the rest of this section.

3.1 Characterization of Nodes

Since we often have to deal with optimal sequences and schedules *with respect to constraints* which do not necessarily yield our desired global optimum for problem $F2|d_i = d|\sum f(C_i)$, we refer to the previous as "best" sequences or schedules in order to provide a distinction. Using Lemma 1 and Properties 1 and 2, we can find a best schedule for a given partition $E\dot{\cup}T$ of N, a job $\tau_1(T)$, and a starting time $s = s_{\tau_1\,2}$ on machine M_2. The cases $E = \emptyset$ and $E \neq \emptyset$ are treated differently. For the former, the best starting time of job τ_1 on M_2 is $\max\{d - p_{\tau_1\,2}, p_{\tau_1\,1}\}$. For $E \neq \emptyset$, the starting time s may be chosen out of an interval $[I_S, d]$ where:

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

In Equation (1), we use $\mathrm{C}^*_{\max}(E)$ for the minimal makespan among all permutations of the jobs of set E which may be determined in $O(n\log n)$ time by Johnson's algorithm [12]. (Indeed, this effort is needed only once for the determination of the Johnson-sequence for the whole job set N — the optimal sequences together with the makespan for subsets are derived in $O(n)$.) The first term guarantees that job τ_1 is not completed before d (because τ_1 has to be tardy), the second term respects the flow shop condition (see Property 1) and the third term is due to the requirement that all early jobs have to be completed before job τ_1 can be processed on M_2.

The first term cannot be altered in $d - p_{\tau_1\,2} + 1$ in general, since the increase of the contribution to the objective function value by τ_1 even for completion at time $d+1$ instead of d can be higher than the reduction of the penalty for the early jobs by the increment of their starting times by 1. This makes sense if one wishes to model a criterion like the minimization of the weighted number of tardy jobs. In this case, the algorithm described in this article may be used by setting $f_i = 0$ for $x \leq d$ and $f_i = w_i$ for $x > d$, where the w_i are sufficiently high. This flexibility of the algorithm has the only disadvantage that a schedule as shown in Figure 2 can belong either to $E = \{1\}$ with $\tau_1 = 2$ or $E = \{1, 2\}$ with $\tau_1 = 3$, since job 2 is completed at time d. In our realization, the enumeration algorithm is not able to recognize this doubling and may consider both cases. However, the description of the bisection search in Section 3.2 will show that the effect on the CPU time requirement is negligible. Equation (1) implies the following two

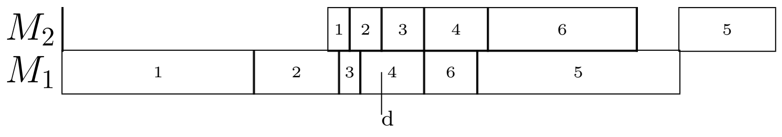

Figure 2: Gantt-chart of an ambigious schedule

conditions for the feasibility of a partition $E\dot{\cup}T$ with $E \neq \emptyset$:

$$\sum_{j \in E} p_{j\,1} + p_{\tau_1\,1} \leq d \tag{2}$$

and

$$\mathrm{C}^*_{\max}(E) \leq d, \tag{3}$$

which we will refer to later.

Let $\tau^*(T, \tau_1, s)$ denote a best sequence of the jobs completed after the common due date d subject to the constraint that E, τ_1, and s are fixed. The best sequence ϵ^* of jobs in set E is dependent on the *starting time* of the first job of T since this is the maximal allowed makespan value for the sequence ϵ^*. Hence, we denote this sequence by $\epsilon^*(E, s)$, s being the starting time of the first job of T, skipping in this notation the job τ_1 itself. The calculation of these sequences and the related schedule are described in detail in Section 4.

The contributions of the sequences $\epsilon^*(E, s)$ and $\tau^*(T, \tau_1, s)$ to the total penalty $F(\pi^*) = F(\epsilon^*) + F(\tau^*)$ are calculated due to Lemma 1 and Properties 1 and 2 as follows:

$$F\left(\epsilon^*\left(E, s\right)\right) \ := \ \sum_{j=1}^{|E|} f_{\epsilon_j}\left(s - \sum_{k=j+1}^{|E|} p_{\epsilon^*_k\,2}\right) \tag{4}$$

$$F\left(\tau^*(T, \tau_1, s)\right) \ := \ \sum_{j=1}^{|T|} f_{\tau_j}(c_{\tau^*_j\,2}) \tag{5}$$

3.2 Enumeration Procedure

As outlined before, we have to enumerate all partitions $N = E\dot{\cup}T$. We call the enumeration of the possible sets E the *main tree* (illustrated in Figure 3). We perform a depth-first search algorithm in the main tree.

At each node (which specifies a certain set E) we now have to enumerate all possible starting times s for the first job of T on M_2. To achieve this, 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 4, where the left node is denoted by ({2,3},1). Using Equation (1), we calculate the left boundary I_S of the possible starting times $s = s_{\tau_1(T)\,2}$. We now describe a procedure, called MINSEARCH, to determine the best starting time s such that the sum of the resulting function values $F(\epsilon^*)$ and $F(\tau^*)$ is minimal for a fixed partition $E\dot{\cup}T$ and a fixed tardy job τ_1.

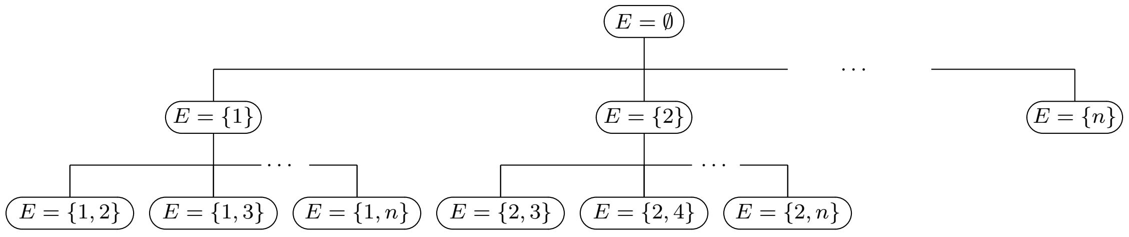

Figure 3: Main tree

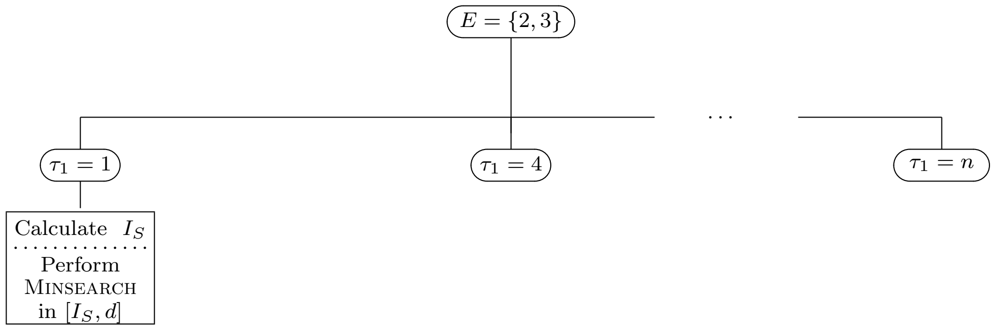

Figure 4: Subtree

Obviously, $F(\epsilon^*(E, s))$ is a non-increasing function (abbreviated as 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 (4) and (5)). But if a function F is decomposable into a non-increasing function dfu and non-decreasing function ifu, one may easily find its minimum value for integer arguments (or any other accuracy) of a given interval by a modified bisection search. It is worth noting that the objective function F does not need to be unimodal since a subinterval will only be excluded from further investigations when the lower bound for this interval is not smaller than the currently best function value (i.e. contrary to classical bisection search, it may happen that both subintervals have to be partitioned further).

