Three, four, five, six or the complexity of scheduling with communication delays
Hoogeveen, J.A.; Lenstra, J.K.; Veltman, B.

Published: 01/01/1993

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.
Three, four, five, six or the Complexity of Scheduling with Communication Delays

J.A. Hoogeveen
J.K. Lenstra
B. Veltman

Eindhoven, July 1993
The Netherlands
Three, four, five, six, or the Complexity of Scheduling with Communication Delays

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

Abstract: A set of unit-time tasks has to be processed on identical parallel processors subject to precedence constraints and unit-time communication delays; does there exist a schedule of length at most \( d \)? The problem has two variants, depending on whether the number of processors is restrictively small or not. For the first variant the question can be answered in polynomial time for \( d = 3 \) and is NP-complete for \( d = 4 \). The second variant is solvable in polynomial time for \( d = 5 \) and NP-complete for \( d = 6 \). As a consequence, neither of the corresponding optimization problems has a polynomial approximation scheme, unless \( P = NP \).

1991 Mathematics Subject Classification: 90B35.
Key Words & Phrases: Identical parallel processors, precedence constraints, communication delays, makespan, complexity, approximation.

1. Introduction

The introduction of parallel computing has created new allocation and scheduling problems that have to be dealt with. They differ from the problems of classical sequencing and scheduling theory mainly in that interprocessor communication delays have to be taken into account: the execution of a task cannot start before all necessary information produced by its predecessors has been transmitted to the processor that has to execute it.

We consider the following type of scheduling problem with communication delays. There are \( n \) tasks that have to be executed by \( m \) identical parallel processors. Each processor is available from time zero onwards and can execute at most one task at a time. Each task \( J_j \) \((j = 1, \ldots, n)\) has a unit processing time and produces information, which may be required by one or more other tasks. These data dependencies define a precedence relation on the task set. It is represented by an acyclic directed graph \( G \) with vertices \( J_1, \ldots, J_n \) and an arc \( J_j \rightarrow J_k \) whenever \( J_k \) needs data from \( J_j \). For each arc \( J_j \rightarrow J_k \), we require that \( J_k \) cannot be started before \( J_j \) has been completed and the information produced by \( J_j \) has been transferred from its processor to the processor of \( J_k \); it is assumed that the transmission of information does not interfere with the availability of the processors. There is a communication delay of one time unit if \( J_j \) and \( J_k \) are executed on different processors; there is no delay if \( J_j \) and \( J_k \) are executed on the same processor. A schedule is an allocation of each task to a time slot of unit length on a processor such that processor availability constraints, precedence constraints and communication delays are taken into account. Given a schedule, let \( S_j \) and \( C_j \) denote the starting and completion time of task \( J_j \), respectively, for \( j = 1, \ldots, n \). We wish to minimize the length or makespan of a schedule, that is, the maximum task completion time, defined as \( C_{\text{max}} = \max_{1 \leq j \leq n} C_j \).

There are two variants of the problem, depending on whether the number of processors is restrictively small or not. With a modification of the three-field notation scheme that was proposed by
Veltman, Lageweg, and Lenstra [1990] as an extension of the terminology introduced by Graham, Lawler, Lenstra, and Rinnooy Kan [1979], we denote the first variant by $P^{\text{prec},c=1,p_j=1|C_{\text{max}}}$ and the second variant by $P^{\infty|\text{prec},c=1,p_j=1|C_{\text{max}}}$.

The $P^{\text{prec},c=1,p_j=1|C_{\text{max}}}$ problem was first addressed by Rayward-Smith [1987], who established NP-hardness and showed that an active schedule is no longer than $3 - 2/m$ times the optimum. A schedule is active if no task can start earlier without increasing the start time of another task. It is a challenging open problem to approximate an optimal solution appreciably better than a factor of 3 in polynomial time.

Papadimitriou and Yannakakis [1990] were among the first to address the unrestricted case. They study a model with general processing times and communication delays that allows for task duplication and derive a 2-approximation algorithm. Colin and Chrétienne [1993] observe that this method generates optimal schedules in case of small communication delays, that is, $\max_{1 \leq i,k \leq n} c_{jk} \leq \min_{1 \leq i \leq n} p_j$.

