Convex Quadratic and Semidefinite Programming Relaxations in Scheduling

Martin Skutella
Technische Universität Berlin

We consider the problem of scheduling unrelated parallel machines subject to release dates so as to minimize the total weighted completion time of jobs. The main contribution of this paper is a provably good convex quadratic programming relaxation of strongly polynomial size for this problem. The best previously known approximation algorithms are based on LP relaxations in time- or interval-indexed variables. Those LP relaxations, however, suffer from a huge number of variables. As a result of the convex quadratic programming approach we can give a very simple and easy to analyze 2-approximation algorithm which can be further improved to performance guarantee 3/2 in the absence of release dates. We also consider preemptive scheduling problems and derive approximation algorithms and results on the power of preemption which improve upon the best previously known results for these settings. Finally, for the special case of two machines we introduce a more sophisticated semidefinite programming relaxation and apply the random hyperplane technique introduced by Goemans and Williamson for the MaxCut problem; this leads to an improved 1.2752-approximation.

Categories and Subject Descriptors: F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems—*sequencing and scheduling*; G.1.6 [**Numerical analysis**]: Optimization—*Convex programming*; G.2.0 [**Discrete mathematics**]: General; G.3 [**Probability and statistics**]: *probabilistic algorithms (including Monte Carlo)*; I.1.2 [**Symbolic and algebraic manipulation**]: Algorithms—*analysis of algorithms*

General Terms: Algorithms, Theory

Additional Key Words and Phrases: Approximation algorithms, convex optimization, performance guarantee, randomized algorithms, scheduling theory, unrelated machines, worst-case ratio

---

Different parts of this work have appeared in a preliminary form in *Proceedings of the 39th Annual IEEE Symposium on Foundations of Computer Science*, pp. 472–481, November 1998, and in J. Nešetřil (ed.), *Proceedings of the 7th Annual European Symposium on Algorithms*, volume 1643 of Lecture Notes in Computer Science, Springer, Berlin, pp. 127–138, July 1999.
This work was partially supported by DONET within the frame of the TMR Programme (contract number ERB FMRX-CT98-0202) while the author was staying at C.O.R.E., Louvain-la-Neuve, Belgium.
Address: Fachbereich Mathematik, MA 6–1, Technische Universität Berlin, Straße des 17. Juni 136, D–10623 Berlin, Germany, e-mail: skutella@math.tu-berlin.de.

1. INTRODUCTION

The last years have witnessed a very fast development in the area of approximation algorithms for NP-hard scheduling problems. Apart from purely combinatorial approaches, linear programming (LP) relaxations have been proved to be a useful tool in the design and analysis of approximation algorithms for several machine scheduling problems, see, e. g., [Lenstra et al. 1990; Shmoys and Tardos 1993; Phillips et al. 1998; Hall et al. 1997; Chakrabarti et al. 1996; Phillips et al. 1998; Möhring et al. 1996; Goemans 1997a; Chudak and Shmoys 1999; Goemans et al. 2000; Schulz and Skutella 1999b; Schulz and Skutella 1999a; Savelsbergh et al. 1998; Munier et al. 1998; Goemans et al. 1999].

In this paper we pursue a somewhat different line of research. We develop approximation algorithms that are not based on polyhedral relaxations but on convex quadratic and semidefinite programming relaxations which have, to the best of our knowledge, never been used in the area of scheduling before. Convex and more specifically semidefinite programming relaxations of combinatorial optimization problems have attracted the attention of many researchers (see, e. g., [Goemans 1997b]). Grötschel, Lovász, and Schrijver [Grötschel et al. 1981] used semidefinite programming to design a polynomial time algorithm for finding the largest stable set in a perfect graph. The use of semidefinite relaxations in the design of approximation algorithms was pioneered by Goemans and Williamson [Goemans and Williamson 1995].

The network scheduling problem. We study the following network scheduling problem: A set J of n jobs has to be scheduled on m unrelated parallel processors or machines that are connected by a network. The jobs continually arrive over time and each job originates at some node of the network. Therefore, before a job can be processed on another machine, it must take the time to travel there through the network. This is modeled by machine-dependent release dates $r_{ij} \geq 0$ which denote the earliest point in time when job j may be processed on machine i. Together with each job j we are given its positive processing requirement which also depends on the machine i job j will be processed on and is therefore denoted by p_{ij}. In *nonpreemptive* schedules, each job j must be processed for the respective amount of time without interruption on one of the m machines. In *preemptive* schedules, a job may repeatedly be interrupted and continued later on another (or the same) machine. For a given job j it may happen that $p_{ij} = \infty$ for some (but not all) machines i such that job j cannot be scheduled on those machines. Every machine can process at most one job at a time. This network scheduling model has been introduced in [Deng et al. 1990; Awerbuch et al. 1992]. We denote the completion time of job j by C_j. The goal is to minimize the total weighted completion time: a weight $w_j \geq 0$ is associated with each job j and we seek to minimize $\sum_{j \in J} w_j C_j$.

To avoid annoying case distinctions and thus to simplify presentation, we always assume in the following that $p_{ij} < \infty$ for all i, j; however, all techniques and results presented in this paper can easily be extended to the case of general processing times.

Notation. In scheduling, it is quite convenient to refer to the respective problems using the standard classification scheme of Graham, Lawler, Lenstra, and Rinnooy

Kan [Graham et al. 1979]. The problems that we consider are special variants of the general problem described above. Each can be denoted by $\alpha\,|\,\beta\,|\,\gamma$ with the following meaning: The machine characteristic α is either 1, P, or R, where $\alpha = 1$ denotes that the scheduling environment provides only one machine; $\alpha = \mathrm{P}$ indicates identical parallel machines; the general case of unrelated parallel machines described above is denoted by $\alpha = \mathrm{R}$. If the number of parallel machines is fixed to a constant m that does not depend on the input, this is indicated by $\alpha = \mathrm{P}m$ and $\alpha = \mathrm{R}m$, respectively. The field β can be empty or contains one or both of the job characteristics r_j (or r_{ij}) and *pmtn*, indicating whether there are nontrivial release dates and whether preemption is allowed. The third field γ refers to the objective function. We are interested in minimizing the total weighted completion time or, for the special case of unit weights, the total completion time denoted by $\gamma = \sum w_j C_j$ and $\gamma = \sum C_j$, respectively.

Approximation algorithms. Since all problems that we will consider in the sequel are NP-hard (see, e.g., [Lawler et al. 1993]), we cannot hope to be able to compute optimal schedules efficiently. Therefore we are interested in how close one can approach the optimum in polynomial time. A (randomized) *α-approximation algorithm* computes in polynomial time a feasible solution to the problem under consideration whose (expected) value is bounded by α times the value of an optimal solution; α is called the *performance guarantee* or *performance ratio* of the algorithm. Notice that all randomized approximation algorithms in this paper can be derandomized.

A family of polynomial time approximation algorithms with performance guarantee $1 + \varepsilon$ for all fixed $\varepsilon > 0$ is called a *polynomial time approximation scheme* (PTAS). If the running times of the approximation algorithms are even bounded by a polynomial in the input size and $1/\varepsilon$, then these algorithms build a *fully polynomial time approximation scheme* (FPTAS).

Known LP based approximations. The first approximation algorithm for the scheduling problem $\mathrm{R}\,|\,r_{ij}\,|\,\sum w_j C_j$ was obtained by Phillips, Stein, and Wein [Phillips et al. 1997] who gave an algorithm with performance guarantee $O(\log^2 n)$. The first constant factor approximation was developed by Hall, Shmoys, and Wein [Hall et al. 1996] (see also [Hall et al. 1997]) whose algorithm achieves performance ratio 16/3. Generalizing a single machine approximation algorithm of Goemans [Goemans 1997a], this result was then improved by Schulz and Skutella [Schulz and Skutella 1999b] to a $(2+\varepsilon)$-approximation algorithm and a $(3/2+\varepsilon)$-approximation algorithm for the problem without release dates $\mathrm{R}\,|\,|\,\sum w_j C_j$. Independently, the latter result has also been obtained by Chudak [Chudak 1999] after reading a preliminary paper of Schulz and Skutella containing the $(2+\varepsilon)$-approximation for $\mathrm{R}\,|\,r_{ij}\,|\,\sum w_j C_j$. All those approximation results rely somehow on (integer) linear programming formulations or relaxations in time-indexed variables. In the following discussion we assume that all processing times and release dates are integral; furthermore, we define $p_{\max} := \max_{i,j} p_{ij}$.

Phillips, Stein, and Wein modeled the network scheduling problem as a hypergraph matching problem by matching each job j to p_{ij} consecutive time intervals of length 1 on a machine i. The underlying graph contains a node for each job and each pair formed by a machine and a time interval $[t, t+1)$ where t is integral and can achieve values in a range of size np_{max}. Therefore, since $p_{\max}$ may be exponential in the input size, the corresponding integer linear program contains exponentially many variables as well as exponentially many constraints. Phillips et al. eluded this problem by partitioning the set of jobs into groups such that the jobs in each group can be scaled down to polynomial size. However, this complicates both the design and the analysis of their approximation algorithm.

The result of Hall, Shmoys, and Wein is based on a polynomial variant of time-indexed formulations which they called *interval-indexed*. The basic idea is to replace the intervals of length 1 by time intervals $[2^k, 2^{k+1})$ of geometrically increasing size. The decision variables in the resulting linear programming relaxation then indicate on which machine and in which time interval a given job completes. Notice, however, that one looses already at least a factor of 2 in this formulation since the interval-indexed variables do not allow a higher precision for the completion times of jobs. The approximation algorithm of Hall et al. relies on Shmoys and Tardos' rounding technique for the generalized assignment problem [Shmoys and Tardos 1993].

