none

Technische Universität Berlin

FAKULTÄT II
MATHEMATIK UND
NATURWISSENSCHAFTEN
Institut für Mathematik

ON THE COMPLEXITY OF SCHEDULING UNIT-TIME JOBS WITH OR-PRECEDENCE CONSTRAINTS

by

BERIT JOHANNES

No. 2003/50

On the Complexity of Scheduling Unit-Time Jobs with OR-Precedence Constraints

Berit Johannes

November 2003

Abstract

AND/OR-networks are an important generalization of ordinary precedence constraints in various scheduling contexts. AND/OR-networks consist of traditional AND-precedence constraints, where a job can only be started after all its predecessors are completed, and OR-precedence constraints, where a job is ready as soon as any of its predecessors is completed. Hence, scheduling problems with AND/OR-constraints inherit the computational hardness of the corresponding problems with AND-precedence constraints. On the other hand, the complexity status of various scheduling problems with OR-constraints has remained open. In this paper, we present several complexity results for scheduling unit-time jobs subject to OR-precedence constraints. In particular, we give a polynomial-time algorithm for minimizing the makespan and the total completion time on identical parallel machines. This algorithm can also be applied if the number of available machines does not decrease over time. In the general case of profile scheduling, scheduling jobs with OR-precedence constraints to minimize the makespan or the total completion time is strongly NP-hard. Furthermore, it is not possible to approximate the makespan with a constant ratio, unless P=NP. In contrast to the makespan and the total completion time, minimizing the total weighted completion time is strongly NP-hard, even on a single machine.

1 Introduction

We consider the following class of scheduling problems. We are given a directed graph on a set of n unit-time jobs that represents OR-precedence constraints between these jobs. This graph is also called an OR-*network*. If there is a path from node i to node j, then i is called a *predecessor* of j and j is a *successor* of i. If there is an arc from i to j, then i is an *immediate predecessor* of j and j is an *immediate successor* of i, formally denoted by $i \prec\cdot\, j$. An assignment of start or completion times to jobs is called a *schedule*. The completion time of a job j is denoted by C_j. A job j satisfies its OR-precedence constraints if it does not start before at least one of its immediate predecessors is completed, that is if $C_j \geqslant \min_{i \prec\cdot j}(C_i) + 1$. Moreover, there are m identical parallel machines. Each job has to be processed on one machine for one unit of time. No more than m jobs may be scheduled at the same time. A schedule is *feasible* if it assigns a start time to every job, does not process more than m jobs simultaneously at any given time, and satisfies the OR-precedence constraints for every job. An instance is feasible if it has a feasible schedule. Note that the precedence graph does not have to be acyclic. Since all jobs have a positive processing time, it is easy to detect whether a given OR-network has a feasible solution or not, as was shown by Igelmund and Radermacher [IR83] for the more general AND/OR-networks, where both kind of precedence constraints coexist (see also Möhring, Skutella, and Stork [MSS]).

In this paper, we consider the objective functions makespan, i.e. the maximum completion time $C_{\max} := \max_j C_j$, the sum of completion times $\sum_j C_j$, and the weighted sum of completion times $\sum_j w_j C_j$, with non-negative weights w_j. Following the common scheduling notation [GLLRK79], we denote the corresponding problems by $\mathrm{P}\,|\,\text{or-prec}, p_j = 1\,|\,C_{\max}$, $\mathrm{P}\,|\,\text{or-prec}, p_j = 1\,|\,\sum C_j$, and

$1\,|\,\text{or-prec}, p_j = 1\,|\,\sum w_j C_j$, respectively. OR-precedence constraints are of interest since they are a special case of AND/OR-networks. AND/OR-networks find use in a variety of applications like resource-constrained project scheduling [Sto00, MS00] and assembly and disassembly problems [GM99].

