New lower and upper bounds for scheduling around a small common due date
Hoogeveen, J.A.; Oosterhout, H.; Velde, van de, S.L.

Published: 01/01/1992

Document Version
Publisher’s PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:
• A submitted manuscript is the author's version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

Citation for published version (APA):

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.
• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Download date: 26. Oct. 2018
Memorandum COSOR 92-18

New lower and upper bounds for scheduling around a small common due date

J.A. Hoogeveen
H. Oosterhout
S.L. van de Velde

Eindhoven, June 1992
The Netherlands
New lower and upper bounds for scheduling around a small common due date

J.A. Hoogeveen
Department of Mathematics and Computing Science, Eindhoven University of Technology
P.O. Box 513, 5600 MB Eindhoven, The Netherlands

H. Oosterhout
Department of Economics, Tilburg University
P.O. Box 90153, 5000 LE Tilburg, The Netherlands

S.L. van de Velde
School of Management Studies, University of Twente
P.O. Box 217, 7500 AE Enschede, The Netherlands

Suppose a set of $n$ jobs has to be scheduled on a single machine, which can handle no more than one job at a time. The problem is to find a schedule that minimizes the sum of the deviations of the job completion times from a given common due date that is smaller than the sum of the processing times. This problem is known to be NP-hard. There exists a pseudo-polynomial algorithm that is able to solve instances up to 1000 jobs. Branch-and-bound algorithms can solve instances up to only 25 jobs. We apply Lagrangian relaxation to find new lower and upper bounds in $O(n \log n)$ time. This upper bound is used to develop a heuristic with both a good worst-case and empirical behavior: its solution value is guaranteed to be no more than $4/3$ times the optimal solution value, and always concurred with the lower bound for $n \geq 50$ in our computational experiments with processing times drawn from different distributions over the integers $1, \ldots, 100$. For the case these bounds do not concur, we present a refinement of the lower bound. This is obtained by solving a subset-sum problem to optimality by a pseudo-polynomial algorithm; this subset-sum problem is of considerably smaller dimension than the common due date problem. If these bounds still do not concur, then we apply branch-and-bound; our computational results show that only little branching is needed. Finally, we indicate to what extent the analysis also applies to the case that all early completions are weighted by a common weight $a$ and all tardy completions by a common weight $b$.

1980 Mathematics Subject Classification (1985 Revision): 90B35.
Key Words & Phrases: single-machine scheduling, common due date, Lagrangian relaxation, approximation algorithm, worst-case behavior, empirical behavior.
1. Introduction

The just-in-time concept for manufacturing has induced a new type of machine scheduling problem in which both early and tardy completions of jobs are penalized. We consider the following single-machine scheduling problem that is associated with this concept.

A set of \( n \) independent jobs has to be scheduled on a single machine, which can handle no more than one job at a time. The machine is assumed to be continuously available from time zero onwards only. Job \( J_j \) requires processing during a given uninterrupted time \( p_j \) and should ideally be completed at a given due date \( d_j \). Without loss of generality, we assume that the processing times and the due dates are integral. We assume furthermore that the jobs are indexed in order of nonincreasing processing times. A schedule \( \sigma \) defines for each job \( J_j \) a completion time \( C_j \), such that the jobs do not overlap in their execution. The earliness and tardiness of \( J_j \) are defined as \( E_j = \max\{d_j - C_j, 0\} \) and \( T_j = \max\{C_j - d_j, 0\} \), respectively. The just-in-time philosophy is reflected in the objective function

\[
f(\sigma) = \sum_{j=1}^{n} (\alpha_j E_j + \beta_j T_j),
\]

where the deviation of \( C_j \) from \( d_j \) is penalized by either \( \alpha_j \) or \( \beta_j \), depending on whether \( J_j \) is early or tardy, for \( j = 1, \ldots, n \). For a review of problems with this type of objective function, we refer to the survey by Baker and Scudder (1990).

An important subclass contains problems with a due date \( d \) that is common to all jobs. The common due date is either specified as part of the problem instance, or is a decision variable that has to be optimized simultaneously with the job sequence. As the first job may start later than time zero, the optimal schedule is identical for both problems unless the common due date \( d \) is restrictively small. The first variant is therefore referred to as the restricted problem and the second variant as the unrestricted problem.

Bagchi, Chang, and Sullivan (1987) propose a branch-and-bound approach for the restricted variant with all earliness penalties equal to \( \alpha \) and with all tardiness penalties equal to \( \beta \). Szwarc (1989) presents a branch-and-bound approach for the case that \( \alpha = \beta \). These branch-and-bound algorithms are able to solve instances up to 25 jobs. Sundararaghavan and Ahmed (1984) present an approximation algorithm for the case \( \alpha = \beta \) that shows a remarkably good performance from an empirical point of view. Lee and Liman (1992) present an approximation algorithm for the case \( \alpha = \beta \) with performance guarantee \( 3/2 \); this means that for any instance their approximation algorithm produces a solution with value no more than \( 3/2 \) times the optimal solution value. Hall, Kubiak and Sethi (1991) and Hoogeveen and Van de Velde (1991) establish the \( \Sigma P \)-hardness of the problem, even if \( \alpha = \beta \), thereby justifying the enumerative and approximative approaches. Hall et al. (1991) propose furthermore a pseudo-polynomial time algorithm running in \( O(n \Sigma p_j) \) time and space for general \( \alpha \) and \( \beta \), and provide computational results for instances up to 1000 jobs for the case \( \alpha = \beta \). Their experiments, however, show that the algorithm is mainly limited by space, not time.

We present a branch-and-bound algorithm for the case \( \alpha = \beta \). Using Lagrangian relaxation, we find new lower and upper bounds in \( O(n \log n) \) time. All our computational experiments with processing times drawn from different distributions over the integers \( 1, \ldots, 100 \) exhibit that for \( n \geq 50 \) the bounds always concur.

For the case that these bounds do not concur, we present a refinement of the lower bound, which is obtained by solving a subset-sum problem to optimality by a pseudo-polynomial algorithm. This can be done very fast, since the subset-sum problem in our application is of a considerably smaller
dimension than the common due date problem. Computational experiments show that, if any, only a small number of nodes are examined in the branch-and-bound algorithm.

In addition, we develop a heuristic that is based upon the Lagrangian upper bound with performance guarantee 4/3. This means that for any instance the heuristic produces a solution with value no more than 4/3 times the optimal value.

This paper is organized as follows. In Section 2, we review Emmons’s matching algorithm (Emmons, 1987) for the unrestricted variant of the common due date problem with general $\alpha$ and $\beta$; it is needed in Section 3, where we develop a lower bound based upon Lagrangian relaxation for the restricted variant with $\alpha = \beta$. In Section 4, we use the insight gained in Section 3 to develop a heuristic for the restricted variant. In Section 5, we show that this heuristic has performance guarantee 4/3. In Section 6, we describe the branch-and-bound algorithm, and in Section 7, we present some computational results. Finally, we briefly indicate to what extent the analysis applies to the case $\alpha \neq \beta$.

