Technische Universität Berlin

FACHBEREICH 3
MATHEMATIK

RANDOM–BASED SCHEDULING

by

ANDREAS S. SCHULZ   MARTIN SKUTELLA

No. 549/1997

Random–Based Scheduling

New Approximations and LP Lower Bounds

(Extended Abstract)

Andreas S. Schulz and Martin Skutella

Technische Universität Berlin,
Fachbereich Mathematik, MA 6–1,
Straße des 17. Juni 136, 10623 Berlin, Germany,
E-mail {schulz,skutella}@math.tu-berlin.de

Abstract. Three characteristics encountered frequently in real-world machine scheduling are jobs released over time, precedence constraints between jobs, and average performance optimization. The general constrained one-machine scheduling problem to minimize the average weighted completion time not only captures these features, but also is an important building block for more complex problems involving multiple machines.

In this context, the conversion of preemptive to nonpreemptive schedules has been established as a strong and useful tool for the design of approximation algorithms. The preemptive problem is already NP-hard, but one can generate good preemptive schedules from LP relaxations in time-indexed variables. However, a straightforward combination of these two components does not directly lead to improved approximations. By showing schedules in slow motion, we introduce a new point of view on the generation of preemptive schedules from LP-solutions which also enables us to give a better analysis.

Specifically, this leads to a randomized approximation algorithm for the general constrained one-machine scheduling problem with expected performance guarantee e. This improves upon the best previously known worst-case bound of 3. In the process, we also give randomized algorithms for related problems involving precedences that asymptotically match the best previously known performance guarantees.

In addition, by exploiting a different technique, we give a simple $3/2$-approximation algorithm for unrelated parallel machine scheduling to minimize the average weighted completion time. It relies on random machine assignments where these random assignments are again guided by an optimum solution to an LP relaxation. For the special case of identical parallel machines, this algorithm is as simple as the one of Kawaguchi and Kyan [KK86], but allows for a remarkably simpler analysis. Interestingly, its derandomized version actually is the algorithm of Kawaguchi and Kyan.

1 Introduction

The main results of this paper are twofold. First, we give an approximation algorithm for the general constrained single machine scheduling problem to minimize the average weighted completion time. It has performance guarantee e whereas the best previously known worst-case bound is 3 [Sch96]. Second, we present another approximation algorithm for the model with unrelated parallel machines (but with independent jobs without non-trivial release dates) that has performance guarantee $3/2$. Previously, a 2–approximation algorithm was known [SS97]. Our first contribution is based on and motivated by earlier work of Hall, Shmoys, and Wein [HSW96], Goemans [Goe96,Goe97], and Chekuri, Motwani, Natarajan, and Stein [CMNS97]; the second one builds on earlier research by the authors [SS97]. All this work was in turn initiated by a paper of Phillips, Stein, and Wein [PSW95].

The clamp spanning our two main results is the use of randomness which in both cases is guided by optimum solutions to related LP relaxations of these problems. Hence, our algorithms actually are randomized approximation algorithms. A randomized ρ–approximation algorithm is an algorithm that produces a feasible solution whose expected value is within a factor of ρ of the optimum; ρ is also called the expected performance guarantee of the algorithm. However, all algorithms given in this paper can be derandomized with no difference in performance guarantee, but at the cost of increased running times. For reasons of brevity, most often we omit the technical details of derandomization.

In the first part, we consider the following model. We are given a set J of n jobs (or tasks) and a disjunctive machine (or processor). Each job j has a positive integral processing time p_j and an integral release date $r_j \geqslant 0$ before which it is not available. In *preemptive* schedules, a job may repeatedly be interrupted and continued at a later point in time. We generally assume that these preemptions only occur at integer points in time. In *nonpreemptive* schedules, a job must be processed in an uninterrupted fashion. We denote the completion time of job j in a schedule by C_j. In addition, $C_j(\alpha)$ for $0 < \alpha \leqslant 1$ denotes the earliest point in time when an α–fraction of j has been completed, in particular, $C_j(1) = C_j$; α–points were first used in the context of approximation by [HSW96]. The starting time of j is denoted by $C_j(0^+)$. We also consider precedence constraints between jobs. If $j \prec k$ for $j,k \in J$, it is required that j is completed before k can start, i. e., $C_j \leqslant C_k(0^+)$. We seek to minimize the average weighted completion time in this setting: a weight $w_j \geqslant 0$ is associated with each job j and the goal is to minimize $\frac{1}{n}\sum_{j\in J} w_j C_j$, or, equivalently, $\sum_{j\in J} w_j C_j$. In scheduling, it is quite convenient to refer to the respective problems using the standard classification scheme of Graham et al. [GLLRK79]. Both problems $1|\,r_j, \mathit{prec}\,|\sum w_j C_j$ and $1|\,r_j, \mathit{pmtn}, \mathit{prec}\,|\sum w_j C_j$ just described are strongly NP-hard. In the second part of this paper, we are given m unrelated parallel machines instead of a single machine. Each job j has a positive integral processing requirement p_{ij} which depends on the machine i job j will be processed on. Each job j must be processed for the respective amount of time on one of the m machines, and may be assigned to any of them. Every machine can process at most one job at a time. In standard notation, this NP-hard problem reads $\mathrm{R}|\,|\sum w_j C_j$.

Chekuri, Motwani, Natarajan, and Stein [CMNS97] give a strong result about converting preemptive schedules to nonpreemptive schedules on a single machine. Consider any preemptive schedule with completion times C_j, $j = 1,\ldots,n$. Chekuri et al. show that if α is selected at random in $(0,1]$ with density function $e^{\alpha}/(e-1)$, then the expected completion time of job j in the schedule produced by sequencing the jobs in nondecreasing order of α–points is at most $\frac{e}{e-1}C_j$. This is a deep and in a sense best possible result. However, in order to get in this manner polynomial-time approximation algorithms for nonpreemptive single machine scheduling problems, one relies on good (exact or approximate) solutions to the respective preemptive version of the problem on hand. The only case where this immediately led to an improved performance guarantee was single machine scheduling with release dates (no precedence constraints) to minimize the average completion time (unit weights). We therefore suggest to convert so-called fractional schedules obtained from certain LP relaxations to obtain provably good nonpreemptive schedules for problems with precedence constraints and arbitrary (nonnegative) weights. The LP relaxation we exploit is weaker than the one of Hall, Shmoys, and Wein [HSW96] which they used to derive the first constant-factor approximation algorithm for $1|\,r_j,\, prec\,|\sum w_jC_j$. In fact, in contrast to their LP, our LP already is a relaxation of the corresponding preemptive problem $1|\,r_j,\, pmtn,\, prec\,|\sum w_jC_j$. Hence, we are also able to give an approximation algorithm for the latter problem. In addition, we can interpret an LP solution as a preemptive schedule, but with preemptions at fractional points in time; schedules of this kind are called *fractional*. Our way of deriving and analyzing good fractional schedules from LP solutions generalizes the techniques of Hall, Shmoys, and Wein [HSW96] to this weaker LP relaxation. A somewhat intricate combination of these techniques with the conversion procedure of Chekuri et al. leads then to approximation algorithms for nonpreemptive single machine scheduling which improve or asymptotically match the best previously known algorithms. Specifically, for single machine scheduling with precedence constraints and release dates so as to minimize the average weighted completion time, we obtain in this way an e–approximation algorithm. The approximation algorithm of Hall, Shmoys, and Wein [HSW96] has performance guarantee 5.83. Inspired by the work of Hall et al., Schulz [Sch96] gave a 3–approximation algorithm based on a different LP.