Scheduling of unit-time jobs with AND-precedence constraints is known to be strongly NP-hard for all three objective functions [Ull75, Law78, LR78]. Of course, these results carry over to AND/OR-networks. Naturally, the question about the complexity of scheduling jobs with OR-precedence constraints has been raised [Woe03]. One might suspect that this problem is at least as hard as scheduling jobs with AND-precedence constraints, since we also have to decide for each job for which immediate predecessor it has to wait. However, we will show that, if we want to minimize the makespan or the total completion time, OR-networks can be reduced to a special case of AND-precedence constraints, making the corresponding scheduling problems solvable in polynomial time. The presented algorithm and its analysis also work if the number of available machines does not increase over time. However, the problem to minimize the makespan or the total completion time becomes strongly NP-hard, if the number of available machines can vary arbitrarily. Actually, we show that the makespan is not approximable within a factor better than $3/2$, unless P=NP.

On the other hand, we show that the minimization of the weighted sum of completion times for jobs in OR-networks is strongly NP-hard even on a single machine. This in fact matches the complexity of the corresponding problem with AND-precedence constraints, for which Lawler [Law78] and Lenstra and Rinnooy Kan [LR78] showed strong NP-hardness.

2 Minimizing the Makespan and Total Completion Time

In this section, we present a polynomial-time algorithm that produces a schedule that is simultaneously optimal for the makespan and the total completion time objective. Let us first give an outline of the algorithm, which proceeds in two steps. We first select for every job j the immediate predecessor for which it has to wait in the solution. In this way we reduce the precedence graph to a subset of precedence constraints such that every job has exactly one immediate predecessor. Hence, the new precedence graph has the form of an outforest and can therefore be interpreted as a set of AND-precedence constraints. We then apply Hu's algorithm [Hu61] for scheduling unit-time jobs subject to AND-precedence constraints in the form of an outtree. For the sake of completeness, we will include a description of Hu's algorithm as well, which amounts to scheduling jobs greedily by giving higher priority to jobs which have a longer tail. Of course, the resulting schedule is also feasible for the original set of OR-precedence constraints. Theorem 2.1 and Corollary 2.2 ensure that the choice of immediate predecessors in Step 1 of the algorithm results in an instance with AND-constraints of the same objective function value. A detailed description of the algorithm is given below.

Algorithm 1: Scheduling unit-time jobs in OR-networks

Input : n jobs of unit-time length with OR-precedence constraints,
m machines
Output : feasible schedule S^m that minimizes the makespan and total completion time

(1) Determine the Earliest Start Time Schedule S^∞ and construct outforest T;
(2a) Let the rank f_j of job j be its height in the tree T;
(2b) List-schedule jobs by considering the precedence constraints in T
and give priority to jobs with higher rank f_j.

In order to select the immediate predecessors in Step 1, we use the concept of *Earliest Start Time Schedule*, which is an optimal but not necessarily feasible schedule since it assumes to have an unlimited number of machines at its disposal. In other words, the restriction of having just m machines is relaxed and only the processing times and precedence constraints are taken into account. The Earliest Start Time Schedule S^∞ can formally be described as the component-wise minimal schedule such that $S_j^\infty \geqslant 0$ and $S_j^\infty \geqslant \min_{i \prec \cdot j} S^\infty(i) + 1$ for all jobs j. It can be determined as follows. For each job j without any OR-predecessor, we set $S_j^\infty := 0$. Every unlabeled job j that has an immediate predecessor with starting time i receives $S_j^\infty := i+1$, and we introduce a red arc between these two jobs. (If an unlabeled job j has two or more immediate predecessors of the same starting time, we only introduce a red arc from one of them to j.) The set of red arcs defines the outforest T of AND-precedence relations. Thus, if there is an arc (i, j) in T, then the job j can start only after the job i has been completed. Once the subset T of precedence constraints is determined, we will exclusively work with these precedence relations and disregard all others.

Clearly, an instance is feasible if and only if there is a feasible Earliest Start Time Schedule, i.e. every job is labeled with a value S_j^∞; see also [IR83, MSS]. Moreover, S_j^∞ is a lower bound for the starting time of job j in any feasible schedule with m machines, for each job j.

Step 2a of the algorithm determines the *rank* f_j of each job j, which will be used as its priority in the list-scheduling procedure in Step 2b. The higher the rank of a job, the higher its priority. The rank f_j of job j corresponds to the length of the longest chain of job j within the outforest T. Thus, the rank of a leaf i in T is $f_i := 1$, and for every other job i we have $f_i := \max_{i \prec \cdot j} f_j + 1$.