Lenstra, Veldhorst, and Veltman [1993] show that the restricted variant in which the precedence relation consists of a collection of trees, $P^{\text{tree},c=1,p_j=1|C_{\text{max}}}$, is NP-hard. Varvarigou, Roychowdhury, and Kailath [1992] show that if, in addition, the number of processors is part of the problem type, then dynamic programming leads to a polynomial-time algorithm. A linear-time algorithm for the case of two processors is given by Lenstra, Veldhorst and Veltman [1993]. The complexity of the problem with a general precedence relation and a fixed number of processors, $P^{m|\text{prec},c=1,p_j=1|C_{\text{max}}}$, is open, even for $m = 2$.

Picouleau [1991A] showed that the problem of deciding whether an instance of $P^{\text{tree},c=1,p_j=1|C_{\text{max}}}$ has a schedule of length at most 3 is decidable in polynomial time. Lenstra and Rinnooy Kan [1978] showed for the case without communication delays, $P^{\text{prec},c=1,p_j=1|C_{\text{max}}}$, that the problem of deciding whether an instance has a schedule of length at most 3 is NP-complete; note that it is trivial to decide if a feasible schedule of length 2 exists.

For the unrestricted case, $P^{\infty|\text{prec},c=1,p_j=1|C_{\text{max}}}$, Picouleau [1991B] established NP-completeness for the problem of deciding whether an instance has a schedule of length at most 8. The case without communication delays, $P^{\infty|\text{prec},c=1,p_j=1|C_{\text{max}}}$, is trivial.

We study the same type of question as investigated by Picouleau, and Lenstra and Rinnooy Kan: for what deadline $d$ can one determine in polynomial time if a schedule of length at most $d$ exists? In Section 2, we give our own proof that the restricted variant is polynomially solvable for $d = 3$ and show NP-completeness for $d = 4$. In Section 3, we show that the unrestricted variant is polynomially solvable for $d = 5$ and NP-complete for $d = 6$.

As a consequence, there exists no polynomial-time algorithm with performance bound smaller than $5/4$ for $P^{\text{prec},c=1,p_j=1|C_{\text{max}}}$, and there exists no polynomial-time algorithm with performance bound smaller than $7/6$ for $P^{\infty|\text{prec},c=1,p_j=1|C_{\text{max}}}$, unless $P=NP$. Neither of these problems has a polynomial approximation scheme, unless $P=NP$.

2. The restricted variant

We start by showing that the problem of deciding whether an instance of $P^{\text{prec},c=1,p_j=1|C_{\text{max}}}$ has a schedule of length at most 3 is decidable in polynomial time. Our algorithm is an alternative to a method previously proposed by Picouleau [1991A]. Next we prove NP-completeness of the problem of deciding whether an instance has a schedule of length at most 4, even for the special case that the precedence relation has the form of a bipartite graph.
Theorem 1. The problem of deciding whether an instance of \( P|\text{prec}, c=1, p_j=1|C_{\text{max}} \) has a schedule of length at most 3 is solvable in polynomial time.

Proof. Given an instance of \( P|\text{prec}, c=1, p_j=1|C_{\text{max}} \), we first check whether some trivial necessary constraints for the existence of a feasible schedule of length at most 3 are satisfied. These are the constraints that there are no paths in the graph of length more than 3, that there are no more than 3\( m \) tasks, and that no two paths of length 3 interfere or share a task. Subsequently, we delete the isolated tasks from the instance; they will be dealt with later.

Our approach to check the existence of a feasible schedule of length at most 3 consists of two steps. We first assign the tasks to time slots. Then the tasks are assigned to the processors by which they have to be executed. The first step proceeds in such a way that the number of processors needed in the second step is minimized.

We first deal with the paths of length 3. The tasks in a path of length 3 are all assigned to a single processor. As no two paths of length 3 interfere, the second task in a chain has only one predecessor and only one successor. The first task in a path of length 3 may be succeeded by several tasks without successors; these tasks are assigned to the third time slot. The third task in a path of length 3 may be preceded by several tasks without predecessors; these tasks are assigned to the first time slot. Furthermore, we assign the tasks with two or more successors to the first time slot and the tasks with two or more predecessors to the third time slot.

The tasks that still have to be assigned either belong to isolated chains of length 2 or are the leaves of a rooted intree or outtree with depth at most equal to 2. In case of a chain of length 2, the tasks can be assigned either to the time slots 1 and 2, or to the time slots 2 and 3, or to the time slots 1 and 3; if a task is assigned to the second time slot, then the other task in the chain has to be executed by the same processor. In case of a rooted intree, at most one of the tasks can be assigned to time slot 2 and all other tasks (except the root) must be assigned to time slot 1, whereas in case of a rooted outtree at most one task can be assigned to time slot 2 and all other tasks (except the root) must be assigned to time slot 3. A straightforward approach finds an assignment of the tasks belonging to chains and trees to time slots such that the maximum number of tasks assigned to a single time slot is minimized. If this number exceeds the number of available processors, then clearly the instance has no schedule of length at most 3.