2. Emmons’s matching algorithm for the unrestricted problem
Kanet (1981) presents an $O(n \log n)$ algorithm for the unrestricted variant with $\alpha = \beta$. Bagchi et al. (1987) and Emmons (1987) propose $O(n \log n)$ algorithms for the case that $\alpha \neq \beta$. We briefly review the concepts of Emmons’s matching algorithm, since they provide the insight needed for the subsequent sections.

Theorem 1 (Kanet, 1981). No optimal schedule has idle time between the execution of the jobs. □

Theorem 2 (Kanet, 1981). There is an optimal schedule for the unrestricted variant in which the due date $d$ coincides with the start time or completion time of the job with the smallest processing time. □

Emmons’s matching algorithm is based upon the concept of positional weights. The scheduling problem reduces then to an assignment problem where jobs have to be assigned to positions. The cost of assigning $J_j$ to the $k$th early position is equal to $\alpha (k - 1) p_j$; the cost of assigning $J_j$ to the $k$th tardy position is equal to $\beta k p_j$. The assignment problem is solved in $O(n \log n)$ time by matching the job that has the $j$th largest processing time with the position that has the $j$th smallest weight, for $j = 1, \ldots, n$.

Emmons’s matching algorithm shows that in any optimal schedule the jobs completed before or at $d$ are scheduled in order of nonincreasing processing times and the jobs started at or after $d$ in order of nondecreasing processing times. Due to this structure, optimal schedules are said to be V-shaped.

Optimal schedules for the restricted variant have the same structure, albeit that there may be one job that is scheduled around $d$. For this particular job, it holds that the early jobs have equal or larger processing times, or the tardy jobs have equal or larger processing times.

3. A new lower bound for the restricted variant
We look upon this $\mathcal{NP}$-hard problem as an ‘easy’ problem complicated by the ‘nasty’ constraint that the machine is only available from time zero onwards. If this constraint were not present, then the problem could easily be solved through Emmons’s algorithm. This is exactly the approach Szwarc (1989) follows in determining a lower bound. The structure of the problem, however, suggests that the technique of Lagrangian relaxation might be more successful. We remove the nasty constraint and put it into the objective function, weighted by a nonnegative Lagrangian multiplier. The resulting
problem is easy to solve. It will be referred to as the Lagrangian problem; its solution provides a lower bound for the original problem.

The nasty constraint can be formulated as

\[ W \leq d, \]

where \( W \) denotes the total amount of work that is processed up to time \( d \). If we introduce a Lagrangian multiplier \( \lambda \geq 0 \) and bring this constraint weighted by \( \lambda \) into the objective function, then we get the following Lagrangian problem, referred to as problem \((L_\lambda)\): find the value \( L(\lambda) \), which is the minimum of

\[
\sum_{j=1}^{n} (E_j + T_j) + \lambda(W - d), \quad (L_\lambda)
\]

for a given \( \lambda \geq 0 \). Obviously, \( L(\lambda) \) is a lower bound for the original problem. There are two questions that immediately arise: Given a value of \( \lambda \), can \( L(\lambda) \) be determined in polynomial time? If so, can the value \( \lambda^* \) that maximizes the lower bound \( L(\lambda) \) be found in polynomial time? The latter problem is referred to as the Lagrangian dual problem. The following two theorems provide affirmative answers to both questions.

**Theorem 3.** For a given \( \lambda \), the Lagrangian problem is solved by applying Emmons's matching algorithm with the weights of the early positions increased by \( \lambda \).

**Proof.** Straightforward arguments show that there exists an optimal schedule for the Lagrangian problem in which some job is completed exactly on time \( d \). Hence, there is an optimal schedule with

\[ W = \sum_{j \in E} p_j, \]

where \( E \) denotes the set of jobs that are scheduled in the early and just-in-time positions. The Lagrangian objective function can then alternatively be written as

\[
\left\{ \sum_{j=1}^{n} (E_j + T_j) + \sum_{j \in E} \lambda p_j \right\} - \lambda d.
\]

Since the last term is a constant for a given \( \lambda \), we need to minimize only the expression inside the braces. This is achieved by applying Emmons's matching algorithm to the case where the weight of the \( k \)th early position is equal to \( k-1+\lambda \).

**Theorem 4.** The optimal value \( \lambda^* \), that is, the value that maximizes the Lagrangian lower bound, is equal to the index \( \lambda \) for which

\[
\left\{ \frac{(k-2)\lambda}{2} \right\} \sum_{j=0}^{k} p_{\lambda+2j} > d \geq \left\{ \frac{(k-1)\lambda}{2} \right\} \sum_{j=0}^{k} p_{\lambda+1+2j},
\]

where \( \left\{ \frac{x}{2} \right\} \) denotes the largest integer smaller than or equal to \( x \). If no such index exists, then \( \lambda^* = 0 \).

**Proof.** Consider an arbitrary value \( \lambda \). If \( \lambda \) is not integral, then all optimal schedules for \((L_\lambda)\) have equal \( W \). If \( \lambda \) is integral, then there are multiple optimal schedules with different \( W \); these are found by breaking ties differently in Emmons’s algorithm. Define for each integer \( \lambda \) (\( \lambda = 0, \ldots, n \)) \( \sigma^\text{min}_\lambda \) as the optimal schedule for the Lagrangian problem \((L_\lambda)\) with \( W \) minimal. In the same fashion, the schedule \( \sigma^\text{max}_\lambda \) is defined as the optimal schedule for the Lagrangian problem \((L_\lambda)\) with \( W \) maximal, for
\( \lambda = 0, \ldots, n \). We define \( W_{\lambda}^{\text{min}} \) and \( W_{\lambda}^{\text{max}} \) as the amount of work processed before time \( d \) in \( \sigma_{\lambda}^{\text{min}} \) and \( \sigma_{\lambda}^{\text{max}} \), respectively. Straightforward calculations show that \( \sigma_{\lambda}^{\text{min}} \) remains optimal if the Lagrangian multiplier is increased by \( \varepsilon \), with \( 0 \leq \varepsilon \leq 1 \); hence, we have that \( \sigma_{\lambda}^{\text{min}} \) is identical to \( \sigma_{\lambda+1}^{\text{max}} \) and \( W_{\lambda}^{\text{min}} = W_{\lambda+1}^{\text{max}} \). This implies that \( L(\lambda) \) is a piecewise-linear and concave function of \( \lambda \). The breakpoints correspond to the integral values \( \lambda = 1, \ldots, n \), and the gradient of the function between the integral breakpoints \( \lambda \) and \( \lambda + 1 \) is equal to \( W_{\lambda}^{\text{min}} - d \), for \( \lambda = 0, \ldots, n - 1 \). The Lagrangian dual problem is therefore solved by putting \( \lambda^* \) equal to the index \( \lambda \) for which \( W_{\lambda}^{\text{max}} > d \geq W_{\lambda}^{\text{min}} \). Due to the indexing of the jobs, the theorem follows.