In Step 2b, we actually assign the jobs to the machines in a greedy fashion. Whenever a machine is available, we schedule an available job with the highest priority, i.e. a job of highest rank among all unscheduled jobs whose immediate predecessors in T are already completed.

Note that Algorithm 1 produces a feasible schedule S^m and runs in polynomial time.

Theorem 2.1. *The schedule S^m returned by Algorithm 1 minimizes the makespan for any feasible instance of the problem $P\,|\,or\text{-}prec, p_j = 1\,|\,C_{\max}$.*

Proof. Consider the schedule S^m. Let $t^\star$ be its length. If all machines are busy in all time periods $1, \dots, t^\star - 1$, then S^m is optimal. Let therefore $[t-1, t)$, with $t < t^\star$, be the first time period when at least one machine is idle.

Claim 1. *There is a job b_i scheduled in time period $t-i$ with rank at most i for every $i = 1, \dots, t-1$.*

Claim 1 is easily proved by induction. There is a job b_1 scheduled in time period $t-1$ that does not have a successor because T is an outforest and at least one machine is idle during period t. Hence, b_1 has rank 1. For $i > 1$, suppose that all jobs in period $t-i-1$ have rank at least $i+1$. Since T is an outforest, this implies that there are at least m different jobs of rank at least i available for scheduling in the next period $t-(i-1)$. However, by induction hypothesis, the algorithm has scheduled the job b_{i-1} with rank at most $i-1$ in that period, which is a contradiction. Hence, there is a job b_{i+1} in period $t-(i+1)$ with rank at most $i+1$.

We will show in the following that all jobs that are completed in S^∞ by time t, will be completed in S^m by time t as well. Suppose there is a job a_1 with completion time at most t in S^∞, which completes in S^m only after time t. Since there is an idle machine in S^m between time $t-1$ and time t, it follows that job a_1 is not yet available at time $t-1$. This implies that there is a predecessor a_2 of a_1 in T that completes at time t. Moreover, we know that a_2 has rank at least 2.

Claim 2. *There is a chain of jobs $(a_{t+1}, a_t, \dots, a_2, a_1)$ such that a_i completes at time $t-i+2$ and a_i is a predecessor of a_{i-1} in T (for $i = 2, \dots, t+1$). In particular, each a_i has rank at least i (for $i = 1, \dots, t+1$).*

We have already constructed a_1 and a_2. Let us more generally assume that we have constructed a chain $(a_i, a_{i-1}, \dots, a_2, a_1)$ with these properties for some $2 \leqslant i \leqslant t$. Since b_{i-1} has a smaller rank than a_i, but is scheduled before a_i, it follows that there must be a predecessor a_{i+1} of a_i in T that is being processed in time period $t-i+1$. Thus, we have extended the chain and an inductive argument implies Claim 2.

Hence, the earliest completion time of job a_1 in any schedule that obeys the precedence constraints is at least $t+1$, in contradiction to our assumption that a_1 is completed in S^∞ by time t. Thus, all jobs that are completed in S^∞ by time t are completed in S^m by time t as well. We can therefore partition the schedules S^m and S^∞ at time t and apply the same argument to all jobs completed after time t. $\square$

Let us now consider the objective function sum of completion times. Since all jobs have the same length and weight, the proof of Theorem 2.1 implies the following result.

Corollary 2.2. *Algorithm 1 solves the problem* $1\,|\,\textit{or-prec}, p_j = 1\,|\,\sum C_j$ *to optimality.*

Thus, Algorithm 1 minimizes both objective functions, the makespan and the sum of completion times simultaneously.

3 OR-Precedence Constraints and Profile Scheduling

In the literature, one is often concerned with the more general *profile scheduling problem*. Instead of m machines being available at all times, we are given a so-called *profile* p, which specifies for all time intervals $[t-1,t)$ the number $p_t \leqslant m$ of available machines. If p_t is non-increasing in t, Algorithm 1 and its analysis still apply to the problems $\mathrm{P_t}\,|\,\text{or-prec}, p_j = 1\,|\,C_{\max}$ and $\mathrm{P_t}\,|\,\text{or-prec}, p_j = 1\,|\,\sum C_j$. However, if p_t is non-decreasing, both problems turn out to be NP-hard.