Our second technique exploits optimum solutions to LP relaxations of parallel machine problems in a different way. The key idea, which has been introduced in [SS97], is to interpret the value of certain LP variables as the probability with which jobs are assigned to machines. For the quite general model of unrelated parallel machines to minimize the average weighted completion time, we show by giving an improved LP relaxation that this leads to a $3/2$–approximation algorithm. The first constant-factor approximation algorithm for this model was also obtained by Hall, Shmoys, and Wein [HSW96] and has performance guarantee $16/3$. This was subsequently improved to 2 [SS97]. One appealing aspect of this technique is that in the special case of identical parallel machines the derandomized version of our algorithm coincides with the algorithm of Kawaguchi and Kyan [KK86]; we therefore get a simple proof that list-scheduling in order of nonincreasing ratios of weight to processing time is a $3/2$–approximation.

The paper is organized as follows. In Sect. 2, we present the first technique in a quite general framework. It is best understood by showing schedules in slow motion. Then, in Sect. 3, we apply this to the general constrained one-machine scheduling problem. Finally, in Sect. 4, we randomly assign jobs to parallel machines.

2 Showing Schedules in Slow Motion

In addition to the setting described above, we consider instances with *soft* precedence constraints which will be denoted by $\prec'$. For $j,k \in J$, $j \prec' k$ requires that $C_j(\alpha) \leqslant C_k(\alpha)$

for $0 < \alpha \leqslant 1$. That is, at each point in time the completed fraction of job k must not be larger than the completed fraction of its predecessor j.

Of course, if we consider a single machine, it only makes sense to talk about soft precedence constraints if we allow preemption. However, even for the preemptive case the step from strong to soft precedence constraints is not a true relaxation for a single machine. This can be seen by considering the following algorithm:

Algorithm SOFT–TO–STRONG
Input: fractional schedule S that obeys release dates and soft precedence constraints.
Output: preemptive schedule that obeys release dates and strong precedence constraints.

sort: Sort the jobs in order of nondecreasing completion times in the given schedule S.

schedule: Construct a new schedule by scheduling at any point in time the first available job in this list.

Notice that Algorithm SOFT–TO–STRONG usually produces a preemptive schedule. Whenever a job is released, the job being processed (if any) will be preempted if the released job has smaller completion time in S. An example for Algorithm SOFT–TO–STRONG is given in the last two rows of Fig. 2.

Lemma 1. *Given a fractional schedule to an instance of* $1|\,r_j,\, pmtn,\, prec'\,|\sum w_jC_j$, *Algorithm* SOFT–TO–STRONG *constructs in time* $O(n\log n)$ *a preemptive schedule obeying the corresponding strong precedence constraints and release dates with no increase in completion times.*

Proof. We suspect the lemma to belong to the folklore of this field; since we could not explicitly find it in the literature, however, we provide a proof for the sake of completeness. The reasoning is quite similar to the one given in [HSSW96, Proof of Lemma 2.8] for a rather different result. W. l. o. g., we assume $r_j \leqslant r_k$ whenever $j \prec k$ for $j,k \in J$. We denote the completion time of a job j in the given schedule S by C'_j and in the constructed schedule by C_j.

By construction, no job is processed before its release date in the new schedule. Moreover, a job is only preempted if another job is released at that time. Therefore, since all the release dates are integral, Algorithm SOFT–TO–STRONG creates preemptions only at integral points in time.

Furthermore, for $j \prec k$ the feasibility of the given schedule yields $C'_j < C'_k$. Thus, because of $r_j \leqslant r_k$ and our ordering of jobs, k is not processed before j is completed and the strong precedence constraints are obeyed.

It remains to show that $C_j \leqslant C'_j$ for each job $j \in J$. Let $t \geqslant 0$ be the earliest point in time such that there is no idle time in $(t, C_j]$ and only jobs k with $C'_k \leqslant C'_j$ are processed on the machine in the period from t to C_j in the new schedule. We denote the set of these jobs by K. By construction $r_k \geqslant t$ for all $k \in K$ and thus $C'_j \geqslant t + \sum_{k\in K} p_k$. On the other hand, the definition of K implies $C_j = t + \sum_{k\in K} p_k$ and therefore $C_j \leqslant C'_j$.

Finally, the running time of Algorithm SOFT–TO–STRONG is dominated by the sorting step and is therefore $O(n\log n)$. □

If all the release dates are 0 the schedule constructed in Algorithm SOFT–TO–STRONG is nonpreemptive. Thus, we get the following corollary.

Corollary 2. *Given a fractional schedule to an instance of* $1|\,pmtn, prec'\,|\sum w_jC_j$*, Algorithm* SOFT–TO–STRONG *computes in time* $O(n\log n)$ *a nonpreemptive schedule obeying the precedence constraints with no increase in completion times.*

It follows that the NP-hard problem $1|\,prec\,|\sum w_jC_j$ [Law78] can be reduced to the problem $1|\,pmtn, prec'\,|\sum w_jC_j$ which is therefore NP-hard, too. In order to develop approximation algorithms for all these problems we now introduce an LP relaxation of $1|\,r_j, pmtn, prec'\,|\sum w_jC_j$ whose optimum value serves as a surrogate for the true optimum in our estimations. The LP relaxation is an immediate extension of a time-indexed LP proposed by Dyer and Wolsey [DW90] for the problem without precedence constraints. Here, time is discretized into the periods $(t,t+1]$, $t=0,1,\ldots,T$ where T is the planning horizon, say $T := \max_{j\in J} r_j + \sum_{j\in J} p_j - 1$. We have a variable y_{jt} for every job j and every time period $(t,t+1]$ representing the fraction of this period that is dedicated to the processing of job j.