Figure 5 illustrates procedure MINSEARCH for the first two steps. We start with the calculation of a lower bound for the function F in the whole interval $I1 = [I_S, d]$ (of the possible starting times of $s_{\tau_1 2}$):

$$\mathrm{LB}_{I1} = \mathrm{LB}_{I1}(\mathrm{ifu}) + \mathrm{LB}_{I1}(\mathrm{dfu}) = \mathrm{ifu}(I_S) + \mathrm{dfu}(d).$$

We then calculate a lower bound for F in the intervals $I2 = \left[I_S, \frac{I_S+d}{2}\right]$ and $I3 = \left[\frac{I_S+d}{2}, d\right]$:

$$\mathrm{LB}_{I2} = \mathrm{LB}_{I2}(\mathrm{ifu}) + \mathrm{LB}_{I2}(\mathrm{dfu}) = \mathrm{ifu}(I_S) + \mathrm{dfu}\left(\frac{I_S+d}{2}\right)$$
$$\mathrm{LB}_{I3} = \mathrm{LB}_{I3}(\mathrm{ifu}) + \mathrm{LB}_{I3}(\mathrm{dfu}) = \mathrm{ifu}\left(\frac{I_S+d}{2}\right) + \mathrm{dfu}\,(d)$$

Interval $I2$ is partitioned further recursively if LB_{I2} is smaller than the currently

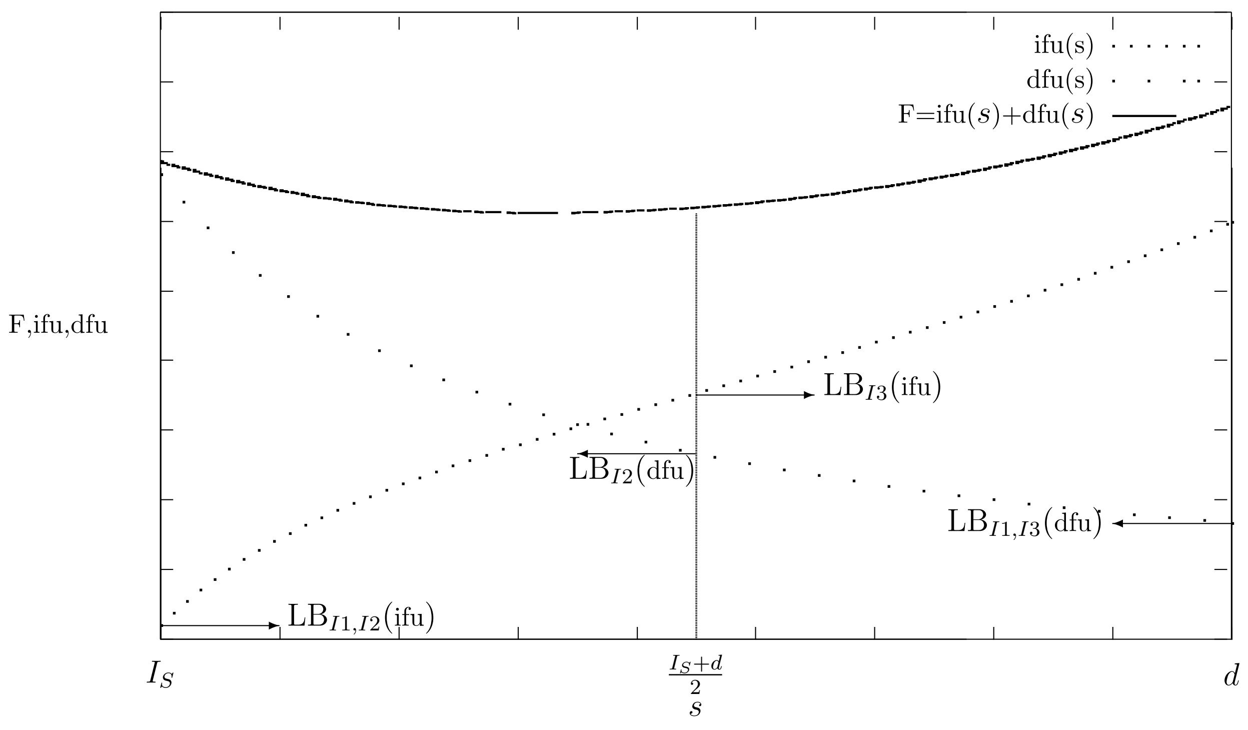

Figure 5: Illustration of the calculation of lower bounds (double indices mean that the values are equal for the two indices).

best function value UB and accordingly, interval $I3$ is partitoned if $LB_{I3} < UB$. This procedure stops if no subinterval has a smaller lower bound than UB.

3.3 Upper Bound Heuristics in Procedure Minsearch

Two upper bounds are used in the procedure, one of them for the branch and bound algorithm for the determination of sequence ϵ^* and the other one for τ^*. The bisection search described above starts with the determination of the sequences $\epsilon^*(E, d)$ and $\tau^*(T, \tau_1, I_s)$ since we need the best (i.e., optimal to the given constraints) objective function values on the boundaries of the interval. By far the most time-consuming part of the whole enumerative algorithm is the calculation of the best sequences and hence objective function values in the course of the bisection search in $[I_S, d]$. Since $\epsilon^*(E, s)$ is also a feasible sequence for $\tilde{s} \geq s$, we save the best sequence ϵ^* for a fixed E and s in order to get an upper bound for the function value $F(\epsilon^*(E, \tilde{s}))$ with $\tilde{s} > s$ by application of the sequencing procedure described in Subsection 4.2. To obtain a good upper bound, our algorithm uses the best sequence calculated for the nearest lower starting time s. In an analogous way, an upper bound for $F(\tau^*(T, \tau_1, \tilde{s}))$ may be generated out of the sequence $\tau^*(T, \tau_1, s)$ provided that $\tilde{s} > s$.

The results described so far can be applied to arbitrary job-dependent penalty functions provided that the monotonicity property (M) is fulfilled. In the remainder of the article, we consider job-independent penalty functions only ($f_i = f$ for all $i \in N$).

3.4 Dominance Criteria for Nodes of the Subtree

In this subsection, we describe dominance criteria that enable us to exclude a certain node E or (E, τ_1) from consideration. A very simple criterion proved to be effective. The flow shop problem is relaxed to a one-machine scheduling problem with the processing times for machine M_2 as input data. One easily gets a lower bound for the best obtainable objective function value of a certain partition $E \dot{\cup} T$ and a fixed job τ_1 of this one-machine problem by calculating the minimal contribution of E via the LPT-rule (i.e. longest processing time first) for a starting time $s = d$ (with β being the SPT-sequence, i.e. shortest processing time first: $p_{\beta_1 2} \leq p_{\beta_2 2} \leq \cdots \leq p_{\beta_e 2}$ and $e = |E|$)

$$\mathrm{LL} = \sum_{j=1}^{e} f\left(d - \sum_{l=1}^{j-1} p_{\beta_l 2}\right)$$

and adding the value of

$$\mathrm{LR} = \sum_{j=1}^{t} f\left(d + \sum_{l=1}^{j} p_{\delta_l 2}\right)$$

(with δ being the SPT-sequence of $T \backslash \{\tau_1\}$: $p_{\delta_1 2} \leq p_{\delta_2 2} \leq \cdots \leq p_{\delta_t 2}$ and $t = |T \backslash \{\tau_1\}|$). This is due to the monotonicity properties (M) of f. We refer to this criterion in the following as VS, since it uses the V-shape property of an optimal schedule of the one-machine case. We also use a criterion called STVS (stronger VS), which even allows to exclude a node of the main tree. The STVS criterion is quite similar to the above criterion with the difference that we choose the tardy job with the longest processing time on M_2 as the first tardy job when we perform the above calculations for LR. This yields a lower bound for the contribution of the tardy jobs to the best objective function value.

The next criterion deals with the dominance of a node (E, τ_1).