Schulz and Skutella generalized an LP relaxation in time-indexed variables that was introduced by Dyer and Wolsey [Dyer and Wolsey 1990] for the corresponding single machine scheduling problem. It contains a decision variable for each triple formed by a job, a machine, and a time interval $[t, t+1)$ which indicates whether the job is being processed in this time interval on the respective machine. The resulting LP relaxation is a 2-relaxation of the scheduling problem under consideration, i. e., the value of an optimal schedule is within a factor 2 of the optimum LP value. However, as the formulation of Phillips et al., this relaxation suffers from an exponential number of variables and constraints. One can overcome this drawback by turning again to interval-indexed variables. However, in order to ensure a higher precision, Schulz and Skutella used time intervals of the form $[(1+\varepsilon)^k, (1+\varepsilon)^{k+1})$ where $\varepsilon > 0$ can be chosen arbitrarily small; this leads to a $(2+\varepsilon)$-relaxation of polynomial size. Notice, however, that the size of the relaxation still depends substantially on $p_{\max}$ and may be huge for small values of ε. The approximation algorithm based on this LP relaxation uses a randomized rounding technique. In the absence of release dates, the quality of the relaxation and the performance guarantee of the algorithm can be improved to $3/2 + \varepsilon$.

New convex quadratic relaxations. For the problem of scheduling unrelated parallel machines in the absence of nontrivial release dates $\mathrm{R} \,|\,|\, \sum w_j C_j$, we introduce a convex quadratic programming relaxation that leads to a simple 3/2-approximation algorithm. One of the basic observations for this result is that in the absence of nontrivial release dates the parallel machine problem can be reduced to an assignment problem of jobs to machines; for a given assignment of jobs to machines the sequencing of the assigned jobs can be done optimally on each machine i by applying Smith's ratio rule [Smith 1956]: schedule the jobs in order of nonincreasing ratios w_j/p_{ij}. Therefore, the problem can be formulated as an integer quadratic program in $n \cdot m$ assignment variables. The quadratic objective function can then be convexified by carefully raising the diagonal entries of the matrix determining the quadratic term until it becomes positive semidefinite and the function thus becomes convex. The resulting convex quadratic programming relaxation together with randomized rounding leads to the approximation result mentioned above. Independently, the same result has later also been derived by Sethuraman and Squillante [Sethuraman and Squillante 1999]. Since many interesting optimization problems can be formulated as quadratic programs, this approach might well prove useful in a more general context.

Unfortunately, for the general network scheduling problem including release dates the situation is more complicated; for a given assignment of jobs to machines, the sequencing problem on each machine is still strongly NP-hard, see [Lenstra et al. 1977]. However, we know that in an optimal schedule a 'violation' of Smith's ratio rule can only occur after a new job has been released; in other words, whenever two successive jobs on machine i can be exchanged without violating release dates, the job with the higher ratio w_j/p_{ij} will be processed first in an optimal schedule. Therefore, the sequencing of jobs that are being started between two successive release dates can be done optimally by Smith's ratio rule. We make use of this insight by partitioning the processing on each machine i into n *time slots* which are essentially defined by the n release dates r_{ij}, $j \in J$; since the sequencing of jobs in each time slot is easy, we have to solve an assignment problem of jobs to time slots and can apply similar ideas as for the problem without release dates. In particular, we derive a convex quadratic programming relaxation in n^2m assignment variables and $O(nm)$ constraints. Randomized rounding based on an optimal solution to this relaxation finally leads to a very simple and easy to analyze 2-approximation algorithm for the general network scheduling problem.

Extensions to preemptive network scheduling. Our technique can be extended to scheduling problems with preemptions. In the context of network scheduling it is reasonable to assume that after a job has been interrupted on one machine, it cannot immediately be continued on another machine; it must again take the time to travel there through the network. We call the delay caused by such a transfer *communication delay*. In a similar context, communication delays between precedence constrained jobs have been studied, see, e. g., [Papadimitriou and Yannakakis 1990].

We give a 3-approximation algorithm for the problem $\mathrm{R} \,|\, r_{ij},\, pmtn \,|\, \sum w_j C_j$ that, in fact, does not make use of preemptions but computes nonpreemptive schedules. Thus, the approximation result also holds for preemptive network scheduling with arbitrary communication delays. Moreover, it also implies a bound on the power of preemption, i. e., one cannot gain more than a factor 3 by allowing preemptions. For the problem without nontrivial release dates $\mathrm{R} \,|\, pmtn \,|\, \sum w_j C_j$, the same technique yields a 2-approximation algorithm. For the preemptive scheduling problems without communication delays, Phillips, Stein, and Wein [Phillips et al. 1998] gave an $(8 + \varepsilon)$-approximation. In [Skutella 1998] the author has achieved slightly worse results than those presented here, based on a time-indexed LP relaxation in the spirit of [Schulz and Skutella 1999b].

Semidefinite programming relaxations. In the last part of the paper we study a semidefinite programming relaxation for the special case of two machines without release dates $\mathrm{R2} \,|\,|\, \sum w_j C_j$ and develop an approximation algorithm in the spirit of Goemans and Williamson's MaxCut approach [Goemans and Williamson 1995]. They formulated the problems MaxCut and MaxDiCut as integer quadratic programs in $\{1,-1\}$-variables and considered a relaxation to a vector program which is equivalent to a semidefinite program. Moreover, they introduced the beautiful idea of rounding a solution to this relaxation to a feasible cut with a random hyperplane through the origin. Their analysis is based on the observation that the probability for an edge to lie in the (directed) cut can be bounded from below in terms of the corresponding coefficient in the vector program. This leads to performance ratios 0.878 and 0.796 for MaxCut and MaxDiCut, respectively.

Feige and Goemans [Feige and Goemans 1995] refined this approach by adding additional constraints to the vector program and by modifying its solution before applying the random hyperplane technique. This leads to an improvement in the performance guarantee from 0.796 to 0.859 for MaxDiCut. More applications of semidefinite programming relaxations in the design of approximation algorithms can for instance be found in [Karger et al. 1998; Chor and Sudan 1998; Frieze and Jerrum 1997].

We contribute to this line of research: The only problems in combinatorial optimization where the random hyperplane technique discussed above has proved useful in the design of approximation algorithms so far are maximization problems. The reason is that up to now only lower bounds on the crucial probabilities involved have been attained, see [Goemans and Williamson 1995, Lemmas 3.2 and 3.4; Lemmas 7.3.1 and 7.3.2]. Inspired by the work of Feige and Goemans we analyze a slightly modified rounding technique and give upper bounds for those probabilities. Together with a more sophisticated semidefinite programming relaxation this leads to the first approximation algorithm for a minimization problem that is based on the random hyperplane approach; it achieves performance ratio 1.276. For the special case of identical parallel machines, this result can be improved within the more general context of an approximation preserving reduction from the problem on a constant number m of identical parallel machines $\mathrm{P}m \,|\,|\sum w_j C_j$ to Max$k$Cut where $k = m$.

Related recent results. Recently, much progress has been made towards a better understanding of the approximability of scheduling problems with average weighted completion time objective. Skutella and Woeginger [Skutella and Woeginger 2000] developed a polynomial time approximation scheme for the problem of scheduling identical parallel machines in the absence of release dates $\mathrm{P}\,|\,|\sum w_j C_j$; this result improves upon the previously best known $(1+\sqrt{2})/2$-approximation algorithm due to Kawaguchi and Kyan [Kawaguchi and Kyan 1986] (back in the seventies, Sahni gave a fully polynomial time approximation scheme for the problem $\mathrm{P}m\,|\,|\sum w_j C_j$ where the number of machines m is constant and does not depend on the input [Sahni 1976]). Subsequently, several research groups have found polynomial-time approximation schemes for problems with release dates such as $1|\,r_j\,|\sum C_j$ and $\mathrm{P}\,|\,r_j\,|\sum w_j C_j$, the preemptive variant $\mathrm{P}\,|\,r_j,\, pmtn\,|\sum w_j C_j$, and also for the corresponding problems on a constant number of unrelated machines $\mathrm{R}m\,|\,r_j\,|\sum w_j C_j$ and $\mathrm{R}m\,|\,r_j,\, pmtn\,|\sum w_j C_j$; see the resulting joint conference proceedings publication [Afrati et al. 1999] for details.

On the other hand, it has been shown by Hoogeveen, Schuurman, and Woeginger [Hoogeveen et al. 1998] that the problems $\mathrm{R}\,|\,r_j\,|\sum C_j$ and $\mathrm{R}|\;|\sum w_j C_j$ are MaxSNP-hard and therefore do not allow a polynomial time approximation scheme, unless P=NP. We give a new and simpler proof for this observation based on a reduction from an MaxSNP-hard variant of Max3Sat.

Recently and inspired by our work, Ageev and Sviridenko [Sviridenko 1999] developed convex quadratic relaxations and approximation algorithms for scheduling problems with additional constraints on the number of jobs on a machine.

The approximation algorithms presented in this paper also lead to the currently best known approximation results for corresponding machine scheduling problems with rejection; in this setting, a job can either be rejected or accepted for processing. However, rejecting a job causes a certain rejection penalty which is added to the objective function of the schedule of accepted jobs. It has been first observed in [Engels et al. 1998] that these problems can be reduced to classical scheduling problems on unrelated parallel machines; notice, that machine-dependent release dates are crucial for this reduction.

Organization of the paper. The paper is organized along the same line as the introduction. In Section 2 we describe the convex quadratic programming relaxation and approximation results for unrelated machine scheduling without release dates. This approach is generalized to network scheduling with machine dependent release dates in Section 3. Extensions to the corresponding preemptive problems are given in Section 4. In Section 5 we present a more sophisticated vector programming relaxation for the special case of two machines and apply the random hyperplane approach of Goemans and Williamson. In Section 6 we discuss relations between MaxCut and scheduling identical parallel machines and present an approximation preserving reduction. Finally, in Section 7 we present a new proof for the MaxSNP-hardness of scheduling an arbitrary number of unrelated machines.

Throughout the paper we will use the following convention: The value of an optimum solution to the scheduling problem under consideration is denoted by Z^*. For a relaxation (R) we denote the optimum value of (R) by Z_R^* and the value of an arbitrary feasible solution a to (R) by $Z_R(a)$.

