PERGAMON

Computers & Operations Research 28 (2001) 787–801

computers & operations research

www.elsevier.com/locate/dsw

Benchmarks for scheduling on a single machine against restrictive and unrestrictive common due dates

Dirk Biskup*, Martin Feldmann

Faculty of Economics and Business Administration, University of Bielefeld, Postfach 10 01 31, 33501 Bielefeld, Germany

Received 1 March 1999; received in revised form 1 September 1999

Abstract

We consider the NP-hard problem of scheduling jobs on a single machine against common due dates with respect to earliness and tardiness penalties. The paper covers two aspects: Firstly, we develop a problem generator and solve 280 instances with two new heuristics to obtain upper bounds on the optimal objective function value. Secondly, we demonstrate computationally that our heuristics are efficient in obtaining near-optimal solutions for small problem instances. The generated problem instances in combination with the upper bounds can be used as benchmarks for future approaches in the field of common due-date scheduling.

Scope and purpose

In connection with just-in-time production and delivery, earliness as well as tardiness penalties are of interest. Thus scheduling against common due dates has received growing attention during the last decade. Many algorithms have been developed to solve the different variants of this problem. But whenever a new algorithm for scheduling against common due dates is proposed, its quality is assessed only on a few self-generated examples. Hence it is difficult to evaluate the various approaches, particularly in comparison with each other. Therefore the goal of this paper is to present numerous benchmark problems together with some upper bounds on the optimal objective function value.

Keywords: Scheduling; Common due date; Benchmark

* Corresponding author. Tel.: + 49-521-106-3929; fax: + 49-521-106-6036.
E-mail addresses: dbiskup@wiwi.uni-bielefeld.de (D. Biskup), mfeldmann@wiwi.uni-bielefeld.de (M. Feldmann).

0305-0548/01/$ - see front matter
PII: S0305-0548(00)00008-3

1. Introduction

Common due-date problems have been studied extensively during the last 20 years. In practice, a common due date occurs, for example, if in an assembly schedule a set of jobs is needed simultaneously, see Kanet [1], or if one customer orders a bundle of perishable goods which have to be delivered at a prespecified time.

When scheduling on a single machine against a common due date, one job at most can be completed exactly at the due date. Hence, some of the jobs have to be completed early, that is prior to the due date, while other jobs must be finished late. In both cases certain costs are incurred: early jobs tie up capital and cause holding costs while the effects of tardy jobs are dissatisfied customers and thus — in the long run — the loss of goodwill and reputation. In this context the goal is to find a schedule which jointly minimizes the sum of earliness and tardiness costs.

Generally speaking, there are two classes of common due-date problems which have proven to be NP-hard, namely if a restrictive common due date is given or if different jobs incur different penalties. In this paper we propose benchmarks for both classes of problems. This is done by the generation of restrictive common due-date problems with distinct and job-individual earliness and tardiness penalties, which seem to be the most difficult problems in this area of research.

The following section introduces the problem formulation and gives a short review of the literature on scheduling against common due dates. In the third section the idea and application of a short Pascal code to generate benchmark problems is described, for which we calculate upper bounds from two heuristics presented in the forth section.

2. Problem formulation

There are n jobs available at time zero, which have to be processed on a single machine. Each of these jobs needs exactly one operation. The processing times p_i of the jobs $i = 1, \ldots, n$ are deterministic and known, preemption of jobs is not allowed. If the completion time C_i of job i is smaller than or equal to the common due date d, which is assumed as given, the jobs' earliness is $E_i = d - C_i$. Accordingly, a job i is tardy with the tardiness $T_i = C_i - d$, if its completion time is greater than the common due date d. As it is not known in advance whether a job will be completed before or after the due date, earliness and tardiness are calculated as $E_i = \max\{d - C_i, 0\}$ and $T_i = \max\{C_i - d, 0\}$ for all jobs $i = 1, \ldots, n$. The per time unit penalties of the job i for being early or tardy are α_i and β_i, respectively. The objective is to jointly minimize the sum of earliness and tardiness penalties

$$f(S) = \sum_{i=1}^{n} \alpha_i E_i + \sum_{i=1}^{n} \beta_i T_i, \tag{1}$$

where S denotes a feasible schedule of the jobs.

To distinguish the terms 'sequence' and 'schedule', which are often used synonymously, we refer to a sequence π as the order in which the jobs are processed. A schedule S on the other hand contains all the information necessary for the manufacture of the jobs, that is, the sequence of the

jobs, the starting time of the first job (it is easy to show that idle times between consecutive jobs cannot be advantageous in common due-date scheduling) and the common due date. Only if these three pieces of information are given, has a schedule been described completely.

In objective function (1), the common due date d might be a decision variable whose value has to be determined, or it might have been given externally. Suppose a due date $d > \sum_{i=1}^{n} p_i$ is given; an (global) optimal schedule S^* with the sequence π^* of the jobs around this due date and its objective function value $f(S^*)$ can be constructed. A common due date is called unrestrictive as long as the optimal schedule S^* (with $f(S^*)$) can be realized. Obviously, an (externally) given common due date, for which $d \geqslant \sum_{i=1}^{n} p_i$ holds, is unrestrictive. Furthermore, the due date is called unrestrictive if it is a decision variable. To summarize, we refer to a common due date as unrestrictive, if the optimal sequence π^* can be constructed without considering the (value of) the due date. Otherwise the common due date is called restrictive. For a good introduction to common due-date problems, see Baker and Scudder [2].

In the following, the most important complexity results for unrestrictive and restrictive common due-date scheduling problems are summarized.