Theorem 1 *Let f fulfill condition (M) and additionally be convex for $x \geq d$. Consider a partition of the set of jobs N in E and T as previously described and a node (E, i), $i = \tau_1(T)$ being the first tardy job. Let $j \in T \backslash \{i\}$ with $p_{j 1} \leq \min_{l \in T \backslash \{i,j\}} \{p_{l 2}\}$. Furthermore, let $[I_S, d]$ be the interval of the feasible starting times of job i (see Equation* (1)*) on M_2 for the node (E, i) as well as for $(E \cup \{j\}, i)$. If*

$$F\left(\tilde{\epsilon}^* \left(E \cup \{j\}, s\right)\right) - F(\epsilon^*(E, s)) \leq f(s + p_{i 2} + \min_{l \in T \backslash \{i,j\}} \{p_{l 2}\}) \tag{6}$$

holds for all $s \in [I_S, d]$, then node (E, i) is dominated by node $(E \cup \{j\}, i)$.

PROOF: We estimate the difference of the costs of the tardy jobs between the best sequences $\tau^*(T,i,s))$ and $\tilde{\tau}^*(T\backslash\{j\},i,s)$. An upper bound for $F(\tilde{\tau}^*(T\backslash\{j\},i,s))$ is obviously obtained by the objective function value for the sequence obtained from τ^* by deleting job j. Since we do not know the position of job j in $\tau^*(T,i,s))$, the estimate has to be correct for any position $k=2,\ldots,|T|$ of job j. Note that the starting times on M_1 of all tardy jobs in $T\backslash\{j\}$ which are processed *before* job j in $\tau^*(T,i,s)$ are incremented by the processing time $p_{j\,1}$, whereas the starting times of the jobs processed after j in $\tau^*(T,i,s)$ are not altered by moving job j from T to E. This increases the completion times of the jobs at positions $2,\ldots,k-1$ in $\tilde{\tau}^*(T\backslash\{j\},i,s)$ by $p_{j\,1}$ at the most. Thus we have for the difference of the costs of the tardy jobs:

$$\begin{aligned}\Delta &= \sum_{l=2,\ldots,k} f(c_{\tau_l\,2}) - \sum_{l=2,\ldots,k-1} f(c_{\tau_l\,2}+p_{j\,1})\\ &= f(c_{\tau_2\,2}) + \sum_{l=2,\ldots,k-1} \left(f(c_{\tau_{l+1}\,2}) - f(c_{\tau_l\,2}+p_{j\,1})\right)\\ &\geq f(c_{\tau_2\,2}) + \sum_{l=2,\ldots,k-1} \left(f(c_{\tau_l\,2}+p_{\tau_l\,2}) - f(c_{\tau_l\,2}+p_{j\,1})\right)\end{aligned}$$

Since $p_{j\,1} \leq \min_{l\in T\backslash\{i,j\}}\{p_{l\,2}\}$, all the terms of the sum are non-negative. Hence,

$$\Delta \geq f(c_{\tau_2\,2}) \geq f(s+p_{i\,2}+\min_{l\in T\backslash\{i,j\}}\{p_{l\,2}\}),$$

from which we obtain Inequality (6). ∎

The application of Theorem 1 in the algorithm is as follows. All jobs j with $p_{j\,1} \leq \min_{l\in T\backslash\{i,j\}}\{p_{l\,2}\}$ are considered. We do not calculate the sequences $\epsilon^*(E,s)$ and $\tilde{\epsilon}^*\,(E\cup\{j\},s)$ exactly, since this is too time-consuming. The value of $F(\epsilon^*(E,s))$ is estimated by an LPT-sequence on M_2, whereas $F\,(\tilde{\epsilon}^*\,(E\cup\{j\},s))$ is estimated by the Johnson-sequence for $E\cup\{j\}$. Furthermore, we ensure that the makespan of this Johnson-sequence does not exceed I_S. Otherwise, the lowest possible starting time $\tilde{I}_S$ of τ_1 for $E\cup\{j\}$ is larger than I_S, violating one assumption of Theorem 1.

4 Bounds and Dominance Criteria for the Sequencing Algorithms

The determination of each function value of ifu(s_1) and dfu(s_2) in the previous section requires the application of a branch and bound algorithm. Subsection 4.1 deals with scheduling the tardy jobs, 4.2 deals with scheduling the early jobs, and in Subsection 4.3, two dominance criteria are derived. Although the branch and bound procedures have some similarity to known algorithms for regular criteria, we also present a new lower bound and a new dominance criterion.

4.1 Obtaining Sequence $\tau^*(T, \tau_1, s)$

The sequence $\tau^*(T, \tau_1, s)$ is obtained by a branch and bound procedure similar to the problem $F2|d_i = d|\sum f(C_i)$ with a regular criterion. The jobs of a partial sequence $\tau^k = (\tau_1, \ldots, \tau_k)$ are scheduled with respect to the starting time $s = s_{\tau_1 2}$ for $i = 2, \ldots, k$ by the following procedure:

- $s_{\tau_i 1} = c_{\tau_{i-1} 1}$
- $s_{\tau_i 2} = \max\{c_{\tau_{i-1} 2}, c_{\tau_i 1}\}$

The first step is due to Property 1. Denote with $\gamma_1, \ldots, \gamma_{t-k}$ and $\delta_1, \ldots, \delta_{t-k}$ the sequence of the remaining jobs of T which are not yet sequenced ($t = |T|$), ordered by non-decreasing processing times on M_1 and M_2, respectively. Then we get a lower bound for the costs of any completion of τ^k by applying the SPT-rule using the following equations:

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

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

with

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

We use ϕ in order to respect the starting time s of the tardy jobs on M_2. Notice that the bounds given above are a generalization of the bounds given by Ignall and Schrage [11] for the $F2||\sum C_i$ problem.

We now introduce a new bound which dominates LB1. Notice that in Equation (7) the *lowest* processing time on M_2 of the jobs not yet scheduled is used exclusively. Now, we consider the constraint that *all* processing times of the unscheduled jobs on M_2 have to be used, but the assignment of the processing times to the set of jobs is relaxed. Additionally, we drop the constraint that only one job may be processed on M_2 at any one time.

Theorem 2 *Among all optimal schedules under the above relaxation, there exists one with an SPT-sequence on M_1.*

The proof is based on an interchange argument. It is obvious from Theorem 2 that the optimal objective function value of the relaxed problem does not exceed the optimal objective function value of the original problem. The relaxed problem can be formulated as an assignment problem. Consider a complete bipartite graph G consisting of $t-k$ nodes representing the unscheduled jobs on M_1 (node set V) and $t-k$ nodes representing the unscheduled jobs on M_2 (node set U). The edges between the two sets of nodes have the weights $c_{i\,j}$ calculated by the following formula ($i \in V$, $j \in U$):

$$c_{i\,j} = f\left(\max\left\{\rho(i,j), c_{\tau_k\,1} + \sum_{q=1}^{i} p_{\gamma_q\,1}\right\} + p_{\delta_j\,2}\right) \tag{9}$$

with

$$\rho(i,j) = \begin{cases} c_{\tau_k\,2} + \sum_{q=1}^{i-1} p_{\delta_q\,2} & i < j \\ c_{\tau_k\,2} + \sum_{q=1}^{j-1} p_{\delta_q\,2} + \sum_{q=j+1}^{i} p_{\delta_q\,2} & i \geq j \end{cases}$$

For $i < j$, the value of $\rho(i,j)$ is just $c_{\tau_k\,2}$ plus the sum of the smallest $i-1$ processing times of the unscheduled jobs on machine M_2. If $i \geq j$, the processing time $p_{\delta_j\,2}$ of job j on M_2 would appear twice in Equation (9), since it is found under the smallest $i-1$ processing times. To avoid this duplication, we replace the value $p_{\delta_j\,2}$ by $p_{\delta_i\,2}$ in the sum, which improves the bound slightly.