2. SCHEDULING UNRELATED MACHINES WITHOUT RELEASE DATES

We start by considering the problem of scheduling unrelated parallel machines in the absence of nontrivial release dates $\mathrm{R}\,|\,|\sum w_j C_j$. It is one of the basic observations for our approach that this parallel machine problem can be reduced to an assignment problem; notice that for a given assignment of jobs to machines the sequencing of the assigned jobs can be done optimally on each machine by applying Smith's ratio rule. Throughout the paper we will use the following convention: Whenever we apply Smith's ratio rule on machine i and $w_k/p_{ik} = w_j/p_{ij}$ for a pair of jobs j, k, the job with smaller index is scheduled first. To simplify notation, we introduce for each machine i a corresponding total order $(J, \prec_i)$ on the set of jobs by setting $j \prec_i k$ if either $w_j/p_{ij} > w_k/p_{ik}$ or $w_j/p_{ij} = w_k/p_{ik}$ and $j < k$.

2.1 An integer quadratic programming formulation

The observation in the last paragraph leads to an integer programming formulation in assignment variables; we introduce for each machine $i = 1, \ldots, m$ and each job $j \in J$ a binary variable $a_{ij} \in \{0, 1\}$ which is set to 1 if and only if job j is being processed on machine i. Lenstra, Shmoys, and Tardos [Lenstra et al. 1990] used the same variables to formulate the problem of minimizing the makespan on unrelated parallel machines as an integer linear program. However, minimizing the average weighted completion time forces quadratic terms and leads to the following integer quadratic program (IQP):

$$\begin{aligned} \text{minimize} \quad & \sum_{j \in J} w_j C_j & & \\ \text{subject to} \quad & \sum_{i=1}^{m} a_{ij} = 1 & \text{for all } j \quad & (1) \\ & C_j = \sum_{i=1}^{m} a_{ij} \cdot \Big(p_{ij} + \sum_{k \prec_i j} a_{ik} \cdot p_{ik} \Big) & \text{for all } j \quad & (2) \\ & a_{ij} \in \{0, 1\} & \text{for all } i, j \quad & (3) \end{aligned}$$

Constraints (1) ensure that each job is assigned to exactly one of the m machines. If job j has been assigned to machine i, its completion time is the sum of its own processing time p_{ij} and the processing times of other jobs $k \prec_i j$ that are also scheduled on machine i. The right hand side of (2) is the sum of these expressions over all machines i weighted by a_{ij}; it is thus equal to the completion time of j. Notice that we could remove constraints (2) and replace C_j in the objective function by the corresponding term on the right hand side of (2).

We denote the quadratic programming relaxation of (IQP) that we get by relaxing the integrality conditions (3) to $a_{ij} \geqslant 0$, for all i, j, by (QP). A feasible solution $\bar{a}$ to (QP) can be turned into a feasible solution to (IQP), i.e., a feasible schedule, by randomized rounding: Each job j is randomly assigned to one of the machines with probabilities given through the values $\bar{a}_{ij}$; notice that $\sum_{i=1}^{m} \bar{a}_{ij} = 1$ by constraints (1). We impose the condition that the random choices are performed pairwise independently for the jobs and refer to this rounding procedure as Algorithm Randomized Rounding. The idea of using randomized rounding in the study of approximation algorithms was introduced by Raghavan and Thompson [Raghavan and Thompson 1987], an overview can be found in [Motwani et al. 1996].

Theorem 2.1. *Given a feasible solution $\bar{a}$ to (QP), the expected value of the schedule computed by Algorithm* Randomized Rounding *is equal to $Z_{QP}(\bar{a})$.*

The proof of Theorem 2.1 relies on the following lemma.

Lemma 2.2. *Consider an algorithm that assigns each job j randomly to one of the m machines. Then, the expected completion time of job j is given by*

$$\mathrm{E}[C_j] \;=\; \sum_{i=1}^{m} \Big(\Pr[j \mapsto i] \cdot p_{ij} + \sum_{k \prec_i j} \Pr[j, k \mapsto i] \cdot p_{ik} \Big)$$

where '$j, k \mapsto i$' ('$j \mapsto i$') denotes the event that both jobs j and k (job j) are assigned to machine i.

Proof. Under the assumption that job j is assigned to the fixed machine i the conditional expectation of j's completion time is

$$\mathrm{E}_{j \mapsto i}[C_j] \;=\; p_{ij} + \sum_{k \prec_i j} \mathrm{Pr}_{j \mapsto i}[k \mapsto i] \cdot p_{ik}$$

where $\mathrm{Pr}_{j \mapsto i}[k \mapsto i]$ denotes the conditional probability that job k is being assigned to machine i. Unconditioning yields

$$\begin{aligned} \mathrm{E}[C_j] \;&=\; \sum_{i=1}^{m} \mathrm{Pr}[j \mapsto i] \cdot \mathrm{E}_{j \mapsto i}[C_j] \\ &=\; \sum_{i=1}^{m} \Big(\mathrm{Pr}[j \mapsto i] \cdot p_{ij} + \sum_{k \prec_i j} \mathrm{Pr}[j, k \mapsto i] \cdot p_{ik} \Big) \end{aligned}$$

which completes the proof. □

Proof of Theorem 2.1. Since for each machine i and each pair of jobs $j \neq k$ the random variables a_{ij} and a_{ik} are drawn independently from each other in Algorithm Randomized Rounding, we get

$$\mathrm{Pr}[j, k \mapsto i] = \mathrm{Pr}[j \mapsto i] \cdot \mathrm{Pr}[k \mapsto i] = \bar{a}_{ij} \cdot \bar{a}_{ik} \;\; .$$

Lemma 2.2 yields the result by constraints (2) and linearity of expectations. □

Algorithm Randomized Rounding can easily be derandomized, e. g., by the method of conditional probabilities. (We refer the reader to [Motwani and Raghavan 1995] for a description and an introduction to this method.) The derandomized version of the algorithm is called Derandomized Rounding. If Algorithm Randomized Rounding starts with an optimum solution to (QP), it computes an integral solution the expected value of which is equal to the optimum value Z^*_{QP} by Theorem 2.1. Thus there must exist at least one random choice that yields a schedule whose value is bounded from above by Z^*_{QP}. On the other hand we know that each feasible solution to (IQP) is by definition also feasible for (QP). This yields the following corollary.

Corollary 2.3. *The optimal values of (IQP) and (QP) are equal. Moreover, given an optimum solution $\bar{a}$ to (QP) one can construct an optimum solution to (IQP) by assigning each job j to an arbitrary machine i with $\bar{a}_{ij} > 0$.*

Bertsimas, Teo, and Vohra [Bertsimas et al. 1996] used similar techniques to establish the integrality of several polyhedra.

It follows from Corollary 2.3 that it is still NP-hard to find an optimum solution to the quadratic program (QP). In the next subsection we consider a relaxation of (IQP) that can be solved in polynomial time. The idea is to convexify the objective function of (QP) in order to get a convex quadratic program. In Section 5 we study an alternative semidefinite programming relaxation of (IQP) for the special case of two machines and extend the ideas developed in [Goemans and Williamson 1995] and [Feige and Goemans 1995] for Max2Sat and MaxDiCut.

2.2 A convex quadratic programming relaxation

Plugging constraints (2) into the objective function, the quadratic program (QP) can be rewritten as

$$\begin{aligned} \text{minimize} \quad & c^T a + \tfrac{1}{2} a^T D a && && (4)\\ \text{subject to} \quad & \sum_{i=1}^{m} a_{ij} = 1 && \text{for } j \in J && (5)\\ & a \geqslant 0 && && (6) \end{aligned}$$

where $a \in \mathbb{R}^{mn}$ denotes the vector consisting of all variables a_{ij} lexicographically ordered with respect to the natural order $1, 2, \ldots, m$ of the machines and then, for each machine i, the jobs ordered according to $\prec_i$. The vector $c \in \mathbb{R}^{mn}$ is given by $c_{ij} = w_j p_{ij}$ and $D = \big(d_{(ij)(hk)}\big)$ is a symmetric $mn \times mn$-matrix given through

$$d_{(ij)(hk)} = \begin{cases} 0 & \text{if } i \neq h \text{ or } j = k\,, \\ w_j p_{ik} & \text{if } i = h \text{ and } k \prec_i j\,, \\ w_k p_{ij} & \text{if } i = h \text{ and } j \prec_i k\,. \end{cases}$$

Because of the lexicographic order of the indices the matrix D is decomposed into m diagonal blocks D_i, $i = 1, \ldots, m$, corresponding to the m machines. If we assume that the jobs are indexed according to $\prec_i$ and if we denote p_{ij} simply by p_j, the i-th block D_i has the following form

$$D_i = \begin{pmatrix} 0 & w_2 p_1 & w_3 p_1 & \cdots & w_n p_1 \\ w_2 p_1 & 0 & w_3 p_2 & \cdots & w_n p_2 \\ w_3 p_1 & w_3 p_2 & 0 & & w_n p_3 \\ \vdots & \vdots & & \ddots & \vdots \\ w_n p_1 & w_n p_2 & w_n p_3 & \cdots & 0 \end{pmatrix} . \tag{7}$$

As an example, consider an instance consisting of 2 jobs where all weights and processing times on the i-th machine are equal to one. In this case we get

$$D_i = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} . \tag{8}$$

In particular, $\det D_i = -1$ and D is not positive semidefinite. It is well known that the objective function (4) is convex if and only if the matrix D is positive semidefinite. Moreover, a convex quadratic program of the form $\min c^T x + \frac{1}{2} x^T D x$ subject to $Ax = b$, $x \geqslant 0$, can be solved in polynomial time (see, e. g., [Kozlov et al. 1979; Chung and Murty 1981]). Thus we get a polynomially solvable relaxation of (QP) if we manage to convexify its objective function. The rough idea is to raise the diagonal entries of D above 0 until D is positive semidefinite.

