Scheduling with Common Due Date, Earliness and Tardiness Penalties for Multi-Machine Problems: A Survey

Volker Lauff and Frank Werner[1]

Otto-von-Guericke-Universität, Fakultät für Mathematik,
PSF 4120, 39016 Magdeburg, Germany

Abstract: The just-in-time production philosophy has led to a growing interest in scheduling problems considering both earliness and tardiness penalties. Most publications considering these so-called non-regular objective functions are devoted to single machine problems. In the case of multi-machine problems, there are some papers dealing with parallel machine problems. However, for multi-operation scheduling problems such as shop scheduling problems, investigations have begun only recently. In this paper, inspired by recent own work (partly yet unpublished), we survey some results on multi-machine scheduling problems with a given common due date, where the focus is on possible approaches for shop scheduling problems.

Key words: Multi-machine scheduling, common due date, nonregular criteria, earliness-tardiness penalties

1 Introduction

Due to the increasing awareness of the just-in-time production philosophy, there is a need for a generalization of the commonly used objective functions in scheduling. Traditionally, tardiness penalties due to delivery after a contractually arranged due date are considered. This approach neglects storage costs due to insurance, theft, perishing, and bounded capital for the case of a completion of the job before the due date. Therefore, tardiness together with earliness penalties are considered in order to meet the requests of practice.

Although there are published a lot of results on single machine problems with earliness and tardiness penalties (see e.g. [1, 2, 3]), there are only some papers dealing with parallel machine problems and only very few papers treating shop scheduling problems with earliness and tardiness penalties.

The problem class considered in this paper is as follows. n jobs $1, 2, \ldots, n$ have to be processed on a set of m machines $M_1, M_2, \ldots, M_m$. In the case of a parallel machine problem, each job consists of a single operation which has to be processed exactly on one of the identical, uniform or heterogeneous machines. In a shop scheduling problem, the jobs consist of several operations. The cases of a job shop (each job has a given technological route in which the jobs are processed by the machines), flow shop (special case of the job shop problem, where each job has the same given technological route which is assumed to be $M_1 \rightarrow M_2 \rightarrow \ldots \rightarrow M_m$) and an open shop (the sequence in which the operations of a job are processed is immaterial) are distinguished in the literature. Without loss of generality, we restrict here to the case that each job has to be processed at most once on a machine, i.e. we can describe an operation

[1]Corresponding author, E-mail: frank.werner@mathematik.uni-magdeburg.de

which represents the processing of job i on machine M_j by (i,j). The processing time p_{ij} of each operation (i,j) is assumed to be fixed and given. Once an operation is started, it may not be interrupted. The starting time of job i on machine M_j is denoted by s_{ij}, the completion time by $c_{ij} = s_{ij} + p_{ij}$. In the case of a single machine problem, the machine index is dropped and the processing time of job i is denoted by p_i. The completion time of the last operation of a job i is denoted by C_i.

For each job i, either a specific due date d_i is assigned by which this job should be ideally completed or a common due date d is assigned to all jobs. In both cases, there is an earliness (tardiness) cost specified by a general (possibly nonlinear) function of the amount of time a job i is completed before (after) the due date. The penalty for some deviation from the due date may be in general different for each job, and also asymmetry of the earliness-tardiness penalty is possible. In most cases, the sum of the earliness and tardiness penalties has to be minimized, but sometimes also the maximum of the earliness and tardiness penalties should be minimal.

A common due date d is assigned to all jobs if there is a unique time by which all the jobs must be ideally completed. Algorithms for scheduling problems with different due dates are fundamentally different from solution strategies for problems with a common due date for all jobs. Moreover, the latter problems are still harder and there are not many results for shop scheduling problems with different due dates so far. Therefore, we restrict our overview to problems with a *common due date*. Such problems also occur as subproblems if the j-th subset of jobs should be ideally completed at the same due date D_j. Among problems with common due date, there are again two types of problems. Either the due date is given in advance, or the due date may be fixed by the scheduler. The latter types of problems are denoted as due date assignment problems, and in addition to the earliness-tardiness penalty terms a due date assignment cost is also considered in the objective function. Recently, Gordon et al. [4] gave an overview on such common date date assignment and scheduling problems, where also problems with given common due date as well as regular optimization criteria are reviewed. However, in the latter paper the focus is on single machine problems.

In this paper, we exclusively deal with *multi-machine* problems with *given* common due date d. In the following we use the 3-parameter classification introduced by Graham et al. [5] in order to describe the problems considered. For sake of brevity, we will not give the condition $d_i = d$ explicitly in the problem notation if the common due date occurs explicitly in the objective function.

The paper is organized as follows. In the literature, problems with restrictive and non-restrictive due date have been considered. Since these terms have to be defined for multi-operation scheduling problems carefully, we discuss these terms in detail in Section 2. In Section 3, some results for parallel machine scheduling problems with given common due date are reviewed. When considering shop scheduling problems, there are two basic models possible. Either, as in the single and parallel machine problems, earliness and tardiness is measured only by the deviation of the completion time of a job from the given due date, or the time the job is in the system is also considered. In the latter case, one possibility is to ‘penalize’ the waiting times between the processing of any two successive operations. This is important if storage or holding costs arise. Both types of problems are considered in Section 4. In Section 5, an outlook and some topics for future work are given.

2 On the Restrictivity of the Common Due Date

In the case of single-operation scheduling problems, the cases of a restrictive and an non-restrictive common due date have been considered. In this section we will discuss this notation a bit more in detail and give a definition for multi-operation problems introduced by Lauff and Werner [6].

For a single machine problem, Kanet [3] assumes in his article that $d \geq \sum_{i=1}^{n} p_i$. In his paper, he calls such a due date "restricted" in order to express that this due date cannot be chosen arbitrarily. Nevertheless, the term "restrictive due date" is used in contradictory meanings in the papers which appeared later. This is due to the fact that the value of the due date relative to the problem data strongly influences the validity of algorithms developed for a certain problem type. Whereas Kanet's algorithm is not valid for relatively small due dates, the branch and bound algorithm developed by Bagchi et al. [1] assumes that the due date is sufficiently small. In general, the value of the due date may influence the computational complexity. From another point of view, we see that under reasonable monotonicity assumptions the optimal objective function value of a certain problem cannot increase by increasing the due date while keeping constant all other problem data.

In the literature, both aspects – computational complexity and optimal objective function value – are considered. Baker and Scudder [2], referring to a result by Bagchi et al. [1], strengthen Kanet's result by calling a due date d for a single machine problem non-restrictive if $d \geq \Delta$, where

$$\Delta = \begin{cases} p_n + p_{n-2} + p_{n-4} + \cdots + p_2, & \text{if } n \text{ is even} \\ p_n + p_{n-2} + p_{n-4} + \cdots + p_1, & \text{if } n \text{ is odd} \end{cases},$$

which is in fact the minimal due date for which a Kanet-schedule remains to be feasible. This is restrictivity rather from the point of view of complexity.

On the other hand, Webster [7] states: "If the optimal cost cannot decrease with increases in the common due date, then the problem is an unrestricted common due date problem... The restricted common due date problem is generally much more difficult to solve."

These definitions are compatible with some problems with a single machine. As an example, consider the problem $1||\sum |C_i - d|$. If $d < \Delta$, the optimal objective function value is in general higher than the objective function value for the Kanet-schedule with $d \geq \Delta$. Nevertheless, the two definitions are different for more complicated problems. The lowest possible optimal objective function value of problem $F4||\sum |C_i - d|$ is the same as the objective function value of the Kanet-schedule with the processing times on M_4 of the jobs as input data. The question whether there exists a schedule with such an objective function value for a certain due date is NP-hard in the strong sense. Consider as an example the following special case: Let the processing times on M_4 be zero for all jobs. The optimal objective function value is zero. Let C^o_{max} be the optimal objective function value of problem $F3||C_{max}$ with the processing times on the first three machines of the jobs of the original problem as input data. Obviously, $d = C^o_{max}$ is the minimal due date for which the optimal objective function value of problem $F4||\sum |C_i - d|$ can be reached. Problem $F3||C_{max}$ is NP-hard in the strong sense [8]. Hence, it appears that Webster's definition is not appropriate for multi-stage scheduling problems.