$$
\begin{aligned}
&& \text{minimize} \quad & \sum_{j\in J} w_jC_j && \\
(LP) && \text{subject to} \quad & \sum_{t=r_j}^{T} y_{jt} = p_j \quad \text{for all } j\in J && (1)\\
&& & \sum_{j\in J} y_{jt} \leqslant 1 \quad \text{for } t=0,\ldots,T && (2)\\
&& & \frac{1}{p_j}\sum_{\ell=r_j}^{t} y_{j\ell} \geqslant \frac{1}{p_k}\sum_{\ell=r_k}^{t} y_{k\ell} \quad \text{for all } j \prec' k \text{ and } t=0,\ldots,T && (3)\\
&& & C_j = \frac{p_j}{2} + \frac{1}{p_j}\sum_{t=0}^{T} y_{jt}\left(t+\tfrac{1}{2}\right) \quad \text{for all } j\in J && (4)\\
&& & y_{jt} = 0 \quad \text{for all } j\in J \text{ and } t=0,\ldots,r_j-1 && (5)\\
&& & y_{jt} \geqslant 0 \quad \text{for all } j\in J \text{ and } t=r_j,\ldots,T && (6)
\end{aligned}
$$

Equations (1) say that all the fractions of a job, which are processed in accordance with the release dates, must sum up to the whole job. Since the machine can process only one job at a time, the machine capacity constraints (2) must be satisfied. Constraints (3) say that at any point $t+1$ in time the completed fraction of job k must not be larger than the completed fraction of its predecessor j.

Consider an arbitrary feasible schedule, where job j is being continuously processed between $C_j - p_j$ and C_j on the machine. Then the expression for C_j in (4) corresponds to the real completion time, if we assign the values to the LP variables y_{jt} as defined above, i. e., $y_{jt}=1$ if j is being processed in the time interval $(t,t+1]$. If j is not being continuously processed but preempted once or several times, the expression for C_j in (4) is a lower bound for the real completion time. Hence, (LP) is a relaxation of the scheduling problem under consideration.

On the other hand, we can interpret every feasible solution y to (LP) in a natural way as a fractional schedule which, by (3), softly respects the precedence constraints: take a linear extension of the precedence constraints and process in each time interval $(t,t+1]$ the occurring jobs j for time y_{jt} each, in this order; see Fig. 1 for an example.

In the following, we always identify a feasible solution to (LP) with a corresponding fractional schedule and vice versa. We mutually use the interpretation that seems more suitable for our purposes.

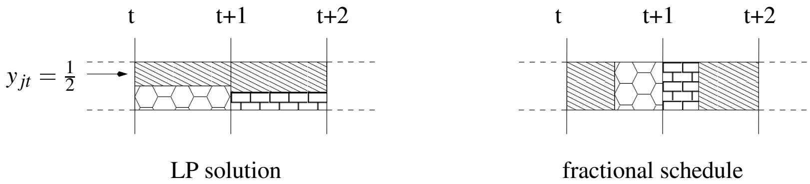

Fig. 1. Interpretation of an LP solution as a fractional schedule and vice versa

Thus, from the LP we get a fractional schedule that satisfies the soft precedence constraints and a lower bound on its total weighted completion time. Of course, this also is a lower bound for preemptive schedules obeying the precedence constraints, and, in turn, of nonpreemptive schedules respecting the precedence constraints. In other words, the LP is a relaxation of $1|\,r_j, pmtn, prec'\,|\sum w_jC_j$, $1|\,r_j, pmtn, prec\,|\sum w_jC_j$, and $1|\,r_j, prec\,|\sum w_jC_j$. The following example shows that it is not better than a 2–relaxation for fractional scheduling (and therefore for the other cases as well) even if all the release dates are 0 and all processing times are 1: let $J = \{1,\dots,n\}$, $w_j = 0$ for $1 \leqslant j \leqslant n-1$, $w_n = 1$ and $j \prec' n$ for $1 \leqslant j \leqslant n-1$. We get a feasible LP solution if we schedule in every time interval $(t, t+1]$ for $0 \leqslant t \leqslant n-1$ a $\frac{1}{n}$–fraction of each job. This yields the LP completion time $(n+1)/2$ for job n, whereas its optimum completion time is obviously n.

The following lemma highlights the relation between the LP completion time and the completion time in the corresponding fractional schedule for a feasible solution to (LP). The observation in part a) is due to Goemans [Goe97]; an analogous result to part b) was already given in [HSW96, Lemma 2.1] for a somewhat different LP.

Lemma 3. *Consider a feasible solution to* (LP) *and let* C_j^{LP} *be the LP completion time of* j *defined in* (4). *Denote the real completion time in the corresponding fractional schedule by* C_j. *Then the following holds:*
a) $\int_0^1 C_j(\alpha)\, d\alpha \leqslant C_j^{LP}$*;*
b) $C_j(\alpha) \leqslant \frac{1}{1-\alpha} C_j^{LP}$ *for any constant* $\alpha \in (0,1)$ *and the given bound is tight.*

Proof. In order to prove part a) we denote by α_t the fraction of job j that is in the fractional schedule completed at time t, for $t = 0,\dots,T+1$. Thus $(\alpha_t)_{t=0}^{T+1}$ is a monotonically nondecreasing sequence with $\alpha_0 = 0$, $\alpha_{T+1} = 1$ and $C_j(\alpha) \leqslant t+1$ for $\alpha \leqslant \alpha_{t+1}$. We can therefore write

$$\int_0^1 C_j(\alpha)\, d\alpha = \sum_{t=0}^{T} \int_{\alpha_t}^{\alpha_{t+1}} C_j(\alpha)\, d\alpha \leqslant \sum_{t=0}^{T} (\alpha_{t+1} - \alpha_t)(t+1)$$
$$= \frac{1}{p_j} \sum_{t=0}^{T} y_{jt}(t+1) = \tfrac{1}{2} + \frac{1}{p_j} \sum_{t=0}^{T} y_{jt}(t + \tfrac{1}{2}) \leqslant C_j^{LP} \quad .$$

The last equality follows from (1). As a consequence of part a) we get for $0 < \alpha \leqslant 1$

$$(1-\alpha) C_j(\alpha) \leqslant \int_\alpha^1 C_j(x)\, dx \leqslant \int_0^1 C_j(x)\, dx \leqslant C_j^{LP} \quad .$$

In order to prove the tightness of this bound, we consider a job j with $p_j = 1$, $r_j = 0$ and an LP solution satisfying $y_{j0} = \alpha - \varepsilon$ and $y_{jT} = 1 - \alpha + \varepsilon$, where $\varepsilon > 0$. This yields $C_j^{LP} = 1 + T(1 - \alpha + \varepsilon)$ and $C_j(\alpha) \geqslant T$. Thus, for ε arbitrarily small and T arbitrarily large, the given bound gets tight. □

As a consequence of Lemma 3 part b) we know that the value of the fractional schedule given by an optimum LP solution can be arbitrarily bad compared to the LP value. Given an arbitrary LP solution and some $\beta \geqslant 1$, the following algorithm computes a new schedule such that all the completion times of jobs are within a constant factor of their LP completion times.