For binary vectors $a \in \{0,1\}^{mn}$ we can rewrite the linear term $c^T a$ in (4) as $a^T \operatorname{diag}(c) a$, where $\operatorname{diag}(c)$ denotes the diagonal matrix whose diagonal entries coincide with the entries of the vector c. We try to convexify the objective function of (QP) by adding a positive fraction $2\gamma \cdot \operatorname{diag}(c)$, $0 < \gamma \leqslant 1$, to D such that $D + 2\gamma \cdot \operatorname{diag}(c)$ is positive semidefinite. This leads to the following modified objective function:

$$\min \quad (1-\gamma)\cdot c^T a + \tfrac{1}{2} a^T \big(D + 2\gamma \cdot \mathrm{diag}(c)\big) a \ . \tag{9}$$

Since $c \geqslant 0$, the value of the linear function $c^T a$ is greater than or equal to the value of the quadratic function $a^T \mathrm{diag}(c) a$ for arbitrary $a \in [0,1]^{mn}$; equality holds for $a \in \{0,1\}^{mn}$. Therefore the modified objective function (9) underestimates (4). Since we want to keep the gap as small as possible and since (9) is nonincreasing in γ for each fixed vector $a \in [0,1]^{mn}$, we are looking for the smallest possible value of γ such that $D + 2\gamma \cdot \mathrm{diag}(c)$ is positive semidefinite.

LEMMA 2.4. *The function*

$$a \ \longmapsto \ (1-\gamma)\cdot c^T a + \tfrac{1}{2} a^T \big(D + 2\gamma \cdot \mathrm{diag}(c)\big) a$$

is convex for arbitrary instances of $\mathrm{R} \,|\, | \sum w_j C_j$ *if and only if* $\gamma \geqslant \frac{1}{2}$.

PROOF. In order to show that the positive semidefiniteness of $D + 2\gamma \cdot \mathrm{diag}(c)$ for all instances implies $\gamma \geqslant \frac{1}{2}$, we consider the example given in (8). Here, the diagonal entries of the i-th block of $D + 2\gamma \cdot \mathrm{diag}(c)$ are equal to 2γ such that this block is positive semidefinite if and only if $\gamma \geqslant \frac{1}{2}$. Thus, we consider the case $\gamma = \frac{1}{2}$ and show that $D + \mathrm{diag}(c)$ is always positive semidefinite. Using the same notation as in (7), the i-th block of $D + \mathrm{diag}(c)$ has the form:

$$A \ = \ \begin{pmatrix} w_1 p_1 & w_2 p_1 & w_3 p_1 & \cdots & w_n p_1 \\ w_2 p_1 & w_2 p_2 & w_3 p_2 & \cdots & w_n p_2 \\ w_3 p_1 & w_3 p_2 & w_3 p_3 & \cdots & w_n p_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ w_n p_1 & w_n p_2 & w_n p_3 & \cdots & w_n p_n \end{pmatrix} \ . \tag{10}$$

We prove that the matrix A is positive semidefinite by showing that the determinants of all its principal sub-matrices are nonnegative. Note that each principal sub-matrix corresponds to a subset of jobs $J' \subseteq J$ and is of the same form as A for the smaller instance induced by the set of jobs J'. Therefore it suffices to show that the determinant of A is nonnegative for all instances.

For $j = 1, \ldots, n$, we multiply the j-th row and column of A by $\frac{1}{p_j} > 0$. Then, for $j = 1, \ldots, n-1$, we iteratively subtract column $j+1$ from column j. The resulting matrix is upper-triangular. The j^{th} diagonal entry is equal to $\frac{w_j}{p_j} - \frac{w_{j+1}}{p_{j+1}} \geqslant 0$, for $j = 1, \ldots, n-1$, and the n^{th} diagonal entry is $\frac{w_n}{p_n} \geqslant 0$.

Since for $\gamma > \frac{1}{2}$ the matrix $D + 2\gamma \cdot \mathrm{diag}(c)$ is the sum of the two positive semidefinite matrices $D + \mathrm{diag}(c)$ and $(2\gamma - 1)\cdot \mathrm{diag}(c)$, the result follows. □

Lemma 2.4 and the remarks above motivate the consideration of the following convex quadratic programming relaxation (CQP):

$$\text{minimize} \quad \frac{1}{2} c^T a + \frac{1}{2} a^T \big(D + \mathrm{diag}(c)\big) a \qquad \text{subject to (5) and (6).}$$

An equivalent formulation of (CQP) using completion time variables C_j as in Subsection 2.1 is: minimize $\sum_j w_j C_j$ subject to (5), (6), and

$$C_j = \sum_{i=1}^{m} a_{ij} \cdot \Big(\frac{1+a_{ij}}{2} p_{ij} + \sum_{k \prec_i j} a_{ik} \cdot p_{ik}\Big) \qquad \text{for all } j. \tag{11}$$

As mentioned above, (CQP) can be solved in polynomial time. If we consider the special case of identical parallel machines $\mathrm{P}\,|\,|\sum w_j C_j$, we can directly give an optimum solution to (CQP).

LEMMA 2.5. *For instances of* $\mathrm{P}\,|\,|\sum w_j C_j$ *the vector* $\bar{a}$*, defined by* $\bar{a}_{ij} := \frac{1}{m}$ *for all* i, j*, is an optimum solution to* (CQP)*. This optimum solution is unique if all ratios* w_j/p_j, $j = 1, \dots, n$*, are different and positive.*

An easy calculation shows that $\bar{a}$ is a Karush-Kuhn-Tucker point and therefore an optimum solution to (CQP). The following more elegant proof of Lemma 2.5 has been proposed by Goemans (personal communication, May 1998).

PROOF. Let $a \neq \bar{a}$ a feasible solution to (CQP). Since (CQP) is symmetric with respect to the m identical machines, we get $m-1$ additional solutions of the same value as a by cyclically permuting the machines $m-1$ times. The convex combination with coefficients $\frac{1}{m}$ of a and these new solutions is precisely $\bar{a}$. Since the objective function of (CQP) is convex, the value of $\bar{a}$ is less than or equal to the value of a. It follows from the proof of Lemma 2.4 that the objective function is strictly convex if all ratios w_j/p_j, $j = 1, \dots, n$, are different and positive. In this case, the value of $\bar{a}$ is strictly less than the value of a and $\bar{a}$ is the unique optimal solution to (CQP). □

2.3 Simple approximation algorithms

Given an optimum solution $\bar{a}$ to (CQP) one can use Algorithm RANDOMIZED ROUNDING to construct a feasible schedule. However, due to the underestimation of the objective function, the expected value $Z_{QP}(\bar{a})$ of the randomly constructed schedule is in general not equal to the optimum value $Z_{CQP}(\bar{a}) = Z^*_{CQP}$ of the relaxation (CQP).

THEOREM 2.6.

a) *Computing an optimum solution* $\bar{a}$ *to* (CQP) *and using* RANDOMIZED ROUNDING *to construct a feasible schedule is a randomized* 2*-approximation algorithm for the problem* $\mathrm{R}\,|\,|\sum w_j C_j$.
b) *Assigning each job independently and uniformly at random to one of the* m *machines is a* $\big(\frac{3}{2} - \frac{1}{2m}\big)$*-approximation algorithm for the problem* $\mathrm{P}\,|\,|\sum w_j C_j$.

PROOF. Notice that the algorithm described in part b) coincides with the algorithm of part a) for the optimum solution $\bar{a}$ to (CQP) given in Lemma 2.5. Theorem 2.1 yields

$$\mathrm{E}\Big[\sum_j w_j C_j\Big] \;=\; Z_{QP}(\bar{a}) \;=\; Z_{CQP}(\bar{a}) + \tfrac{1}{2}\big(c^T\bar{a} - \bar{a}^T \mathrm{diag}(c)\bar{a}\big) \;\leqslant\; 2 \cdot Z_{CQP}(\bar{a}) \;.$$

The inequality follows from $Z_{CQP}(\bar{a}) \geqslant \frac{1}{2} c^T \bar{a}$ and $\bar{a}^T \mathrm{diag}(c)\bar{a} \geqslant 0$. Since $\bar{a}$ can be computed in polynomial time and $Z_{CQP}(\bar{a}) = Z^*_{CQP}$ is a lower bound on Z^*, we have found a 2-approximation algorithm.

To prove part b) we use a second lower bound on Z^*. For the case of identical parallel machines constraints (5) imply $c^T a = \sum_j w_j p_j \leqslant Z^*$ for every feasible solution a to (CQP). Since $\bar{a}^T \mathrm{diag}(c)\bar{a} = \frac{1}{m} c^T \bar{a}$, the same arguments as above yield $\mathrm{E}\big[\sum_j w_j C_j\big] \leqslant (\frac{3}{2} - \frac{1}{2m}) \cdot Z^*$. □

The second part of the theorem can also be proved based on an LP relaxation in time indexed variables, see [Schulz and Skutella 1999b].

In Theorem 2.6 we have proved a bound on the value of the computed schedule. At the same time, however, we have also derived a bound on the value of an optimum solution to the relaxation (CQP) in terms of an optimum solution to the scheduling problem.

Corollary 2.7. *For instances of* $\mathrm{R}\,|\,|\sum w_j C_j$, *the value of an optimal schedule is within a factor* 2 *of the optimum solution to the relaxation* (CQP). *This bound is tight even for the case of identical parallel machines* $\mathrm{P}\,|\,|\sum w_j C_j$.