Lauff and Werner [6] propose the following definition, which provides that both aspects are preserved for multi-stage systems:

Definition 1 *For a scheduling problem with certain input data denote with F^*_d the optimal objective function value for a due date value d. Further, let $\mathcal{S}_d$ be the set of all optimal schedules* *for a due date d. A due date $\tilde{d}$ is called non-restrictive if*

$$F^*_{\tilde{d}} = \min_d \{F^*_d\}$$

and if there exists a schedule $\tilde{S} \in \mathcal{S}_{\tilde{d}}$ with the property that there is no d which fulfills the two conditions

- *$F^*_{\tilde{d}} = F^*_d$ and*
- *there is a schedule $S_d \in \mathcal{S}_d$ which can be determined with lower computational complexity than $\tilde{S}$.*

Speaking informally: A due date is non-restrictive, if and only if by increasing the due date neither the objective function value nor the complexity of the optimization problem can be reduced.

3 Parallel Machine Problems

Early results about parallel machine problems have already been summarized by Baker and Scudder [2] in 1990. It is well known that in the case of minimizing the sum of the absolute deviations from a non-restrictive common due date, on each machine there is no inserted idle time and the optimal schedule is V-shaped. Moreover, on each machine one job completes precisely at the due date. The non-restrictive problem on parallel identical machines has been investigated by Hall [9] and Sundaraghaghavan and Ahmed [10], and an algorithm that generalizes Kanet's algorithm has been given. The longest jobs are assigned to different machines and are sequenced first on the corresponding machine. The next $2m$ largest jobs should be assigned in any order to the second and last positions on the machines, the next $2m$ jobs should occupy the third and second-to-last positions, and so on. If the final set of jobs has less than $2m$ jobs, they can be assigned to any subset of the next $2m$ positions. It is worth noting that this procedure generates a large number of alternative optima.

Emmons [11] extends this analysis to the case of different earliness and tardiness penalties, where earliness and tardiness are penalized by two different weights a for late completion and b for early completion. Moreover, he considered also problems with uniform parallel machines, i.e. each machine has its own efficiency factor. The procedure sketched above can be easily adapted to this case by utilizing the smallest-to-largest matching of coefficients (i.e. positional weights) and processing times, taking into account that the objective function F can be written as

$$F = \sum_{k=1}^{m} \left[\sum_{j=1}^{n_{1k}} \frac{b(j-1)}{s_k} p_{[jk]} + \sum_{j=1}^{n_{2k}} \frac{aj}{s_k} p_{(jk)} \right],$$

where s_k is the speed of machine M_k, n_{1k} is the number of jobs processed on M_k before the due date, n_{2k} is the number of jobs processed on M_k after the due date, $[jk]$ is the j-th job in the sequence (processed before the due date) and (jk) is the j-th last job in the sequence (processed after the due date) on M_k $(k = 1, 2, \ldots, m)$. Emmons also considers secondary criteria problems such as the minimization of makespan or the attainment of a minimum due date, and for special cases of the latter problems solutions are provided.

Most results for parallel machine problems in the 90ies have been obtained for identical machines. Federgruen and Mosheiev [12, 13] consider such problems first with a general (but symmetric) convex penalty function that is non-decreasing in the absolute deviation of the completion time of a job from the given common due date. In [12], the case of minimizing the sum of the penalty functions is considered. The authors derive a tight lower bound computable in $O(n \log n)$ time. Then the Alternating Schedule heuristic is presented which uses the idea given above for the problem of minimizing the sum of absolute deviations from the due date and works as follows. Assume that the jobs are numbered according to non-decreasing processing times and for simplicity, we assume that the number of jobs n is a multiple of the number of machines m (otherwise dummy jobs of zero processing time are added). Then this heuristic schedules the jobs in subsets each of size m. In each subset, the j-th job is scheduled on machine M_j. In the case that n is an odd multiple of m, the first m (i.e. the shortest) jobs are scheduled on the corresponding machine such that each job completes precisely at the due date. Then the second subset (i.e. jobs $m+1, m+2, \ldots, 2m$) are scheduled such that they are processed directly after the already scheduled jobs on the machines. Then the jobs of the next subset (i.e. jobs $2m+1, 2m+2, \ldots, 3m$) are scheduled directly before the first job on each machine, and so on. If n is an even multiple of m, the first m jobs are scheduled such that exactly half of the processing time of each of these jobs is performed until the due date, the remaining jobs are then scheduled alternatingly at the head and tail of the partial schedule as described before. The Alternating Schedule heuristic requires $O(n \log n)$ time.

Finally, the presented heuristic is generalized to the case of a possibly restrictive due date and asymmetric, possibly non-convex earliness-tardiness penalty functions. In this case, penalties are specified by two distinct non-negative functions, a function F non-decreasing in the earliness of job i (i.e. non-increasing in the completion time $C_i \le d$) and a function G non-decreasing in the tardiness. The Modified Alternating Schedule heuristic MASH for this problem distinguishes again the cases when the number of jobs is an odd or an even multiple of the number of machines m. In the first case, the first m scheduled jobs complete precisely at the due date. The remaining jobs $m+1, m+2, \ldots, n$ are scheduled such that each job is either directly preceding or directly following the partial schedule constructed so far, wherever among the $2m$ possible choices the resulting cost for this job is minimal. In the case that n is an even multiple of m, only the scheduling of the first m jobs is different. Job j processed on machine M_j is scheduled such that it completes at C_j with $d \le C_j \le d + p_j$ such that

$$G(C_j - d) + F(d - C_j + p_j) = \min_x \{G(x - d) + F(d - x + p_j)\}.$$

Thus, in both cases the jobs are scheduled in a greedy manner and finally, the schedule on each machine is possibly shifted by varying its starting time to the optimal value. Since in the case of a restrictive due date no efficiently computable bounds are available, the MASH heuristic has only been tested on single machine problems (with up to 200 jobs).

In [13], Federgruen and Mosheiev consider similar problems but with minimizing the maximum of the absolute deviations of the completion times from the common due date. For a non-restrictive due date it is shown that the problem decomposes into two parts: Each of the m largest jobs is assigned to a different machine to be processed first. All remaining jobs are assigned to the machine so as to minimize their makespan (i.e. the maximum of the workload should be minimal). The latter problem is NP-hard, and therefore, heuristics can be used for finding an approximate solution. One such heuristic is the LPT rule, i.e. the jobs are assigned in decreasing order of their processing times to a machine with the smallest assigned workload so far. Finally, on each machine M_j the optimal starting time of the first job has to be found which is obtained as the root of the equation

$$F(d-x) = G(x + S^j - d),$$

where S^j is the total workload (except the long job assigned in the first step) of machine M_j, provided that $F(d) \geq G(S^j - d)$. Otherwise, the first job on machine M_j starts at time zero.

For a restrictive due date, the situation is a bit more complicated in the sense that any set of jobs may need to be processed first on the m machines (not necessarily the m largest jobs). In the heuristic given in [13], $n-m$ such sets are considered, namely assuming that the jobs are numbered according to non-increasing processing times in the jth run, $j \in \{1, 2, \ldots, n-m)$, jobs $j, j+1, \ldots, j+m-1$ are taken as the initial jobs on the m machines, and as before, then the remaining jobs are scheduled so as to minimize their makespan value (e.g. by using the LPT rule for a heuristic solution of this hard problem). For both cases considered above, numerical results are given for problems with up to $n = 500$ jobs and $m = 5$ machines. It has been found that the average optimality gap for each type of problems considered is always below 4.7 %.

Mosheiov and Shadman [14] consider the problem of minimizing the maximum of the weighted deviation from the given common due date on identical parallel machines under the assumption of unit processing times for all jobs. In the case of a non-restrictive common due date, an optimal solution can be found in time which is polynomial in the number of jobs. In the case of a restrictive due date, an optimal solution can be found still in time which is polynomial in the number of jobs, however exponential in the number of machines. Both procedures are based on finding the optimal early part $\alpha_j \in [0, 1]$ of the 'crossing due date' job on machine M_j $(j = 1, 2, \ldots, m)$.