For $\alpha_i = \beta_i = 1$ and d being unrestrictive problem (1) is polynomially solvable in $O(n \log n)$ time by a matching algorithm; see Kanet [1]. Even with $\alpha \neq \beta$ where $\alpha_i = \alpha$ and $\beta_i = \beta$ for all jobs $i = 1, \ldots, n$ the problem remains polynomially solvable, see Panwalkar et al. [3]. The unrestrictive common due-date problem becomes NP-hard if different penalties for the jobs are considered. For the case that the earliness and tardiness penalties are equal, that is $\alpha_i = \beta_i$ for $i = 1, \ldots, n$, it is possible to construct a dynamic optimization algorithm. Its running time $O(n\sum_{i=1}^{n} p_i)$ is pseudopolynomial as it depends on the sum of processing times; see Hall and Posner [4]. In the context of common due-date scheduling this problem is typically referred to as the weighted problem. The problem with an unrestrictive common due date and no restrictions upon the penalties (called general problem) is NP-hard. A pseudopolynomial algorithm is only known for the special case where $p_i/\alpha_i \geqslant p_k/\alpha_k$ implies $p_i/\beta_i \geqslant p_k/\beta_k$, see Lee et al. [5]. These authors conjecture that the problem is NP-hard in the strong sense if this special case does not hold. In any event, the complexity status of the general problem still remains open. Dileepan [6], De et al. [7] and Van den Akker et al. [8] propose branch-and-bound algorithms for solving it.

The restrictive common due-date problem is NP-hard even if $\alpha_i = \beta_i = 1$ for $i = 1, \ldots, n$, which has been proven independently by Hall et al. [9] and Hoogeveen and Van de Velde [10], who construct a pseudopolynomial dynamic optimization algorithm whose running time $O(n\sum_{i=1}^{n} p_i)$ again depends on the sum of processing times. Kahlbacher [11] extended the results of Hall et al. [9] and Hoogeveen and Van de Velde [10] to the restrictive problem with $\alpha_i = \alpha$ and $\beta_i = \beta$ for $i = 1, \ldots, n$. To the best of our knowledge no optimizing solution procedures exist for the restrictive common due-date problem with different earliness and tardiness penalties.

The complexity results stated in the former paragraph are summarized in Table 1.

Due to the complexity results it is most unlikely to find efficient algorithms for the general problem with a restrictive due date. To obtain optimal schedules S^* for small instances of this problem the following mixed-integer programming formulation is applied. Let s_i and x_{ik} be the decision variables which (together with d) determine the schedule S^*; s_i is the starting time of job i and x_{ik} takes the value 1 if job i is sequenced (not necessarily directly) prior to job k and $x_{ik} = 0$ otherwise. Further, let R be a sufficiently large number. The objective is to find a schedule S^* which

Table 1
An overview of complexity results for restrictive and unrestrictive common due-date problems

	d is unrestrictive	d is restrictive
$\alpha_i = \beta_i = 1$ for $i = 1, \ldots, n$	Polynomially solvable	NP-hard, pseudopolynomially solvable
$\alpha_i = \alpha$ and $\beta_i = \beta$ for $i = 1, \ldots, n$	Polynomially solvable	NP-hard, pseudopolynomially solvable
$\alpha_i = \beta_i$ for $i = 1, \ldots, n$ (Weighted problem)	NP-hard, pseudopolynomially solvable	NP-hard, pseudopolynomially solvable
α_i, β_i (General problem)	NP-hard, open	NP-hard, open

minimizes (1) subject to the following restrictions:

$$T_i \geqslant s_i + p_i - d, \quad i = 1, \ldots, n, \tag{2}$$

$$E_i \geqslant d - s_i - p_i, \quad i = 1, \ldots, n, \tag{3}$$

$$s_i + p_i \leqslant s_k + R(1 - x_{ik}), \quad i = 1, \ldots, n-1;\ k = i+1, \ldots, n, \tag{4}$$

$$s_k + p_k \leqslant s_i + Rx_{ik}, \quad i = 1, \ldots, n-1;\ k = i+1, \ldots, n, \tag{5}$$

$$T_i, E_i, s_i \geqslant 0, \quad i = 1, \ldots, n, \tag{6}$$

$$x_{ik} \in \{0, 1\}, \quad i = 1, \ldots, n-1;\ k = i+1, \ldots, n. \tag{7}$$

The values for earliness and tardiness are calculated by restrictions (2) and (3). Restrictions (4) and (5) determine the starting times of the jobs: If job i is sequenced prior to job k the restriction $s_i + p_i \leqslant s_k$ only holds if $x_{ik} = 1$. Because of the addition of R, (5) is not restrictive with $x_{ik} = 1$. On the other hand, for $x_{ik} = 0$ restriction (5) gives $s_k + p_k \leqslant s_i$ and (4) is not restrictive. Note that this formulation uses the order-dependent binary variables proposed by Manne [12]. However, even with this formulation, the problem remains computationally demanding.

Example 1. Given an instance with 8 jobs ($n = 8$) and the following data (Table 2). The optimal objective function value can be obtained by applying the above problem formulation (1)–(7). The example instance has been solved by LINDO for all due dates taking integer values between 0 and 70, i.e. $d = 0, 1, 2, \ldots, 70$, as 70 equals the sum of the processing times. The resulting 71 objective function values are connected and plotted in Fig. 1.