Theorem 3.1. *The problems* $P_t\,|\,\textit{or-prec}, p_j = 1\,|\,C_{\max} \leq 2$ *and* $P_t\,|\,\textit{or-prec}, p_j = 1\,|\,\sum C_j$ *are strongly NP-hard.*

Proof. We give a reduction from the MINIMUM COVER problem, which is NP-complete [Kar72]. An instance can be described as follows. We are given a collection C of subsets of a finite set S and a positive integer k with $k < |C|$ and $k < |S|$. The question is whether C contains a cover for S of size k or less, i.e. a subset $C' \subseteq C$ with $|C'| \leqslant k$ such that every element of S belongs to at least one member of C'. We assume w.l.o.g. that there is no element in S that is not contained in one of the sets in C.

We define the corresponding instance of the profile scheduling problem as follows. The number of available machines in time period 1 is k, and it is $|C|+|S|-k$ thereafter. For every subset in C we introduce a *C-job* and for each element in S we introduce an *S-job*. If the element of S that corresponds to the S-job j is contained in the subset associated with the C-job i, we introduce an arc from i to j. The resulting graph represents the OR-precedence constraints. For the makespan problem, we ask whether there exists a feasible schedule of length at most 2. For the sum of completion times objective, we are interested in a schedule of cost at most $2|C|+2|S|-k$.

Let us assume that the MINIMUM COVER instance has a solution, i.e. there exists a subset $C' \subseteq C$ with $|C'| \leqslant k$ such that every element of S belongs to at least one member of C'. We construct a solution of makespan 2 to the corresponding scheduling instance as follows. Every C-job whose corresponding subset in C is also part of C' is scheduled in the first time period. Since $|C'| \leqslant k$ there are indeed sufficiently many machines available for these jobs. In case that $|C'| < k$ we fill the remaining machines with any unscheduled C-jobs. Since $k \leqslant |C|$ it is assured that every machine in time period 1 receives a C-job. All other jobs will be scheduled in time period 2. Since the total number of available spots equals the number of jobs, every job is assigned a time-period and a machine. Since no C-job has any predecessor, the precedence constraints for the C-jobs are automatically satisfied. All S-jobs are scheduled in time period 2 and for every S-job there is an immediate predecessor C-job that completes in time period 1, since the C-jobs that correspond to the MINIMUM COVER solution are scheduled in time period 1. Hence, all precedence constraints are satisfied and the schedule is feasible. The makespan of this schedule is 2, and its total completion time is $k+2(|C|+|S|-k) = 2|C|+2|S|-k$.

For the other direction, assume that there is a feasible schedule for the constructed scheduling instance of makespan 2 (or of total completion time $2|C|+2|S|-k$). It follows that all precedence constraints are satisfied and all jobs are scheduled in the time periods 1 and 2. Since every S-job has at least one predecessor C-job, only C-jobs are scheduled in time period 1. Since no job is scheduled after time period 2, there must be a predecessor scheduled in time period 1 for each S-job in time-period 2. Since there are only k machines available in time period 1, the C-jobs in that period correspond to a cover of size k of the elements in S. Hence, the MINIMUM COVER instance has a solution of value at most k.

□

Note that the profile of the instance constructed in the preceding proof is non-decreasing in t. If one changes the construction slightly by reducing the number of available machines in time periods 3 to $|C|+|S|$ to zero, one obtains the following corollary.

Corollary 3.2. *The problem $P_t \,|\, \text{or-prec}, p_j = 1 \,|\, C_{\max}$ cannot be approximated within a factor smaller than $3/2$ for non-decreasing profiles and it does not have a polynomial-time approximation algorithm with constant performance guarantee for general profiles, unless P=NP.*

In fact, even the feasibility problem is NP-complete for general profiles.

4 Minimizing the Total Weighted Completion Time

The problem of minimizing the weighted sum of completion times is strongly NP-hard, even on a single machine.