Algorithm SLOW–MOTION
Input: fractional schedule S obeying release dates and soft precedence constraints, and a parameter $\beta \geqslant 1$.
Output: fractional schedule that obeys release dates and soft precedence constraints.

slow motion Consider the given schedule S as a movie and display it β times slower than in real time. Mathematically spoken, we map every point t in time onto $\beta \cdot t$. This defines in a natural way a new schedule S' where job j is being processed for $\beta \cdot p_j$ time units.

cut Let t_j be the earliest point in time when job j has been processed for p_j time units in S'. Convert S' to a feasible schedule S'' by leaving the machine idle whenever it processed job j after t_j.

The output of Algorithm SLOW–MOTION still allows fractional preemption and only obeys the soft precedence constraints. But we can overcome these drawbacks if we use Algorithm SOFT–TO–STRONG as a postprocessing step. Figure 2 shows the action of Algorithm SLOW–MOTION and the postprocessing with Algorithm SOFT–TO–STRONG for an example: we are given three jobs $J = \{1,2,3\}$ with $p_1 = p_2 = p_3 = 2$, $r_1 = 0$, $r_2 = r_3 = 1$ and $2 \prec' 3$. The parameter β is set to $\frac{3}{2}$ in this example.

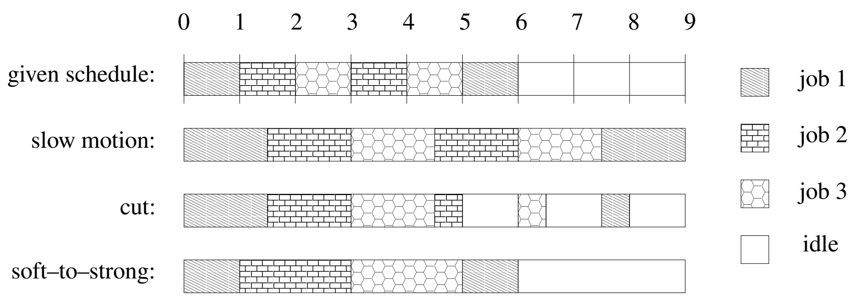

Fig. 2. The two steps in Algorithm SLOW–MOTION together with postprocessing by Algorithm SOFT–TO–STRONG.

Lemma 4. *Algorithm* SLOW–MOTION *with input* $\beta \geqslant 1$ *computes a feasible fractional schedule. The completion time of job j equals* $\beta \cdot C_j(\frac{1}{\beta})$ *where* C_j *is the completion time of job j in the given schedule.*

Proof. Let C_j^β denote the completion time of job j in the schedule constructed by Algorithm SLOW–MOTION. By construction $C_j^\beta(\alpha) = \beta \cdot C_j(\alpha/\beta)$ for $0 < \alpha \leqslant 1$. In particular $C_j^\beta(0^+) = \beta \cdot C_j(0^+) \geqslant \beta \cdot r_j \geqslant r_j$ and $C_j^\beta = \beta \cdot C_j(\frac{1}{\beta})$. Moreover, since S respects the soft precedence constraints we get

$$C_j^\beta(\alpha) = \beta \cdot C_j(\alpha/\beta) \leqslant \beta \cdot C_k(\alpha/\beta) = C_k^\beta(\alpha)$$

for $j \prec' k$ and $0 < \alpha \leqslant 1$. Thus the constructed schedule is feasible. □

We would like to mention that Lemma 4 is related to Theorem 2.1 in [HSW96]. As an easy corollary of Lemma 1, Lemma 3 b), and Lemma 4, Algorithm SLOW–MOTION followed by Algorithm SOFT–TO–STRONG has a performance guarantee of $\beta^2/(\beta-1)$ for the problem $1|\,r_j, pmtn, prec'\,|\sum w_j C_j$. An optimal choice seems to be $\beta = 2$ yielding the performance guarantee 4. But one can in fact do better by choosing β randomly:

Theorem 5. *If one invokes Algorithm* SLOW–MOTION *with input* β *such that* $1/\beta$ *is randomly drawn from* $(0,1]$ *with density function* $f(x) = 2x$*, then the expected completion time of a job* j *in the resulting schedule is bounded from above by twice its LP completion time* C_j^{LP}.

Proof. By Lemma 3 a) and Lemma 4 the expected completion time of j is

$$\int_0^1 f(x) \frac{1}{x} C_j(x)\, dx = 2 \int_0^1 C_j(x)\, dx \leqslant 2 \cdot C_j^{LP} \quad .$$ □

Since the value of an optimum LP-solution is a lower bound on the value of any feasible schedule, this randomized algorithm has performance guarantee 2 if we start with an optimum LP-solution. Using Lemma 1 and Corollary 2 we have thus found randomized 2–approximation algorithms for the following problems:

$$1|\,r_j, pmtn, prec'\,|\sum w_j C_j, \quad 1|\,r_j, pmtn, prec\,|\sum w_j C_j, \quad \text{and} \quad 1|\,prec\,|\sum w_j C_j$$

Moreover, we have shown that (LP) is a 2–relaxation for these problems and that this is best possible for (LP). For both problems $1|\,r_j, pmtn, prec\,|\sum w_j C_j$ and $1|\,prec\,|\sum w_j C_j$ 2–approximation algorithms based on different LP relaxations and without any usage of randomness were known before; see [HSSW96] for both algorithms. In addition, Goemans [Goe96] applied the very same technique as in Theorem 5 to improve the 4–approximation algorithm of Hall, Shmoys, and Wein [HSW96] for $1|\,prec\,|\sum w_j C_j$ to a 2–approximation algorithm.

Notice that we cannot immediately solve (LP) in polynomial time since there are exponentially many variables y_{jt}. Thus, up to now, we have only given pseudo-polynomial approximation algorithms. However, we can overcome this drawback by introducing new variables which are not associated with exponentially many time intervals of length 1, but rather with a polynomial number of intervals of geometrically increasing size. This idea was introduced earlier by Hall, Shmoys, and Wein [HSW96] and since then used in several settings, e. g., in [SS97].