Let \( \sigma^* \) be an optimal schedule for the Lagrangian dual problem. If \( \lambda^* = 0 \), then \( \sigma^* = \sigma_{\lambda=0}^{\text{min}} \) is feasible for the original problem, and hence optimal. Note that this also implies that \( d \geq p_1 + p_3 + \cdots + p_n \) if \( n \) is odd, and \( d \geq p_1 + p_3 + \cdots + p_{n-1} \) if \( n \) is even. This agrees with the result by Bagchi et al. (1987) that the schedules \((J_1, J_3, \ldots, J_n, J_{n-1}, \ldots, J_2)\) and \((J_1, J_3, \ldots, J_{n-1}, J_n, \ldots, J_2)\) are optimal under the respective conditions.

In the remainder, we assume that \( \lambda^* \geq 1 \). Depending on whether \( n - \lambda^* \) is odd or even, \( \sigma^* \) has the following structure. First, suppose \( n - \lambda^* \) is odd. Then the jobs \( J_1, \ldots, J_{\lambda^*-1} \) occupy the last \( \lambda^* - 1 \) positions in \( \sigma^* \), the pair \( \{J_{\lambda^*}, J_{\lambda^*+1}\} \) occupies the first early position and the \( \lambda^* \)th tardy position, the pair \( \{J_{\lambda^*+2}, J_{\lambda^*+3}\} \) occupies the second early position and the \( (\lambda^*+1) \)th tardy position, and so on. Finally, the pair \( \{J_{n-1}, J_n\} \) occupies the positions around the due date. Second, if \( n - \lambda^* \) is even, then \( \sigma^* \) has the same structure, except that \( J_n \) is positioned between \( J_{n-2} \) and \( J_{n-1} \), and is started somewhere in the interval \([d - p_n, d]\).

**Proposition 1.** If there exists a schedule \( \sigma^* \) that is optimal for the Lagrangian dual problem in which the first job is started at time zero, then the Lagrangian lower bound \( L(\lambda^*) \) is tight and \( \sigma^* \) is an optimal schedule for the original problem.

**Proof.** In this case we have \( L(\lambda^*) = \Sigma(E_j + T_j) + \lambda^*(W - d) = \Sigma(E_j + T_j) = f(\sigma^*) \).

If no such schedule \( \sigma^* \) exists, then there is a gap between the optimal value for the original problem and the Lagrangian lower bound. We get a better lower bound, however, by solving the modified Lagrangian problem, which is to find a schedule that minimizes

\[
\sum_{j=1}^{n} |C_j - d| + \lambda^*(W - d) + |W - d|.
\]

Clearly, the modified Lagrangian problem yields a lower bound for the original problem for any \( \lambda^* \geq 1 \).

**Theorem 5.** The modified Lagrangian problem is solved by a schedule from among the optimal schedules for the Lagrangian dual problem that has minimal \( |W - d| \).

**Proof.** Suppose that \( \pi \) is a schedule that has minimal Lagrangian cost from among the optimal schedules for the modified Lagrangian problem; suppose further that \( \pi \) is not optimal for the Lagrangian dual problem. Then either the jobs are not assigned to the optimal set of positions, or there are at least two jobs \( J_i \) and \( J_j \) with \( p_i > p_j \) that are not optimally assigned. As to the first case, assigning \( J_i \) to a
position with smaller weight decreases the Lagrangian cost by at least $p_i$, while $|W - d|$ is increased by at most $p_i$. As to the second case, the interchange of $J_i$ and $J_j$ decreases the Lagrangian cost by at least $p_i - p_j$, while $|W - d|$ is increased by at most $p_i - p_j$. Therefore, in both cases $\pi$ is easily transformed into a schedule $\pi$ that is also optimal for the modified Lagrangian problem but that has smaller Lagrangian cost than $\pi$. This contradicts the assumption that $\pi$ has minimal Lagrangian cost. Hence, $\pi$ must be also optimal for the Lagrangian dual problem.

The problem of minimizing $|W - d|$ is transformed into a considerably smaller instance of subset-sum in the following way. Renumber the jobs such that $J_{k+1} \lambda^*$ becomes $J_k$ for $k = 1, \ldots, n-\lambda^*+1$; $n$ becomes equal to $n-\lambda^*+1$; the jobs previously denoted by $J_1, \ldots, J_{\lambda^*-1}$ are now simply referred to as the 'remaining' jobs. Hence, the jobs $\{J_{2k-1}, J_{2k}\}$ form a pair in the Lagrangian dual for $k = 1, \ldots, I$, with $I = 1, \ldots, \lfloor n/2 \rfloor$. Define $a_j$ as the difference in processing time between the jobs of the $j$th pair ($j = 1, \ldots, I$), and define $D = d - W_{\min}$. Remove the values $a_j$ that are zero; suppose that $m$ of them remain. Define $A$ as the multiset containing the $m$ remaining $a_j$ values; let $a_{(j)}$ denote the $j$th largest element in $A$.

If $n$ is even, then the problem of minimizing $|W - d|$ is equivalent to determining a subset $A \subseteq A$, whose sum is as close to $D$ as possible. If $n$ is odd, then an optimal schedule for the Lagrangian dual problem is optimal for the original problem in case $W \in [d - p_n, d]$. Finding such a schedule is equivalent to determining a subset $A \subseteq A$ whose sum falls in the interval $[D - p_n, D]$. If no such subset exists, then the goal is to find a subset $A$ whose sum is as close as possible to either $D - p_n$ or $D$. This problem, known as the optimization version of subset-sum, is $\mathcal{NP}$-hard in the ordinary sense (Garey and Johnson, 1979).

The instance of subset-sum can then be solved to optimality by dynamic programming requiring $O(mD)$ time and space. Note that $D \leq \Sigma a_j \leq p_{\max}$; hence, the subset-sum problem is of a smaller dimension than the underlying common due date problem.

4. A new upper bound for the restricted variant
Consider an optimal schedule for the Lagrangian dual problem. If $W \leq d$, then it is also a feasible schedule for the common due date problem; if $W > d$, then we defer the schedule to make it feasible. The analysis in the previous section suggests that we should look for an optimal schedule for the Lagrangian dual problem with $|W - d|$ minimal. Recall that $W = d$ is a sufficient condition for also having an optimal schedule for the common due date problem.

We develop an approximation algorithm for the common due date problem based upon Johnson’s approximation algorithm (Johnson, 1974) for subset-sum, which runs in $O(m)$ time.