The resulting function is non-increasing in d. However, $f(S(d))$ is not concave and its slope differs; even for some intervals of d the function $f(S(d))$ is parallel to the abscissa. The example instance is unrestrictive for $d \geqslant 51$, as for these values the (global) optimal value of $f(S(d)) = 438$ can be realized.

3. Generating benchmark problems

For the restrictive general problem, i.e. that of the south-east corner of Table 1, we propose benchmark problems with 10, 20, 50, 100, 200, 500 and 1000 jobs. For each of these problem sizes

Table 2
Example instance for $n = 8$

	1	2	3	4	5	6	7	8
P_i	7	1	18	6	13	14	5	6
α_i	2	8	4	9	5	5	7	4
β_i	14	7	8	9	7	9	5	14

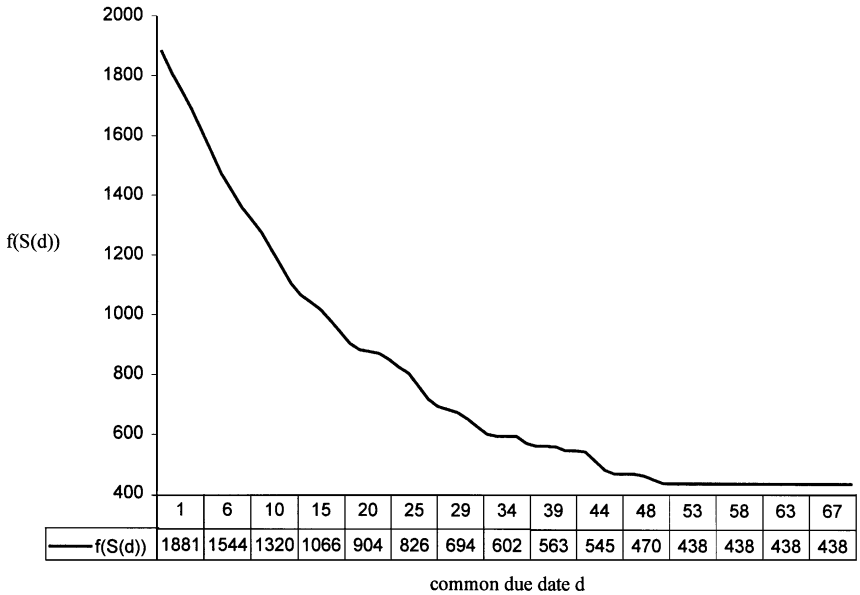

Fig. 1. The objective function values for Example 1.

10 different instances are generated, hence we obtain 70 problem instances altogether. The instances are generated by using the well-known linear congruential random number generator developed by Lehmer [13].

In the following, we give a brief description of the random number generator used for a computer with a 32-bit word. For further details refer Knuth [14] or Sedgewick [15]. The linear congruential generator is based on the recursion

$$X_{n+1} = (cX_n + e) \bmod m,$$

where X_n denotes the nth pseudo-random number generated and c, e and m are constants. Their values can be set by the user subject to the following restrictions; see Knuth [14]: The value of c should end with $\ldots y21$, where y is even. Furthermore, m should be large but smaller than or equal to $2^{31} - 1$ and $0 \leqslant e < m$. The initial seed X_0 can be chosen arbitrarily because it does not provoke a special pattern of random numbers.

In our formulation, the generator is initialized by the sum of an arbitrarily chosen constant (i.e. 3,794,612) plus k plus n, where $1 \leqslant k \leqslant 10$ denotes the kth instance of the problem size n. For example, the third 50-job instance is unequivocally determined by the number $50 + 3(+ 3{,}794{,}612)$. We set the following data for the problem generator:

$$c = 31{,}415{,}821; \quad m = 100{,}000{,}000; \quad e = 1$$

and

$$X_0 = 3{,}794{,}612 + n + k \text{ for the } k\text{th problem instance with } n \text{ jobs.}$$

Since multiplying cX_n can easily cause overflow errors, c and X_n are divided into $10^4 c_1 + c_2$ and $10^4 X_{n1} + X_{n2}$, respectively. It follows:

$$cX_n = (10^4 c_1 + c_2)(10^4 X_{n1} + X_{n2}) = 10^8 c_1 X_{n1} + 10^4 (c_1 X_{n2} + c_2 X_{n1}) + c_2 X_{n2}.$$

As only the last eight digits are of interest for the linear congruential random number generator, the term $10^8 c_1 X_{n1}$ and the last four digits of the second term are ignored:

$$X_{n+1} = [(c_1 X_{n2} + c_2 X_{n1}) + c_2 X_{n2} + 1] \bmod 100{,}000{,}000.$$

The modulus division restricts the value of X_{n+1} to numbers which consist of eight or fewer digits. To obtain an integer s of the interval [1, range] from the random numbers X_n, the first four digits of X_n are multiplied with the range and the result is divided by 10,000 again. Since this would give an integer which lies in the interval [0, range-1], the value 1 is added:

$$s = \left\lfloor \frac{\lfloor X_n/10{,}000 \rfloor \text{range}}{10{,}000} \right\rfloor + 1,$$

where $\lfloor x \rfloor$ denotes the biggest integer smaller than or equal to x. Repeating this procedure leads to uniformly distributed integers of the interval [1, range], see, Knuth [14].

We assume for the proposed benchmark problems that the processing times are integers of the interval [1, 20], the earliness penalties are of [1, 10] and the tardiness penalties are of [1, 15]. The output of the problem generator has to be interpreted in the following way: the first integer states the processing time, the second integer the earliness penalty and the third integer the tardiness penalty of job 1, respectively. The processing time, earliness and tardiness penalty of job 2 are given by the following three integers, etc.