However, in order to get polynomial-time approximation algorithms in this way, we have to pay for with a slightly worse performance guarantee. For any constant $\varepsilon > 0$ we get polynomial-time approximation algorithms with performance guarantee $2+\varepsilon$ for the scheduling problems listed above.

3 An *e*–Approximation Algorithm for the General Constrained One–Machine Scheduling Problem

As mentioned earlier, the randomized conversion technique of [CMNS97] transforms any preemptive schedule into a nonpreemptive one such that the expected completion time of job j in the nonpreemptive schedule is at most a factor of $\frac{e}{e-1}$ of its completion time in the preemptive schedule. In addition, it is easy to see that, if the preemptive schedule obeys given precedence constraints the same holds for the produced nonpreemptive schedule. Hence, if we first use SLOW–MOTION followed by SOFT–TO–STRONG to get a 2–approximation for $1|\,r_j, pmtn, prec\,|\sum w_jC_j$ and then invoke the conversion algorithm of Chekuri et al., we get a $\frac{2e}{e-1}$–approximation algorithm for $1|\,r_j, prec\,|\sum w_jC_j$. Unfortunately, this does not immediately lead to a better performance guarantee than the 3–approximation algorithm known before [Sch96]. However, when we combine both algorithms in a somewhat more intricate way we get a considerably improved performance guarantee. The key idea is that instead of using the algorithms independently we make the choice of the random variable in the algorithm of Chekuri et al. dependent on the choice of β in Algorithm SLOW–MOTION.

Consider the following algorithm that makes use of density functions f and g_β which are subsequently defined.

Algorithm: *e*–APPROXIMATION
Input: Instance of $1|\,r_j, prec\,|\sum w_jC_j$.
Output: Feasible schedule.
1) Compute an optimum solution to (LP). Call the corresponding fractional schedule S.
2) Draw $\frac{1}{\beta}$ randomly from $(0,1]$ with density function f.
3) Let S' be the fractional schedule produced by SLOW–MOTION with input (S,β).
4) Draw α randomly from $(0,1]$ with density function g_β.
5) Construct a nonpreemptive schedule by scheduling the jobs as early as possible in nondecreasing order of $C'_j(\alpha)$.

Here, C'_j is the completion time of job j in the schedule S'. Recall that SLOW–MOTION guarantees that $C'_j(\alpha) \leqslant C'_k(\alpha)$ for $j \prec k$ and any $\alpha \in (0,1]$. Consequently the schedule produced by Algorithm *e*–APPROXIMATION is indeed feasible.

Lemma 6. *Let* $\beta \geqslant 1$ *be a constant and* $g_\beta(x) = \frac{1/\beta}{e^{1/\beta}-1} \cdot e^{x/\beta}$, *for* $x \in (0,1]$. *Then, for each job* $j \in J$, *its expected completion time* $E_\beta(C_j)$ *in the schedule constructed by e–*APPROXIMATION *is at most* $\frac{1}{\beta} \cdot \frac{e^{1/\beta}}{e^{1/\beta}-1} \cdot C'_j$.

Proof. (*Sketch.*) Our analysis almost follows the analysis of [CMNS97] for their conversion technique with the small, but crucial exception that we utilize the structure of the fractional schedule S' produced by Algorithm SLOW–MOTION. In fact, as in [CMNS97] and for the purpose of a more accessible analysis, we do not analyze the schedule produced by e–APPROXIMATION but rather a more structured one which is, however, not better than the output of e–APPROXIMATION. This schedule is obtained by replacing Step 5 with the following procedure:

5') Take the fractional schedule S' produced by Algorithm SLOW–MOTION. Consider the jobs $j \in J$ in nonincreasing order of $C'_j(\alpha)$ and iteratively change the current preemptive schedule by applying the following steps:
   i) remove the $\alpha \cdot p_j$ units of job j that are processed before $C'_j(\alpha)$ from the machine and leave it idle within the corresponding time intervals; we say that this idle time is caused by job j;
   ii) postpone the whole processing that is done later than $C'_j(\alpha)$ by p_j;
   iii) remove the remaining $(1-\alpha)$–fraction of job j from the machine and shrink the corresponding time intervals;
   iv) process job j in the released time interval $(C'_j(\alpha), C'_j(\alpha)+p_j]$.

Consider an arbitrary but fixed $\alpha \in (0,1]$ and a job j. We denote with C^α_j its completion time in this new schedule. Let K_α be the set of jobs that complete an α–fraction in S' before $C'_j(\alpha)$, i. e., $K_\alpha = \{k \in J : C'_k(\alpha) \leqslant C'_j(\alpha)\}$. Moreover, for a job $k \notin K$ let $\eta_k(\alpha)$ be the fraction of k that is completed in S' by time $C'_j(\alpha)$. Then, because of Step 5', it is not too difficult to see that

$$C^\alpha_j \leqslant T_j + (1+\alpha) \sum_{k \in K_\alpha} p_k + \sum_{k \notin K_\alpha} \eta_k(\alpha) \cdot p_k$$

where T_j is the total idle time in the schedule S' before j completes. The factors α and $\eta_k(\alpha)$ purely result from the idle time caused by the respective jobs. This is the moment we can improve the analysis by exploiting the structure of S'. First, observe that the necessity of idle time in the schedule is only caused by release dates. Second, because of the slow motion, no job j starts before time $\beta \cdot r_j$ in S'. Consequently, we may further modify the output of Step 5' by reshrinking the idle time caused by job j by a factor of β, for all $j \in J$, without violating the feasibility. Therefore we get

$$C^\alpha_j \leqslant T_j + (1+\frac{\alpha}{\beta}) \sum_{k \in K_\alpha} p_k + \sum_{k \notin K_\alpha} \frac{\eta_k(\alpha)}{\beta} \cdot p_k \ .$$

Hence,

$$E_\beta(C_j) \leqslant \int_0^1 \left(T_j + (1+\frac{\alpha}{\beta}) \sum_{k \in K_\alpha} p_k + \sum_{k \notin K_\alpha} \frac{\eta_k(\alpha)}{\beta} \cdot p_k \right) g_\beta(\alpha)\, d\alpha \tag{7}$$

and a similar argument as the one given by Chekuri et al. [CMNS97] gives the desired bound. □

Lemma 6 together with Lemma 4 yields the following upper bound on the expected completion time of any job j in the schedule produced by Algorithm e–APPROXIMATION:

$$E(C_j) \leqslant \int_0^1 f(x) \cdot \frac{e^x}{e^x - 1} \cdot C_j^S(x)\, dx \ ,$$

where C_j^S is the completion time of job j in schedule S and $\frac{1}{\beta}$ is replaced by x. An optimal choice of the density function f is $f(x) = e \cdot (1 - e^{-x})$. Hence,

$$E(C_j) \leqslant e \int_0^1 C_j^S(x)\, dx \leqslant e \cdot C_j^{LP}$$

by Lemma 3 a). This eventually gives the following result.