Johnson’s algorithm
Step 1. $A = \emptyset$; $j \leftarrow 1$.
Step 2. If $a_{(j)} \leq D$, then $A \leftarrow A \cup \{a_{(j)}\}$ and $D \leftarrow D - a_{(j)}$.
Step 3. $j \leftarrow j + 1$; if $j \leq m$, then go to Step 1.

Using an approximation algorithm for subset-sum rather than an optimization algorithm does not affect the worst-case behavior; see Section 5. As to the empirical behavior, our computational results suggest that the loss in accuracy, if any, is small.
Furthermore, we can identify a class of instances for which Johnson’s algorithm always finds a solution value equal to the target sum \( D \). This class comprises the instances possessing the so-called divisibility property; this class is important in our application, as many instances can be expected to belong to it.

**Definition.** A multiset of values \( \{a_1, \ldots, a_m\} \), with \( 1 = a_1 \leq a_2 \leq \cdots \leq a_m \) is said to possess the divisibility property if for every \( j (j = 1, \ldots, m) \) and for every value \( D \in \{1, 2, \ldots, \Sigma_{i=1}^m a_i\} \) there exists a subset \( A \subseteq \{a_1, \ldots, a_j\} \), whose sum is equal to \( D \).

**Theorem 6.** A multiset of values \( \{a_1, \ldots, a_m\} \), with \( 1 = a_1 \leq a_2 \leq \cdots \leq a_m \) possesses the divisibility property if and only if \( a_{j+1} \leq \Sigma_{i=1}^m a_i + 1 \), for \( j = 1, \ldots, n-1 \).

**Theorem 7.** If an instance of subset-sum satisfies the divisibility property, then Johnson’s algorithm finds a subset with sum equal to \( D \).

In our application, each \( a_j \) is equal to the difference in processing times between two successive jobs in the shortest processing time order. If the number of jobs with different processing times is not too small, then the values \( a_j \) tend to be small. Hence, for a randomly generated instance the likelihood of possessing the divisibility property increases with the number of jobs.

Johnson’s algorithm always yields a subset with sum no more than \( D \). This handicap is overcome by applying the algorithm also to the target sum \( D = \Sigma_{j=1}^m a_j - D \) and taking the complement of the resulting subset with respect to \( \mathcal{A} \). We use the subscripts 1 and 2 to distinguish the approximation from below and from above: \( A_1 \) and \( D_1 \) denote the resulting subset and the gap for the approximation from below, and \( A_2 \) and \( D_2 \) denote the resulting subset and the gap for the approximation from above.

If both \( D_1 > 0 \) and \( D_2 > 0 \), then we apply the next algorithm to derive feasible schedules for the common due date problem from the subsets \( A_1 \) and \( A_2 \).

**Algorithm Transform**

**Step 1.** Consider \( A_1 \). Starting with \( \sigma_{\text{min}} \), interchange the jobs that correspond to \( a_j \in A_1 \) for \( j = 1, \ldots, m \), thereby increasing \( W \) by \( a_j \) per interchange. Determine the schedule corresponding to \( A_2 \) in a similar fashion, starting from \( \sigma_{\text{max}} \). Let the resulting schedules be \( \sigma_1 \) and \( \sigma_2 \).

**Step 2.** The schedule \( \sigma_1 \) is started at time \( D_1 \). Shift the schedule to the left until the first job is started at time 0 or the number of jobs completed before or at \( d \) exceeds the number of jobs completed after \( d \) by two. Rearrange the jobs to make the schedule V-shaped again. The resulting schedule is denoted by \( \sigma_1 \).

**Step 3.** The schedule \( \sigma_2 \) is started at time \( -D_2 \). Defer the schedule such that the first job is started at time zero, and rearrange the jobs to make the schedule V-shaped again; this schedule is denoted by \( \sigma_2 \). If some \( J_k \) is scheduled around \( d \), then defer \( \sigma_2 \) until \( J_k \) is started exactly at \( d \). Rearrange the jobs to make the schedule V-shaped; let the resulting schedule be \( \sigma_2 \).

We now present our approximation algorithm for the common due date problem; in the remainder, we
refer to it as the Even-Odd Heuristic.

**Even-Odd Heuristic**

Step 0. Given an instance of the common due date problem, solve the Lagrangian dual problem, and apply Johnson's algorithm to the corresponding instance of subset-sum.

Step 1. If \( D_1 \leq D_2 \), then apply Algorithm Transform; go to Step 5.

Step 2. Let \( Q = \{ a_j | a_j \geq D_1 \} \). If \( Q \neq \{ a_1 \} \), then apply Algorithm Transform, and go to Step 5.

Step 3. If \( p_1 > d \), then apply Algorithm Transform to determine \( D_2 \). Furthermore, solve the Lagrangian dual problem under the condition that \( J_1 \) and all the 'remaining' jobs occupy the last positions; go to Step 5.

Step 4. Solve the Lagrangian dual problem under the condition that \( J_1 \) and the 'remaining' jobs are assigned to positions after \( d \), and solve the Lagrangian dual problem with \( J_1 \) assigned to a position before \( d \). Apply Johnson’s algorithm and Algorithm Transform to all these solutions.

Step 5. Choose a schedule with minimal cost.

**5. Worst-case behavior**

For any instance \( I \) of the common due date problem, let \( EOH(I) \) denote the solution value determined by the Even-Odd Heuristic, and let \( OPT(I) \) denote the optimal solution value. We define \( \rho \) as

\[
\rho = \sup_I EOH(I)/OPT(I).
\]

In this section, we prove that \( \rho \leq 4/3 \), that is, that the Even-Odd Heuristic has performance guarantee 4/3.

Suppose first that Johnson’s algorithm does not solve the corresponding instance of subset-sum to optimality, that is, \( D_1 \) or \( D_2 \) is not minimal. This means that we do not know the minimal value of \( W-d \), and therefore cannot use the strengthened lower bound in our analysis.

**Lemma 1.** If Johnson's algorithm does not solve the resulting instance of subset-sum to optimality, then \( \rho \leq 8/7 \).

**Proof.** A straightforward analysis shows that, if Johnson's algorithm leaves a gap \( G \) that is not minimal, then at least three \( a_j \) values greater than \( G \) are involved; this means that at least six jobs with processing times at least equal to \( 3G, 2G, G, G, G, \) and 0, respectively. Furthermore, due to the structure of the solution of the Lagrangian problem, the \( \lambda* - 1 \) 'remaining' jobs must have processing times at least \( 3G \).