Theorem. *The problem $1 \,|\, \text{or-prec}, p_j = 1 \,|\, \sum w_j C_j$ is strongly NP-hard.*

Proof. The proof borrows ideas from the NP-hardness proof of $1 \,|\, \text{prec}, p_j = 1 \,|\, \sum w_j C_j$ by a reduction from MINIMAL LINEAR ARRANGEMENT due to Lawler [Law78] and Lenstra and Rinnooy Kan [LR78].

Consider the decision version of the MAXIMAL LINEAR ARRANGEMENT problem: given a graph $G = (V,E)$ and a positive integer K, is there a one-to-one function $\pi : V \to \{1,2,\dots,|V|\}$ such that $\sum_{\{i,j\}\in E} |\pi_i - \pi_j| \geqslant K$? Let $\overline{G} = (V,\overline{E})$ be the complement of G. For every graph G on n vertices and every permutation π, the expression $\sum_{\{i,j\}\in E} |\pi_i - \pi_j| + \sum_{\{i,j\}\in \overline{E}} |\pi_i - \pi_j|$ is constant. Therefore, solving the MAXIMAL LINEAR ARRANGEMENT in G is equivalent to solving the MINIMAL LINEAR ARRANGEMENT problem in $\overline{G}$. Garey, Johnson and Stockmeyer [GJS76] showed that the MINIMAL LINEAR ARRANGEMENT problem is NP-complete. Hence, the MAXIMAL LINEAR ARRANGEMENT problem is NP-complete as well.

To show the NP-hardness of $1 \,|\, \text{or-prec}, p_j = 1 \,|\, \sum w_j C_j$, we give a reduction from MAXIMAL LINEAR ARRANGEMENT. Let a graph $G = (V,E)$ and a positive integer K be given. Let n be the number of vertices, m the number of edges, and let d_i be the degree of vertex i, for all $i \in V$. We define the following instance of the scheduling problem $1 \,|\, \text{or-prec}, p_j = 1 \,|\, \sum w_j C_j$. For every vertex $i \in V$, we introduce a chain of t unit-time jobs $i_1,\dots,i_t$ with job i_k being an OR-predecessor of job i_{k+1}, $k = 1,\dots,t-1$. All but the last job of the chain have weight 0 and the last job i_t has weight $w_{i_t} = m - d_i$. For every edge $\{i,j\} \in E$, we introduce an edge job $J_{\{i,j\}}$ with $p_{\{i,j\}} = 1$ and $w_{\{i,j\}} = 2$. The edge job $J_{\{i,j\}}$ has two OR-predecessors, i_t and j_t. Let $t = (n+3)m^2$ and $Y = \frac{n(n+1)}{2} mt - t(K-1) - m^2$. It is easy to see that in every optimal schedule the chain $i_1,\dots,i_t$ will be scheduled uninterruptedly for all $i \in V$. In our analysis we therefore consider the chain $i_1,\dots,i_t$ as one job J_i with processing time t and weight $w_i = m - d_i$ for all $i \in V$. Let C_i and $C_{\{i,j\}}$ denote the completion time of job J_i and $J_{\{i,j\}}$, respectively. The following Lemma 4.1 provides more useful information about the structure of any optimal schedule.

Lemma 4.1. *Let $J_{\{i,j\}}$ be any edge job with the vertex jobs J_i and J_j as its* OR*-predecessors. Assume w.l.o.g. that J_i completes before J_j. There is no vertex job scheduled between J_i and $J_{\{i,j\}}$ in every optimal schedule.*

Proof. Assume that in an optimal schedule there is a vertex job J_k scheduled between J_i and $J_{\{i,j\}}$. By processing $J_{\{i,j\}}$ immediately after J_i and consequently delaying all jobs between C_i and $C_{\{i,j\}}$, including job J_k, by one time unit, the objective function value is increased by

$$\sum_{\{\ell: C_i < C_\ell < C_{\{i,j\}}\}} w_\ell \; - \; 2 \sum_{\{\ell: C_i < C_\ell < C_{\{i,j\}}\}} p_\ell \; < \; \sum_{\ell \in V \cup E} w_\ell - 2p_k < \sum_{\ell \in V \cup E} m - 2t = nm + m^2 - 2(n+3)m^2 < 0\,.$$