In standard literature on combinatorial optimization (see for instance [?]), this assignment problem is formulated as a linear program:

$$\begin{aligned} \sum_{i,j} c_{i\,j} x_{i\,j} &\to \min! \\ \sum_{j=1}^{n} x_{i\,j} &= 1 \text{ for } i = 1,\ldots,n \\ \sum_{i=1}^{n} x_{i\,j} &= 1 \text{ for } j = 1,\ldots,n \\ x_{i\,j} &\geq 0 \end{aligned}$$

with the dual

$$\begin{aligned} \sum_{i=1}^{n} \alpha_i + \sum_{j=1}^{n} \beta_j &\to \max! \\ \alpha_i + \beta_j &\leq c_{i\,j} \text{ for all } i = 1,\ldots,n \text{ and } j = 1,\ldots,n \\ \alpha_i, \beta_j &\lesseqgtr 0 \end{aligned}$$

Here $x_{ij} = 1$ means that the processing time $p_{\delta_j\,2}$ is assigned to the job with rank i in the sequence $(\tau_{k+1},\ldots,\tau_t)$. It is well-known that this problem may be solved in $O(n^3)$ time by the Hungarian method. In order to save computation time, we construct a feasible but not necessarily optimal solution of the dual and calculate this matching bound LBM as follows:

$$\begin{aligned} \alpha_i &= c_{i\,1} - c_{1\,1} \text{ for all } i \\ \beta_j &= \min_i\{c_{i\,j} - \alpha_i\} \text{ for all } j \\ \text{LBM} &= \sum_{k=1}^{n} (\alpha_k \beta_k) \end{aligned} \tag{10}$$

This calculation can be done in $O(n^2)$ time. According to duality theory, this solution of the dual provides a lower bound for the optimal objective function value of the primal.

Notice that bound LB1 described at the beginning of the section is dominated by the matching criterion. Hence, the first upper bound calculated is LB2 in $O(n)$ time (recall that we do not need to sort the processing times in each step due to our data structure). If the value generated by this calculation is not high enough to cut the node, the matching criterion is applied and LBM is calculated.

4.2 Obtaining Sequence $\epsilon^*(E, s)$

In general, a branch and bound algorithm has to be applied to determine the sequence $\epsilon^*(E, s)$. However, for cases satisfying Lemma 2 below, the application of the LPT-rule generates the sequence $\epsilon^*(E, s)$.

Lemma 2 *For fixed E, τ_1 and s, the sequence $\epsilon^*(E, s)$ is obtained by scheduling the jobs in E on machine M_2 according to the LPT-rule provided the makespan of this sequence does not exceed s.*

Proof: This follows easily by contradiction: Assume job i being scheduled immediately after job j with $p_{i\,2} > p_{j\,2}$ and $C_i \leq d$ in an optimal schedule S. Denote with S' the schedule obtained from S by the exchange of jobs i and j and setting $C'_i = C_i - p_{j\,2}$ and $C'_j = C_i$ as completion times for the jobs i and j (note that by Property 2 we have $C_i = C_j + p_{i\,2}$). Since $f(x)$ is non-increasing for $x \leq d$, we get

$$\begin{aligned} \sum_{i \in N} f(C_i) - \sum_{i \in N} f(C'_i) &= f(C_j) - f(C_i - p_{j\,2}) \\ &= f(C_i - p_{i\,2}) - f(C_i - p_{j\,2}) \geq 0. \end{aligned}$$

■

Note that Lemma 2 above does not necessarily hold for job-specific penalty functions f_i. However, it can be adapted to the case of a linear weighted penalty function $f_i = w_i T_i + h_i E_i$ where w_i and h_i are the weights assigned to the earliness (E_i) and tardiness (T_i) of job i, respectively. In this case, there exists an optimal schedule where the early jobs are processed according to non-increasing ratios $p_{i\,2}/h_i$, provided the makespan of this schedule does not exceed s.

For the determination of sequence $\epsilon^*(E, s)$ we assign the jobs of E successively, beginning with the latest job processed, testing all possible partial sequences. Denote $e = |E|$ and $p_E = \sum_{i \in E} p_{i\,1}$. The starting times (i.e., the completion times of the last early jobs) for the branch and bound algorithm are p_E and s on M_1 and M_2, respectively. A partial schedule is generated from the sequence $(\epsilon_{e-k}, \ldots, \epsilon_e)$ in the following manner which takes into account that each job of the sequence completes as late as possible:

- set $c_{\epsilon_e\,1} = p_E$ and $c_{\epsilon_e\,2} = s$;

- for $i = e-1, \ldots, e-k$ set $c_{\epsilon_i\,2} = s_{\epsilon_{i+1}\,2}$ and $c_{\epsilon_i\,1} = s_{\epsilon_{i+1}\,1}$.

The bound calculation is slightly different from the one for the tardy jobs. Let $\delta_1, \ldots, \delta_{e-k-1}$ be a sequence of the unscheduled jobs of set E where jobs are arranged according to non-increasing processing times at M_2.

Then, by application of the LPT-rule, we get the following lower bound:

$$\mathrm{LBE} \;=\; \sum_{j=e-k}^{e} f(c_{\epsilon_j\,2}) + \sum_{j=1}^{e-k-1} f\left(s_{\epsilon_{e-k}\,2} - \sum_{l=1}^{j-1} p_{\delta_l\,2}\right) \tag{11}$$

We also check by Johnson's algorithm whether the minimal makespan of the unscheduled jobs $\{\delta_1, \ldots, \delta_{e-k-1}\}$ does not exceed the maximal allowed makespan, hence the inequality

$$\mathrm{C}^*_{\max}(\{\delta_1, \ldots, \delta_{e-k-1}\}) \leq s_{\epsilon_{e-k}\,2} \tag{12}$$

This check will be denoted as criterion EMSP. Otherwise, we do not branch further since all possible schedules completing the partial sequence $(\epsilon_{e-k}, \ldots, \epsilon_{e-n})$ will not be feasible. On the other hand, we stop branching if the makespan of the LPT-sequence of the unscheduled jobs is lower than or equal to $s_{\epsilon_{e-k}\,2}$, since the bound is tight in this case (we call this criterion EXCT).

4.3 Dominance Criteria

In this section, we describe dominance criteria that allow us to exclude nodes from the search when determining the sequences $\epsilon^*(E, s)$ and $\tau^*(T, \tau_1, s)$ after having fixed the sets E and T, the first tardy job τ_1 and its starting time s. For brevity, we introduce C_τ for the completion time of the last job of a tardy sequence. Note that this is in general *not* the makespan of sequence τ.

A dominance criterion for determining $\tau^*(T, \tau_1, s)$ can be given as follows:

Theorem 3 *Consider a node $\tau^k = (\tau_1, \ldots, \tau_k)$ in the branching tree for determining $\tau^*(T, \tau_1, s)$ and node (τ^k, i). If there exists a job $\tau_m \in \{\tau_1, \ldots, \tau_k\}$ such that:*

$$C_{(\tau_1,\ldots,\tau_{m-1},i,\tau_{m+1},\ldots,\tau_k,\tau_m)} \leq C_{(\tau^k,i)} \text{ and}$$
$$F(\tau_1, \ldots, \tau_{m-1}, i, \tau_{m+1}, \ldots, \tau_k, \tau_m) \leq F(\tau^k, i),$$

then every completion of the partial sequence $\tilde{\tau} = (\tau_1, \ldots, \tau_{m-1}, i, \tau_{m+1}, \ldots, \tau_k, \tau_m)$ leads to a schedule with an objective function value which is lower than or equal to the one of the analogous completion of sequence (τ^k, i).

The above theorem motivates the following Algorithm SUCC for determining jobs to be added to the partial sequence τ^k in order to generate the successor nodes. It is unlikely that Theorem 3 will be satisfied if $p_{\tau_m\,2} < p_{i\,2}$. Therefore, Algorithm SUCC considers only such m for which $p_{\tau_m\,2} \geq p_{i\,2}$. For all such already sequenced jobs τ_m satisfying the above inequality, we compare sequence (τ^k, i) with $\tilde{\tau}$. Only if Theorem 3 is not satisfied, successor node (τ^k, i) is considered. If $C_{\tilde{\tau}} = C_{\tau^k,i}$ and $F(\tilde{\tau}) = F(\tau^k, i)$, we have to ensure that one of the two possible sequences is retained. This is accomplished by the use of a linked list data structure. An analogous dominance criterion for the determination of $\epsilon^*(E, s)$ can be given (for details see [8, 15]).