Determining a problem instance solely by the values of n and k has two advantages. Firstly, it is possible to generate a certain instance without transcribing a particular random seed number (even without looking into this paper again). Secondly, a simple *for*-loop can be implemented, so that all 10 instances of a certain problem size are generated automatically. We hope to minimize the potential numbers of transmission errors by this procedure. An executable version of the problem generator is given in the appendix.

Generally more or less restrictive due dates are imaginable and reasonable. The lower the value of d, the higher the expected percentage of late jobs. Besides, for $d = 0$ the problem becomes trivial, because it would be optimal to sequence the jobs according to non-decreasing ratios p_i/β_i, which is a straightforward consequence of Property 1 stated further on. Hence, for each instance four different values for d are given, that is $d = \lfloor h \sum_{i=1}^{n} p_i \rfloor$ with $h = 0.2,\ 0.4,\ 0.6$ and 0.8. For each of the

70 problem instances these four common due date values are taken into account, so that altogether 280 benchmark problems are proposed.

Note that these benchmark problems might easily be modified to obtain different (easier) problems. For example, setting $d = \sum_{i=1}^{n} p_i$ gives the unrestrictive version of the general problem. Using only the proposed earliness penalties, irrespective of whether the jobs are early or tardy, leads (with $d = \sum_{i=1}^{n} p_i$) to the weighted problem.

Furthermore, it is possible to neglect the different values of the earliness and tardiness penalties proposed and to tackle the instances of the restrictive common due-date problem with $\alpha_i = \beta_i = 1$. It is obvious that all versions of NP-hard common due-date problems can be conducted easily from the different instances. The advantage of using the benchmarks proposed for other (easier) problems as well lies again in the possibility it offers of making the results verifiable for the readers and comparable to other algorithms.

4. Heuristical approaches to the restrictive general problem

As mentioned before an optimization algorithm for the restrictive common due-date problem with general penalties has never been proposed. The only researchers who have studied this problem have been Lee and Kim [16] and James [17], in both cases using metaheuristics, namely parallel genetic algorithms and tabu search. We decided not to adopt their approaches for two reasons. Firstly, both limit their search space to schedules in which the first job starts at time zero, although this might exclude optimal schedules a priori, see Szwarc [18]. Secondly the application of a metaheuristic to a special (undocumented) instance is hardly reproducible. For benchmark problems in particular, it is desirable to tackle them by means of deterministic approaches, which are easy to understand and reproduce. Therefore, we constructed two straightforward heuristics to obtain upper bounds on the optimal objective function values for the proposed benchmarks. As these heuristics make use of the following properties, a short analysis of the problem is given before the heuristics are presented.

It is obvious that an optimal schedule cannot have any idle times between consecutive jobs; for a general proof of this property, see Cheng and Kahlbacher [19]. Furthermore, the following two properties hold, both well known in the area of common due-date scheduling.

Property 1. For the restrictive common due-date problem with general penalties an optimal schedule, which has the so-called V-shaped property, exists. This means the jobs i which are completed at or before the due date ($C_i \leqslant d$) are ordered according to non-increasing ratios p_i/α_i. The jobs i whose processing starts at or after the due date ($C_i - p_i \geqslant d$), are sequenced in non-decreasing order of the ratios p_i/β_i.

Proof. The proof can be made by the standard interchange argument; see for example, Baker and Scudder [2]. □

Note that Property 1 leaves open the possibility that an optimal schedule with a so-called straddling job exists. A job is called 'straddling' if its processing is started before and finished after

the due date. If in an optimal schedule a straddling job j exists, its ratios p_j/α_j and p_j/β_j can be greater than that of the remaining jobs; see, for example, Hoogeveen and Van de Velde [10].[1]

Property 2. For the restrictive common due-date problem with general penalties an optimal schedule exists in which either the processing of the first job starts at time zero or one job is completed at the due date.

Proof. The proof is similar to that of Hoogeveen and Van de Velde [10] for the restrictive common due-date problem with weighted penalties. □

Note that the Property 2 leaves open the possibility that an optimal schedule starts at time zero and the completion time of one job coincides with the due date. Nevertheless, in view of Property 2, the search for an optimal schedule should not be restricted to sequences starting at time zero.

Two different heuristics are proposed to tackle the problem. In both of them we restrict the search to V-shaped schedules, in which the completion time of one job coincides with the due date and the first job is not necessarily started at time zero (i.e. to schedules without a straddling job). Further, we carried out computational tests on a third heuristic, which arranges the jobs in the time interval $(0, \sum_{i=1}^{n} p_i)$, so that a straddling job might occur, but the results were disappointing. Hence, we decided not to introduce it.

Let A be the ordered set of jobs which are started after the due date and B be the ordered set of jobs which are completed before. The idea of the first heuristic presented is due to that of Dileepan [6].

Heuristic I. In a first step all jobs are assumed to be late, that is $A = \{1, 2, \ldots, n\}$. The objective function value f is calculated by sequencing the jobs according to the Property 1 and starting the first job at time d. In each of the following steps of the Heuristic I job k is selected and shifted from set A to set B, which achieves the greatest reduction of the objective function value. Note that only one reasonable possibility for inserting job k into the ordered set B exists, namely according to the V-shaped property. Hence, in each step of the Heuristic I it is necessary to consider $|A|$ potential jobs to be moved, where $|x|$ denotes the (remaining) number of elements in set x. The algorithm terminates as soon as the objective function value cannot be decreased anymore, which could occur for two reasons. Firstly, it might simply be disadvantageous to shift another job from set A to set B. Secondly, there might not be enough idle time to insert a job into set B, that is, the difference $d - \sum_{i \in B} p_i$ is smaller than the smallest processing time of all jobs of the present set A, for which a removal would decrease the objective function value. Obviously, an upper bound on the running time for this algorithm is given by $O(n^2)$.