Given an assignment of tasks to time slots, a feasible schedule is constructed in the following way. First assign each task that is to be processed in the second time slot to a processor and assign its predecessor or successor to the same processor. The remaining tasks can be scheduled on arbitrary processors according to the time slot assignment. Finally, the isolated tasks can be used to fill the empty slots. \( \square \)

Theorem 2. The problem of deciding whether an instance of \( P|\text{prec}, c=1, p_j=1|C_{\text{max}} \) has a schedule of length at most 4 is NP-complete, even for bipartite precedence relations.

Proof. Our proof is based on a reduction from the NP-complete problem Clique and extends the proof by Lenstra and Rinnooy Kan [1978] for the variant without communication delays. The Clique problem is defined as follows:

Clique: Given a graph \( G=(V,E) \) and an integer \( k \), does \( G \) have a complete subgraph on \( k \) vertices?
Given an instance of Clique, define the number of edges in a clique of size \( k \) by \( 1 = \frac{k(k-1)}{2} \) and define \( m = \max \{ |V|+l-k, |E|-1 \} \). We construct the following instance of \( P|\text{prec}, c=1, p_j=1|C_{\text{max}} \). There are \( m = 2(m+1) \) processors, which have to process \( 4m \) tasks. Each vertex \( v \in V \) corresponds to a pair of vertex tasks \( J_v \) and \( K_v \), and each edge \( e \in E \) corresponds to an edge task \( L_e \); we introduce precedence constraints \( J_v \rightarrow K_v \) and \( J_v \rightarrow L_e \) if \( v \) is incident to \( e \). In addition, we define \( 4m-2|V|-|E| \) dummy tasks: there are \( m-k \) tasks of type \( W \), \( m-|V| \) of type \( X \), \( m-|V|+k-1 \) of type \( Y \), and \( m-|E|+l \) of type \( Z \). The precedence constraints between these dummy tasks are such that all \( W \) tasks should precede all \( Y \) and \( Z \) tasks, and all \( X \) tasks should precede all \( Z \) tasks.

Suppose that \( G \) contains a clique of size \( k \). Then a schedule of length at most 4 is obtained by scheduling the tasks according to the pattern given in Figure 1. Here \( J, K, \) and \( L \) stand for the tasks of type \( J_v, K_v, \) and \( L_e \), respectively, \( J_{\text{clique}} \) (\( K_{\text{clique}} \)) denotes the set of tasks of type \( J \) (\( K \)) corresponding to the clique vertices, and \( L_{\text{clique}} \) denotes the set of tasks of type \( L \) corresponding to the clique edges.

![Figure 1. Schedule of length 4.](image)

Conversely, suppose that there exists a feasible schedule \( \sigma \) of length at most 4. We will show that in any such schedule the non-dummy tasks processed in time slot 1 correspond to the vertices of a clique of size \( k \). The \( W \) tasks are processed in time slot 1 in \( \sigma \), since they must precede all of the tasks of types \( Y \) and \( Z \), of which there are at least \( m+2 \). A similar argument shows that the \( Z \) tasks are processed in time slot 4 in \( \sigma \). It follows immediately from these observations that the tasks of types \( X \) and \( Y \) are processed in \( \sigma \) in the time periods \([0, 2]\) and \([2, 4]\), respectively.