This contradicts the optimality of the schedule. Note that the resulting schedule is feasible since $J_{\{i,j\}}$ is scheduled after one of its OR-predecessors. □

Suppose that J_i is scheduled in position π_i among the vertex jobs. Thus we have

$$\begin{aligned} &t\pi_i \leqslant C_i \leqslant t\pi_i + m\ , \\ &t|\pi_i - \pi_j| \leqslant |C_i - C_j| \leqslant t|\pi_i - \pi_j| + m\ , \\ &\min\{C_i, C_j\} < C_{\{i,j\}} \leqslant \min\{C_i, C_j\} + m\ . \end{aligned}$$

Hence $\quad t|\pi_i - \pi_j| \leqslant |C_i - C_j| = -2\min\{C_i,C_j\} + C_i + C_j \leqslant -2C_{\{i,j\}} + 2m + C_i + C_j$, and

$$2C_{\{i,j\}} - C_i - C_j > 2\min\{C_i,C_j\} - C_i - C_j = -|C_i - C_j| \geqslant -t|\pi_i - \pi_j| - m\ .$$

If $\sum_{\{i,j\}\in E} |\pi_i - \pi_j| \geqslant K$, then we have

$$\begin{aligned} \sum w_j C_j &= m\sum_{i\in V} C_i + \sum_{\{i,j\}\in E} (2C_{\{i,j\}} - C_i - C_j) \\ &\leqslant m\sum_{i\in V} (t\pi_i + m) - \sum_{\{i,j\}\in E} (t|\pi_i - \pi_j| - 2m) \\ &= \frac{n(n+1)}{2} mt + (n+2)m^2 - t\sum_{\{i,j\}\in E} |\pi_i - \pi_j| \\ &\leqslant \frac{n(n+1)}{2} mt - t(K-1) - m^2 = Y\ . \end{aligned}$$

If $\sum_{\{i,j\}\in E} |\pi_i - \pi_j| \leqslant K-1$, then we have

$$\begin{aligned} \sum w_j C_j &= m\sum_{i\in V} C_i + \sum_{\{i,j\}\in E} (2C_{\{i,j\}} - C_i - C_j) \\ &> m\sum_{i\in V} t\pi_i - \sum_{\{i,j\}\in E} (t|\pi_i - \pi_j| + m) \\ &= \frac{n(n+1)}{2} mt - m^2 - t\sum_{\{i,j\}\in E} |\pi_i - \pi_j| \\ &\geqslant \frac{n(n+1)}{2} mt - t(K-1) - m^2 = Y\ . \end{aligned}$$

This completes the proof. □

Acknowledgements The author would like to thank Gerhard Woeginger for posing the question about the complexity of the problem P$|$or-prec, $p_j = 1\,|\,C_{\max}$, and Martin Skutella for bringing it to her attention. Furthermore, the author is also grateful to Gerhard Woeginger for pointing out the application of Algorithm 1 and its analysis to profile scheduling.

References

[GJS76] M. R. Garey, D. S. Johnson, and L. Stockmeyer. Some simplified NP-complete graph problems. *Theoretical Computer Science*, 1:237–267, 1976.

[GLLRK79] R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. H. G. Rinnooy Kan. Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of Discrete Mathematics*, 5:287–326, 1979.

[GM99] M. H. Goldwasser and R. Motwani. Complexity measures for assembly sequences. *International Journal of Computational Geometry and Applications*, 9:371–418, 1999.

[Hu61] T. C. Hu. Parallel sequencing and assembly line problems. *Operations Research*, 9:841–848, 1961.

[IR83] G. Igelmund and F. J. Radermacher. Algorithmic approaches to preselective strategies for stochastic scheduling problems. *Networks*, 13:29–48, 1983.

[Kar72] R. M. Karp. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, New York, 1972.

[Law78] E. L. Lawler. Sequencing jobs to minimize total weighted completion time subject to precedence constraints. *Annals of Discrete Mathematics*, 2:75–90, 1978.