First, assume \( D_1 \leq D_2 \). Then we have for any instance \( I \) that

\[
EOH(I) \leq f(\sigma_1) = L(\lambda*) + \lambda* D_1 \leq OPT(I) + \lambda* D_1.
\]

Inspecting \( \sigma_{max}^* \), we see that \( L(\lambda*) \geq D_1 (5 + 3\lambda* (\lambda* + 1)/2) \). Hence, 

\[
\rho \leq 1 + (2\lambda* / (10 + 3\lambda* (\lambda* + 1))) \leq 8/7 \text{ for any } \lambda* \geq 1.
\]

Second, assume \( D_1 > D_2 \). If \( D_1 \) is not minimal, then we use the above analysis and find \( \rho \leq 8/7 \). If \( D_1 \) is minimal, then \( D_2 \) is not. Consider an element \( a_j A_2 \) and suppose that \( a_j < D_1 + D_2 \). This implies that

\[
\overline{D} < \sum_{k \in A_2} a_k + a_j < \overline{D} + D_1;
\]
as a consequence, the sizes of the elements in \( A - A_2 - \{ a_j \} \) add up to a value between \( D - D_1 \) and \( D \), contradicting the minimality of \( D_1 \). Hence, \( a_j \geq D_1 + D_2 \), and the above analysis can be applied to establish \( \rho \leq 8/7 \). \( \square \)

So, if Johnson's algorithm does not give minimal values \( D_1 \) and \( D_2 \), then we surely have \( \rho \leq 4/3 \). From now on, we assume that \( D_1 \) and \( D_2 \) are minimal; hence, we can now use the strengthened lower bound.

**Lemma 2. If** \( D_1 \leq D_2 \), **then** \( \rho \leq 4/3 \).

**Proof.** Again, we have that \( EOH (I) \leq L (\lambda *) + \lambda * D_1 \). Furthermore, from Theorem 5 it follows that \( OPT (I) \geq L (\lambda *) + D_1 \). Every element \( a_j A_1 \) must have size \( a_j \geq D_1 + D_2 \geq 2D_1 \). Inspecting \( \sigma_{\lambda * \max} \), we see that \( L (\lambda *) \geq \lambda * (\lambda * - 1) D_1 \); this gives \( \rho \leq 1 + ((\lambda * - 1)/(1+\lambda * (\lambda * - 1))) \leq 4/3 \) for any \( \lambda \geq 1 \). \( \square \)

Now suppose that \( D_1 > D_2 \). It is easy to show \( \rho = 4/3 \) if there exists an element \( a_k \geq D_1 \) with \( k \geq 3 \). If no such element exists, then we consider the costs of all schedules determined by Algorithm Transform. To that end, we need an upper bound on \( \Delta = f (\sigma_2) - f (\sigma_2) \).

**Proposition 3. Suppose that the first job in \( \sigma_2 \) has processing time no more than \( d \). Then \( \Delta \) is no more than the sum of the positional costs in \( \sigma_2 \) of the last \( k \) jobs before \( d \) and the first \( k + 1 \) jobs after \( d \), where \( k \) is the number of jobs that have been transferred from a position before \( d \) to a position after \( d \).**

**Proof.** Without loss of generality, we assume that \( n \) is even; if not, then we add a dummy job with zero processing time. For matter of convenience, renumber the jobs temporarily such that \( J_1, \ldots, J_k \) are the jobs that are transferred from positions before \( d \) to positions after \( d \) (\( J_k \) is completed at time \( d \)), and \( J_{2k}, \ldots, J_{k+1} \) are the first \( k + 1 \) jobs after \( d \) (\( J_{2k} \) is started at time \( d \)). Note that the jobs \( J_i \) and \( J_{k+i} \) \( (i = 1, \ldots, k) \) form a pair in the Lagrangian dual; hence, we must have that \( \min \{ p_{i+1}, p_{k+i+1} \} \geq \max \{ p_i, p_{k+i} \} \), for \( i = 1, \ldots, k-1 \).

Suppose that \( J_0 \) occupies position \( \lambda * + \mu \) in \( \sigma_2 \), with \( \mu \geq 0 \). Twice the positional cost of the jobs \( J_0, \ldots, J_{2k} \) in \( \sigma_2 \) is then equal to

\[
2((\mu+1)p_1 + \cdots + (\mu+k)p_k + (\lambda * + \mu)p_0 + \cdots + (\lambda * + \mu + k)p_{2k}) \geq \\
(\lambda * + \mu)p_0 + ((\lambda * + \mu)p_0 + (\lambda * + \mu + 1)p_{k+1} + 2p_1) + \cdots \\
+ (\lambda * + \mu + k - 1)p_{2k-1} + (\lambda * + \mu + k)p_{2k} + 2kp_k) \geq \\
(\lambda * + \mu)p_0 + (\lambda * + \mu + 1)(p_1 + p_{k+1}) + \min \{ p_1, p_{k+1} \} + (\lambda * + \mu + 3)(p_2 + p_{k+2}) \\
+ \min \{ p_2, p_{k+2} \} + \cdots + (\lambda * + \mu + 2k - 1)(p_k + p_{2k}) + \min \{ p_k, p_{2k} \}.
\]

The last expression is exactly equal to the positional cost due to the jobs \( J_0, \ldots, J_{2k} \) in \( \sigma_2 \). \( \square \)

**Lemma 3. Suppose that** \( a_1 \) and \( a_2 \) **are the only elements larger than** \( D_1 \). **Then** \( \rho \leq 4/3 \).

**Proof.** First, suppose that \( p_1 + p_3 \leq d \). Partition the jobs into two subsets: the first one is \( \{ J_3, \ldots, J_n \} \), the second one consists of \( J_1, J_2 \), and the 'remaining' jobs. As \( p_1 + p_3 \leq d \), it follows immediately
from Proposition 3 that for \( a_2 \) the sum of the positional costs of the jobs in \( \{ J_3, \ldots, J_n \} \) is at least equal to \( \Delta \). The sum of the positional costs of the jobs in the other subset is at least \((1+\lambda^*^2)D_1 \geq 2\lambda^*D_1 \). Hence, \( \text{OPT}(I) \geq 2\lambda^*D_1 + \Delta \), implying that \( p \leq 4/3 \).

Second, suppose that \( p_1 + p_3 > d \). As \( a_1 \) and \( a_2 \) are the only two elements greater than \( D_1 \), it follows immediately that \( D_1 = d - p_1 - p_4 - p_5 - \cdots \) if \( a_1 \geq a_2 \), and that \( D_2 = p_1 + p_3 + p_6 + \cdots - d \). An easy interchange argument, validated by the inequality \( D_1 > D_2 \), proves that \( J_1, J_3, J_n, J_{n-1}, \ldots \) is an optimal schedule for the case that \( J_1 \) and \( J_3 \) are started before time \( d \). Hence, we are done unless \( J_1 \) or \( J_3 \) is started at or after time \( d \) in any optimal schedule. In this case, however, we impose the additional constraint to the common due date problem that \( J_1 \) or \( J_3 \) is started at or after time \( d \). Consider the modified Lagrangian problem with such an additional constraint. Along the lines of the proof of Theorem 5, we can show that this problem is solved by an optimal schedule for the Lagrangian dual problem with \( J_1 \) or \( J_3 \) scheduled after \( d \) for which \( |W - d| \) is minimal; this is exactly the schedule \( \sigma_1 \). We have therefore that \( \text{OPT}(I) \geq L(\lambda^*) + D_1 \geq (\lambda^*^2 + 2)D_1 \). As \( \text{EOH}(I) \leq L(\lambda^*) + \lambda^*D_1 \), we obtain \( p \leq 1 + ((\lambda^* - 1)/(\lambda^*^2 + 2)) < 4/3 \).  