As the number of tasks is exactly equal to \( 4m \), \( \sigma \) does not contain any idle time; hence, next to the tasks of type \( W \) and \( X \), exactly \( k+|V|-1 \) vertex tasks must be processed in time period \([0, 2]\). As the vertex tasks of type \( J \) have to precede the corresponding vertex tasks of type \( K \), we know that no more than \( |V|-1 \) vertex tasks are processed in time slot 2 in \( \sigma \). This observation, combined with the observation that all \( X \) tasks are processed in time period \([0, 2]\), implies that \( \sigma \) processes all \( X \) tasks in time slot 2, that \( k \) vertex tasks of type \( J \) are processed in time slot 1, and that the corresponding vertex tasks of type \( K \) and the remaining vertex tasks of type \( J \) are processed in time slot 2. The set of tasks that are processed in time slot 3 consists of \( Y \) tasks, edge tasks \( L \) that have both predecessors processed in time slot 1, vertex tasks of type \( K \), and \( L \) tasks with one predecessor in time slot 1 and one predecessor in time slot 2; the total number of these tasks is equal to \( m \), as \( \sigma \) contains no idle time. Note that both the \( K \) tasks and the \( L \) tasks with one predecessor in time slot 2 must be scheduled immediately after their preceding task of type \( J \), implying that the number of these tasks is at most \( |V|-k \). Hence, there
are at least $l$ edge tasks with both predecessors processed in time slot 1, implying that the $k$ vertices corresponding to the $k$ vertex tasks that are processed in time slot 1 induce a complete subgraph of $G$.

**Corollary 1.** For $P|\text{prec},c=1,p_j=1|C_{\text{max}}$ there exists no polynomial-time algorithm with performance bound smaller than $5/4$, unless $P=NP$. □

3. The unrestricted variant

We now consider the variant for which the number of processors is not restrictively small. We first show that the problem of deciding whether an instance has a schedule of length at most 5 is solvable in polynomial time. Next we show that the problem of deciding whether an instance has a schedule of length at most 6 is NP-complete.

**Theorem 3.** The problem of deciding whether an instance of $P\infty|\text{prec},c=1,p_j=1|C_{\text{max}}$ has a schedule of length at most 5 is solvable in polynomial time.

**Proof.** Given an arbitrary instance of the problem $P\infty|\text{prec},c=1,p_j=1|C_{\text{max}}$, we first check whether some obviously necessary constraints hold. These are that the graph contains no path of length more than 5 and that there are no two interfering paths of length 5. Suppose that these constraints are satisfied. Then it is easy to see that each task that does not belong to a path of length 4 can be assigned to a processor and time slot without violating any constraint. We now present a polynomial-time algorithm that checks whether a given set of paths of length 4 fits into a feasible schedule of length at most 5.

Let $J_1 \rightarrow J_2 \rightarrow J_3 \rightarrow J_4$ denote a path of length 4. Without loss of generality, $J_1$ and $J_4$ can be processed in the first and last time slot, respectively. We develop an algorithm to check in polynomial time whether there exists a feasible assignment of the middle tasks $J_2$ and $J_3$ to time slots, while observing the constraint that two dependent tasks that are assigned to two consecutive time slots must be performed by the same processor. We distinguish a number of cases.

Suppose that $J_2$ has at least two predecessors, implying that $J_2$ cannot start before time 2; then we have to assign $J_2, J_3,$ and $J_4$ to the last three time slots and they have to be performed by the same processor. Similarly, if $J_3$ has at least two successors, then $J_3$ has to be executed in time slot 3 and $J_1$ and $J_2$ have to be executed by the same processor in the first and second time slot, respectively.

The other cases require a more intricate procedure. From now on, $J_2$ has one predecessor and $J_3$ has one successor. For each unscheduled task $J_j$, we define the depth $d_j$ as the number of tasks, in a path of length 4, that precede this task; thus $d_2 = 1$ and $d_3 = 2$. As $J_j$ starts at time $d_j$ or $d_j + 1$ in any feasible schedule of length at most 5, we have that $S_j = d_j + x_j$, with $x_j \in \{0, 1\}$. The problem of assigning feasible start times to the tasks $J_j$ can thus be formulated as a problem of assigning feasible binary values to the variables $x_j$.

Consider the case depicted in Figure 2a. Let $V$ denote the set of immediate successors of $J_1$ that belong to a path of length 4; in this case we have $V = \{J_2, J_2a, J_2b\}$. As at most one of the tasks in $V$ can be executed in time slot 2, the $x$ variables corresponding to the tasks in $V$ must satisfy the constraint $\sum_{j \in V} x_j \geq |V| - 1$.

Three other cases we distinguish are illustrated in Figures 2b-d. Analogous observations lead to similar constraints for each case; these constraints are shown next to the graph. Note that in case 2b
we have already assigned $J_2$ to time slot 2, and that in case 2c task $J_3$ has already been assigned to time slot 4. If two dependent tasks $J_2$ and $J_3$ have not yet been assigned to time slots, then we have to add the constraint $x_3 - x_2 \geq 0$ to ensure consistency. Note that each binary solution that satisfies all constraints derived for the cases 2a through 2d induces a feasible schedule of length at most 5; let $Ax \leq b$ denote the set of constraints.