The underlying idea of the second heuristic is to schedule the jobs with relatively high tardiness costs early. Therefore, the jobs are ordered according to decreasing ratios β_i/α_i. Let

[1] The example of Hoogeveen and Van de Velde [10, p. 238] has to be extended by a fourth job with, for example, $p_4 = 9$ and $\alpha_4 = \beta_4 = 4$, to obtain the desired result; otherwise, schedule (2, 1, 3) is not optimal.

$P^{\alpha\beta} := \{\beta_{[1]}/\alpha_{[1]}, \beta_{[2]}/\alpha_{[2]}, \ldots, \beta_{[n]}/\alpha_{[n]}\}$ with $\beta_{[1]}/\alpha_{[1]} \geqslant \beta_{[2]}/\alpha_{[2]} \geqslant, \ldots, \geqslant \beta_{[n]}/\alpha_{[n]}$ and all jobs with equal ratios are ordered according to decreasing values of β. That means for all jobs i of $P^{\alpha\beta}$: if $\beta_{[i]}/\alpha_{[i]} = \beta_{[i+1]}/\alpha_{[i+1]}$ then $\beta_{[i]} \geqslant \beta_{[i+1]}$; $[i]$ denotes the job occupying the ith position in $P^{\alpha\beta}$.

Heuristic II. Take the first job of $P^{\alpha\beta}$, assign it to set B and delete it from $P^{\alpha\beta}$. Continue with this iteration until one of the following situations occurs:

(a) The time gap $d - \sum_{i\in B} p_i$ is too small to assign the first job of $P^{\alpha\beta}$ to B. If in this case the time gap is greater than zero search for another job of $P^{\alpha\beta}$ which fits into the gap. As soon as the time gap is zero or no more job of $P^{\alpha\beta}$ fits into the gap, assign the remaining jobs of $P^{\alpha\beta}$ to set A, sequence all jobs in A and B according to the V-shaped property and calculate the objective function value, say f^{II}.

(b) $n/2$ jobs have been assigned to set B. In this situation a short (deterministic) local search is carried out: in each step set B is enlarged by the first of the remaining jobs of $P^{\alpha\beta}$ if (and only if) this decreases the objective function value. As soon as the objective function value increases by adding one more job to B, the local search is terminated and the lowest, i.e. the penultimate objective function value, is saved as f^{II}.

Suppose Heuristic II was applied without considering situation (b). The objective function value f^{II} would become a convex function in d: if d is very small, f^{II} decreases with an increasing d. As soon as d exceeds a particular value it is no longer advantageous to assign jobs to set B. Therefore, the local search procedure (b) is implemented. It is evident that the local search primary applies for the larger values of h.

To verify the program given in the appendix we state the sum of processing times of the problem instances with 10 jobs in the Table 3 and demonstrate the heuristics by means of the following example:

Example 1 *(continued)*. Generating the first instance of an 8 job problem ($n = 8$, $k = 1$) should give the data of Example 1. With $\sum_{i=1}^{8} p_i = 70$ and $h = 0.2$ the common due date is $d = 14$.

Applying the Heuristic I yields (after one iteration) sequence (6, 2, 8, 1, 4, 7, 5, 3), in which job 6 is completed at the due date and (accidentally) started at time zero. The objective function value of this schedule is $f^{\mathrm{I}} = 1320$.

Applying Heuristic II yields the sequence (1, 8, 2, 4, 7, 6, 5, 3), in which job 2 is completed at the due date and job 1 is (accidentally) started at time zero: see Fig. 2. The objective function value of this schedule is $f^{\mathrm{II}} = 1066$.

As a consequence, 1066 is taken as the upper bound on the optimal objective function value. Note that this objective function value has been found to be optimal (by solving mixed-integer formulation (1)–(7)).

The best objective function values for the 280 instances yielded by the two heuristics are stated in Tables 3–9. They are upper bounds on the optimal objective function value for the particular instance. We index them as I and/or or II in order to show by which of the two heuristics the particular upper bound has been obtained. The optimal objective function values for the instances with 10 jobs are given in square brackets, if they are not obtained by one of the heuristics; otherwise, they are indicated by asterisk.

Table 3
The upper bounds and optimal objective function values for the 10 job examples

10 jobs ($n = 10$)	$\sum_{i=1}^{10} p_i$	$h = 0.2$	$h = 0.4$	$h = 0.6$	$h = 0.8$
$k = 1$	116	2009^{I} [1936]	$1057^{I/II}$ [1025]	$841^{*I/II}$	$818^{*I/II}$
$k = 2$	129	1125^{II} [1042]	615^{*I}	615^{*I}	615^{*I}
$k = 3$	125	$1731^{I/II}$ [1586]	931^{I} [917]	$793^{*I/II}$	$793^{*I/II}$
$k = 4$	102	2392^{I} [2139]	1251^{II} [1230]	$815^{*I/II}$	$815^{I/II}$ [803]
$k = 5$	94	1220^{I} [1187]	661^{I} [630]	$521^{*I/II}$	$521^{*I/II}$
$k = 6$	88	1623^{I} [1521]	908^{*II}	$755^{*I/II}$	$755^{*I/II}$
$k = 7$	103	2269^{I} [2170]	1374^{*I}	$1102^{I/II}$ [1101]	$1083^{*I/II}$
$k = 8$	79	1774^{II} [1720]	1104^{II} [1020]	610^{*I}	540^{*I}
$k = 9$	92	1792^{I} [1574]	876^{*I}	582^{*I}	$554^{*I/II}$
$k = 10$	127	1934^{II} [1869]	1173^{I} [1136]	$711^{I/II}$ [710]	$671^{*I/II}$