For a non-restrictive due date, the case $n < 2m$ has a trivial solution. For the case $n \geq 2m$, it is proven that on all machines, inequality $\alpha_j \in [0.5, 1]$ holds. Assuming that n is a multiple of m (possibly obtained by adding dummy jobs), it is shown that there exists a cyclic optimal schedule as follows provided that jobs and machines are numbered such that for the given job weights w_i inequalities $w_1 \geq w_2 \geq \ldots \geq w_n$ hold and $1 \geq \alpha_1 \geq \alpha_2 \geq \ldots \geq \alpha_m \geq 0.5$:

$$\begin{array}{rcllllll}
\pi^1 & = & (\ldots, & 4m, & 2m, & 1, & 2m+1, & \ldots), \\
\pi^2 & = & (\ldots, & 4m-1, & 2m-1, & 2, & 2m+2, & \ldots), \\
\ldots & & & \ldots & \ldots & & & \\
\pi^{m-1} & = & (\ldots, & 3m+2, & m+2, & m-1, & 3m-1, & \ldots), \\
\pi^m & = & (\ldots, & 3m+1, & m+1, & m, & 3m, & \ldots),
\end{array}$$

where π^j denotes the job sequence on machine M_j. In this way, m single machine problems are obtained, each with n/m jobs. It remains to determine the optimal α_j value for each machine which reduces to a two-variable linear programming problem solvable in $O(n)$ time. Thus, the overall algorithm including the initial sorting of the jobs requires $O(n \log n)$ time.

In the case of a restrictive due date, starting processing on all machines at time zero does not guarantee optimality (in the case that a restrictive due date implies starting processing on all machines at time zero, a simple cyclic schedule allows a polynomial solution). In the general case, three subregions of the interval $\alpha_j \in [0, 1]$ have to be investigated separately for each machine M_j yielding altogether 3^m combinations. For each combination, a linear time effort is required to find the exact α-values and having all these values, the schedule is completely determined. The procedure can be arranged in such a way that the total running time reduces from $O(n3^m)$ to $O(nm2^m)$. Moreover, a lower bound is given and based on the cyclic schedule (presented above for the problem with non-restrictive due date) around the due date, a heuristic is given as follows: Jobs are scheduled according to the cyclic schedule until the first job hits time zero, i.e. its starting time becomes negative. Then all remaining jobs must be processed after the due date and in order to preserve a necessary optimality condition, at each iteration m jobs are assigned to m different machines in the same order of machines (i.e. if r jobs are assigned by the cyclic schedule and job $r+j, j \in \{1, \ldots, m\}$, has been assigned to machine M_j, then also jobs $r+m+j, r+2m+j, \ldots$ are assigned to machine M_j).

The corresponding α_j values for all machines are determined heuristically. The complexity of the heuristic is $O(n \log n)$. It is shown that the heuristic has a constant bound of two on the worst case error and that this bound is tight. Numerical results with this heuristic are presented for problems with up to 500 jobs and 20 machines indicating that the average optimality gap is rather small.

Sun and Wang [15] consider the problem of scheduling n jobs with given proportional job weights (i.e. the job weight is proportional to the processing time of this job, where without loss of generality, one is taken as factor of proportionality) on m identical parallel machines such that the sum of the weighted absolute deviations from the common due date is minimized. It is shown that even for $m = 2$, the problem with a non-restrictive due date is NP-hard in the ordinary sense. For the problem under consideration, a dynamic programming algorithm with time complexity $O(nmd^mP^{m-1})$ is given, where P is the sum of the processing times of all jobs. Moreover, two list scheduling heuristics (namely the LPT and the modified LPT rule) are investigated. In both cases, the next job according to the list is scheduled on the machine available first. For the modified LPT rule, after assigning the longest m jobs to the machines, the availability time for machine M_l is defined by $\max\{0, p_l - d\}$ assuming that jobs are numbered according to non-increasing processing times. If the job sequences on the machines have been determined, it remains to find the optimal starting time of the first job on each machine. It is shown that the worst case relative error of the LPT heuristic is bounded by m. In the case that the largest processing time of a job is not smaller than the given due date, the modified LPT heuristic has a worst case ratio bounded by 25/12.

Bank and Werner [16] consider heuristic constructive and iterative algorithms for the problem of minimizing the sum of weighted linear earliness and tardiness penalties on unrelated parallel machines subject to given release dates, i.e. the objective function to be minimized has the form

$$F = \sum (w_i \max\{0, C_i - d\} + h_i \max\{0, d - C_i\}),$$

where w_i and h_i are given job weights. All the constructive heuristics presented in that paper are two-stage procedures. In the first stage, all operations have to be assigned to a machine on which they are processed. It turned out that distributing the jobs according to non-increasing largest slack times s_i^* to the machines and choosing in each step a machine that yields the smallest makespan value for the jobs distributed so far (obtained by scheduling the jobs on a certain machine according to non-decreasing release dates) worked best. Here the largest slack time for a certain job i is given by

$$s_i^* = \max\{s_{ij} | j = 1, \ldots, m\},$$

where

$$s_{ij} = d - (r_{ij} + p_{ij}),\ j = 1, \ldots, m,$$

denotes the slack on machine M_j. Then, in the second step all the resulting single machine problems are heuristically solved. A scheduling heuristic is presented which schedules the jobs initially around the due date, starting with the job that finishes precisely at the due date (if possible). The procedure applied tries to schedule the jobs in such a way that a job i is preferably inserted into the set of early jobs if $h_i \leq w_i$, otherwise preferably into the set of tardy jobs on the corresponding machine. Furthermore, a procedure that balances the total processing time before and after the due date is applied to decide finally whether the job currently considered is scheduled early or tardy on the chosen machine. When sequencing a particular job tardy on some machine, a local optimality condition guarantees that the resulting schedule satisfies a half V-shape property (i.e. the jobs started after d are ordered according to non-increasing ratios w_i/p_{ij}), and if a job is scheduled early, it is also inserted into a partial sequence such that a local optimality condition (for details see Theorem 1, part (a) in [16]) is satisfied. Finally, it is checked whether shifting the whole job sequence on some machine to the left (so that not necessarily one job completes precisely at the due date) improves the objective function value.

All the iterative algorithms presented in [16] perform local search among the set of d-schedules (which are schedules where on each machine, either a job completes precisely at the due date d or the set of early jobs is empty). The authors investigate different neighborhoods among the set of d-schedules. The neighborhood which turned out to be the best in the computations is a shift neighborhood where in each neighbor generation, exactly one job is moved to another machine and reinserted at the best position within the job sequence (SH^* neighborhood). It remains to be open whether the underlying neighborhood graph is connected. In that paper it has been proven that, if the neighbor generation is extended in such a way that an arbitrary neighbor is generated by performing either a shift of a job to another machine as described above or by selecting an arbitrary job and reinserting it on the best position on the same machine ($SH^* \cup SHM^*$ neighborhood), this guarantees connectivity of the neighborhood (so that the best d-schedule can be reached within the search from any initial d-schedule by a sequence of neighbor generations). The additional consideration of pairwise exchanges of two jobs scheduled on different machines is only advantageous for some problem classes with small number of machines. The authors report on computational results for problems with up to 500 jobs and 20 machines, where none of the metaheuristics tested is superior to the remaining ones. In the tests, good results have been obtained with a threshold accepting heuristic (either with a linear threshold reduction scheme or with a flexible threshold scheme, i.e. in the latter case the threshold may decrease or increase in the course of the computations in dependence on the results).

4 Shop Scheduling Problems

When considering multi-operation scheduling problems, there are two reasonable generalizations in comparison with single machine problems. Either, the objective function is kept the same as in the single machine problem, where the completion time is now the completion time of the last operation of the job, or a second penalty is added to this term which takes into account the storage costs of the intermediate. Such intermediate storage costs contribute considerably to overall costs e.g. in chemical industry, if the intermediate is not stable. In the first subsection, we treat problems without intermediate storage costs. In Subsection 4.2, we summarize the few results for problems with intermediate storage costs.