The analysis of the case that \( a_2 \) is the only element greater than \( D_1 \) proceeds along the same lines.

**Lemma 4.** Suppose \( D_1 > D_2 \), \( a_1 \) is the only element greater than \( D_1 \), and \( p_1 > d \). Then \( \text{EOH}(I) = \text{OPT}(I) \).

**Proof.** An easy interchange argument, validated by the inequality \( D_1 > D_2 \), proves that in any optimal schedule \( J_1 \) is either started at time 0 or scheduled immediately before the ‘remaining’ jobs. The inequality \( D_1 > D_2 \) also implies that Emmons’s matching algorithm determines a feasible and hence optimal schedule for the case that \( J_1 \) and the ‘remaining’ jobs are scheduled at or after \( d \) for which \( |W - d| \) is minimal; this is exactly the schedule \( \sigma_1 \). We have therefore that \( \text{OPT}(I) \geq L(\lambda^*) + D_1 \geq (\lambda^*^2 + 2)D_1 \). As \( \text{EOH}(I) \leq L(\lambda^*) + \lambda^*D_1 \), we obtain \( p \leq 1 + ((\lambda^* - 1)/(\lambda^*^2 + 2)) < 4/3 \).  

If \( p_1 \leq d \), then we solve both the Lagrangian dual problem with the additional constraint that \( J_1 \) and all ‘remaining’ jobs are scheduled after \( d \) and the Lagrangian dual problem with the additional constraint that \( J_1 \) is scheduled before \( d \).

**Lemma 5.** Suppose that \( D_1 > D_2 \), that \( a_1 \) is the only element greater than \( D_1 \), and that \( p_1 \leq d \). Then we have \( p \leq 4/3 \).

**Proof.** First, suppose that there is an optimal schedule in which \( J_1 \) and the ‘remaining’ jobs are started at or after \( d \). Suppose that solving the Lagrangian dual problem under the condition that \( J_1 \) and all ‘remaining’ jobs are assigned to positions after \( d \) gives \( \bar{\lambda}^*, \bar{D}_1 \) and \( \bar{D}_2 \). If \( \bar{\lambda}^* = 0 \), then we have found an optimal schedule. If \( \bar{\lambda}^* \geq 1 \), then the schedule that corresponds to \( \bar{D}_2 \) must begin with \( J_2, J_3, J_4; \) if not, then \( W \) does not sum up to \( d + D_2 \). Hence, we have \( a_1 \geq p_4 \). This gives \( \text{EOH}(I) \leq L(\bar{\lambda}^*) + \bar{\lambda}^*\bar{D}_1 \), \( \text{OPT}(I) \geq L(\bar{\lambda}^*) \), and \( L(\bar{\lambda}^*) \geq (3 + ((\bar{\lambda}^* + 1)(\bar{\lambda}^* + 2)/2))D_1 \), from which \( p \leq 4/3 \) follows.

Second, suppose there is an optimal schedule in which \( J_1 \) or some ‘remaining’ job is not started after \( d \). The optimal solution for the Lagrangian dual problem with the additional constraint that \( J_1 \) or some ‘remaining’ job is not started after \( d \) is such that \( J_1 \) is started before \( d \) and all the ‘remaining’ jobs after \( d \); this is easily proven by an interchange argument. Suppose that solving this Lagrangian problem gives \( \bar{\lambda}^*, \bar{D}_1 \), and \( \bar{D}_2 \). Consider the schedule \( \sigma \) that corresponds to \( \bar{D}_2 \). Since \( \bar{\lambda}^* \geq \lambda^* + 1 \), the
first job after \( J_1 \) must be some \( J_k \) with \( k \geq 4 \); hence, we have \( D_1 \leq p_4 \). The case \( D_1 \leq D_2 \) is easy to handle; assume therefore that \( D_1 > D_2 \). Along the lines of Lemma 3, it can then be proven that \( p \leq 4/3 \). \( \square \)

**Theorem 8.** The Even-Odd Heuristic has performance guarantee 4/3, and this bound can be approximated arbitrarily closely.

**Proof.** The first part follows immediately from Lemmas 1 to 5. The following example, based upon the case that only \( a_2 > D_1 \), shows that we can get arbitrarily close to this bound. Let \( D \) be an arbitrary positive integer. There are \( n = 2D + 6 \) jobs \( \{ J_1, \ldots, J_n \} \) with processing times

\[
\begin{align*}
p_1 &= p_2 = p_3 = D^2 + 2D, \\
p_4 &= p_5 = p_6 = D, \\
p_{6+i} &= 1 \text{ for } i = 1, \ldots, 2D,
\end{align*}
\]

and with common due date

\[ d = 2D^2 + 5D. \]

The Even-Odd Heuristic gives the schedules \( J_1, J_4, J_5, J_7, \ldots, J_n, J_6, J_3, J_2 \) with \( J_1 \) started at time \( D^2 \), and \( J_1, J_3, J_5, J_7, \ldots, J_n, J_6, J_4, J_2 \) with \( J_1 \) started at time zero. Both schedules have cost \( 4D^2 + 18D \). The optimal schedule \( J_1, J_3, J_7, \ldots, J_n, J_6, J_5, J_4, J_2 \), has cost \( 3D^2 + 19D \), however. Hence, we get arbitrarily close to 4/3 by choosing \( D \) sufficiently large. \( \square \)

6. **Branch-and-bound**

First, we solve the Lagrangian dual problem. If \( \lambda^* = 0 \), then \( \sigma^* = \sigma_0^{\text{min}} \) is an optimal solution for the common due date problem, and we are done. Otherwise, we determine upper bounds as described in Section 4; we also apply the heuristic presented by Sundararaghavan and Ahmed. If the lower and the best upper bound do not concur, then we solve the subset-sum problem to optimality by dynamic programming. If the bounds still do not concur, then we apply branch-and-bound.

For the design of the search tree we make use of the V-shapedness of optimal schedules. Assume that the jobs have been reindexed in order of nonincreasing processing times. A node at level \( j \) \( (j = 1, \ldots, n) \) of the search tree corresponds to a partial schedule in which the completion times of the jobs \( J_1, \ldots, J_j \) are fixed. Each node at level \( j \) has at most \( n - j \) descendants. In the \( k \)th descendant \( (k = 1, \ldots, n - j) \), \( J_k \) is started before \( d \) and the jobs \( J_{j+1}, \ldots, J_{j+k-1} \) are to be completed after \( d \). Given the partial schedule for \( J_1, \ldots, J_j \), a partial schedule for \( J_1, \ldots, J_{j+k} \) can easily be computed.