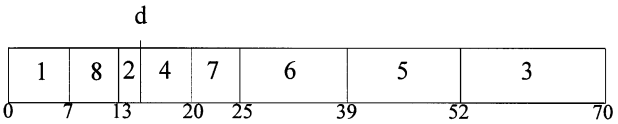

Fig. 2. The schedule for the $n = 8$, $k = 1$ problem instance obtained by Heuristic II.

It is obvious that for the larger problem instances the Heuristic II yields better objective function values than the Heuristic I. As regards the results of the 10 job examples the bounds for the instances with more restrictive due dates are worse than those of the less restrictive problems. This observation coincides with the complexity results of Section 2 and probably holds for all the generated bounds.

However, as it is not easy to find an optimal solution, especially for the larger problem instances, the quality of the upper bounds is hard to ascertain. For instances with up to 10 jobs it was possible

Table 4
The upper bounds for the 20 job examples

20 jobs ($n = 20$)	$h = 0.2$	$h = 0.4$	$h = 0.6$	$h = 0.8$
$k = 1$	4431^{II}	3066^{I}	2986^{I}	2986^{I}
$k = 2$	8567^{I}	4897^{II}	3260^{I}	$2980^{I/II}$
$k = 3$	6331^{II}	3883^{II}	3600^{I}	3600^{I}
$k = 4$	9478^{I}	5122^{II}	3336^{II}	3040^{I}
$k = 5$	4340^{II}	2571^{II}	2206^{I}	2206^{I}
$k = 6$	6766^{II}	3601^{I}	3016^{II}	3016^{II}
$k = 7$	$11{,}101^{I}$	6357^{II}	4175^{II}	3900^{II}
$k = 8$	4203^{II}	2151^{II}	1638^{I}	1638^{I}
$k = 9$	3530^{II}	2097^{II}	1992^{I}	1992^{I}
$k = 10$	5545^{II}	3192^{I}	2116^{II}	1995^{I}

Table 5
The upper bounds for the 50 job examples

50 jobs ($n = 50$)	$h = 0.2$	$h = 0.4$	$h = 0.6$	$h = 0.8$
$k = 1$	$42{,}363^{II}$	$24{,}868^{II}$	$17{,}990^{II}$	$17{,}990^{II}$
$k = 2$	$33{,}637^{II}$	$19{,}279^{II}$	$14{,}231^{II}$	$14{,}132^{II}$
$k = 3$	$37{,}641^{I}$	$21{,}353^{II}$	$16{,}497^{II}$	$16{,}497^{II}$
$k = 4$	$30{,}166^{II}$	$17{,}495^{I}$	$14{,}105^{II}$	$14{,}105^{II}$
$k = 5$	$32{,}604^{II}$	$18{,}441^{II}$	$14{,}650^{II}$	$14{,}650^{II}$
$k = 6$	$36{,}920^{I}$	$21{,}497^{I}$	$14{,}251^{I}$	$14{,}075^{II}$
$k = 7$	$44{,}277^{II}$	$23{,}883^{II}$	$17{,}715^{II}$	$17{,}715^{II}$
$k = 8$	$46{,}065^{II}$	$25{,}402^{II}$	$21{,}367^{I}$	$21{,}367^{I}$
$k = 9$	$36{,}397^{I}$	$21{,}929^{I}$	$14{,}298^{II}$	$13{,}952^{I}$
$k = 10$	$35{,}797^{I}$	$20{,}048^{II}$	$14{,}377^{II}$	$14{,}377^{II}$

Table 6
The upper bounds for the 100 job examples

100 jobs ($n = 100$)	$h = 0.2$	$h = 0.4$	$h = 0.6$	$h = 0.8$
$k = 1$	$156{,}103^{I}$	$89{,}588^{II}$	$72{,}019^{II}$	$72{,}019^{II}$
$k = 2$	$132{,}605^{II}$	$74{,}854^{II}$	$59{,}351^{II}$	$59{,}351^{II}$
$k = 3$	$137{,}463^{II}$	$85{,}363^{II}$	$68{,}537^{I}$	$68{,}537^{I}$
$k = 4$	$137{,}265^{II}$	87.730^{II}	$69{,}231^{II}$	$69{,}231^{II}$
$k = 5$	$136{,}761^{II}$	$76{,}424^{II}$	$55{,}291^{II}$	$55{,}277^{II}$
$k = 6$	$151{,}938^{I}$	$86{,}724^{I}$	$62{,}519^{II}$	$62{,}519^{II}$
$k = 7$	$141{,}613^{II}$	$79{,}854^{II}$	$62{,}213^{II}$	$62{,}213^{II}$
$k = 8$	$168{,}086^{II}$	$95{,}361^{II}$	$80{,}844^{I}$	$80{,}844^{I}$
$k = 9$	$125{,}153^{II}$	$73{,}605^{II}$	$58{,}771^{II}$	$58{,}771^{II}$
$k = 10$	$124{,}446^{II}$	$72{,}399^{II}$	$61{,}419^{II}$	$61{,}419^{II}$