The second type of dominance criterion we use has, to our knowledge, never been applied before. Nevertheless, it can be used in any branch and bound algorithm in principle, dealing with scheduling or other combinatorial optimization problems. We describe it for tardy jobs. We identify each partial tardy sequence $\tau = (\tau_1, \tau_2, \dots, \tau_k)$ by the sequence obtained out of τ by sorting the jobs according to increasing *numbers*; recall that each job is identified by a number $i \in \{1, 2, \dots, n\}$. Denote the partial sequence obtained by this order with ν. Let $A = \{\tau^1, \tau^2, \dots, \tau^a\}$ be the set of all a permutations of ν which had been considered before τ. To each such permutation τ^j we assign a tuple (C^j_ν, F^j_ν) $(j = 1, 2, \dots, a)$ of completion time and contribution to the objective function value, hence:

$$C^j_\nu = C_{\tau^j}$$

$$F^j_\nu = \sum_{i=1}^{k} f(c_{\tau^j_i\,2}).$$

If there exists a permutation τ^j, $j \in \{1, 2, \dots, a\}$, with

$$C^j_\nu \le C_\tau \text{ and}$$

$$F^j_\nu \le \sum_{i=1}^{k} f(c_{\tau_i\,2}),$$

then sequence τ is dominated by sequence τ^j. An analogous statement holds for the early jobs. Since there are $\binom{|T|}{k}$ possible sequences ν of length k, we restrict the number of saved tuples (C^j_ν, F^j_ν) in our algorithmical realization to seven, which leads to a memory requirement of roughly 40 MB for running the program. We call this criterion MODS (*memory organized dominating sets*). Since for MODS an efficient implementation is particularly important, the interested reader is referred to [15] for some hints.

Note that MODS cannot replace the criteria based on Theorems 3 and the analogous theorem for determining $\epsilon^*(E, s)$. For instance, the latter are able to cut off a current node by comparison with a node which has not been considered so far. Computational experience justifies the application of both methods.

5 Comparison with Known Results for Regular Problems

In this section, we briefly discuss why the results for $F2||\sum C_i$ are mostly not used in the new algorithm. Consider the determination of sequence τ^* and assume, for simplicity, $d = 0$ in the following. This means that a function fulfilling (M) is just non-decreasing and $f(0) = 0$. For a tardy job sequence, we will now develop a lower bound which is "best" in the sense that it dominates all others which apply to an objective function satisfying (M).

Problem Choice: Given are a job set $\{1, \dots, n\}$ and starting times s_1 and s_2 for the earliest possible start of the first job scheduled on M_1 and M_2, respectively. (If $d = 0$, as we assume in this section, $s_1 = s_2 = 0$). For all $k = 1, \dots, n$ solve the problem: Choose a subset J_k with k jobs of $\{1, \dots, n\}$ such that the resulting completion time of the last scheduled job is minimal over all subsets with k elements.

Problem Choice is NP-hard, as it can be easily seen in what follows. The corresponding decision problem is: For given $k < n$, s_1 and s_2, does there exist a sequence τ consisting of k (different) jobs of $\{1, \dots, n\}$ with $C_{\tau_k} \leq \gamma$? For $s_1 = s_2 = 0$, we get the special case: For given $k < n$, does there exist a sequence τ consisting of k (different) jobs of $\{1, \dots, n\}$ with $C_{\tau_k} = C_{\max}(\tau) \leq \gamma$? An alternative formulation might be as follows: Does there exist a sequence of the jobs $\{1, \dots, n\}$, such that the number of late jobs for a given due date γ is no greater than $n - k$? This question is the corresponding decision problem to $F2|d_i = d|\sum U_i$. The latter has been proven to be NP-hard in the ordinary sense by Jozefowska et al. [13].

Theorem 4 *Let γ_i for $i = 1, \dots, n$ be the minimal completion times mentioned in Problem Choice. Then there is no lower bound which dominates*

$$\sum_{i=1}^{n} f(\gamma_i)$$

for problem $F2||\sum f(C_i)$ which is valid for all f fulfilling (M).

Proof: Suppose, there is another lower bound which dominates the bound given in the theorem. Denote the sequence of non-decreasing (estimated) completion times which is obtained by the former for a certain input by $\delta = (\delta_1, \delta_2, \dots, \delta_n)$. It follows that there is at least one instance for which there exists an index i with $\delta_i > \gamma_i$. We will now construct a counterexample by defining penalty function f as follows:

$$f(x) = \begin{cases} 1 & x > \gamma_i \\ 0 & x \leq \gamma_i \end{cases} .$$

■

It should have become clear from above that the bounds used in the general case are *dominated* by the bounds used in the well-known linear case. It follows that we cannot expect to be able to solve the problem sizes which are tackled in the linear case.

Concerning known lower bounds and dominance criteria for the $F2||\sum C_i$ problem, we only summarize here that most of them cannot be applied to the general case of earliness-tardiness penalty functions and those that can be applied appear to be not competitive in the general case. A detailed comparison and discussion is given in [15].

6 Computational Results

We now discuss in detail the computational results of empirical experiments conducted to determine the effectiveness and efficiency of the proposed dominance conditions and enumerative algorithm in solving reasonably sized problems. In the first subsection, we describe the generation of the test problems. Then, in Subsection 6.2, we investigate the effectiveness of the dominance criteria and give recommendations in which order they should be applied. The efficiency of the proposed algorithm for two different objective functions and different problem types is discussed in Subsection 6.3. Finally a brief discussion of the results for special problems with regular criteria is given. A numerical example for illustrating the presented algorithm is discussed at the end of this section.

6.1 Test Problems

The job processing times are uniformly distributed integers from the interval $[1, p_{\max}]$, where $p_{\max} \in \{5, 20, 50\}$. The common due date was taken to be a function of the maximum load P on the two machines given by:

$$P = \max \left\{ \sum_{i \in N} p_{i\,1}, \sum_{i \in N} p_{i\,2} \right\}.$$

To control the restrictiveness of the due date, the following values of the common due date were used:

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

For the early or tardy completion of a job, we studied the following two penalty functions:

- a linear-quadratic (LQ) penalty function:

$$f(x) = \begin{cases} d - x, & x \le d \\ (x - d)^2, & x > d \end{cases}; \tag{13}$$

- an asymmetric linear-linear (LL) penalty function

$$f(x) = \begin{cases} d - x - t, & x < d - t \\ 0, & x \in [d - t, d] \\ 5(x - d), & x > d \end{cases} \tag{14}$$

  with a time window of width $t = 0.25d$.

6.2 Effectiveness of Dominance Criteria

To evaluate the effectiveness of the proposed dominance criteria, we use ten instances with $n = 18$ jobs each. The processing times are generated from uniformly distributed integers in the range $p_{i\,j} \in [1, 20]$. The objective function is the LQ penalty function given by Equation (13).

6.2.1 Criteria for Nodes of the Main and Subtree

The hierarchy of the criteria for excluding nodes of the main tree or subtree is:

1. MKSP (makespan condition): We check whether for a partition (E, T) of N Inequality (3) is fulfilled; otherwise we do not branch this node of the main tree further;

2. STVS (stronger V-shape condition, see Section 3.4);

3. IS: If $I_S > d$, Inequality (2) is violated, which means that the interval $[I_S, d]$ of the possible starting times of the first tardy job is empty;

4. VS (V-shape condition, see Section 3.4);

5. THEO1: If the conditions of Theorem 1 are satisfied, the node (E, τ_1) is excluded;

6. LFT: This criterion checks whether the lowest possible contribution of the tardy jobs to the objective function value, $F(\tau^*(T, i, I_S))$, plus the costs of an LPT-sequence of the early jobs on M_2 (for $s_{\tau_1 2} = d$) exceeds the best objective function value evaluated so far.