It is easily verified that every column of $A$ contains at most one +1 and at most one -1 entry, implying that $A$ is a network matrix [Schrijver, 1986]. Hence, if we add the inequalities $0 \leq x_j \leq 1$, then the constraint matrix remains totally unimodular and the polyhedron $\{ x \mid 0 \leq x \leq 1; Ax \leq b \}$ is integral. As we can decide in polynomial time whether the polyhedron is empty, the problem whether a given instance of $P_{\infty} | prec, c = 1, p_j = 1 | C_{\text{max}}$ has a schedule of length at most 5 is decidable in polynomial time. \(\square\)
Theorem 4. The problem of deciding whether an instance of \( P^{\infty}\text{prec},c=1,p_j=1\mid C_{\text{max}} \) has a schedule of length at most 6 is NP-complete.

Proof. Our proof is based on a reduction from the NP-complete problem 3-Satisfiability, which is defined as follows:

3-Satisfiability: Given a set \( U \) of variables and a collection \( C \) of clauses over \( U \) such that each clause \( c \in C \) has \( |c| = 3 \), does there exist a truth assignment for \( C \)?

Given an instance \((U,C)\) of 3-Satisfiability, we construct the following instance of \( P^{\infty}\text{prec},c=1,p_j=1\mid C_{\text{max}} \). For each variable \( x \) we introduce six tasks: \( x_1, x_2, x_3, x, \bar{x}, \) and \( x_6 \); the precedence constraints between these tasks are given in Figure 3. For each clause \( c=(x_c,y_c,z_c) \), where the literals \( x_c, y_c, \) and \( z_c \) are occurrences of negated or unnegated variables, we introduce thirteen tasks: \( \hat{x}_c, \tilde{y}_c, \check{z}_c, x_c, \bar{x}_c, y_c, \bar{y}_c, z_c, \bar{z}_c, x_c y_c, x_c z_c, y_c z_c, \) and \( c \); the precedence constraints between these tasks are also given in Figure 3. We further introduce precedence constraints between the variable tasks and the clause tasks. If the occurrence of variable \( x \) in \( c \) is unnegated, then \( x_c \) precedes the variable task \( x \) and \( x_c \) precedes the variable task \( x \), as illustrated in Figure 3. If the occurrence of variable \( x \) in \( c \) is negated, then \( x_c \) precedes the variable task \( x \) and \( x_c \) precedes the variable task \( x \). Thus, \( x_c \) represents the occurrence of variable \( x \) in clause \( c \); it precedes the corresponding variable task.

We start by making two essential observations. First, note that in a schedule of length at most 6 there are exactly two ways to schedule the tasks corresponding to a variable \( x \), depending upon

![Figure 3. Variable tasks and clause tasks.](image-url)
whether variable task \( x \) is scheduled in time slot 4 and variable task \( \bar{x} \) in time slot 5 or the other way around. In both cases, the tasks \( x_1, x_2, \) and \( x_3 \) have to be performed by the same processor as the variable task that is scheduled in time slot 4 and the task \( x_6 \) has to be performed by the same processor as the variable task scheduled in time slot 5. Second, note that in order to schedule the clause tasks corresponding to clause \( c = (x_c, y_c, z_c) \) within six time units at least one of the tasks \( x_c, y_c, \) and \( z_c \) must be scheduled in time slot 2.

Suppose that a truth assignment for \( C \) exists. Then a schedule of length at most 6 is obtained by scheduling the variable task \( x \) in time slot 4 if variable \( x \) is true and in time slot 5 otherwise. If the literal \( x \) occurring in clause \( c \) is true, then \( x_c \) is scheduled in time slot 2 on the same processor as \( \bar{x}_c \), and \( \bar{x}_c \) is scheduled in time slot 3; if the literal \( x \) in \( c \) is false, then the task \( x_c \) is scheduled in time slot 3 and \( \bar{x}_c \) in time slot 2. The other tasks are scheduled in a greedy manner. As every clause \( c \) contains at least one true literal, each clause task \( c \) will be completed by time 6.

Conversely, suppose that there exists a schedule of length at most 6. We will show that there exists a truth assignment for the instance of 3-Satisfiability. Define a variable \( x \) as true if the corresponding variable task \( x \) is processed in time slot 4, and false otherwise. Without loss of generality, suppose that variable task \( x \) is executed in time slot 4. Each unnegated occurrence of \( x \) must be scheduled in time slot 2 and each negated occurrence of \( x \) in slot 3, implying that all literals are assigned values consistently. As each clause task \( c \) has been completed at time 6, we know that each clause contains at least one true literal.