Theorem 7. *Algorithm e–*APPROXIMATION *is an e–approximation algorithm for single machine scheduling subject to release dates and precedence constraints so as to minimize the average weighted completion time.*

To summarize where the improvement on $\frac{2e}{e-1}$ originates from, observe that the bound (7) is the better the larger β is chosen since the idle time caused by jobs is shrunken by the factor β. On the other hand, Lemma 4 and Lemma 3 b) show that the bound on the completion time of any specific job increases as $\beta \geqslant 2$ increases. It is the balancing of these two effects that leads to the better approximation.

As a consequence of Lemma 4 we know that $C'_j(\alpha) = \beta \cdot C_j^S(\alpha/\beta)$. Hence, Algorithm e–APPROXIMATION nonpreemptively schedules the jobs as early as possible in nondecreasing order of $C_j^S(\gamma)$ where $\gamma = \alpha/\beta$ is randomly drawn from $(0,1]$ with a certain density function implicitly defined by f and g_β.

Finally, observe that (LP) therefore is an e–relaxation of $1|\,r_j, prec\,|\sum w_j C_j$. Due to the huge number of variables in this LP relaxation, e–APPROXIMATION also is only a pseudo-polynomial algorithm. Again, however, we may replace this LP with a similar one in interval-indexed variables and this results in a polynomial-time $(e+\varepsilon)$–approximation algorithm for the general constrained one-machine scheduling problem $1|\,r_j, prec\,|\sum w_j C_j$.

4 List Scheduling with Random Assignments to Parallel Machines

We consider the scheduling problem $\mathrm{R}|\,|\sum w_j C_j$. That is, we are given m machines and the processing time of each job $j \in J$ may depend on the machine i it is processed on; it is therefore denoted by p_{ij}. Each job must be processed on one of the machines, and may be assigned to any of them. In contrast to [SS97], we use a tighter LP relaxation in order to get a lower bound for the problem which we then use to give an improved approximation algorithm. Independently, this improvement has also been derived by Fabián A. Chudak (communicated to us by David B. Shmoys, March 1997) after reading a preliminary version of [SS97] which contained a 2–approximation algorithm for $\mathrm{R}|\,r_{ij}\,|\sum w_j C_j$.

Let $T = \sum_{j \in J} \max_i p_{ij} - 1$ and introduce for every job $j \in J$, every machine $i = 1, \ldots, m$, and every point $t = 0, \ldots, T$ in time a variable y_{ijt} which represents the processing time of job j on machine i within the time interval $(t, t+1]$. Equivalently, one can say that a y_{ijt}/p_{ij}–fraction of job j is being processed on machine i within the time interval $(t,t+1]$. The LP is as follows:

$$
(LP_R) \qquad \begin{aligned}
&\text{minimize} && \sum_{j\in J} w_j C_j \\
&\text{subject to} && \sum_{i=1}^{m}\sum_{t=0}^{T} \frac{y_{ijt}}{p_{ij}} = 1 \quad \text{for all } j \in J && (8)\\
& && \sum_{j\in J} y_{ijt} \leqslant 1 \quad \text{for } i = 1,\dots,m \text{ and } t = 0,\dots,T && (9)\\
& && C_j \geqslant \sum_{i=1}^{m}\sum_{t=0}^{T}\Big(\frac{y_{ijt}}{p_{ij}}\big(t+\tfrac{1}{2}\big) + \tfrac{1}{2} y_{ijt}\Big) \quad \text{for all } j \in J && (10)\\
& && C_j \geqslant \sum_{i=1}^{m}\sum_{t=0}^{T} y_{ijt} \quad \text{for all } j \in J && (11)\\
& && y_{ijt} \geqslant 0 \quad \text{for } i = 1,\dots,m,\ j \in J,\ t = 0,\dots,T && (12)
\end{aligned}
$$

Equations (8) say that all the fractions of a job must sum up to the processing requirement of the whole job. Since each machine can process only one job at a time, the machine capacity constraints (9) must be satisfied. Consider an arbitrary feasible schedule, where job j is being continuously processed between $C_j - p_{hj}$ and C_j on machine h. Then the right hand side of (10) corresponds to the real completion time, if we assign the values to the LP variables y_{ijt} as defined above, i. e., $y_{ijt} = 1$ if $i = h$ and $t \in [C_j - p_{hj}, C_j - 1]$, and $y_{ijt} = 0$ otherwise. Moreover, the right hand side of (11) equals the processing time p_{hj} of j on machine h in this case and is therefore a lower bound on the completion time. Hence, (LP_R) is a relaxation of the scheduling problem under consideration.

We can divide the problem to construct a feasible schedule for a given instance into two steps: in the first step we assign each job to a machine it should be processed on; in the second step we decide in which order the jobs are processed on each of the m machines. In fact, we only have to worry about the first step since the second step can be solved optimally by Smith's rule [Smi56]: sequence for each machine i the jobs that were assigned to it in order of nonincreasing w_j/p_{ij}. So it remains to decide which job should be processed on which machine. This is the point where we can make use of an optimum solution to (LP_R). We may interpret the LP solution as follows: for job j a fraction of it given by $f_{ij} = \frac{1}{p_{ij}} \sum_{t=0}^{T} y_{ijt}$ should be processed on machine i. By (8) these fractions some up to 1 over all machines. Since we are not allowed to divide j into fractions but have to process it continuously on one machine, we interpret the f_{ij} as probabilities instead. This yields the following algorithm:

Algorithm RANDOM–ASSIGNMENT
Input: Instance of $R\,|\,|\sum w_j C_j$.
Output: Feasible schedule.
1) Compute an optimum solution to (LP_R).
2) For each job $j \in J$, assign job j to machine i with probability f_{ij}.
3) Schedule on each machine i the jobs that were assigned to it in order of nonincreasing w_j/p_{ij}.

Theorem 8. *Algorithm* RANDOM–ASSIGNMENT *has performance guarantee* $\frac{3}{2}$.

In order to prove Theorem 8 we in fact do not consider Algorithm RANDOM–ASSIGNMENT but a modified version which is slightly more complicated, not better than the original algorithm, but easier to analyze. We replace Steps 2 and 3 by

2') Assign job j to a machine-time pair (i,t) with probability y_{ijt}/p_{ij} and set $t_j := t$.
3') Schedule on each machine i the jobs that were assigned to it in order of nondecreasing t_j.