4.1 Shop Scheduling Problems without Intermediate Storage Costs

Concerning problems with non-restrictive due date according to Definition 1, the following results have been mentioned by Lauff and Werner [6]. The non-restrictive job shop problem $Jm||\sum|C_i - d|$ can be solved as follows. Let N_j be the set of operations which are the last operation of some job according to the technological route and which have to be processed on machine M_j ($j = 1, \ldots, m$). Then the job shop problem reduces to (at most) m independent single machine problems with the processing times of the corresponding set of operations as input data. Each of these m problems can be solved by Kanet's algorithm [3] in $O(|N_j|\log|N_j|)$ time. The remaining (non-last) operations can be scheduled feasibly in an arbitrary way. Therefore, the overall complexity is $O(mn\log n)$. Similarly, the optimal objective function value for the flow shop problem $Fm||\sum|C_i - d|$ can be calculated in $O(n\log n)$ time (equal to the optimal objective function value for the single machine problem on the last machine M_m), and the complete scheduling problem can be solved in $O(n\log n + nm)$ time for a non-restrictive common due date.

The open shop problem $Om||\sum|C_i - d|$ with a non-restrictive due date reduces to the problem $Rm||\sum|C_i - d|$ with unrelated parallel machines (see [17]). The latter problem can be solved in polynomial time. First one has to decide which operation of each job is processed as the last one, and on which machine and at which time it is scheduled (this can be found by an optimal solution to problem $Rm||\sum|C_i - d|$). This partial schedule can be completed by sequencing the remaining (non-last) operations of all jobs arbitrarily in a feasible way.

The only shop environment which has been considered until now for the restrictive case are flow-shop scheduling problems, where most existing papers deal with two-machine problems. Sarper [18] was the first dealing with such a problem, where the objective is the minimization of the sum of the absolute deviations from the common due date (SAD problem). Sarper presents a mixed integer programming formulation (which however cannot be used for practical problems since the solution of a 6-job problem with use of the LINDO package took about 75 minutes) and gives three heuristics for the approximate solution of this problem. The first one is based on a few starting solutions and for each of them, a limited neighborhood search is performed. To generate neighbors, special pairwise interchanges are allowed (namely the first job can be interchanged with the last job, the second job with the second-to-last one and so on). Additionally, also adjacent pairwise interchanges are used for generating neighbors. The second and third heuristics are insertion-type algorithms. Insertion techniques are very common for constructive heuristics and were successfully applied to many scheduling problems (see e.g. [19]). The jobs are ordered in a list according to a certain criterion (Sarper uses the SPT and LPT orders of the processing times on machine M_2). The sequence is built up successively by inserting the jobs into a partial sequence according to this order. There are $i + 1$ possible positions for a job which is inserted into a sequence consisting of i jobs. Among these $i + 1$ resulting sequences, the one with the lowest objective function value is selected. The underlying idea is that an optimal decision for a part of the sequence is at least a good decision for the whole sequence. Having a (partial) sequence fixed, this approach requires a procedure for fixing the starting times of all operations. Although this subproblem can be solved by means of linear programming, it is rather time consuming and Sarper uses a rough heuristic (we do not discuss this method here further since later a substantially better procedure is presented).

The quality of the three heuristics is compared with the exact solution by applying a mixed integer program for (very) small problem size with $n \in \{5, 6\}$ jobs and relative to each other for problems with up to 20 jobs. None of the heuristics is superior to the other ones. We only mention that even for small problems with $n \in \{5, 6\}$ jobs, the maximal average percentage deviation of the heuristics for one data set is already 9.80 % for the first heuristic, 13.33 % for the second heuristic and 25.09 % for the third heuristic. For the problems with 20 jobs and large due dates, each of the heuristics required 30 minutes or more for certain data sets. The heuristics suggested by Sarper have some drawbacks which will be discussed later when comparing with some other algorithms.

In [20, 21], Gupta et al. develop the first enumerative algorithm for the two-machine flow shop problem $F2|d_i = d| \sum f_i(C_i)$ with a common due date and general earliness and tardiness penalties. The only requirement on functions f_i is a monotonicity property as follows. It is assumed that $f_i(d) = 0$ and

$$\begin{array}{ll} \text{M1: } f_i(x) \text{ is non-increasing for } x \leq d, & \\ \text{M2: } f_i(x) \text{ is non-decreasing for } x \geq d, & \end{array} \qquad \text{(M)}$$

for $i = 1, \ldots, n$. Note that this assumption includes regular criteria as a special case, since by setting $f_i(x) = 0$ for $x \leq d$ every arbitrary regular criterion minimizing total penalty may be modeled. Note also that convexity of the functions f_i is not required. It is worth noting that for the problem with a non-restrictive common due date, the starting times of operations may be non-integers even if the processing times and the due date are integer. Nevertheless, for ease of presentation, the authors restrict to integer starting times of the jobs and mention that the algorithm is general enough to solve each problem with given accuracy (see description of bisection search later).

The enumeration strategy in [20] is fundamentally different from those used in branch and bound algorithms for problems with regular criteria and we sketch some components of this algorithm. The algorithm for solving problem $F2|d_i = d| \sum f_i(C_i)$ uses the following properties of an optimal schedule:

- The set of all permutation schedules contains an optimal schedule.
- There exists an optimal schedule in which the first job starts at time zero and there is no inserted idle time on M_1.
- 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.

The algorithm presented partitions the job set into the set E of early jobs and the set T of tardy jobs, and in the first level of the algorithm, the possible sets E of early jobs are enumerated (the latter enumeration is denoted as main tree). Based on the above properties, for a particular node of the main tree, one has to find both the best starting time s of the first tardy job τ_1 on M_2 and the best sequences of the jobs of the sets E and T. To determine s, one has to perform a search over an interval the length of which depends on the first tardy job. Therefore, for fixed set E, several values of s and jobs τ_1 have to be considered, and for each combination, the best sequences of the jobs in E and $T \setminus \{\tau_1\}$ have to be determined by separate branch and bound algorithms.

Let $\tau^*(T,\tau_1,s)$ denote a best sequence of the jobs completed after the common due date d subject to the constraints that E,τ_1, and s are fixed. The best sequence ϵ^* of jobs in set E is completely determined by the *starting time* of the first job of T since this is the maximal allowed makespan value for the sequence ϵ^*. This sequence is denoted by $\epsilon^*(E,s)$, s being the starting time of the first job of T.

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 Propositions 1 - 3 as follows:

$$\begin{aligned} F\left(\epsilon^*\left(E,s\right)\right) &:= \sum_{j=1}^{|E|} f_{\epsilon_j}\left(s - \sum_{k=j+1}^{|E|} p_{\epsilon^*_k,2}\right) \\ F\left(\tau^*(T,\tau_1,s)\right) &:= \sum_{j=1}^{|T|} f_{\tau_j}(c_{\tau^*_j,2}) \end{aligned}$$

In the main tree, a depth-first search is performed. At each node of the main tree, every job in T is successively considered as the first tardy job τ_1 on machine M_2, obtaining different subnodes (E,τ_1). 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:

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

In Equation (1), we use $C^*_{max}(E)$ 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.