Proof. The positive result follows from the proof of Theorem 2.6. To prove the tightness of this result, consider an instance with one job of size and weight one and m identical parallel machines. The value Z^* of an optimum schedule is equal to one; by Lemma 2.5 we get $Z^*_{CQP} = \frac{m+1}{2m}$. Thus, if m goes to infinity the ratio Z^*/Z^*_{CQP} converges to 2. □

2.4 Improving the relaxation and approximation

Unfortunately, we cannot directly carry over the $\frac{3}{2}$-approximation from Theorem 2.6 b) to the setting of unrelated parallel machines. The reason is that $c^T a$ is not necessarily a lower bound on Z^* for every feasible solution a to (CQP). However, the value of each binary solution a is bounded from below by $c^T a$. The idea for an improved approximation result is to add this lower bound as a constraint to (CQP). It leads to the following strengthened relaxation (CQP'):

$$
\begin{aligned}
\text{minimize} \quad & Z_{CQP'} \\
\text{subject to} \quad & \sum_{i=1}^{m} a_{ij} = 1 && \text{for all } j \\
& Z_{CQP'} \geqslant \tfrac{1}{2} c^T a + \tfrac{1}{2} a^T \big(D + \mathrm{diag}(c)\big) a && (12) \\
& Z_{CQP'} \geqslant c^T a && (13) \\
& a \geqslant 0
\end{aligned}
$$

Unfortunately, it is not clear whether (CQP') can be solved to optimality in polynomial time. Consider a simple example consisting of three jobs with weights $w_1 = w_2 = w_3 = 1$ and two machines such that $p_{11} = p_{22} = 1$, $p_{12} = p_{21} = \infty$, and $p_{13} = p_{23} = 6$. Job 1 can only be processed on the first machine and job 2 only on the second machine. Therefore we get $a_{11} = a_{22} = 1$ and $a_{12} = a_{21} = 0$ for every feasible solution with finite value and (CQP') can be rewritten as minimize Z subject to $Z \geqslant 6a_{13}^2 - 6a_{13} + 9$, $Z \geqslant 8$, and $a_{13} \geqslant 0$. The optimum value of this program is 8, but the only optimal solutions are $a_{13} = \frac{1}{2}\big(1 + 1/\sqrt{3}\big)$ and $a_{13} = \frac{1}{2}\big(1 - 1/\sqrt{3}\big)$ which are both irrational.

On the other hand, (CQP') is a convex program and can be solved within an additive error of ε in polynomial time, for example through the ellipsoid algorithm, see [Grötschel et al. 1988].

LEMMA 2.8. *Given an arbitrary feasible solution $\bar{a}$ to (CQP'), Algorithm* RANDOMIZED ROUNDING *computes a schedule whose expected value is bounded from above by $\frac{3}{2} \cdot Z_{CQP'}(\bar{a})$.*

PROOF. The same arguments as in the proof of Theorem 2.6 together with (13) yield

$$\mathrm{E}\Big[\sum_j w_j C_j\Big] \leqslant Z_{CQP'}(\bar{a}) + \tfrac{1}{2}\big(c^T\bar{a} - \bar{a}^T \mathrm{diag}(c)\bar{a}\big) \leqslant \tfrac{3}{2} \cdot Z_{CQP'}(\bar{a})$$

since $\bar{a}^T \mathrm{diag}(c)\bar{a} \geqslant 0$. □

COROLLARY 2.9. *For instances of* $R \,|\,|\sum w_j C_j$*, the value of an optimal schedule is within a factor $\frac{3}{2}$ of the optimum solution to the relaxation (CQP').*

We get a randomized approximation algorithm with expected performance ratio $\frac{3}{2}+\varepsilon$ if we apply Algorithm RANDOMIZED ROUNDING to an almost optimal solution to (CQP') which can be computed in polynomial time. We can prove a slightly better bound for the derandomized version.

THEOREM 2.10. *Computing a near optimal solution to the relaxation (CQP') and using Algorithm* DERANDOMIZED ROUNDING *to get a feasible schedule is a $\frac{3}{2}$-approximation algorithm for* $R \,|\,|\sum w_j C_j$.

PROOF. We compute a feasible solution $\bar{a}$ to (CQP') satisfying $Z_{CQP'}(\bar{a}) < Z^*_{CQP'} + \frac{1}{3}$. By Lemma 2.8 Algorithm DERANDOMIZED ROUNDING converts this solution into a feasible schedule whose value is bounded by

$$\tfrac{3}{2} \cdot Z_{CQP'}(\bar{a}) < \tfrac{3}{2} \cdot Z^*_{CQP'} + \tfrac{1}{2} \leqslant \tfrac{3}{2} \cdot Z^* + \tfrac{1}{2} \ .$$

Since all weights and processing times are integral, the same holds for Z^*. The value of our schedule can therefore be bounded by $\frac{3}{2} \cdot Z^*$. □

Notice that the performance ratios given in Lemma 2.8 and Theorem 2.10 are only tight if (13) is tight for the solution $\bar{a}$ to (CQP'). In general, if $Z_{CQP'}(\bar{a})$ is much larger than $c^T\bar{a}$, we get a better performance guarantee, see Figure 1.

COROLLARY 2.11. *For any feasible solution $\bar{a}$ to (CQP') Algorithm* RANDOMIZED ROUNDING *computes a feasible schedule whose expected value is bounded from above by $\big(1 + \frac{c^T\bar{a}}{2Z_{CQP'}(\bar{a})}\big) \cdot Z_{CQP'}(\bar{a})$.*

We will make use of this observation in Section 5 in order to prove better bounds for the special case of two machines.

3. SCHEDULING UNRELATED MACHINES SUBJECT TO RELEASE DATES

Our approach in the last section was led by the insight that in the absence of nontrivial release dates it suffices to find a reasonable assignment of jobs to machines; afterwards, the problem of finding an optimal sequence for the jobs on the machines is easy. Unfortunately, for the general network scheduling problem including release dates the situation is more complicated; for a given assignment of jobs to machines, the sequencing problem on each machine is still strongly NP-hard, see [Lenstra et al. 1977].

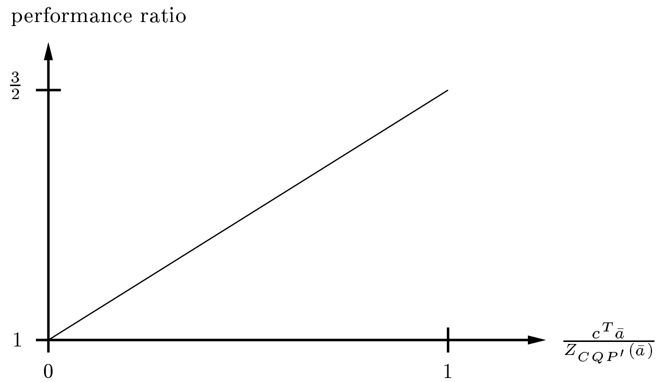

Fig. 1. The performance of RANDOMIZED ROUNDING depends on $\frac{c^T \bar{a}}{Z_{CQP'}(\bar{a})}$.

3.1 Scheduling in time slots

Notice, however, that in an optimal schedule a 'violation' of Smith's ratio rule can only occur after a new job has been released; in other words, whenever two successive jobs on machine i can be exchanged without violating release dates, the job with higher ratio w_j/p_{ij} is processed first in an optimal schedule. Therefore, the sequencing of jobs that are being started between two successive release dates can be done optimally by Smith's ratio rule. We make use of this insight by partitioning the processing on each machine i into n *time slots* which are essentially defined by the n release dates r_{ij}, $j \in J$. Each job is being processed on one machine in one of its time slots and we make sure that job j can only be processed in a slot that starts after its release date. This reduces the problem to finding a good assignment of jobs to time slots and we can apply ideas similar to those in Section 2.

Let $\rho_{i_1} \leqslant \rho_{i_2} \leqslant \cdots \leqslant \rho_{i_n}$ be an ordering of the release dates r_{ij}, $j \in J$, on machine i; moreover, we set $\rho_{i_{n+1}} := \infty$. For a given feasible schedule we say that i_k, the k^{th} time slot on machine i, contains all jobs j that are started within the interval $[\rho_{i_k}, \rho_{i_{k+1}})$ on machine i; we denote this by $j \in i_k$. We may assume that there is no idle time between the processing of jobs in one time slot, i. e., all jobs in a slot are processed one after another without interruption. Moreover, as a consequence of Smith's ratio rule we can without loss of generality restrict to schedules where the jobs in time slot i_k are sequenced according to $\prec_i$:

LEMMA 3.1. *In an optimal solution to the scheduling problem under consideration, the jobs in each time slot i_k are scheduled without interruption in order of nonincreasing ratios w_j/p_{ij}. Furthermore, there exists an optimal solution where the jobs are sequenced according to $\prec_i$ in each time slot i_k.*

Notice that there may be several empty time slots i_k. This happens in particular if $\rho_{i_k} = \rho_{i_{k+1}}$. Therefore it would be sufficient to introduce only q_i time slots for machine i where q_i is the number of different values r_{ij}, $j \in J$. For example, if there are no nontrivial release dates (i. e., $r_{ij} = 0$ for all i and j), we only need to introduce one time slot $[0, \infty)$ on each machine. For this special case our approach coincides with the one given in the last section.

Up to now we have described how a feasible schedule can be interpreted as a feasible assignment of jobs to time slots. We call an assignment *feasible* if each job j is being assigned to a time slot i_k with $\rho_{i_k} \geqslant r_{ij}$. On the other hand, for a given feasible assignment of the jobs in J to time slots we can easily construct a corresponding feasible schedule: Sequence the jobs in time slot i_k according to $\prec_i$ and start it as early as possible after the jobs in the previous slot on machine i are finished but not before ρ_{i_k}. In other words, the starting time s_{i_k} of time slot i_k is given by $s_{i_1} := \rho_{i_1}$ and $s_{i_{k+1}} := \max\{\rho_{i_{k+1}}, s_{i_k} + \sum_{j \in i_k} p_{ij}\}$, for $k = 1, \ldots, n-1$.

LEMMA 3.2. *Given its assignment of jobs to time slots, we can reconstruct an optimal schedule meeting the properties described in Lemma 3.1.*

PROOF. Given the feasible assignment of jobs to time slots, we construct a new feasible schedule as described above. We show that the new schedule coincides with the optimal schedule defining the assignment. By definition of the assignment and by construction of the new schedule, the sequence of jobs coincides for the two schedules on each machine. Thus, it suffices to show that the completion time of each job in the new schedule is less than or equal to its completion time in the optimal schedule; the other direction then follows from optimality.