The random assignment of job j to one of the machines in Step 2' is exactly the same as in the original algorithm. In addition, we create a random order through the random variables t_j, ties are also broken randomly. Of course, this random order cannot be better than the optimum order given by Smith's rule. Thus the modified algorithm cannot be better than the original one and Theorem 8 is a corollary of the following lemma.

Lemma 9. *The expected completion time of job j in the schedule constructed by the modified algorithm is bounded from above by $\frac{3}{2}$ times the optimum LP completion time C_j^{LP}, for all $j \in J$.*

Proof. We first consider the conditional expectation of j's completion time under the assumption that j was assigned to the machine-time pair (i,t). We denote this conditional expectation by $E_{it}(C_j)$ and get

$$\begin{aligned} E_{it}(C_j) &= p_{ij} + \sum_{k \neq j} p_{ik} \cdot \Pr(k \text{ on } i \text{ before } j) = p_{ij} + \sum_{k \neq j} p_{ik} \cdot \Big(\sum_{\ell=0}^{t-1} \frac{y_{ik\ell}}{p_{ik}} + \frac{1}{2} \cdot \frac{y_{ikt}}{p_{ik}} \Big) \\ &= p_{ij} + \sum_{\ell=0}^{t-1} \sum_{k \neq j} y_{ik\ell} + \frac{1}{2} \sum_{k \neq j} y_{ikt} \leqslant p_{ij} + t + \frac{1}{2} \quad . \end{aligned}$$

The last inequality follows from the machine capacity constraints (9). Summing over all possible assignments of j to intervals and machines we finally bound the expectation of j's completion time as follows:

$$\begin{aligned} E(C_j) &= \sum_{i=1}^{m} \sum_{t=0}^{T} \frac{y_{ijt}}{p_{ij}} \cdot E_{it}(C_j) \leqslant \sum_{i=1}^{m} \sum_{t=0}^{T} \Big(\frac{y_{ijt}}{p_{ij}} (t + \tfrac{1}{2}) + y_{ijt} \Big) \\ &= \sum_{i=1}^{m} \sum_{t=0}^{T} \Big(\frac{y_{ijt}}{p_{ij}} (t + \tfrac{1}{2}) + \tfrac{1}{2} y_{ijt} \Big) + \frac{1}{2} \sum_{i=1}^{m} \sum_{t=0}^{T} y_{ijt} \leqslant \frac{3}{2} \cdot C_j^{LP} \quad . \end{aligned}$$

The last inequality follows from (10) and (11). □

Again, we cannot directly solve (LP_R) in polynomial time since there are exponentially many variables y_{ijt}. Thus the running time of Algorithm RANDOM–ASSIGNMENT is only pseudo-polynomial. However, once more we can overcome this difficulty by replacing the time-indexed variables with interval-indexed variables. Hence, we get a polynomial-time approximation algorithm with performance guarantee $\frac{3}{2} + \varepsilon$ for any constant $\varepsilon > 0$.

As a special case of the scheduling problem discussed above we now consider identical parallel machines: the processing time of job j does not depend on the machine it is processed on and it is therefore denoted by p_j. A generalization of Smith's rule to identical parallel machines is the following list scheduling algorithm: sort the jobs in order of nonincreasing w_j/p_j; whenever a machine becomes available, the list is scanned for the first not yet executed job, which is then assigned to the machine. Kawaguchi and Kyan [KK86] have shown that this simple algorithm has performance guarantee $(\sqrt{2}+1)/2$ and that this is best possible for the algorithm. Since their analysis is somewhat complicated we present here a randomized algorithm which is as simple as the one of Kawaguchi and Kyan but much easier to analyze. Moreover, its derandomization by the method of conditional probabilities is precisely the algorithm of Kawaguchi and Kyan.

Algorithm RANDOM–KK
Input: Instance of $\mathrm{P}\,|\,|\sum w_j C_j$.
Output: Feasible schedule.
1) Assign each job randomly (with probability $\frac{1}{m}$) to one of the machines.
2) Schedule on each machine i the jobs that were assigned to i in order of nonincreasing w_j/p_j.

Theorem 10.
a) Algorithm RANDOM–KK *has performance guarantee* $\frac{3}{2}$. *Moreover, there exist instances for which this bound is asymptotically tight.*
b) Derandomization by the method of conditional probabilities precisely gives the algorithm of Kawaguchi and Kyan.

Proof. Because of Theorem 8 and the construction of Algorithm RANDOM–ASSIGNMENT it suffices to show that there exists an optimum solution to (LP_R) such that each of the m machines processes an $\frac{1}{m}$ fraction of every job. This can be easily seen in the following way: take an optimum solution y to (LP_R) and construct a new solution y' by setting $y'_{ijt} = \frac{1}{m}\sum_{h=1}^{m} y_{hjt}$ for each i, j and t. This is obviously a feasible solution to (LP_R) with the same objective value and therefore optimal. If we choose this solution in Step 1 of Algorithm RANDOM–ASSIGNMENT, it obviously does the same as Algorithm RANDOM–KK.

In order to show that the performance guarantee $\frac{3}{2}$ is best possible, we consider instances with m identical parallel machines and m jobs of unit length and weight. We get an optimal schedule with value m by assigning one job to each machine. On the other hand we can show that the expected completion time of a job in the schedule constructed by Algorithm RANDOM–KK is $\frac{3}{2} - \frac{1}{2m}$ which converges to $\frac{3}{2}$ for increasing m. Since the fraction w_j/p_j equals 1 for all jobs, we can w. l. o. g. schedule on each machine the jobs that were assigned to it in a random order. Consider a fixed job j and the machine i it has been assigned to. The probability that a job $k \neq j$ was assigned to the same machine is $\frac{1}{m}$. In this case k is processed before j on the machine with probability $\frac{1}{2}$. We therefore get $E(C_j) = 1 + \sum_{k\neq j} \frac{1}{2m} = \frac{3}{2} - \frac{1}{2m}$.

The derandomization of Algorithm RANDOM–KK by the method of conditional probabilities yields: iteratively consider the jobs in order of nonincreasing w_j/p_j and assign job j to the machine with the smallest load so far. This is just another formulation of Kawaguchi and Kyan's algorithm. The smallest load is bounded from above by the average load $\frac{1}{m}\sum_{k<j} p_k$ where we sum over all jobs k that we considered before j. Thus the completion time of j is bounded by $p_j + \frac{1}{m}\sum_{k<j} p_k$ which exactly equals the expected completion time of j in Algorithm RANDOM–KK. □

As a consequence of the tightness result in Theorem 10 a) we know that the bound in Theorem 8 is also tight.