Then Gupta et al. [20, 21] apply 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 $E\cup T$ and a fixed tardy job τ_1. 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. 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 bisection-type 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 if 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 1 illustrates procedure MINSEARCH for the first two steps. It starts with the calculation of a lower bound for function F in the whole interval $I1=[I_S,d]$ (of the possible starting times of $s_{\tau_1 2}$):

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

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

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

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

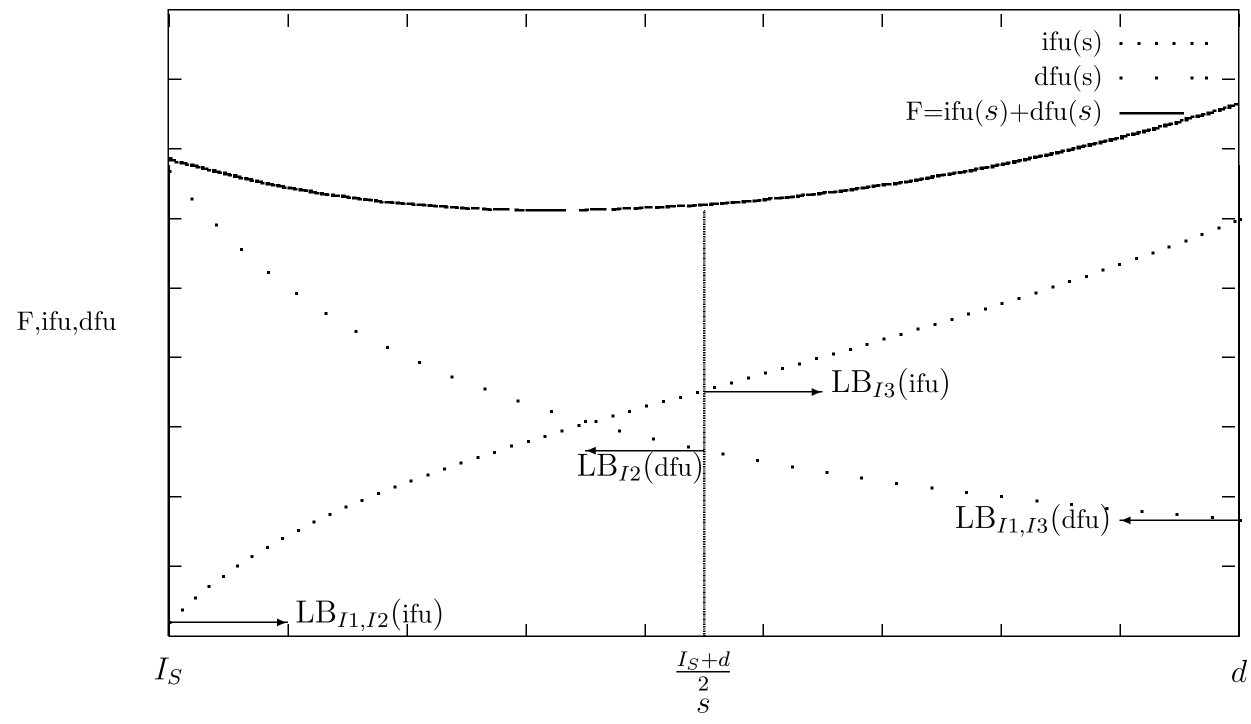

Figure 1: Illustration of the calculation of lower bounds (double indices mean that the values are equal for the two indices).

Interval I2 is partitioned further recursively if LB_{I2} is smaller than the currently best function value UB and accordingly, interval I3 is partitioned if $LB_{I3} < UB$. This procedure stops if no subinterval has a smaller lower bound than UB.

The determination of each function value of ifu(s_1) and dfu(s_2) above requires the application of a branch and bound algorithm. Although these branch and bound procedures (described in [20] for the case of job-independent penalty functions, i.e. $f_i = f$ for all jobs i) have some similarity to known algorithms for regular criteria, the authors also present a new lower bound and a new dominance criterion. For details the reader is referred to [20], where also a detailed numerical example for illustration is given.

We describe the generation of the test problems used in the latter paper and also in other tests (see later [22]). The problem data of one instance consists of the number of jobs, the processing times of all operations and the common due date. The job processing times are uniformly distributed integers from the interval $[1, p_{\max}]$. In the experiments, Gupta et al. used $p_{\max} \in \{5, 20, 50\}$. The common due date was taken to be a function of the maximum load P on the two machines given by:

$$P = \max\left\{\sum_{i\in N} p_{i1}, \sum_{i\in N} p_{i2}\right\}.$$

To test the influence of the relative value of the due date compared with the other input data, the following values of the common due date were used:

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

Two objective functions have been used in the tests:

- an asymmetric linear-linear (LL) penalty function:

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

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

- a linear-quadratic (LQ) penalty function:

$$f(x) = \begin{cases} d - x, & x \leq d \\ (x-d)^2, & x > d \end{cases}$$

The algorithm developed has been tested on problems with up to 20 jobs. The following results have been found:

- For both asymmetric penalty functions LL and LQ, problems with up to 20 jobs can be satisfactorily solved within a time limit of 15 minutes.
- For the problem with the LL penalty function, problems with $d = 0.75P$ are the most difficult ones while for the LQ function, problems with $d = 0.25P$ turned out to be the hardest problems.
- Problems with $d = 1.0P$ are easy to solve for both objective functions.

Since this class of two-machine problems can be solved exactly only for moderate problem size, the development of efficient heuristics is important. Lauff and Werner [22] developed and compared different heuristic algorithms for the same problem type as considered in [20, 21] but with identical penalty functions for each job (i.e. $f_i = f$). In [22], a feasible schedule is represented by the job sequence π and by the starting time $s_{\pi_1,2}$ of the *first* job on M_2 (notice that this a bit different in comparison with the enumerative procedure in [20]). The latter is based on the property that every schedule for a fixed job sequence $\pi = (\pi_1, \ldots, \pi_n)$ with idle times in-between the processing of jobs π_1 to π_μ, where

$$\mu = \max\{i \in \{1, \ldots, n\} | C_{max}(\pi_1, \ldots, \pi_i) - p_{\pi_i,2} \leq d\},$$

and $C_{max}(\pi_1, \ldots, \pi_i)$ is the makespan value of sequence $(\pi_1, \ldots, \pi_i)$, can be changed into a schedule without idle time in-between these jobs, such that the objective function value does not increase (note that sequence $(\pi_{\mu+1}, \ldots, \pi_n)$ always constitutes a sequence of tardy jobs for the fixed job sequence $\pi = (\pi_1, \ldots, \pi_n)$). The possible interval of the starting time of job π_1 on M_2 for a fixed job sequence π is

$$\left[\max\{\kappa, d\} - \sum_{i=1}^{\mu} p_{\pi_i,2}, d \right],$$

where $\kappa = C_{max}(\pi_1, \ldots, \pi_\mu)$. In the above interval, a bisection-type search for the optimal starting time $s_{\pi_1,2}$ is performed (again it may happen that both subintervals have to be partitioned further).

Compared to the algorithm given in [18] for finding a schedule for some fixed job sequence, there are the following differences. Sarper does not attempt to remove or insert idle time between jobs as is done by Lauff and Werner by computing the best function value for all the different values of $s_{\pi_1,2}$ considered in the course of the bisection search. It has been observed that this is a *crucial point in order to find good heuristic solutions.* Further, Lauff and Werner apply a bisection search instead of examining every reasonable integer starting time $s_{\pi_1,2}$, which significantly reduces the computational effort. In the approach chosen by Sarper only *two* possible starting times are possible anyway for the SAD-criterion which he studied. Since the algorithm in [18] starts with the schedule providing the lowest starting times on M_2 possible for a given sequence, there are only two possibilities. Either, the relative value of the due date is low compared with the other input data, then this schedule is already the optimal one for this sequence, or, the due date is rather loose, then the schedule with $s_{\pi_{\lceil n/2 \rceil},2} = d$ yields an optimal value for the sequence. This follows by a similar argument as given in [3] and by the fact that the algorithm of Sarper does not remove idle times.