By contradiction assume that there exists a job j on machine i whose completion time has been increased in the new schedule. Moreover, let j be the first job on machine i with this property. By construction, j must be the first job in its time slot i_k since otherwise its predecessor in this slot would have also been delayed. Moreover, j must have been started strictly later than ρ_{i_k} since its starting time in the optimal schedule is at least ρ_{i_k}. Thus, the start of time slot i_k has been delayed by slot i_{k-1} and job j has been started immediately after the completion of the last job j' in slot i_{k-1}. This, however, implies that j' must have been finished later than in the optimal schedule which is a contradiction to the choice of j. $\square$

Notice that a job that is being assigned to time slot i_k is not necessarily started within the interval $[\rho_{i_k}, \rho_{i_{k+1}})$. In particular, several feasible assignments of jobs to time slots may lead to the same feasible schedule. Consider, e. g., an instance consisting of three jobs of unit length and unit weight that have to be scheduled on a single machine; jobs 1 and 2 are released at time 0, while job 3 becomes available at time 1. We get an optimal schedule by processing the jobs without interruption in order of increasing numbers. This schedule corresponds to five different feasible assignments of jobs to time slots. We can assign job 1 to one of the first two slots, job 2 to the same or a later slot, and finally job 3 to slot 3.

3.2 Quadratic programming formulations and relaxations

In the previous subsection we have reduced the scheduling problem under consideration to finding an optimal assignment of jobs to time slots. Generalizing the approach described in Section 2, we give a formulation of $\mathrm{R} \,|\, r_{ij} \,|\, \sum w_j C_j$ in assignment variables. However, it will turn out that the subject of convexification is slightly more complicated in this general setting.

For each job j and each time slot i_k we introduce a variable $a_{i_k j} \in \{0,1\}$ where $a_{i_k j} = 1$ if job j is being assigned to time slot i_k, and $a_{i_k j} = 0$ otherwise. This leads to the following integer quadratic program:

$$
\begin{aligned}
\text{minimize} \quad & \sum_j w_j C_j \\
\text{subject to} \quad & \sum_{i,k} a_{i_k j} = 1 && \text{for all } j && (14)\\
& s_{i_1} = \rho_{i_1} && \text{for all } i && (15)\\
& s_{i_{k+1}} = \max\{\rho_{i_{k+1}},\, s_{i_k} + \sum_j a_{i_k j} p_{ij}\} && \text{for all } i,\, k && (16)\\
& C_j = \sum_{i,k} a_{i_k j}\Big(s_{i_k} + p_{ij} + \sum_{j' \prec_i j} a_{i_k j'} p_{ij'}\Big) && \text{for all } j && (17)\\
& a_{i_k j} = 0 && \text{if } \rho_{i_k} < r_{ij} && (18)\\
& a_{i_k j} \in \{0,1\} && \text{for all } i,\, k,\, j
\end{aligned}
$$

Constraints (14) ensure that each job is being assigned to exactly one time slot. In constraints (15) and (16) we set the starting times of the time slots as described in Subsection 3.1. If job j is being assigned to time slot i_k, its completion time is the sum of the starting time s_{i_k} of this slot, its own processing time p_{ij}, and the processing times of other jobs $j' \prec_i j$ that are also scheduled in this time slot. The right hand side of (17) is the sum of these expressions over all time slots i_k weighted by $a_{i_k j}$; it is thus equal to the completion time of j. Finally, constraints (18) ensure that no job is being processed before its release date.

It follows from our considerations in Subsection 3.1 that we could replace (18) by the stronger constraints

$$a_{i_k j} \;=\; 0 \qquad\qquad \text{if } \rho_{i_k} < r_{ij} \text{ or } \rho_{i_k} = \rho_{i_{k+1}}$$

which reduces the number of available time slots on each machine. For the special case $\mathrm{R}\,|\,|\sum w_j C_j$ this leads exactly to the integer quadratic program (IQP) that has been introduced in Section 2. Thus, as a consequence of Corollary 2.3, it is still NP-hard to solve the continuous relaxation of our integer quadratic program.

In Section 2 we convexified the objective function of the continuous relaxation (QP) by replacing constraints (2) with the new constraints (11). This motivates the study of the following quadratic programming relaxation for the general problem including release dates: minimize $\sum_j w_j C_j$ subject to (14), (15), (16), (18), and the following two constraints:

$$
\begin{aligned}
C_j \;&=\; \sum_{i,k} a_{i_k j}\Big(s_{i_k} + \frac{1 + a_{i_k j}}{2} p_{ij} + \sum_{j' \prec_i j} a_{i_k j'} p_{ij'}\Big) && \text{for all } j && (19)\\
a_{i_k j} \;&\geqslant\; 0 && \text{for all } i,\, k,\, j && (20)
\end{aligned}
$$

Notice that a solution to this program is uniquely determined by giving the values of the assignment variables $a_{i_k j}$. In contrast to the case without nontrivial release dates, we cannot directly prove that this quadratic program is convex. Nevertheless, in the remaining part of this subsection we will show that it can be solved in polynomial time. The main idea is to show that one can restrict to solutions satisfying $s_{i_k} = \rho_{i_k}$ for all i and k. Adding these constraints and thus getting rid of the variables s_{i_k} then leads to a convex quadratic program.

LEMMA 3.3. *For all instances of* $\mathrm{R}\,|\,r_{ij}\,|\sum w_j C_j$ *there exists an optimal solution to the quadratic programming relaxation satisfying* $s_{i_k} = \rho_{i_k}$ *for all* i *and* k.

PROOF. We show how an arbitrary optimal solution to the quadratic program can be iteratively turned into one satisfying $s_{i_k} = \rho_{i_k}$ for all i and k. Consider the machines one after another. For a given machine i determine the smallest k satisfying $s_{i_k} > \rho_{i_k}$; if such a k does not exist, we proceed to the next machine until we are finished. Consider the job $\hat{j}$ with $a_{i_{k-1}\hat{j}} > 0$ that is maximal with respect to $\prec_i$. Let $\delta := \min\{s_{i_k} - \rho_{i_k}, a_{i_{k-1}\hat{j}}\} > 0$ and move a δ-fraction of job $\hat{j}$ from slot i_{k-1} to slot i_k, i.e., modify the current solution as follows:

$$a_{i_{k-1}\hat{j}} := a_{i_{k-1}\hat{j}} - \delta \ , \qquad a_{i_k\hat{j}} := a_{i_k\hat{j}} + \delta \ .$$

All other assignment variables remain unchanged. Observe that this defines a new feasible solution to the quadratic program where s_{i_k} has been decreased to $s_{i_k} - \delta p_{i\hat{j}}$ and all other $s_{i'_{k'}}$ have remained unchanged. Thus, a short computation shows that the 'completion time' of job $\hat{j}$ given in (19) has changed by

$$\delta \sum_{j' \prec_i \hat{j}} a_{i_k j'} p_{ij'} \ .$$

Moreover, it can be seen that the only other jobs j' whose 'completion time' (19) could have been changed are those which satisfy $j' \prec_i \hat{j}$. Again, a short computation shows that this change is given by

$$-a_{i_k j'} \delta p_{i\hat{j}} \ .$$

Thus, the total change of the objective value is given by

$$\delta \sum_{j' \prec_i \hat{j}} a_{i_k j'} \left(w_{\hat{j}} p_{ij'} - w_{j'} p_{i\hat{j}}\right) \ .$$

Since, by definition of $\prec_i$, all terms in this sum are nonpositive, the new solution is optimal as well. Notice that in the new solution $s_{i_k} = \rho_{i_k}$ or $a_{i_{k-1}\hat{j}} = 0$ by the choice of δ. Therefore, after at most n iterations of the described procedure we arrive at a solution satisfying $s_{i_k} = \rho_{i_k}$ and we can proceed to the next time slot or the next machine. $\square$

As a consequence of Lemma 3.3 we can replace the variables s_{i_k} by the constants ρ_{i_k} if we change constraints (16) to:

$$\sum_j a_{i_k j} p_{ij} \ \leqslant \ \rho_{i_{k+1}} - \rho_{i_k} \qquad \text{for all } i,\ k. \tag{21}$$

Furthermore, if we remove constraints (19) and replace C_j in the objective function by the right hand side of (19), we get the following convex quadratic programming relaxation which we denote by (CQP) since it generalizes the convex quadratic program developed in Section 2:

$$\text{minimize} \quad b^T a + \tfrac{1}{2} c^T a + \tfrac{1}{2} a^T \big(D + \operatorname{diag}(c)\big) a \tag{22}$$

subject to (14), (21), (18), and (20). Here, $a \in \mathbb{R}^{mn^2}$ denotes the vector consisting of all variables $a_{i_k j}$ lexicographically ordered with respect to the natural order $1_1, 1_2, \ldots, m_n$ of the time slots and then, for each slot i_k, the jobs ordered according to $\prec_i$. The vectors $b, c \in \mathbb{R}^{mn^2}$ are given by $b_{i_k j} = w_j \rho_{i_k}$, $c_{i_k j} = w_j p_{ij}$, and $D = \big(d_{(i_k j)(i'_{k'} j')}\big)$ is a symmetric $mn^2 \times mn^2$-matrix given through

$$d_{(i_k j)(i'_{k'} j')} = \begin{cases} 0 & \text{if } i_k \neq i'_{k'} \text{ or } j = j'\,, \\ w_{j'} p_{ij} & \text{if } i_k = i'_{k'} \text{ and } j \prec_i j'\,, \\ w_j p_{ij'} & \text{if } i_k = i'_{k'} \text{ and } j' \prec_i j\,. \end{cases}$$

Because of the lexicographic order of the indices the matrix $D + \operatorname{diag}(c)$ is again decomposed into mn diagonal blocks corresponding to the mn time slots. If we assume that the jobs are indexed according to $\prec_i$ and if we denote p_{ij} simply by p_j, each block corresponding to a time slot on machine i has the form of the matrix A given in (10). In particular, $D + \operatorname{diag}(c)$ is positive semidefinite, the objective function (22) is convex, and (CQP) can be solved in polynomial time.