References

[CMNS97] C. Chekuri, R. Motwani, B. Natarajan, and C. Stein. Approximation techniques for average completion time scheduling. In *Proceedings of the 8th ACM–SIAM Symposium on Discrete Algorithms*, pages 609 – 618, 1997.

[DW90] M. E. Dyer and L. A. Wolsey. Formulating the single machine sequencing problem with release dates as a mixed integer program. *Discrete Applied Mathematics*, 26:255 – 270, 1990.

[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.

[Goe96] M. X. Goemans, June 1996. Personal communication.

[Goe97] M. X. Goemans. Improved approximation algorithms for scheduling with release dates. In *Proceedings of the 8th ACM–SIAM Symposium on Discrete Algorithms*, pages 591 – 598, 1997.

[HSSW96] L. A. Hall, A. S. Schulz, D. B. Shmoys, and J. Wein. Scheduling to minimize average completion time: Off–line and on–line approximation algorithms. Preprint 516/1996, Department of Mathematics, Technical University of Berlin, Berlin, Germany, 1996. To appear in *Mathematics of Operations Research*.

[HSW96] L. A. Hall, D. B. Shmoys, and J. Wein. Scheduling to minimize average completion time: Off–line and on–line algorithms. In *Proceedings of the 7th ACM–SIAM Symposium on Discrete Algorithms*, pages 142 – 151, 1996.

[KK86] T. Kawaguchi and S. Kyan. Worst case bound of an LRF schedule for the mean weighted flow–time problem. *SIAM Journal on Computing*, 15:1119 – 1129, 1986.

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

[PSW95] C. Phillips, C. Stein, and J. Wein. Scheduling jobs that arrive over time. In *Proceedings of the Fourth Workshop on Algorithms and Data Structures*, number 955 in Lecture Notes in Computer Science, pages 86 – 97. Springer, Berlin, 1995.

[Sch96] A. S. Schulz. Scheduling to minimize total weighted completion time: Performance guarantees of LP–based heuristics and lower bounds. In W. H. Cunningham, S. T. McCormick, and M. Queyranne, editors, *Integer Programming and Combinatorial Optimization*, number 1084 in Lecture Notes in Computer Science, pages 301 – 315. Springer, Berlin, 1996. Proceedings of the 5th International IPCO Conference.

[Smi56] W. E. Smith. Various optimizers for single–stage production. *Naval Research and Logistics Quarterly*, 3:59 – 66, 1956.

[SS97] A. S. Schulz and M. Skutella. Scheduling–LPs bear probabilities: Randomized approximations for min–sum criteria. Preprint 533/1996, Department of Mathematics, Technical University of Berlin, Berlin, Germany, November 1996; revised March 1997.

Reports from the group

"Algorithmic Discrete Mathematics"

of the Department of Mathematics, TU Berlin

554/1997 *Rolf H. Möhring and Matthias Müller–Hannemann:* Complexity and Modeling Aspects of Mesh Refinement into Quadrilaterals

551/1997 *Hans Bodlaender, Jens Gustedt and Jan Arne Telle:* Linear-Time Register Allocation for a Fixed Number of Registers and no Stack Variables

549/1997 *Andreas S. Schulz and Martin Skutella:* Random-Based Scheduling: New Approximations and LP Lower Bounds

536/1996 *Cynthia A. Phillips, Andreas S. Schulz, David B. Shmoys, Cliff Stein, and Joel Wein:* Improved Bounds on Relaxations of a Parallel Machine Scheduling Problem

535/1996 *Rainer Schrader, Andreas S. Schulz, and Georg Wambach:* Base Polytopes of Series-Parallel Posets: Linear Description and Optimization

533/1996 *Andreas S. Schulz and Martin Skutella:* Scheduling–LPs Bear Probabilities: Randomized Approximations for Min–Sum Criteria

530/1996 *Ulrich H. Kortenkamp, Jürgen Richter-Gebert, Aravamuthan Sarangarajan, and Günter M. Ziegler:* Extremal Properties of 0/1-Polytopes

524/1996 *Elias Dahlhaus, Jens Gustedt and Ross McConnell:* Efficient and Practical Modular Decomposition

523/1996 *Jens Gustedt and Christophe Fiorio:* Memory Management for Union-Find Algorithms

520/1996 *Rolf H. Möhring, Matthias Müller-Hannemann, and Karsten Weihe:* Mesh Refinement via Bidirected Flows: Modeling, Complexity, and Computational Results

519/1996 *Matthias Müller-Hannemann and Karsten Weihe:* Minimum Strictly Convex Quadrangulations of Convex Polygons

517/1996 *Rolf H. Möhring, Markus W. Schäffter, and Andreas S. Schulz:* Scheduling Jobs with Communication Delays: Using Infeasible Solutions for Approximation

516/1996 *Leslie A. Hall, Andreas S. Schulz, David B. Shmoys, and Joel Wein:* Scheduling to Minimize Average Completion Time: Off-line and On-line Approximation Algorithms

515/1996 *Christophe Fiorio and Jens Gustedt:* Volume Segmentation of 3-dimensional Images

514/1996 *Martin Skutella:* Approximation Algorithms for the Discrete Time-Cost Tradeoff Problem

509/1996 *Soumen Chakrabarti, Cynthia A. Phillips, Andreas S. Schulz, David B. Shmoys, Cliff Stein, and Joel Wein:* Improved Scheduling Algorithms for Minsum Criteria

508/1996 *Rudolf Müller and Andreas S. Schulz:* Transitive Packing

506/1996 *Rolf H. Möhring and Markus W. Schäffter:* A Simple Approximation Algorithm for Scheduling Forests with Unit Processing Times and Zero-One Communication Delays

505/1996 *Rolf H. Möhring and Dorothea Wagner:* Combinatorial Topics in VLSI Design: An Annotated Bibliography

504/1996 *Uta Wille:* The Role of Synthetic Geometry in Representational Measurement Theory

502/1996 *Nina Amenta and Günter M.Ziegler:* Deformed Products and Maximal Shadows of Polytopes

500/1996 *Stephan Hartmann and Markus W. Schäffter and Andreas S. Schulz:* Switchbox Routing in VLSI Design: Closing the Complexity Gap

498/1996 *Ewa Malesinska, Alessandro Panconesi:* On the Hardness of Allocating Frequencies for Hybrid Networks

496/1996 *Jörg Rambau:* Triangulations of Cyclic Polytopes and higher Bruhat Orders

Reports may be requested from:

S. Marcus
Fachbereich Mathematik, MA 6–1
TU Berlin
Straße des 17. Juni 136
D-10623 Berlin – Germany

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

Reports are available via anonymous ftp from:

ftp.math.tu-berlin.de
cd pub/Preprints/combi
file Report–<number>–<year>.ps.Z