For constructing a job sequence, Lauff and Werner [22] studied priority rules as well as insertion techniques. Priority rules are applied to sort the jobs in a list, e.g. in SPT-order on M_1. The sequence is built up by appending the next job of the list to the end of the current partial sequence (algorithm Append). In addition to the standard insertion algorithm, Lauff and Werner have also studied an alternative to the variant described earlier. This is motivated by the generality of the penalty function defined in (M). For instance, asymmetry is possible, and the functions considered in [22] are asymmetric. An early deviation from the due date is penalized less than a tardy one with the same absolute value. For an insertion according to the SPT-order, this means that nearly all jobs are sequenced early at the beginning with the consequence that the jobs with larger processing times are sequenced tardy, when there are no more possibilities to schedule them early. This contradicts the idea behind the insertion heuristics, since the optimal decision for a partial sequence is not advantageous from the global point of view. Informally speaking, the standard insertion heuristic recognizes the difficulty too late. To overcome this problem, Lauff and Werner have changed the due date d used in the calculation procedure in each step. Let the current sequence consist of i jobs, and the next job according to the list enters the sequence. Then the due date in this $(i+1)$-th insertion step is $d_{i+1} = (i+1)d/n$. This procedure will be referred to as *due date modification.*

Lauff and Werner also develop a procedure Smooth which is first motivated and then sketched. For single machine problems, a sequence is called V-shaped, if there are no three jobs i, j, k, where i is processed before j and j before k, with $p_i < p_j$ and $p_k < p_j$. Transferred to the flow shop problem considered here, this means that the early jobs should strive for an LPT-order, and the tardy jobs for an SPT-order with respect to the processing times on M_2. Deviations from the strict V-shape sequence on M_2 may reduce the objective function value due to the flow shop constraint. These observations lead to the following criterion for interchanging two adjacent jobs π_i and π_{i+1} in a permutation π. An adjacent pairwise interchange of both jobs *does not increase* the objective function value if there are either adjacent

- early jobs π_i, π_{i+1} with $p_{\pi_i,2} < p_{\pi_{i+1},2}$ or
- tardy jobs π_i, π_{i+1} with $p_{\pi_i,2} > p_{\pi_{i+1},2}$;

and the inequality

$$c_{\pi_{i+1},2} \geq \max(c_{\pi_i,1} + p_{\pi_{i+1},2}, c_{\pi_{i+1},1}) + p_{\pi_i,2}$$

holds. Note that one cannot be sure that the objective function value is indeed *reduced* due to the generality of the penalty function as given in condition (M).

Procedure Smooth works as follows. First, the starting time $s_{\pi_1,2}$ is optimized by applying the bisection-type search procedure. By this, one can identify the early and tardy jobs of the sequence. Then, the conditions given above are checked successively for all indices $i \in \{1, 2, \dots, n-1\}$ of permutation π and adjacent pairwise interchanges are performed if one condition is fulfilled. The best schedule of the resulting modified sequence is again determined by bisection search, and these steps are repeated until a sequence is obtained where both two conditions are not fulfilled for any two adjacent jobs.

The authors studied 12 constructive heuristics and a randomly generated sequence for comparison. For details the reader is referred to [22]. Here we discuss only the influence of procedure Smooth for the LL penalty function a bit more in detail. The results for a selection of the heuristics are given in Table 1. We use the following notation in order to distinguish the methods. The first letter is either an I (Insertion) or an A (Append). We append an S, if procedure Smooth is applied. The name is accomplished by two acronyms in brackets. The first represents the order according to which the jobs are considered in the construction process, for instance SPT1 stands for the SPT-order with respect to the processing times on M_1. If the due date modification is applied, the second expression in brackets is an D, and N otherwise. Thus, as an example, I(SPT2,N) means that the jobs are inserted according to non-increasing processing times on M_2 without due date modification and without procedure Smooth.

It can be seen from Table 1 that the use of procedure Smooth can improve the results obtained with the insertion and appending procedures considerably. This holds particularly for the variants, where without procedure Smooth bad results are obtained. Without applying procedure Smooth, the appending procedure A(SPT1) is not competetive, however, with applying procedure Smooth, the results are partially even better than those with the insertion algorithm (this is particularly true for the large problems with $n = 200$ jobs).

When comparing the constructive heuristic solutions with the exact solution on small problems, Lauff and Werner found that for the LL penalty function and problems with 14 jobs, variant IS(SPT2,D) was the best with an average percentage deviation of no more than 4.03 % (the largest percentage deviations were obtained for problems with $d/P = 0.75$). For the problems with the LQ penalty function and 20 jobs, this percentage deviation from the optimum became larger, namely almost up to 10 % for the problems with $d/P = 0.75$ (in this case variant IS(SPT1,D) turned out to be the best).

The iterative algorithms presented by Lauff and Werner are based on composite neighborhoods, where a neighbor is either generated by a shift of one chosen job or by an interchange of two arbitrary jobs in the current job sequence π. In [22], iterative improvement, simulated annealing and multi-descent algorithms are applied. Concerning iterative improvement, some investigations about scanning the (large) number of neighbors in the composite shift/pairwise interchange neighborhood in order to improve the convergence to a local optimum are done. For simulated annealing, a very thorough testing of the parameters is performed (this is particularly important for the asymmetric objective functions considered there since usually parameters are found for very specialized assumptions, e.g. randomly determined processing times from a specific interval). The multi-descent heuristic applies iterative improvement repeatedly with different starting solutions.

The iterative algorithms developed have been tested on problems with 40 and 200 jobs relative to each other and for smaller problems with $n \in \{14, 16, 18, 20\}$ jobs, the results have

40 jobs without Smooth						
d/P	0	0.1	0.25	0.5	0.75	1.0
I(SPT1,D)	0.54 (117)	1.08 (0)	2.67 (0)	6.68 (0)	6.37 (5)	1.45 (30)
I(SPT2,D)	0.75 (7)	1.14 (0)	1.88 (2)	3.03 (12)	1.92 (41)	0.35 (128)
I(SPT1,N)	0.54 (117)	0.73 (93)	1.51 (45)	4.07 (13)	7.33 (0)	10.02 (0)
I(SPT2,N)	0.75 (7)	1.96 (1)	2.77 (1)	5.13 (9)	5.39 (43)	0.32 (57)
A(SPT1)	18.24 (0)	21.95 (0)	25.68 (0)	31.30 (0)	43.72 (0)	69.60 (0)

40 jobs with Smooth						
d/P	0	0.1	0.25	0.5	0.75	1.0
IS(SPT1,D)	0.54 (119)	0.95 (51)	1.67 (33)	3.12 (23)	4.13 (35)	0.33 (120)
IS(SPT2,D)	0.48 (140)	0.73 (100)	0.90 (126)	1.28 (163)	1.23 (169)	0.32 (132)
IS(SPT1,N)	0.54 (119)	0.72 (107)	1.45 (90)	3.62 (90)	4.55 (96)	0.76 (47)
IS(SPT2,N)	0.48 (140)	1.54 (48)	2.15 (71)	4.67 (83)	5.30 (65)	0.32 (57)
AS(SPT1)	0.78 (137)	1.25 (91)	2.23 (80)	6.29 (30)	9.66 (6)	7.42 (3)

200 jobs without Smooth						
d/P	0	0.1	0.25	0.5	0.75	1.0
I(SPT1,D)	0.27 (126)	1.27 (0)	3.91 (0)	7.87 (0)	3.98 (0)	0.12 (5)
I(SPT2,D)	0.43 (0)	1.32 (0)	3.64 (0)	6.28 (0)	3.52 (0)	0.01 (197)
I(SPT1,N)	0.27 (126)	0.82 (1)	1.86 (0)	3.14 (0)	7.55 (0)	11.54 (0)
I(SPT2,N)	0.43 (0)	0.87 (0)	1.20 (0)	2.45 (1)	5.98 (9)	0.00 (136)
A(SPT1)	23.15 (0)	27.96 (0)	34.20 (0)	40.97 (0)	50.95 (0)	68.38 (0)

200 jobs with Smooth						
d/P	0	0.1	0.25	0.5	0.75	1.0
IS(SPT1,D)	0.27 (126)	1.07 (1)	2.71 (0)	3.91 (0)	2.36 (48)	0.01 (59)
IS(SPT2,D)	0.32 (63)	1.00 (3)	2.42 (0)	3.07 (11)	1.44 (86)	0.01 (206)
IS(SPT1,N)	0.27 (126)	0.78 (7)	1.61 (12)	1.85 (48)	1.91 (118)	0.18 (1)
IS(SPT2,N)	0.32 (63)	0.65 (14)	0.82 (100)	2.21 (153)	5.85 (41)	0.00 (136)
AS(SPT1)	0.23 (211)	0.12 (60)	0.22 (287)	1.46 (188)	2.30 (104)	1.65 (0)