The criteria are ordered in this sequence according to increasing amount of information needed for their application. For example, LFT can only be applied after the selection of the partition (E, T), and after choosing a first tardy job i. However, the STVS criterion applies for nodes of the main tree. The average numbers of nodes cut by the criteria are shown in Table 1.

d/P	MKSP	STVS	IS	VS	THEO1	LFT
0	262,142	0	0	0	0	0
0.1	203,867	0	187	65	0	88
0.25	210,190	375	13,184	10,013	8	7,771
0.5	144,866	39,353	179,310	233,576	94	364,053
0.75	9,115	208,950	25,676	149,312	3	127,424
1.0	1	261,669	0	967	0	0

Table 1: Number of nodes cut by the different criteria

In Table 1 only the nodes are given which MKSP cuts directly, *not* the nodes following in this branch of the main tree. The IS criterion dominates MKSP since every node cut by MKSP would also be cut by IS. The number of nodes cut by IS is smaller only because MKSP is checked first. With the due date becoming less restrictive, the number of nodes excluded decreases clearly. Note that between $d/P = 0.25$ and $d/P = 0.75$, criterion IS is important due to its stricter check for feasibility.

Table 1 also shows that STVS and VS are complementary to MKSP and IS. VS also dominates STVS. An analogous argument as above applies. Apparently, VS is most important in the region where STVS is not strict enough ($d/P = 0.25$ and $d/P = 0.5$). The importance of these criteria increases with increasing d/P because the influence of the flow shop condition decreases. The problem reduces more and more to a one-machine problem which is the relaxation underlying the criteria.

Among the six criteria above, LFT requires the highest computational effort. But if a node cannot be excluded due to this criterion, the value of $F(\tau^*(T, i, I_S))$ is needed in the course of the procedure MINSEARCH anyway. The criterion is most helpful in the region where the feasibility criteria are no more and the one-machine relaxation is not yet very relevant. Criterion THEO1 is — at least for this type of objective function — not of great importance and gives a small contribution only around $d/P = 0.5$.

6.2.2 Criteria for the Branch and Bound Procedures of the Subtree Nodes

Additionally, we study criteria for accelerating the branch and bound algorithms for determining the values $F(\tau^*(T, i, s))$ and $F(\epsilon^*(E, s))$. For the determination of the best sequences of the tardy jobs, we hierarchically implement and investigate

1. Algorithm SUCC;
2. LB2 (see Equation (8));
3. LBM (see Equation (10)).

Note that we do this analysis without applying MODS, since this criterion is not based on *structural* properties. In the branch and bound algorithm, MODS is applied after Algorithm SUCC and before LB2. The analogue is true for the early jobs. For the tardy jobs and $d = 0$, Table 2 shows the nodes cut by the lower bounds LB2 and LBM as well as the ones considered (CONS) in the branch and bound procedures.

Recall that bound LB2 is calculated *before* LBM. Table 2 shows that LBM cuts many more nodes than LB2, except for $k = 2$ (k as in Equation (8)). The ratio of nodes cut by LBM to nodes considered is increasing with k. For $k \geq 5$, LBM cuts more nodes than there remain to be considered. This explains its significant influence on the performance. Table 3 gives the total number of nodes cut by the three criteria for the different values of d/P. Each value is the average of the results of ten instances. Bound LBM does not cut any node for $d/P = 1$. This is due to the underlying relaxation. If the due date becomes less restrictive, the problem reduces more and more to a one-machine problem with the processing times on M_2 as input data. In this case, the SPT-rule with respect to the jobs on M_2 gets tight, whereas the matching relaxation leads to exactly the opposite job sequence but with overlap of the operations. This second bound is significantly lower and checked after the first one.

Algorithm SUCC additionally cuts approximately 10% of all nodes, independently from the restrictiveness of the due date.

depth	2	3	4	5	6
LB2	129	326	2,331	12,287	47,362
LBM	32	545	6,153	36,841	167,668
CONS	145	1,160	6,247	25,114	65,485
depth	7	8	9	10	11
LB2	127,789	260,038	331,486	313,897	183,317
LBM	424,734	888,848	1,165,000	1,263,610	1,013,600
CONS	136,013	186,494	207,028	169,522	112,138
depth	12	13	14	15	16
LB2	71,763	25,380	6,920	2,852	356
LBM	636,508	399,982	153,803	47,775	7,323
CONS	76,695	34,810	13,326	2,678	356

Table 2: Efficiency of the lower bounds for the tardy jobs depending on the number of nodes already sequenced (k in Equation (8))

d/P	0	0.1	0.25	0.5	0.75	1
LB2	1,386,230	5,493,130	8,860,500	5,480,560	290,414	2,141
LBM	6,212,420	17,999,900	22,809,300	31,281,900	1,070,350	0
CONS	1,037,210	3,208,250	4,802,490	6,265,000	200,209	345

Table 3: Efficiency of the lower bounds for the tardy jobs depending on the restrictiveness

Finally, we discuss the following criteria used for the determination of the best sequences of the early jobs (hierarchy as given):

1. Algorithm SUCC in a variant for the early jobs;
2. LBE (see Equation (11));
3. EMSP (see Equation (12));
4. EXCT (see end of Section 4.2).

The number of nodes cut by the last three criteria as well as the number of nodes considered (CONS) are given in Table 4.

The values are the average of ten instances. Obviously, the number of nodes considered increases dramatically with the increase of d/P from 0 to 0.75. For $d/P = 1$, the number is lower than for $d/P = 0.75$. The reason for this is the efficiency of the criteria for the nodes of the main tree and the subtree, especially STVS (see the analysis in the preceding subsection). The most effective criteria are LBE and EMSP.

The variant of Algorithm SUCC adapted for the early jobs showed a different behaviour than the variant for the tardy jobs. In Table 5, we give the number of nodes which were tested and which were cut, each as average over 10 instances.

d/P	0	0.1	0.25	0.5	0.75	1
LBE	0	1	365	157,184	9,692,170	1,585,700
EMSP	0	1	447	103,782	5,907,260	511,360
EXCT	0	0	125	1,752	11,183	0
CONS	0	1	455	98,548	4,831,230	545,451

Table 4: Efficiency of the criteria for the early job sequences

d/P	0	0.1	0.25	0.5	0.75	1
CONS	0	0	693	227,480	11,318,900	1,579,920
CUT	0	0	54	47,460	3,626,880	198,004

Table 5: Number of nodes tested and cut by a variant of Algorithm SUCC for the early jobs

6.3 Efficiency of the Proposed Enumerative Algorithm

To test the efficiency of the proposed enumerative algorithm, we generated 2,880 problem instances, representing 20 instances for each value of the number of jobs $n \in \{14, 16, 18, 20\}$, the maximum job processing time $p_{\max}$, the due date d and each objective function. The time limit of the computation was 15 minutes for each instance.

The computational results are given in Tables 6 and 7 without and with criterion MODS for an Intel Celeron 466MHz Linux-PC. SP is the number of solved instances within the time limit and AVE the average computation time in seconds for the instances solved within the time limit. Unsolved instances are not considered in the computation times.

We first discuss the results in absence of criterion MODS, given in Table 6. Nearly all instances with the LL penalty function up to a value of $d = 0.5P$ can be solved. Problems with $d = 0.75P$ are most difficult. Except for this case, the influence of the range of processing times is rather insignificant. This result confirms the efficiency of the bisection search applied in the enumeration procedure. Thus, for the LL penalty function, problems with restrictive due dates, which should be of highest practical interest, are easier to solve than problems with a less restrictive due date.

In contrast, problems with the LQ penalty function are especially hard to solve for $d = 0.25P$, whereas the instances with $d = 0.75P$ are by far less difficult than with the LL penalty function. Furthermore, for $d = 0.75P$, the algorithm can solve more problems with the LQ penalty function than with the LL penalty function. The influence of the restrictiveness on the performance is less than in the LL case.