The algorithm that we propose is of the 'depth-first' type. We employ an active node search: at each level we choose one node to branch from. We consistently choose the node, whose job has the smallest remaining index. A simple but powerful rule to restrict the growth of the search tree is the following. A node at level \( j \) \( (j = 1, \ldots, n) \) corresponding to some \( J_k \) can be discarded if another node at the same level corresponding to some \( J_l \) with \( p_k = p_l \) has already been considered. This rule obviously avoids duplication of schedules.

In the nodes of the tree, we only compute the lower bound \( L(\lambda^*) \); we neither solve the modified Lagrangian dual problem nor compute additional upper bounds.
7. Computational results
We considered two types of instances, depending on the distributions used to generate the processing
times. Computational experiments were performed with \( d = \left\lceil \sum p_i \right\rceil \) for \( t = 0.1, 0.2, 0.3, 0.4 \), respectively, and with the number of jobs ranging from 10 to 1000. For each combination of \( n \) and \( t \) we generated 100 instances. The algorithm was coded in the computer language C; the experiments were conducted on a Compaq-386 personal computer.

Table 1 shows some of the results for instances with the processing times drawn from the discrete uniform distribution \([1, 100]\). The design of Table 1 reflects our three-phase approach. The third column '# \( O(n \log n) \)' shows the number of times (out of 100) that the Even-Odd Heuristic finds a schedule with cost equal to the Lagrangian lower bound \( L(\lambda^*) \); this is the number of times that the common due date problem was provably solved to optimality in \( O(n \log n) \) time. The fourth column '# DP' shows how many of the remaining instances were provably solved to optimality by dynamic programming applied to subset-sum. The fifth column 'maximum # nodes' shows the maximum number of nodes needed by the branch-and-bound algorithm. The sixth column '# Even-Odd optimal' shows the number of times that the Even-Odd Heuristic found an optimal schedule. The seventh column '# SA optimal' gives the same information for the approximation algorithm presented by Sundaaraghavan and Ahmed. The last column '# LB tight' shows the number of times that the lower bound (strengthened or not) was equal to the optimal solution value. All these instances, including those with 1000 jobs, were solved in less than 1 second, if the jobs were already sorted in order of non-decreasing processing times.

Instances with a large number of different processing times can be expected to possess the divisibility property. In this sense, the success of the algorithm for instances generated from a uniform distribution may be mainly due to the divisibility property. We therefore applied our algorithm also to instances for which the divisibility property was expected to play a less prominent role. We generated these instances in the following way. We partitioned the jobs into subsets, whereafter all jobs in the same subset got the same processing time, drawn from the uniform distribution \([1, 100]\). The results exhibited essentially the same pattern as for the first type of instances, albeit that the number of jobs to reach a 100% score went up a little.

8. Extensions
The lower bound approach can be extended to the restricted variant of each problem that is solvable by Emmons's matching algorithm. The most important problem in this context is the \( 1 \mid d_j = d \mid \Sigma (\alpha E_j + \beta T_j) \) problem.

Without loss of generality, we assume that \( \alpha \) and \( \beta \) are integral and relatively prime. A similar analysis shows that the optimal value \( \lambda^* \) is the value \( \lambda^* \in \{1, \ldots, n\beta \} \) for which \( W_{\lambda^*}^{\max} \geq d > W_{\lambda^*}^{\min} \). Furthermore, Theorem 5 still holds.

It is straightforward to develop a heuristic for the common due date problem with \( \alpha \neq \beta \) by applying Johnson's algorithm and Algorithm Transform; its worst-case performance, however, is still an open question.

Acknowledgement
The authors are grateful to Jan Karel Lenstra for his helpful comments.
<table>
<thead>
<tr>
<th>n</th>
<th>t</th>
<th>#O(n \log n)</th>
<th>#DP</th>
<th>maximum # nodes</th>
<th>#Even-Odd optimal</th>
<th>#SA optimal</th>
<th>#LB tight</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0.1</td>
<td>66</td>
<td>20</td>
<td>12</td>
<td>72</td>
<td>77</td>
<td>86</td>
</tr>
<tr>
<td>10</td>
<td>0.2</td>
<td>69</td>
<td>20</td>
<td>22</td>
<td>72</td>
<td>58</td>
<td>89</td>
</tr>
<tr>
<td>10</td>
<td>0.3</td>
<td>68</td>
<td>23</td>
<td>22</td>
<td>68</td>
<td>59</td>
<td>93</td>
</tr>
<tr>
<td>10</td>
<td>0.4</td>
<td>82</td>
<td>1</td>
<td>40</td>
<td>85</td>
<td>62</td>
<td>85</td>
</tr>
<tr>
<td>20</td>
<td>0.1</td>
<td>81</td>
<td>12</td>
<td>94</td>
<td>84</td>
<td>51</td>
<td>94</td>
</tr>
<tr>
<td>20</td>
<td>0.2</td>
<td>94</td>
<td>5</td>
<td>167</td>
<td>94</td>
<td>43</td>
<td>99</td>
</tr>
<tr>
<td>20</td>
<td>0.3</td>
<td>99</td>
<td>0</td>
<td>320</td>
<td>100</td>
<td>42</td>
<td>99</td>
</tr>
<tr>
<td>20</td>
<td>0.4</td>
<td>99</td>
<td>1</td>
<td>0</td>
<td>99</td>
<td>35</td>
<td>100</td>
</tr>
<tr>
<td>30</td>
<td>0.1</td>
<td>100</td>
<td>0</td>
<td>0</td>
<td>100</td>
<td>50</td>
<td>100</td>
</tr>
<tr>
<td>30</td>
<td>0.2</td>
<td>98</td>
<td>2</td>
<td>0</td>
<td>98</td>
<td>51</td>
<td>100</td>
</tr>
<tr>
<td>30</td>
<td>0.3</td>
<td>100</td>
<td>0</td>
<td>0</td>
<td>100</td>
<td>57</td>
<td>100</td>
</tr>
<tr>
<td>30</td>
<td>0.4</td>
<td>100</td>
<td>0</td>
<td>0</td>
<td>100</td>
<td>68</td>
<td>100</td>
</tr>
<tr>
<td>40</td>
<td>0.1</td>
<td>100</td>
<td>0</td>
<td>0</td>
<td>100</td>
<td>63</td>
<td>100</td>
</tr>
<tr>
<td>40</td>
<td>0.2</td>
<td>100</td>
<td>0</td>
<td>0</td>
<td>100</td>
<td>64</td>
<td>100</td>
</tr>
<tr>
<td>40</td>
<td>0.3</td>
<td>100</td>
<td>0</td>
<td>0</td>
<td>100</td>
<td>63</td>
<td>100</td>
</tr>
<tr>
<td>40</td>
<td>0.4</td>
<td>100</td>
<td>0</td>
<td>0</td>
<td>100</td>
<td>54</td>
<td>100</td>
</tr>
<tr>
<td>50</td>
<td>0.1</td>
<td>100</td>
<td>0</td>
<td>0</td>
<td>100</td>
<td>72</td>
<td>100</td>
</tr>
<tr>
<td>50</td>
<td>0.2</td>
<td>100</td>
<td>0</td>
<td>0</td>
<td>100</td>
<td>63</td>
<td>100</td>
</tr>
<tr>
<td>50</td>
<td>0.3</td>
<td>100</td>
<td>0</td>
<td>0</td>
<td>100</td>
<td>69</td>
<td>100</td>
</tr>
<tr>
<td>50</td>
<td>0.4</td>
<td>100</td>
<td>0</td>
<td>0</td>
<td>100</td>
<td>75</td>
<td>100</td>
</tr>
<tr>
<td>100</td>
<td>0.1</td>
<td>100</td>
<td>0</td>
<td>0</td>
<td>100</td>
<td>81</td>
<td>100</td>
</tr>
<tr>
<td>100</td>
<td>0.2</td>
<td>100</td>
<td>0</td>
<td>0</td>
<td>100</td>
<td>86</td>
<td>100</td>
</tr>
<tr>
<td>100</td>
<td>0.3</td>
<td>100</td>
<td>0</td>
<td>0</td>
<td>100</td>
<td>78</td>
<td>100</td>
</tr>
<tr>
<td>100</td>
<td>0.4</td>
<td>100</td>
<td>0</td>
<td>0</td>
<td>100</td>
<td>78</td>
<td>100</td>
</tr>
</tbody>
</table>