Table 1: The influence of applying procedure Smooth for the LL penalty function

been compared with the exact solution found by the enumerative algorithm given in [20]. The problems have been generated in the same way as in [20] (see above in this section), but the maximum processing time p_{max} was set to be equal to 100 for the large problems with 40 and 200 jobs. We summarize the most important results from [22]:

- For the problems with 40 and 200 jobs, it has been found that algorithms simulated annealing and multi-descent gave the best results. Both procedures deviate from the best solution found by some variant on average by no more than 0.12 % for the problems with 40 jobs and by no more than 0.35 % for the problems with 200 jobs.
- For the problems with 20 jobs and the LQ penalty function, algorithms multi-descent and simulated annealing gave the best results, too, where the multi-descent algorithm is slightly superior. Except for instances with $d/P = 0.75$, both algorithms produced results that deviate from the optimal value by no more than 1.2 % on average. In contrast to iterative improvement procedures, the simulated annealing and multi-descent algorithms solved all instances with 14 jobs and the LL penalty function to optimality. Except for instances with $d/P = 0.75$, the variants of iterative improvement produced solutions with an average percentage deviation from the optimal solution not greater than 1.3 % for the problems with 14 jobs and not greater than 2.3 % for the problems with 20 jobs.
- The most difficult problems are those with $d/P = 0.75$. For problems with 20 jobs (where almost all optimal solutions are known from the algorithm in [20]), the multi-descent algorithm was the best with an average percentage deviation of 2.36 % followed by the simulated annealing algorithm with a deviation of 2.44 %. However, even all IT variants tested deviated by no more than 3.25 % from the optimal value on average in this case.
- In general, the quality of the solutions found by the heuristics becomes worse with increasing ratio d/P until $d/P = 0.75$. In contrast, for $d/P = 1.0$, all iterative algorithms generated solutions with a function value very close to the optimum.

In comparison to simulated annealing, the multi-descent algorithm has the advantage that *no parameter optimization is required* after fixing the number of iterations. We emphasize that papers [20, 22] have been discussed more in detail since they are still unpublished yet.

Sung and Min [23] consider a two-machine flow shop problem with one or two batching machines such that the sum of the absolute deviations of the completion times from the given common due date is minimized. It is assumed that each batch can process simultaneously up to c jobs and that each batch has the same processing time t. Moreover, the common due date d is assumed to be greater than or equal to the time required for processing all jobs on the first machine (this time depends on whether M_1 is a batching machine or not).

Sung and Min consider three different configurations of the two-machine flow shop:

- Only the second machine M_2 is a batch processing machine.
- Both machines are a batch processing machine.
- Only the first machine M_1 is a batch processing machine.

For the first two cases polynomial algorithms are given (which can be used also for single machine problems with a batching machine, where release dates are involved which are not greater than the given due date). These algorithms are based on structural properties of an optimal schedule. We sketch the algorithm for the first configuration. Observe that there exists an optimal schedule with no idle time inserted between any two consecutive jobs and batches on the machines, respectively. Moreover, there exists an optimal schedule having the SPT sequence starting at time zero on M_1. So it remains to determine the batch schedule for M_2.

In the non-restrictive case, there exists an optimal schedule such that some batch is exactly completed at d on M_2. Moreover, the N_e early jobs are scheduled in batches such that the first batch contains the first $[N_e - (\lceil N_e/c \rceil - 1)\, c]$ jobs and the next $(\lceil N_e/c \rceil - 1)$ batches are full of jobs, i.e. each of them contains c jobs. Further, the N_t tardy jobs are scheduled in batches such that the first $(\lceil N_t/c \rceil - 1)$ batches are full of jobs and the last batch contains the remaining $[N_t - (\lceil N_t/c \rceil - 1)c]$ jobs. It is shown that there exists a large number of optimal schedules and that among them, there is one having the smallest number of early jobs with

(a) $\left[\left(\lceil \frac{n}{2c} \rceil\right) c\right]$ early jobs if $\lceil n/c \rceil$ is even and

(b) $\left[n - \left(\lceil \frac{n}{2c} \rceil - 1\right) c\right]$ early jobs if $\lceil n/c \rceil$ is odd.

In the restrictive case, there exists an optimal schedule with N_e early jobs and N_t tardy jobs, which are scheduled according to the same strategy as described above and $N_e \in [N_e^{max} - c + 1, N_e^{max}]$, where N_e^{max} is the largest number of jobs that can be processed before the due date. Moreover, there exists an optimal schedule with the property that at least one of the following properties is satisfied:

(a) some batch is completed at d,

(b) the starting time ST of the first batch on M_2 is given by

$$ST = d - kt - \alpha, \quad \alpha = \min_{1 \le i \le k+1} \{d - (k - i + 1)t - R_i\},$$

where $k = \lceil N_e/c \rceil$ and R_i denotes the largest completion time of a job, which is contained in the i-th batch on M_2, on machine M_1. Using the above properties of an optimal schedule, the number of early jobs in an optimal schedule can be found in at most c iterations. The second configuration can be solved by slight modifications of the algorithm sketched above. The time complexity of both algorithms is $O(n(c + \log n))$.

The third problem is shown to be NP-hard (it follows directly from NP-hardness of the single machine problem of minimizing the sum of the absolute deviations from a given restrictive due date). For the latter problem, a pseudopolynomial algorithm is given which is a slight modification of the *split* algorithm proposed by Ventura and Weng [24] for the single machine problem.

Gowrishankar et al. [25] consider the problem of minimizing the sum of squares of the deviations of the completion times from a given common due date (and the related problem of minimizing the variance of the completion times of jobs - CTV problem) in the case of a flow shop problem with an arbitrary number of machines (i.e. problem $F|| \sum(C_i - d)^2$). Contrary to [20, 21, 22], the authors consider only a special form of the given due date, namely it is assumed to be equal to the best mean completion time of a schedule. Due to this, the authors consider *only schedules starting at time zero* which have no inserted idle time. It means that contrary to the papers mentioned earlier, no optimization algorithm is required which determines for some job sequence the best schedule. In this sense, the problem considered in [25] is a simplification of the (two-machine) problem considered in [20, 21, 22].

In the branch and bound algorithm given in [25], the authors restrict to permutation schedules. The branching strategy is based on the generation of partial job sequences (i.e. in the first level, the first job is fixed, in the second level, the second job is fixed and so on), and a lower bound for all schedules starting with some fixed job sequence is given. This lower bound is based on two terms, one of them is a lower bound for a fixed partial job sequence for the CTV problem which is also a lower bound for the minimization of the sum of the squared deviations from a given due date. The other term uses a lower bound for the positional completion times of the unscheduled jobs and for the makespan value of a completed partial sequence.

Computational results are presented for problems with up to 10 jobs and 25 machines. For the problems with 10 jobs, the branch and bound algorithm generates on average between 40,000 and 73,000 nodes (where the largest number of nodes is generated for a small number of machines m). Accordingly, among the problems with $m \in \{5, 10, 15, 20, 25\}$ considered, the largest mean computation time of about 7.5 minutes per instance has been obtained for $m = 5$. It is worth noting that the problem of minimizing the squared deviations from the given due date is usually harder than the corresponding CTV problem. Finally, heuristics are given both for problem $F||(C_i - d)^2$ and the CTV problem. For the problem with given common due date, the standard insertion algorithm presented by Nawaz et al. [19] is applied, where the jobs are inserted according to non-increasing sums of their processing times. Then, all the possible $n-1$ neighbors in the adjacent pairwise interchange neighborhood are generated and the best neighbor is accepted as new starting solution if it provides an improvement of the objective function value. Then again all neighbors are checked until no further improvement of the objective function value has been obtained during one cycle. For small problems with 10 jobs, the average percentage deviation from the optimal function value is between 9 and 12 %. For larger problems with $n \in \{30, 40, 50\}$ jobs, the heuristic solutions are unfortunately only compared to a random search procedure with a time limit of one minute, making it difficult to judge about the performance of the proposed algorithm.