[LR78] J. K. Lenstra and A. H. G. Rinnooy Kan. Complexity of scheduling under precedence constraints. *Operations Research*, 26:22–35, 1978.

[MS00] R. H. Möhring and F. Stork. Linear preselective policies for stochastic project scheduling. *Mathematical Methods of Operations Research*, 52:501–515, 2000.

[MSS] R. H. Möhring, M. Skutella, and F. Stork. Scheduling with AND/OR precedence constraints. *SIAM Journal on Computing*. To appear.

[Sto00] F. Stork. A branch-and-bound algorithm for minimizing expected makespan in stochastic project networks with resource constraints. Technical Report 613, Technische Universität Berlin, Institut für Mathematik, Berlin, Germany, 1998, revised July 2000.

[Ull75] J. D. Ullman. NP-complete scheduling problems. *Journal of Computer and System Science*, 10:384–393, 1975.

[Woe03] G. Woeginger. Personal communication with M. Skutella, February 2003. Technische Universität Berlin, Berlin, Germany.

Reports from the group

"Combinatorial Optimization and Graph Algorithms"

of the Department of Mathematics, TU Berlin

2004/08 *Marco E. Lübbecke and Jacques Desrosiers:* Selected Topics in Column Generation

2003/49 *Christian Liebchen and Rolf H. Möhring:* Information on MIPLIB's timetab-instances

2003/48 *Jacques Desrosiers and Marco E. Lübbecke:* A Primer in Column Generation

2003/47 *Thomas Erlebach, Vanessa Kääb, and Rolf H. Möhring:* Scheduling AND/OR-Networks on Identical Parallel Machines

2003/43 *Michael R. Bussieck, Thomas Lindner, and Marco E. Lübbecke:* A Fast Algorithm for Near Cost Optimal Line Plans

2003/42 *Marco E. Lübbecke:* Dual Variable Based Fathoming in Dynamic Programs for Column Generation

2003/37 *Sándor P. Fekete, Marco E. Lübbecke, and Henk Meijer:* Minimizing the Stabbing Number of Matchings, Trees, and Triangulations

2003/25 *Daniel Villeneuve, Jacques Desrosiers, Marco E. Lübbecke, and François Soumis:* On Compact Formulations for Integer Programs Solved by Column Generation

2003/24 *Alex Hall, Katharina Langkau, and Martin Skutella:* An FPTAS for Quickest Multicommodity Flows with Inflow-Dependent Transit Times

2003/23 *Sven O. Krumke, Nicole Megow, and Tjark Vredeveld:* How to Whack Moles

2003/22 *Nicole Megow and Andreas S. Schulz:* Scheduling to Minimize Average Completion Time Revisited: Deterministic On-Line Algorithms

2003/16 *Christian Liebchen:* Symmetry for Periodic Railway Timetables

2003/12 *Christian Liebchen:* Finding Short Integral Cycle Bases for Cyclic Timetabling

762/2002 *Ekkehard Köhler and Katharina Langkau and Martin Skutella:* Time-Expanded Graphs for Flow-Dependent Transit Times

761/2002 *Christian Liebchen and Leon Peeters:* On Cyclic Timetabling and Cycles in Graphs

752/2002 *Ekkehard Köhler and Rolf H. Möhring and Martin Skutella:* Traffic Networks and Flows Over Time

739/2002 *Georg Baier and Ekkehard Köhler and Martin Skutella:* On the *k*-splittable Flow Problem

736/2002 *Christian Liebchen and Rolf H. Möhring:* A Case Study in Periodic Timetabling

Reports may be requested from:

Sekretariat MA 6–1
Fakultt II – Institut fr Mathematik
TU Berlin
Straße des 17. Juni 136
D-10623 Berlin – Germany

e-mail: klink@math.TU-Berlin.DE

Reports are also available in various formats from

`http://www.math.tu-berlin.de/coga/publications/techreports/`

and via anonymous ftp as

`ftp://ftp.math.tu-berlin.de/pub/Preprints/combi/Report-`*number-year*`.ps`