Table 7
The upper bounds for the 200 job examples

200 jobs ($n = 200$)	$h = 0.2$	$h = 0.4$	$h = 0.6$	$h = 0.8$
$k = 1$	$526{,}666^{II}$	$301{,}449^{II}$	$254{,}268^{II}$	$254{,}268^{II}$
$k = 2$	$566{,}643^{II}$	$335{,}714^{II}$	$266{,}028^{II}$	$266{,}028^{II}$
$k = 3$	$529{,}919^{I}$	$308{,}278^{II}$	$254{,}647^{II}$	$254{,}647^{II}$
$k = 4$	$603{,}709^{II}$	$360{,}852^{II}$	$297{,}269^{II}$	$297{,}269^{II}$
$k = 5$	$547{,}953^{II}$	$322{,}268^{II}$	$260{,}455^{II}$	$260{,}455^{II}$
$k = 6$	$502{,}276^{II}$	$292{,}453^{II}$	$236{,}160^{II}$	$236{,}160^{II}$
$k = 7$	$479{,}651^{II}$	$279{,}576^{II}$	$247{,}555^{II}$	$247{,}555^{II}$
$k = 8$	$530{,}896^{II}$	$288{,}746^{II}$	$225{,}572^{II}$	$225{,}572^{II}$
$k = 9$	$575{,}353^{I}$	$331{,}107^{II}$	$255{,}029^{II}$	$255{,}029^{II}$
$k = 10$	$572{,}866^{II}$	$332{,}808^{II}$	$269{,}236^{II}$	$269{,}236^{II}$

to find optimal solutions in a reasonable time. We generated 80 problem instances with $n = 9$ and 10 ($k = 1, 2, \ldots, 10$ and $h = 0.2, 0.4, 0.6$ and 0.8) and solved them optimally with LINDO: 22 of the 40 upper bounds of the 9 job examples turned out to be optimal and the average percentage deviation from the optimal objective function value of the 40 bounds was 2.36%. With regard to the instances with 10 jobs (see Table 3) 21 of the 40 upper bounds were optimal and the average deviation from the optimal objective function value is 2.28%.

There is no reason, moreover, why the quality of the two heuristics and hence the quality of the upper bounds should worsen with an increasing number of jobs.

Table 8
The upper bounds for the 500 job examples

500 jobs ($n = 500$)	$h = 0.2$	$h = 0.4$	$h = 0.6$	$h = 0.8$
$k = 1$	3,113,088[II]	1,839,902[II]	1,581,233[II]	1,581,233[II]
$k = 2$	3,569,058[II]	2,064,998[II]	1,715,332[II]	1,715,322[II]
$k = 3$	3,300,744[II]	1,909,304[II]	1,644,947[II]	1,644,947[II]
$k = 4$	3,408,867[II]	1,930,829[II]	1,640,942[II]	1,640,942[II]
$k = 5$	3,377,547[II]	1,881,221[II]	1,468,325[II]	1,468,325[II]
$k = 6$	3,024,082[II]	1,658,411[II]	1,413,345[II]	1,413,345[II]
$k = 7$	3,381,166[II]	1,971,176[II]	1,634,912[II]	1,634,912[II]
$k = 8$	3,376,678[II]	1,924,191[II]	1,542,090[II]	1,542,090[II]
$k = 9$	3,617,807[II]	2,065,647[II]	1,684,055[II]	1,684,055[II]
$k = 10$	3,315,019[II]	1,928,579[II]	1,520,515[II]	1,520,515[II]

Table 9
The upper bounds for the 1000 job examples

1000 jobs ($n = 1000$)	$h = 0.2$	$h = 0.4$	$h = 0.6$	$h = 0.8$
$k = 1$	15,190,371[II]	8,570,154[II]	6,411,581[II]	6,411,581[II]
$k = 2$	13,356,727[II]	7,592,040[II]	6,112,598[II]	6,112,598[II]
$k = 3$	12,919,259[II]	7,313,736[II]	5,985,538[II]	5,985,538[II]
$k = 4$	12,705,290[II]	7,300,217[II]	6,096,729[II]	6,096,729[II]
$k = 5$	13,276,868[II]	7,738,367[II]	6,348,242[II]	6,348,242[II]
$k = 6$	12,236,080[II]	7,144,491[II]	6,082,142[II]	6,082,142[II]
$k = 7$	14,160,773[II]	8,426,024[II]	6,575,879[II]	6,575,879[II]
$k = 8$	13,314,723[II]	7,508,507[II]	6,069,658[II]	6,069,658[II]
$k = 9$	12,433,821[II]	7,299,271[II]	6,188,416[II]	6,188,416[II]
$k = 10$	13,395,234[II]	7,617,658[II]	6,147,295[II]	6,147,295[II]

5. Conclusions

In this paper we presented 280 benchmarks for the restrictive common due-date problem with general earliness and tardiness penalties. We hope that future approaches to tackling this NP-hard problem will use the proposed benchmarks in order to make the results and hence the quality of the algorithms comparable, see Feldmann and Biskup [20].

The authors are gladly willing to distribute a longer and more user-friendly version of the problem generator by email.

Acknowledgements

We wish to thank two anonymous referees for their helpful comments on an earlier version of this paper.