The convex quadratic programming relaxation (CQP) is in some sense similar to the linear programming relaxation in time-indexed variables that has been introduced in [Schulz and Skutella 1999b]. Without going into the details, we give a rough idea of the common underlying intuition of both relaxations (a more detailed discussion of this matter can be found in [Skutella 1998, Section 3.3.4]): a job may be split into several parts (corresponding to fractional values $a_{i_k j}$ in (CQP)) who can be scattered over the machines and over time. The completion time of a job in such a 'fractional schedule' is somehow related to its mean busy time; the mean busy time of a job is the average point in time at which its fractions are being processed (see (19) where C_j is set to the average over the terms in brackets on the right hand side weighted by $a_{i_k j}$). However, in contrast to the time-indexed LP relaxation, the construction of the convex quadratic program (CQP) contains more insights into the structure of an optimal schedule. As a result, (CQP) is of strongly polynomial size while the LP relaxation contains an exponential number of time-indexed variables and constraints.

3.3 A simple 2-approximation algorithm

A natural generalization of algorithm Randomized Rounding to problems including release dates is the following: Given a feasible solution a to (CQP), we compute an integral solution $\bar{a}$ by setting for each job j exactly one of the variables $\bar{a}_{i_k j}$ to 1 with probabilities given through $a_{i_k j}$. Although the integral solution $\bar{a}$ does not necessarily fulfill constraints (21), it represents a feasible assignment of jobs to time slots and thus a feasible schedule. For our analysis we require again that the random choices are performed pairwise independently for the jobs. We prove the following result.

Theorem 3.4. *Computing an optimal solution to (CQP) and using randomized rounding to turn it into a feasible schedule is a 2-approximation algorithm for the problem* $\mathrm{R} \mid r_{ij} \mid \sum w_j C_j$.

Theorem 3.4 follows from the next lemma which gives a slightly stronger result including job-by-job bounds.

LEMMA 3.5. *Using randomized rounding in order to turn an arbitrary feasible solution to* (CQP) *into a feasible assignment of jobs to time slots yields a schedule such that the expected completion time of each job is bounded by twice the corresponding value* (19) *in the given solution to* (CQP).

PROOF. The computed random assignment of jobs to machines is denoted by $\bar{a}$ such that $\Pr[\bar{a}_{i_k j} = 1] = a_{i_k j}$ by definition. The starting time s_{i_k} of time slot i_k in the schedule corresponding to $\bar{a}$ is given by (15) and (16). The completion time C_j of job j is given by (17).

We consider a fixed job j. First, we also consider a fixed assignment of j to time slot i_k. Since the random choices are performed pairwise independently for the jobs, the conditional probability for job $j' \neq j$ to be assigned to an arbitrary time slot $i'_{k'}$ is given by $\Pr_{j \mapsto i_k}[\bar{a}_{i'_{k'} j'} = 1] = a_{i'_{k'} j'}$.

We start by showing that the conditional expectation of the starting time of time slot i_k is bounded from above by $2\rho_{i_k}$. Notice that by definition there is no idle time in the time interval between ρ_{i_k} and the starting time of time slot i_k; in other words, this interval is completely covered by the processing of jobs $j' \neq j$ that have been assigned to earlier time slots $i_1, \ldots, i_{k-1}$. Therefore, the conditional expectation of the starting time of slot i_k can be bounded by

$$\begin{aligned} \mathrm{E}_{j \mapsto i_k}[s_{i_k}] &\leqslant \rho_{i_k} + \sum_{k'=1}^{k-1} \sum_{j' \neq j} \Pr_{j \mapsto i_k}[\bar{a}_{i'_{k'} j'} = 1] p_{ij'} \\ &= \rho_{i_k} + \sum_{k'=1}^{k-1} \sum_{j' \neq j} a_{i_{k'} j'} p_{ij'} \leqslant 2\rho_{i_k} \ ; \end{aligned}$$

the last inequality follows from constraints (21). We now turn to the expected completion time of job j. Using (17), the formula of total expectation yields

$$\begin{aligned} \mathrm{E}[C_j] &= \mathrm{E}\Big[\sum_{i,k} \bar{a}_{i_k j}\Big(s_{i_k} + p_{ij} + \sum_{j' \prec_i j} \bar{a}_{i_k j'} p_{ij'}\Big)\Big] \\ &= \sum_{i,k} \Pr[j \mapsto i_k]\Big(\mathrm{E}_{j \mapsto i_k}[s_{i_k}] + p_{ij} + \sum_{j' \prec_i j} \mathrm{E}_{j \mapsto i_k}[\bar{a}_{i_k j'}] p_{ij'}\Big) \\ &\leqslant 2\sum_{i,k} a_{i_k j}\Big(\rho_{i_k} + \frac{1 + a_{i_k j}}{2} p_{ij} + \sum_{j' \prec_i j} a_{i_k j'} p_{ij'}\Big) \ . \end{aligned}$$

This completes the proof. □

Since the value of an optimal solution to (CQP) is a lower bound on the value of an optimal schedule, Theorem 3.4 follows from Lemma 3.5 and linearity of expectations. As a result of our considerations we can state the following generalization of Corollary 2.7.

COROLLARY 3.6. *For instances of* $\mathrm{R} \,|\, r_{ij} \,|\, \sum w_j C_j$, *the value of an optimal schedule is within a factor* 2 *of the optimum solution to the relaxation* (CQP). *This bound is tight even for the case of identical parallel machines without release dates* $\mathrm{P} \,|\,|\, \sum w_j C_j$.

4. EXTENSIONS TO SCHEDULING WITH PREEMPTIONS

In this section we discuss the preemptive problem $\mathrm{R}\,|\,r_{ij},\, pmtn\,|\sum w_j C_j$ and generalizations to network scheduling. In contrast to the nonpreemptive setting, a job may now repeatedly be interrupted and continued later on another (or the same) machine. In the context of network scheduling, it is reasonable to assume that after a job has been interrupted on one machine it cannot be continued on another machine until a certain communication delay is elapsed that allows the job to travel through the network to its new machine.

The ideas and techniques presented in the last sections can be generalized to this setting. However, since we have to use a somewhat weaker relaxation in order to capture the possibility of preemptions, we only get a 3-approximation algorithm. This result can be improved to performance guarantee 2 in the absence of nontrivial release dates $\mathrm{R}\,|\,pmtn\,|\sum w_j C_j$ but with arbitrary communication delays.

Although the convex quadratic programming relaxation (CQP) allows to break a job into fractions and thus to 'preempt' it by choosing fractional values $a_{i_k j}$, it is not necessarily a relaxation of $\mathrm{R}\,|\,r_{ij},\, pmtn\,|\sum w_j C_j$. The following example shows that this is even true for the case without nontrivial release dates.

Example 4.1. *We are given two jobs and two machines with* $p_{1,1} = 2$, $p_{2,1} = \infty$, $w_1 = 2$, $p_{1,2} = 2$, $p_{2,2} = 4$, *and* $w_2 = 1$. *An optimal preemptive schedule processes the first job on the first machine for two time units starting at time* 0. *Meanwhile, the second job is processed on the second machine, preempted at time* 2, *and then scheduled on the first machine for one time unit. The value of this preemptive schedule is* 7. *It is an easy observation that this fractional assignment to machines also defines an optimal solution to* (CQP) $(a_{1,1} = 1,\ a_{2,1} = 0,\ a_{1,2} = a_{2,2} = \frac{1}{2})$. *However, the corresponding expression* (11) *for the completion time of the second job is* $\frac{13}{4}$ *instead of* 3 *and the value of the solution is equal to* 7.25.

For instances of $\mathrm{R}\,|\,r_{ij},\, pmtn\,|\sum w_j C_j$ and for a given preemptive schedule, we can always associate a feasible solution a of (CQP): let $a_{i_k j}$ the fraction of job j that is being processed on machine i within the interval $[\rho_{i_k}, \rho_{i_{k+1}})$. The following technical lemma is the key to a convex quadratic programming relaxation for the preemptive variants of the scheduling problems under consideration.

Lemma 4.2. *Consider an arbitrary preemptive schedule and let a denote the corresponding feasible solution to (CQP) as defined above. Then,*

$$\sum_j w_j C_j \;\geqslant\; \sum_j w_j \sum_{i,k} a_{i_k j}\Big(\rho_{i_k} + \frac{a_{i_k j}}{2} p_{ij} + \sum_{j' \prec_i j} a_{i_k j'} p_{ij'}\Big) \tag{23}$$

and

$$\sum_j w_j C_j \;\geqslant\; \sum_j w_j \sum_{i,k} a_{i_k j} p_{ij} \ . \tag{24}$$

Proof. The second bound is obvious since the expression on the right hand side is the weighted sum of processing times in the given preemptive schedule. Proving the first bound is more complicated.

Let $(J, \prec)$ denote a total order of the set of jobs according to nondecreasing completion times in the given preemptive schedule. For the following considerations we slightly relax the scheduling problem by allowing a job to be simultaneously processed on more than one machine. We show that the first bound is even true for feasible solutions to this relaxed problem.

We first show that for each time interval $[\rho_{i_k}, \rho_{i_{k+1}})$ on machine i and for each job j we can assume that the fraction of job j of size $a_{i_kj}p_{ij}$ is being processed without interruption within this time interval: Modify the given schedule by processing in each time slot the corresponding fractions of jobs one after another according to the order given by $\prec$; notice that the completion times of jobs and thus the value of the schedule cannot increase during this process. (It might however happen that a job is processed on more than one machine at a time after this modification.)

Before proving the bound (23), we prove the following modified version where we have replaced $\prec_i$ by $\prec$ on the right hand side:

$$\sum_j w_j C_j \;\geqslant\; \sum_j w_j \sum_{i,k} a_{i_kj}\Big(\rho_{i_k} + \frac{a_{i_kj}}{2}p_{ij} + \sum_{j'\prec j} a_{i_kj'}p_{ij'}\Big) \tag{25}$$

Notice that for each time slot i_k and each job j with $a_{i_kj} > 0$, the term in brackets on the right hand side is a lower bound on C_j since the fractions of jobs are scheduled according to $\prec$ in each time slot. The contribution of job j to the right hand side is w_j times a convex combination of the terms in brackets and is thus bounded by w_jC_j which proves the bound.