Table 1. Computational results.

References


<table>
<thead>
<tr>
<th>Number</th>
<th>Month</th>
<th>Author</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>92-01</td>
<td>January</td>
<td>F.W. Steutel</td>
<td>On the addition of log-convex functions and sequences</td>
</tr>
<tr>
<td>92-02</td>
<td>January</td>
<td>P. v.d. Laan</td>
<td>Selection constants for Uniform populations</td>
</tr>
<tr>
<td>92-03</td>
<td>February</td>
<td>E.E.M. v. Berkum</td>
<td>Data reduction in statistical inference</td>
</tr>
<tr>
<td></td>
<td></td>
<td>H.N. Linssen</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>D.A. Overdijk</td>
<td></td>
</tr>
<tr>
<td>92-04</td>
<td>February</td>
<td>H.J.C. Huijberts</td>
<td>Strong dynamic input-output decoupling:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>H. Nijmeijer</td>
<td>from linearity to nonlinearity</td>
</tr>
<tr>
<td>92-05</td>
<td>March</td>
<td>S.J.L. v. Eijndhoven</td>
<td>Introduction to a behavioral approach of continuous-time systems</td>
</tr>
<tr>
<td></td>
<td></td>
<td>J.M. Šoethoudt</td>
<td></td>
</tr>
<tr>
<td>92-06</td>
<td>April</td>
<td>P.J. Zwietering</td>
<td>The minimal number of layers of a perceptron that sorts</td>
</tr>
<tr>
<td></td>
<td></td>
<td>E.H.L. Aarts</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>J. Wessels</td>
<td></td>
</tr>
<tr>
<td>92-07</td>
<td>April</td>
<td>F.P.A. Coolen</td>
<td>Maximum Imprecision Related to Intervals of Measures and Bayesian Inference with Conjugate Imprecise Prior Densities</td>
</tr>
<tr>
<td>92-08</td>
<td>May</td>
<td>I.J.B.F. Adan</td>
<td>A Note on “The effect of varying routing probability in</td>
</tr>
<tr>
<td></td>
<td></td>
<td>J. Wessels</td>
<td>two parallel queues with dynamic routing under a</td>
</tr>
<tr>
<td></td>
<td></td>
<td>W.H.M. Zijm</td>
<td>threshold-type scheduling”</td>
</tr>
<tr>
<td>92-09</td>
<td>May</td>
<td>I.J.B.F. Adan</td>
<td>Upper and lower bounds for the waiting time in the</td>
</tr>
<tr>
<td></td>
<td></td>
<td>G.J.J.A.N. v. Houtum</td>
<td>symmetric shortest queue system</td>
</tr>
<tr>
<td></td>
<td></td>
<td>J. v.d. Wal</td>
<td></td>
</tr>
<tr>
<td>92-10</td>
<td>May</td>
<td>P. v.d. Laan</td>
<td>Subset Selection: Robustness and Imprecise Selection</td>
</tr>
<tr>
<td>92-11</td>
<td>May</td>
<td>R.J.M. Vaessens</td>
<td>A Local Search Template</td>
</tr>
<tr>
<td></td>
<td></td>
<td>E.H.L. Aarts</td>
<td>(Extended Abstract)</td>
</tr>
<tr>
<td>92-12</td>
<td>May</td>
<td>F.P.A. Coolen</td>
<td>Elicitation of Expert Knowledge and Assessment of Imprecise Prior Densities for Lifetime Distributions</td>
</tr>
<tr>
<td>92-13</td>
<td>May</td>
<td>M.A. Peters</td>
<td>Mixed $H_2/H_\infty$ Control in a Stochastic Framework</td>
</tr>
<tr>
<td></td>
<td></td>
<td>A.A. Stoorvogel</td>
<td></td>
</tr>
<tr>
<td>Number</td>
<td>Month</td>
<td>Author</td>
<td>Title</td>
</tr>
<tr>
<td>-------</td>
<td>-------</td>
<td>--------</td>
<td>-------</td>
</tr>
<tr>
<td>92-14</td>
<td>June</td>
<td>P.J. Zwietering, E.H.L. Aarts, J. Wessels</td>
<td>The construction of minimal multi-layered perceptrons: a case study for sorting</td>
</tr>
<tr>
<td>92-15</td>
<td>June</td>
<td>P. van der Laan</td>
<td>Experiments: Design, Parametric and Nonparametric Analysis, and Selection</td>
</tr>
<tr>
<td>92-16</td>
<td>June</td>
<td>J.J.A.M. Brands, F.W. Steutel, R.J.G. Wilms</td>
<td>On the number of maxima in a discrete sample</td>
</tr>
<tr>
<td>92-17</td>
<td>June</td>
<td>S.J.L. v. Eijndhoven, J.M. Soethoudt</td>
<td>Introduction to a behavioral approach of continuous-time systems part II</td>
</tr>
<tr>
<td>92-18</td>
<td>June</td>
<td>J.A. Hoogeveen, H. Oosterhout, S.L. van der Velde</td>
<td>New lower and upper bounds for scheduling around a small common due date</td>
</tr>
</tbody>
</table>