Appendix

```
Program Generator_for_Due_Date_Problems;
CONST m = 100000000; m1 = 10000; b = 31415821;

VAR  seed :      LONGINT;
     n :         INTEGER;
     k :         INTEGER;
     range_p :   BYTE;
     range_a :   BYTE;
     range_b :   BYTE;
     i :         INTEGER;
     problem :   ARRAY [1..20000] of BYTE;

{##################################################}
FUNCTION Mult(p,q: LONGINT) : LONGINT;
VAR p1,p0,q1,q0: LONGINT;
BEGIN
   p1:= p DIV m1; p0:= p MOD m1;
   q1:= q DIV m1; q0:= q MOD m1;
   mult:= (((p0*q1 + p1*q0) MOD m1)*m1 + p0*q0) MOD m;
END;

{##################################################}
FUNCTION Randomint(range: BYTE) : BYTE;
BEGIN
   seed:= (Mult (seed,b) + 1) MOD m;
   Randomint:= (((seed DIV m1)*range)DIV m1) + 1
END;

{##################################################}
PROCEDURE Generate_Problem;

CONST start_seed = 3794612;

BEGIN;
   WRITE ('PROBLEM SIZE: ');                READLN (n);
   WRITE ('INSTANCE NUMBER: ');             READLN (k);
```

```
   WRITE ('RANGE PROCESSING TIME: ');      READLN (range_p);
   WRITE ('RANGE EARLINESS PENALTY: ');    READLN (range_a);
   WRITE ('RANGE LATENESS PENALTY: ');     READLN (range_b);
   i := 0;
   seed := start_seed + n + k;
REPEAT
   BEGIN
   INC(i); problem[i] := Randomint(range_p);
   INC(i); problem[i] := Randomint(range_a);
   INC(i); problem[i] := Randomint(range_b);
   END
UNTIL (i = 3*n);
END;

{##########################################################}
```

References

[1] Kanet JJ. Minimizing the average deviation of job completion times about a common due date. Naval Research Logistics Quarterly 1981;28:643–51.

[2] Baker KR, Scudder GD. Sequencing with earliness and tardiness penalties: a review. Operations Research 1990;38:22–36.

[3] Panwalkar SS, Smith ML, Seidmann A. Common due date assignment to minimize total penalty for the one machine scheduling problem. Operations Research 1982;30:391–9.

[4] Hall NG, Posner ME. Earliness-tardiness scheduling problems, I: weighted deviation of completion times about a common due date. Operations Research 1991;39:836–46.

[5] Lee C-Y, Danusaputro SL, Lin C-S. Minimizing weighted number of tardy jobs and weighted earliness-tardiness penalties about a common due date. Computers & Operations Research 1991;18:379–89.

[6] Dileepan P. Common due-date scheduling problem with separate earliness and tardiness penalties. Computers & Operations Research 1993;20:179–84.

[7] De P, Ghosh JB, Wells CE. Solving a generalized model for CON due date assignment and sequencing. International Journal of Production Economics 1994;34:179–85.

[8] Van den Akker M, Hoogeveen H, van de Velde S. A column generation algorithm for common due-date scheduling. Memorandum COSOR 97-01, Eindhoven University of Technology, Eindhoven, Netherlands, 1997.

[9] Hall NG, Kubiak W, Sethi SP. Earliness-tardiness scheduling problems, II: Deviation of completion times about a restrictive common due date. Operations Research 1991;39:847–56.

[10] Hoogeveen JA, van de Velde SL. Scheduling around a small common due date. European Journal of Operational Research 1991;55:237–42.

[11] Kahlbacher HG. Termin- und Ablaufplanung - ein analytischer Zugang, Ph.D. thesis, University of Kaiserslautern, Kaiserslautern, Germany, 1992 (in German).

[12] Manne AS. On the job-shop scheduling problem. Operations Research 1960;8:219–23.

[13] Lehmer DH. Mathematical methods in large-scale computing units. Proceedings on the Second Symposium on Large-Scale Digital Calculating Machinery, Harvard University Press, Cambridge, UK, 1951, p. 141–6.

[14] Knuth DE. The art of computing, vol. 2, seminumerical algorithms, 3rd ed. Addison-Wesley, Amsterdam, Netherlands.

[15] Sedgewick R. Algorithms in C, Parts 1–4, 3rd ed.. Addison-Wesley, Amsterdam, Netherlands.

[16] Lee C-Y, Kim SJ. Parallel genetic algorithms for the earliness-tardiness job scheduling problem with general penalty weights. Computers & Industrial Engineering 1995;28:231–43.

[17] James RJW. Using tabu search to solve the common due date early/tardy machine scheduling problem. Computers & Operations Research 1997;24:199–208.
[18] Szwarc W. Single-machine scheduling to minimize absolute deviation of completion times from a common due date. Naval Research Logistics 1989;36:663–73.
[19] Cheng TCE, Kahlbacher HG. A proof for the longest-job-first policy in one-machine scheduling. Naval Research Logistics 1991;38:715–20.
[20] Feldmann M, Biskup D. Single-machine scheduling for minimizing earliness and tardiness penalties by meta-heuristic approaches. Discussion Paper No. 425, University of Bielefeld, Bielefeld, Germany, 1999.

Dirk Biskup obtained a degree in Economics and Business Administration from the University of Hamburg. At the moment he is working as a Research Assistant at the Chair of Production, Operations Management and Managerial Accounting (Professor Jahnke) at the University of Bielefeld, Germany. The theme of his doctorial thesis is scheduling against due dates. His research interests are in single and parallel machine scheduling, lotsizing and cost allocation.

Martin Feldmann obtained a Ph.D. in Economics and Business Administration from the University of Bielefeld. At the moment he is working as an Assistant Professor at the Chair of Operations Research and Business Administration (Professor Kistner) at the University of Bielefeld, Germany. His research interests are in production and Operations Management, Logistics and Metaheuristics.