4.2 Shop Scheduling Problems with Intermediate Storage Costs

The second model of earliness-tardiness multi-stage problems considers intermediate storage costs in the objective function. These costs contribute considerably to overall costs in chemical industry, if the intermediate is not stable. Such problems have been considered by Lauff and Werner [6]. In the latter paper, the simplest case of the two-machine problem with intermediate storage costs is assumed, namely that the intermediate storage cost for each job is equal to the waiting time between the completion of the first operation and the start of the second operation. For the resulting two-machine flow shop problem with minimizing the sum of the absolute deviations from the common due date, the following properties have been given in [6]:

- For problem $F2||\sum|C_i - d| + \sum(s_{i2} - c_{i1})$ and a non-restrictive due date d, there are instances for which no permutation schedule is optimal.
- For problem $F2||\sum|C_i - d| + \sum(s_{i2} - c_{i1})$ with $d = 0$, there are instances where no permutation schedule is optimal.

Moreover, in [6] an algorithm has been given that delivers for fixed job sequences π^1 and π^2 on both machines the resulting best schedule. By checking all possible combinations of π^1 and

π^2, optimal schedules for small problems until seven jobs can be found in two minutes.

Concerning the complexity of shop scheduling problems, the following results have been obtained by Lauff and Werner [6]:

- Problem $F2||\sum|C_i - d| + \sum(s_{i2} - c_{i2})$ with a non-restrictive common due date is NP-hard in the strong sense.
- Problem $F2||\sum|C_i - d| + \sum(s_{i2} - c_{i2})$ with $d = 0$ is NP-hard in the strong sense.
- Problem $O2||\sum|C_i - d| + \sum(s_{i[2]} - c_{i[1]})$ with a non-restrictive common due date is NP-hard in the strong sense.
- Problem $O2||\sum|C_i - d| + \sum(s_{i[2]} - c_{i[1]})$ with $d = 0$ is NP-hard in the strong sense.

In the latter two problems the notations $s_{i[2]}$ and $c_{i[1]}$ denote the starting time of the second operation and the completion time of the first operation of job i, respectively. In all cases above, the reduction is done from 3-partition.

5 Outlook and Further Work

The investigation of shop scheduling problems with nonregular criteria has begun only recently. We reviewed some recent papers in this field for multi-machine problems and discussed some own work [20, 22] still unpublished yet a bit more in detail. Concerning problems with a *given common due date*, we give the following topics for further investigations:

- Until now, there are only algorithms for flow shop scheduling with a common due date available, but there do not exist exact algorithms for job and open shop scheduling problems in the two-machine environment. It can be expected in analogy to scheduling problems with regular criteria that both job and open shop problems are computationally harder than the corresponding flow shop problem.
- The algorithm given in [20] can be improved for the special case of the SAD problem since the branch and bound algorithms for finding the best sequences of early and tardy jobs can be refined by incorporating similar criteria as those used in the algorithms for problem $F2||\sum C_i$.
- The development of an exact algorithm for the two-machine flow-shop problem with intermediate storage costs is of interest. From our own experience with developing an exact algorithm for the problem without intermediate storage costs (see [20]), this is expected to be a challenging task and the size of the problems that can be solved exactly is probably less than 20. The development of heuristics for the latter type of problems is an interesting topic.
- Concerning the development of algorithms for the problems without considering storage costs and a larger number of machines than two, there do not exist algorithms up to now (except the algorithm given in [25] which, however, considers only schedules starting at time zero). The generalization of the algorithm from [20, 21] can cause problems since some of the properties used depend on the two-machine problem. The development of

general strategies for solving m-machine flow shop problems is of interest (but, due to the difficulty of such problems, the problem sizes that can be handled are expected to be rather small).

It is also worth investigating whether algorithms for problems with a common due date can be used for the harder problems with 'similar' due dates and how good the results will be in this case for the original problem. For the case that the common due date is not given in advance but a decision variable, there are some results for parallel machine problems. However, no results are available for the case of shop scheduling problems. The development of lower bounds, enumerative and heuristic algorithms is of interest.

References

[1] U. Bagchi, R. Sullivan and Y. Chang, Minimizing Absolute and Squared Deviations of Completion Times with Different Earliness and Tardiness Penalties and a Common Due Date. *Naval Res. Logist. Quart.* **34**, 739 - 751, (1987).

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

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

[4] V. Gordon, J.-M. Proth and C. Chu, A Survey of the State-of-the-Art of Common Due Date Assignment and Scheduling Research. *European J. Oper. Res.* **139** (1), 1 - 25, (2002).

[5] 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).

[6] V. Lauff and F. Werner, On the Complexity of Some Properties of Multi-Stage Scheduling Problems with Earliness and Tardiness Penalties. *Comput. Oper. Res.* (2003), (to appear).

[7] S.T. Webster, The Complexity of Scheduling Job Families About a Common Due Date. *Oper. Res. Lett.* **20**, 65 - 74, (1997).

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

[9] N.G. Hall, Single and Multi-Processor Models for Minimizing Completion Time Variance. *Naval Res. Logist. Quart.* **33**, 49 - 54, (1986).

[10] P. Sundararaghavan and M. Ahmed, Minimizing the Sum of Absolute Lateness in Single Machine and Multimachine Scheduling. *Naval Res. Logist. Quart.* **31**, 325 - 333, (1984).

[11] H. Emmons, Scheduling to a Common Due-Date on Parallel Uniform Processors. *Naval Res. Logist.* **34**, 803 - 810, (1987).

[12] A. Federgruen and G. Mosheiov, Heuristics for Multimachine Minmax Scheduling Problems with General Earliness and Tardiness Costs. *Naval Res. Logist.* **44**, 287 - 299, (1997).

[13] A. Federgruen and G. Mosheiov, Heuristics for Multimachine Scheduling Problems with Earliness and Tardiness Costs. *Manage. Sci.* **42**, 1544 - 1555, (1996) (erratum **43**, 735, (1997)).

[14] G. Mosheiov and M. Shadmon, Minmax Earliness-Tardiness Costs with Unit Processing Times. *European J. Oper. Res.* **130**, 638 - 652, (2001).

[15] H. Sun and G. Wang, Parallel Machine Earliness and Tardiness Scheduling with Proportional Weights. *Comput. Oper. Res.* **30**, 801 - 808, (2003).

[16] J. Bank and F. Werner, Heuristic Algorithms for Unrelated Parallel Machine Scheduling with a Common Due Date, Release Dates, and Linear Earliness and Tardiness Penalties. *Mathl. Comput. Modelling* **33**, 363 - 383, (2001).

[17] W. Kubiak, S. Lou and S. Sethi, Equivalence of Mean Flow Time Problems and Mean Absolute Deviation Problems. *Oper. Res. Lett.* **9**(6), 371 - 374, (1990).

[18] 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).

[19] M. Nawaz, E.E. Enscore and I. Ham, A Heuristic Algorithm for the m-Machine, n-Job Flow Shop Sequencing Problem. *OMEGA* **11**, 91 - 95, (1983).

[20] 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) (under submission).

[21] J.N.D. Gupta, V. Lauff and F. Werner, On the Solution of 2-Machine Flow Shop Problems with a Common Due Date. In Operations Research Proceedings 1999 (SOR) (Edited by K. Inderfurth *et al.*), Magdeburg, September 1 - 3, pp. 383 - 388, (2000).

[22] 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) (under submission).

[23] C.S. Sung and J.I. Min, Scheduling in a Two-Machine Flow Shop with Batch Processing Machine(s) for Earliness/Tardiness Measure Under a Common Due Date. *European J. Oper. Res.* **131**, 95 - 106, (2001).

[24] J.A. Ventura and M.X. Weng, An Improved Dynamic Programming Algorithm for the Single Machine Mean Absolute Deviation Problem with a Restrictive Common Due Date. *Oper. Res. Lett.* **17**, 149 - 152, (1995).

[25] 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).