The improvements due to applying MODS (see Table 7) are impressive for $d = 0, 0.1P, 0.75P, 1.0P$. For $d = 0.25P$ and $d = 0.5P$, the performance of the algorithm with MODS is more or less comparable to the performance without

n	$d=0$		$d=0.1P$		$d=0.25P$		$d=0.50P$		$d=0.75P$		$d=1.0P$	
	SP	AVE	SP	AVE	SP	AVE	SP	AVE	SP	AVE	SP	AVE
	$p_{i\,j} \in [1,5]$, LQ											
14	20	1	20	1	20	1	20	2	20	7	20	37
16	20	2	20	2	20	3	20	5	19	17	18	41
18	20	2	20	3	20	7	20	26	13	90	19	21
20	18	110	14	242	7	104	8	98	10	316	16	58
	$p_{i\,j} \in [1,20]$, LQ											
14	20	1	20	1	20	4	20	2	20	2	20	3
16	20	8	20	38	20	106	20	33	20	21	20	15
18	20	46	19	176	11	141	20	221	19	83	20	107
20	16	208	9	190	6	136	8	62	15	315	16	114
	$p_{i\,j} \in [1,50]$, LQ											
14	20	1	20	1	20	3	20	2	20	1	20	2
16	20	7	20	28	20	89	20	35	20	4	20	9
18	20	43	19	192	13	189	19	209	20	166	20	100
20	12	148	8	188	6	48	9	169	19	138	16	144
	$p_{i\,j} \in [1,5]$, LL											
14	20	1	20	1	20	1	20	2	19	6	20	37
16	20	2	20	2	20	3	20	5	19	17	18	41
18	20	2	20	3	20	7	20	26	11	106	10	2
20	20	14	20	31	19	61	19	248	8	203	14	8
	$p_{i\,j} \in [1,20]$, LL											
14	20	1	20	1	20	1	20	2	20	29	20	66
16	20	1	20	2	20	2	20	9	16	164	18	62
18	20	4	20	7	20	17	20	50	3	29	9	40
20	20	35	19	59	18	109	16	334	2	183	8	7
	$p_{i\,j} \in [1,50]$, LL											
14	20	1	20	1	20	1	20	2	20	25	20	39
16	20	2	20	2	20	4	20	19	14	147	16	187
18	20	3	20	4	20	9	19	64	8	144	10	8
20	20	20	20	59	19	109	17	392	3	163	8	23

Table 6: Computational results without criterion MODS

n	$d=0$		$d=0.1P$		$d=0.25P$		$d=0.50P$		$d=0.75P$		$d=1.0P$	
	SP	AVE	SP	AVE	SP	AVE	SP	AVE	SP	AVE	SP	AVE
	$p_{i\,j} \in [1,5]$, LQ											
14	20	1	20	2	20	4	20	6	20	2	20	1
16	20	2	20	4	20	31	20	46	20	10	20	2
18	20	4	20	14	20	147	20	246	20	47	20	3
20	20	22	19	125	9	272	9	229	20	316	20	8
	$p_{i\,j} \in [1,20]$, LQ											
14	20	1	20	2	20	4	20	4	20	2	20	2
16	20	2	20	7	20	48	20	37	20	8	20	2
18	20	9	20	45	18	288	20	233	20	37	20	7
20	20	54	17	281	6	175	8	86	20	146	20	12
	$p_{i\,j} \in [1,50]$, LQ											
14	20	1	20	2	20	3	20	3	20	2	20	2
16	20	3	20	8	20	49	20	40	20	4	20	2
18	20	7	20	38	17	245	20	240	20	18	20	6
20	20	44	18	294	6	58	8	124	19	61	20	13
	$p_{i\,j} \in [1,5]$, LL											
14	20	1	20	1	20	1	20	2	20	2	20	1
16	20	1	20	2	20	4	20	8	20	9	20	2
18	20	2	20	3	20	10	20	38	20	49	20	4
20	20	7	20	20	18	39	18	272	19	193	20	14
	$p_{i\,j} \in [1,20]$, LL											
14	20	1	20	1	20	1	20	2	20	6	20	3
16	20	1	20	2	20	3	20	11	20	32	20	4
18	20	3	20	5	20	20	20	52	19	166	20	21
20	20	10	20	36	19	142	17	265	12	302	20	39
	$p_{i\,j} \in [1,50]$, LL											
14	20	1	20	1	20	1	20	2	20	5	20	2
16	20	2	20	2	20	5	20	11	20	35	20	4
18	20	3	20	4	20	11	20	57	17	153	20	17
20	20	9	20	32	19	125	19	323	11	321	20	37

Table 7: Computational results with criterion MODS

i	1	2	3	4	5	6
M_1	18	10	2	6	19	5
M_2	2	3	4	6	9	14

Table 8: Processing times of the example

MODS. With MODS, all instances for $d = 0$ can be solved even for the LQ objective function. Also, *all* instances with $d = 1.0P$ can be solved for both objective functions.

For problems with the LL penalty function, the inclusion of specific properties as described in the literature on problem $F2||\sum C_i$ can speed up the algorithm. Della Croce et al. [4] were able to solve problems up to 30 jobs for the regular problem $F2||\sum C_i$ within a time limit of one minute for each instance with processing times from the interval [1,10]. In general, the algorithm presented in this article is able to solve most problems with up to 20 jobs satisfactorily for very general objective functions.

6.4 Performance for Regular Problems with Larger Inputs

We tested the enumerative algorithm for regular problems, i.e. $d = 0$, also for larger input sizes. For that we modified the algorithm in two ways. First, we included job τ_1 in the search tree for the tardy jobs. Second, criterion MODS saves the tuples (C^j_ν, F^j_ν) for nodes with up to only seven jobs due to the restricted memory size on the PC. For the linear penalty function, we could solve all 20 instances with $n = 22$ jobs. For $n = 24$, we obtained all optimal solutions for the instances with processing times from the interval $[1, 5]$. For the interval $[1, 20]$ ($[1, 50]$), the algorithm did not find an optimal solution in two (one) cases.

For the quadratic penalty function, the number of solved instances decreased clearly with increasing width of the interval of processing times. For $n = 22$ and the intervals $[1, 5]/[1, 20]/[1, 50]$, the algorithms found an optimal solution in 17/17/14 cases within the time limit of 15 minutes. The corresponding values for $n = 24$ are 10/6/3.

6.5 A Numerical Example

Finally we illustrate the components of our algorithm by briefly describing the solution of a six-job numerical example. The processing times of the six jobs are given in Table 8. The jobs are ordered according to increasing processing times on M_2.

The penalty function is of the linear-quadratic form given by Equation (13). Let $d = 30$ be the common due date. According to Figure 3, we have to start with $E = \emptyset$. As pointed out in Section 3.1, there are differences in the treatment of the nodes with $E = \emptyset$ and $E \neq \emptyset$. For ease of implementation, we check *all* nodes with $E = \emptyset$ at first, whereas for $E \neq \emptyset$ we perform a depth-first search as described in Figures 3 and 4. Furthermore, we do not check all our sophisticated dominance criteria for $E = \emptyset$. Most of these criteria are especially effective if the best calculated objective function value so far is near the optimum. This is unlikely to happen for the first calculated nodes in general. This simplification is justified by the computational results. For high d/P, the bound LB2 in the branch and bound algorithm is tight for $E = \emptyset$. Therefore, the computation time spent on nodes with $E = \emptyset$ is negligible in this case. For $d = 0$ or small d/P, the algorithm performs very well anyway (see the discussion of Table 6).

The first job τ_1 we consider is 1. The algorithm yields the schedule displayed in Figure 6. The idle time on M_2 is relatively small, which shows that LB2 is

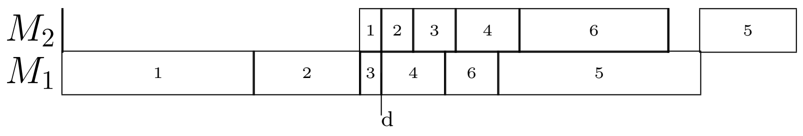

Figure 6: First Gantt-chart of the example

rather tight as mentioned above.

The first node of the subtree which is cut is $E = \{1\}$ and $\tau_1 = 5$. This is due to criterion IS: Inequality (2) is violated since $p_{1\,1} + p_{5\,1} = 37 > d = 30$.

The following node of the subtree is $E = \{1\}$ and $\tau_1 = 6$. Despite the fact that this node is feasible, it is excluded from consideration before procedure MINSEARCH is performed: The lowest possible starting time of job $\tau_1 = 6$ on M_2 is (see Equation (1))

$$\max\{30 - 14, 18 + 5, 20\} = 23.$$

Hence, we calculate the lowest possible contribution of the tardy jobs to the objective function value via the previously described branch and bound procedure with a starting time of $23 + p_{6\,2} = 37$ which yields 2217. To this value, we have to add the function value $f(37) = 7^2 = 49$, the penalty for the tardiness of job 6. The lowest possible contribution of the one early job is 0. This is *always* the case if E contains only one element, since in this case the completion time can be set to the due date. Criterion LFT (see Item 6 in Section 6.2.1) checks whether this value exceeds the best objective function value calculated so far, 2249. Hence, this node is cut due to criterion LFT.