**Corollary 2.** For \( P=\Pi_1 \) and \( c=1 \), \( p_j=\Pi_1 \) \( C_{\max} \) there exists no polynomial-time algorithm with performance bound smaller than \( 7/6 \), unless \( P=NP \).

**References**


<table>
<thead>
<tr>
<th>Number</th>
<th>Month</th>
<th>Author</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>93-01</td>
<td>January</td>
<td>P. v.d. Laan, C. v. Eeden</td>
<td>Subset selection for the best of two populations: Tables of the expected subset size</td>
</tr>
<tr>
<td>93-03</td>
<td>February</td>
<td>Jan Beirlant, John H.J. Einmahl</td>
<td>Asymptotic confidence intervals for the length of the shortest under random censoring.</td>
</tr>
<tr>
<td>93-04</td>
<td>February</td>
<td>E. Balas, J. K. Lenstra, A. Vazacopoulos</td>
<td>One machine scheduling with delayed precedence constraints</td>
</tr>
<tr>
<td>93-05</td>
<td>March</td>
<td>A.A. Stoorvogel, J.H.A. Ludlage</td>
<td>The discrete time minimum entropy $H_\infty$ control problem</td>
</tr>
<tr>
<td>93-06</td>
<td>March</td>
<td>H.J.C. Huijberts, C.H. Moog</td>
<td>Controlled invariance of nonlinear systems: nonexact forms speak louder than exact forms</td>
</tr>
<tr>
<td>93-07</td>
<td>March</td>
<td>Marinus Veldhorst</td>
<td>A linear time algorithm to schedule trees with communication delays optimally on two machines</td>
</tr>
<tr>
<td>93-08</td>
<td>March</td>
<td>Stan van Hoesel, Antoon Kolen</td>
<td>A class of strong valid inequalities for the discrete lot-sizing and scheduling problem</td>
</tr>
<tr>
<td>93-09</td>
<td>March</td>
<td>F.P.A. Coolen</td>
<td>Bayesian decision theory with imprecise prior probabilities applied to replacement problems</td>
</tr>
<tr>
<td>93-10</td>
<td>March</td>
<td>A.W.J. Kolen, A.H.G. Rinnooy Kan, C.P.M. van Hoesel, A.P.M. Wagelmans</td>
<td>Sensitivity analysis of list scheduling heuristics</td>
</tr>
<tr>
<td>93-11</td>
<td>March</td>
<td>A.A. Stoorvogel, J.H.A. Ludlage</td>
<td>Squaring-down and the problems of almost-zeros for continuous-time systems</td>
</tr>
<tr>
<td>93-12</td>
<td>April</td>
<td>Paul van der Laan</td>
<td>The efficiency of subset selection of an $\varepsilon$-best uniform population relative to selection of the best one</td>
</tr>
<tr>
<td>93-13</td>
<td>April</td>
<td>R.J.G. Wilms</td>
<td>On the limiting distribution of fractional parts of extreme order statistics</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>93-14</td>
<td>May</td>
<td>L.C.G.J.M. Habets</td>
<td>On the Genericity of Stabilizability for Time-Day Systems</td>
</tr>
<tr>
<td>93-15</td>
<td>June</td>
<td>P. van der Laan, C. van Eeden</td>
<td>Subset selection with a generalized selection goal based on a loss function</td>
</tr>
<tr>
<td>93-16</td>
<td>June</td>
<td>A.A. Stoorvogel, A. Saberi, B.M. Chen</td>
<td>The Discrete-time $H_{\infty}$ Control Problem with Strictly Proper Measurement Feedback</td>
</tr>
<tr>
<td>93-17</td>
<td>June</td>
<td>J. Beirlant, J.H.J. Einmahl</td>
<td>Maximal type test statistics based on conditional processes</td>
</tr>
<tr>
<td>93-18</td>
<td>July</td>
<td>F.P.A. Coolen</td>
<td>Decision making with imprecise probabilities</td>
</tr>
<tr>
<td>93-19</td>
<td>July</td>
<td>J.A. Hoogeveen, J.K. Lenstra, B. Veltman</td>
<td>Three, four, five, six or the Complexity of Scheduling with Communication Delays</td>
</tr>
</tbody>
</table>