Another dominance criterion can be demonstrated by node $E = \{2\}$,$\tau_1 = 5$. The current upper bound is 1775. The lowest possible starting time of job 5 on M_2 is 29. Now, criterion VS is applied. The contribution of the early job to the objective function value is 0 again. The contribution of the tardy jobs to the objective function value is estimated via the SPT-rule for the jobs on M_2 after job 5. Since the SPT-sequence is (1,3,4,6) and the lowest possible starting time of job 5 is 29, we get as an estimate

$$8^2 + (8 + 2)^2 + (8 + 2 + 4)^2 + (8 + 2 + 4 + 6)^2 + (8 + 2 + 4 + 6 + 14)^2 = 1916$$

which is higher than the current upper bound. Continuing the above process throughout the search tree results in an optimal schedule displayed in Figure 7. Its objective function value is 1383.

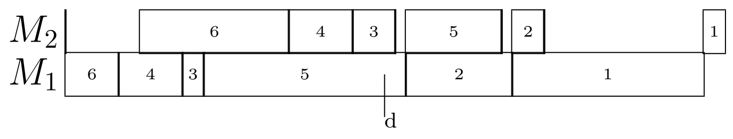

Figure 7: An optimal schedule

7 Conclusions

This article considered the two-machine flow shop problem with a non-regular optimization criterion. Since the problem is NP-hard in the strong sense, we developed and tested an enumerative algorithm for the two-machine flow shop problem to minimize an arbitrary function of the weighted sum of earliness and tardiness penalties. Depending on the job characteristics, the algorithm proposed in this paper can satisfactorily solve problems with up to 20 jobs. This is a significant improvement compared to results obtained by Majaj [18] who was able to solve instances with six jobs in 75 minutes of CPU time on a PRIME mainframe computer using the LINDO [20] package; samples with five jobs took about 12 minutes.

Apart from our restriction to a common due date for all jobs, the considered penalty function fulfilling a weak monotonicity property allows to model all reasonable criteria. In this sense, the problem considered in this article is rather general. Even on the field of regular sum criteria this paper provides new results, since, to the best of our knowledge, only sum objective functions, based on *linear* terms in completion times have been studied so far. Most of the bounds and dominance criteria which have been developed for this special case do not apply to the more general case of a non-decreasing penalty function. A new dominance criterion, MODS, yields an impressive improvement. In principle, it can be used in *any* enumerative algorithm, but the storage requirement is high. Thus, it is particularly useful in case of problems for which only small input sizes can be tackled.

For special penalty functions, the algorithm may be refined. For example, considering linear earliness and tardiness penalties, the branch and bound algorithm for determining sequences ϵ^* and τ^* can be made more effective as especially designed dominance criteria can be included and the computation of bounds used in the bisection search can be simplified.

It suggests itself to develop powerful heuristics to tackle larger instances for the two-machine flow shop problem. This has been done in another paper [16].

The present work also indicates some directions for future research. Firstly, the generalization towards job-specific due dates, and the use of different concepts of job earliness that incorporate the different values of work-in-process at various stages should be explored. It should be possible to use the algorithm given in this article for local optimization in a heuristic for some of these problems, for instance in problems where the set of jobs may be divided in several subsets of jobs which have to be scheduled around the same or similar due dates. Secondly, the development of strategies to solve m-stage flow shop problems with non-regular criteria is both interesting and useful. Finally, developing techniques for more complex multi-stage, multi-machine scheduling problems, like those in the open and job shop environments, would enhance the practicability of scheduling theory.

References

[1] N.R. Achuthan, J. Grabowski and J. Sidney, Optimal Flow-Shop Scheduling with Earliness and Tardiness Penalties. *Opsearch* **18**, 117 - 138, (1981).

[2] K.R. Baker and G.D. Scudder, Sequencing with Earliness and Tardiness Penalties, *Operations Research* **38**, 22 - 36, (1989).

[3] P. Brandimarte and M. Maiocco, Job Shop Scheduling with a Non-regular Objective: a Comparison of Neighbourhood Structures Based on a Sequencing/Timing Decomposition. *Intern. J. of Prod. Res.* **37** (8), 1697 - 1715, (1999).

[4] F. Della Croce, V. Narayan and R. Tadei, The Two-Machine Total Completion Time Flow Shop Problem. *Eur. J. Oper. Res.* **90**, 227 - 237, (1996).

[5] M.R. Garey, D.S. Johnson and R. Sethi, The Complexity of Flowshop and Jobshop Scheduling. *Math. Oper. Res.* **1**, 117 - 129, (1976).

[6] R.L. Graham, E.L. Lawler, J.K. Lenstra and A.H.G. Rinnooy Kan, Optimization and Approximation in Deterministic Sequencing and Scheduling. *Annals of Discr. Math.* **5**, 287 - 326, (1979).

[7] K. Gowrishankar, C. Rajendran and G. Srinivasan, Flow Shop Scheduling Algorithms for Minimizing the Completion Time Variance and the Sum of Squares of Completion Time Deviations from a Common Due Date. *European J. Oper. Res.* **132**, 643 - 665, (2001).

[8] 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. Preprint 33/99, FMA, Otto-von-Guericke-Universität Magdeburg, FMA, (1999).

[9] N. Hall, W. Kubiak and S. Sethi, Earliness-Tardiness Scheduling Problems, ii: Deviation of Completion Times About a Common Due Date. *Oper. Res.* **5**, 847 - 856, 1991.

[10] N. Hall and M. Posner, Earliness-Tardiness Scheduling Problems, i: Weighted Deviation of Completion Times About a Common Due Date. *Oper. Res.* **5**, 836 - 846, (1991).

[11] 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).

[12] S.M. Johnson, Optimal Two and Three Stage Production Schedules with Setup Times Included. *Naval Res. Logist. Quart.* **1**(1), 61 - 68 (1954).

[13] J. Jozefowska, B. Jurisch and W. Kubiak, Scheduling Shops to Minimize the Weighted Number of Late Jobs. *Oper. Res. Lett.* **16**(5), 277 - 283, (1994)

[14] J.J. Kanet, Minimizing the Average Deviations of Job Completion Times About a Common Due Date. *Naval Res. Logist. Quart.* **28**, 643 - 651, (1981).

[15] V. Lauff, Multi-Stage Scheduling Problems with Non-Regular Optimization Criteria. PhD Thesis, Otto-von-Guericke-University Magdeburg, Germany, (2001).

[16] V. Lauff and F. Werner, Heuristics for Two-Machine Flow Shop Problems with Earliness and Tardiness Penalties. Otto-von-Guericke-Universität Magdeburg, FMA, Preprint 01/01, (2001) (to appear in International Journal of Operations and Quantitative Management).

[17] V. Lauff and F. Werner, On the Complexity of Some Properties of Multi-Stage Scheduling Problems with Earliness and Tardiness Penalties. *Comput. Oper. Res.* **31**, 317 - 345, (2004).

[18] S.A. Majaj, Just-in-Time Production for the Two Machine Common Due Date Scheduling Problem. Master's Thesis, Univ. of Southern Colorado, Pueblo, CO, (1991).

[19] H. Sarper, Minimizing the Sum of Absolute Deviations About a Common Due Date for the Two-Machine Flow Shop Problem. *Appl. Math. Modelling* **19**, 153 - 161, (1995).

[20] L. Schrage, *Linear, Interactive and Discrete Optimization (LINDO)*, (1989).

[21] Y.N. Sotskov, T. Tautenhahn and F. Werner, On the Application of Insertion Techniques for Job-Shop Problems with Setup Times. *RAIRO Rech. Oper.* **33**, 209 - 245, (1999).