It remains to show that the right hand side of (25) is an upper bound on the right hand side of (23). Consider a time slot i_k and its contribution to the right hand sides. It suffices to show that

$$\sum_j a_{i_kj}w_j\Big(\rho_{i_k} + \frac{a_{i_kj}}{2}p_{ij} + \sum_{j'\prec_i j} a_{i_kj'}p_{ij'}\Big) \leqslant \sum_j a_{i_kj}w_j\Big(\rho_{i_k} + \frac{a_{i_kj}}{2}p_{ij} + \sum_{j'\prec j} a_{i_kj'}p_{ij'}\Big)$$

for each slot i_k. This is equivalent to

$$\sum_j a_{i_kj}w_j \sum_{j'\prec_i j} a_{i_kj'}p_{ij'} \;\leqslant\; \sum_j a_{i_kj}w_j \sum_{j'\prec j} a_{i_kj'}p_{ij'}$$

which follows by a simple exchange argument — in fact, this is exactly Smith's ratio rule applied to the fractions $a_{i_kj}p_{ij}$ of jobs of weight $a_{i_kj}w_j$. $\square$

As a result of Lemma 4.2 the following convex quadratic program, which we denote by (CQP'_p), is a relaxation of the preemptive problem $\mathrm{R}\,|\,r_{ij},\, pmtn\,|\sum w_jC_j$:

$$\begin{aligned}
\text{minimize}\quad & Z && \\
\text{subject to}\quad & \sum_{i,k} a_{i_kj} = 1 && \text{for all } j\\
& \sum_j a_{i_kj}p_{ij} \leqslant \rho_{i_{k+1}} - \rho_{i_k} && \text{for all } i,\ k.\\
& Z \geqslant b^Ta + \tfrac{1}{2}a^T\big(D + \mathrm{diag}(c)\big)a && (26)\\
& Z \geqslant c^Ta && (27)\\
& a \geqslant 0 &&
\end{aligned}$$

The vectors b, c and the matrix D are defined as above. Notice that the right hand sides of constraints (26) and (27) are equal to the right hand sides of (23) and (24), respectively.

In order to turn a solution to this relaxation into a feasible schedule we apply exactly the same randomized rounding heuristic as in the nonpreemptive case. In particular, we do not make use of the possibility to preempt jobs but compute a nonpreemptive schedule. Therefore, our results hold for the case of arbitrary communication delays.

LEMMA 4.3. *Given a feasible solution $\bar{a}$ to (CQP'_p), Algorithm* RANDOMIZED ROUNDING *computes a nonpreemptive schedule whose expected value is bounded from above by $3 \cdot Z_{CQP'_p}(\bar{a})$. In the absence of nontrivial release dates this bound can be improved to $2 \cdot Z_{CQP'_p}(\bar{a})$.*

PROOF. Notice that the objective function (22) of the program (CQP) is equal to the right hand side of (26) plus half of the right hand side of (27). This yields $2 \cdot Z_{CQP}(\bar{a}) \leqslant 3 \cdot Z_{CQP'_p}(\bar{a})$ and the first bound follows from the proof of Theorem 3.4.

To get the improved bound in the absence of release dates, notice that the objective function (4) of the quadratic program (QP) is equal to the sum of the right hand sides of (26) and (27). This yields $Z_{QP}(\bar{a}) \leqslant 2 \cdot Z_{CQP'_p}(\bar{a})$ and the result follows from Theorem 2.1. $\square$

COROLLARY 4.4. *For instances of $\mathrm{R} \,|\, r_{ij},\, pmtn \,|\, \sum w_j C_j$, the value of an optimal solution to the relaxation (CQP'_p) is within a factor 3 of the value of an optimal schedule. For instances of $\mathrm{R} \,|\, r_{ij},\, pmtn \,|\, \sum w_j C_j$, this bound can be improved to 2.*

As a result of Lemma 4.3 we get randomized approximation algorithms with expected performance guarantee $3+\varepsilon$ and $2+\varepsilon$ if we apply Algorithm RANDOMIZED ROUNDING to an almost optimal solution to (CQP'_p) which can be computed in polynomial time. Using the same idea as in the proof of Theorem 2.10, we can prove slightly better bounds for the derandomized version.

THEOREM 4.5. *Computing a near optimal solution to the relaxation (CQP'_p) and using Algorithm* DERANDOMIZED ROUNDING *to get a feasible schedule is a 3-approximation algorithm for the problem $\mathrm{R} \,|\, r_{ij},\, pmtn \,|\, \sum w_j C_j$ and a 2-approximation algorithm for $\mathrm{R} \,|\, pmtn \,|\, \sum w_j C_j$.*

Our considerations also yield bounds on the power of preemption. Since we can compute a nonpreemptive schedule whose value is bounded by 3 respectively 2 times the value of an optimal preemptive schedule, we have derived upper bounds on the ratios of optimal nonpreemptive to optimal preemptive schedules.

COROLLARY 4.6. *For instances of the problem $\mathrm{R} \,|\, r_{ij} \,|\, \sum w_j C_j$, the value of an optimal nonpreemptive schedule is at most a factor 3 above the value of an optimal preemptive schedule. In the absence of nontrivial release dates, this bound can be improved to 2.*

5. A SEMIDEFINITE RELAXATION FOR TWO MACHINES

In this section we consider the problem of scheduling two unrelated parallel machines in the absence of nontrivial release dates. We introduce a semidefinite programming relaxation for this problem and apply the random hyperplane technique in order to compute provably good feasible schedules. In contrast to the MAXCUT result of Goemans and Williamson, however, the analysis turns out to be much more complicated in our case since we are considering a minimization problem; while the very elegant analysis of Goemans and Williamson is based on simple lower bounds on certain probabilities related to the random hyperplane, we have to prove upper bounds which requires both a stronger semidefinite programming relaxation and a more sophisticated rounding technique.

We start with a reformulation of the integer quadratic program (IQP) from Section 2 in variables $x_j \in \{1,-1\}$, for $j \in J$. To keep the notation as simple as possible we assume that the two machines are numbered 0 and -1 and introduce corresponding variables $x_0, x_{-1} \in \{1,-1\}$ with $x_{-1} = -x_0$. The new variables have the following meaning: Job j is being assigned to machine 0 if $x_j = x_0$ and to machine -1 otherwise. Notice that we have introduced the variable x_{-1} only to keep notation simple; it could as well be replaced by $-x_0$. Observe that the assignment variables a_{ij} in (IQP) can be replaced by $(1 + x_i x_j)/2$ and the quadratic terms $a_{ij}a_{ik}$ by $(x_j x_k + x_i x_j + x_i x_k + 1)/4$.

We get a relaxation of (IQP) to a vector program (VP) if we replace the one-dimensional variables x_j with absolute value 1 by vectors $v_j \in \mathbb{R}^{n+1}$ of unit length:

$$
\begin{aligned}
\text{minimize} \quad & Z \\
\text{subject to} \quad & Z \geqslant \sum_j w_j \sum_{i=-1}^{0} \Big(\frac{1 + v_i v_j}{2} \cdot p_{ij} + \sum_{k \prec_i j} \frac{v_j v_k + v_i v_j + v_i v_k + 1}{4} \cdot p_{ik} \Big) \qquad (28) \\
& v_{-1} v_0 = -1 \\
& v_j v_j = 1 \qquad \text{for } j \in J \cup \{0,-1\}
\end{aligned}
$$

Here $v_j v_k$ denotes the scalar product of the vectors v_j and v_k. It is well known that such a program is equivalent to a semidefinite program in variables corresponding to the scalar products (see, e.g., [Goemans and Williamson 1995]). This program can be strengthened by adding the constraints

$$
v_j v_k + v_i v_j + v_i v_k + 1 \;\geqslant\; 0 \qquad \text{for } i \in \{0,-1\} \text{ and } j,k \in J. \tag{29}
$$

Observe that those constraints are always fulfilled for $\{1,-1\}$-variables. Constraints of the same form have been used by Feige and Goemans [Feige and Goemans 1995] to improve some of the approximation results of Goemans and Williamson [Goemans and Williamson 1995].

It is one of the key insights of this section that the vector program can be further strengthened with the quadratic cut (12) from the convex quadratic program (CQP') in Section 2. For a feasible solution v to (VP) we denote by $a = a(v)$ the corresponding solution to (CQP), i.e.,

$$
a_{ij} = \frac{1 + v_i v_j}{2} \qquad \text{for } i \in \{0,-1\} \text{ and } j \in J, \tag{30}
$$

and add the constraint

$$Z \;\geqslant\; \tfrac{1}{2}c^T a + \tfrac{1}{2}a^T\big(D + \mathrm{diag}(c)\big)a \tag{31}$$

$$= \sum_j w_j \sum_{i=-1}^{0} a_{ij} \cdot \Big(\frac{1 + a_{ij}}{2} p_{ij} + \sum_{k \prec_i j} a_{ik} \cdot p_{ik}\Big) \tag{32}$$

to the vector program (VP). The resulting program with the additional constraints (29), (30) and (31) is denoted by (SDP). Since the right hand side of constraint (31) is convex quadratic, (SDP) can be interpreted as a semidefinite program in variables corresponding to the scalar products $v_j v_k$. For a feasible solution to (SDP) we consider the random hyperplane rounding that was introduced by Goemans and Williamson:

Algorithm Random Hyperplane

1) Draw a random vector r uniformly distributed from the unit-sphere of $\mathbb{R}^{n+1}$.
2) For each job j, assign j to the machine i with $\mathrm{sgn}(v_j r) = \mathrm{sgn}(v_i r)$.

In the second step ties can be broken arbitrarily; they occur with probability zero. The random vector r can be interpreted as the normal vector of a random hyperplane through the origin which partitions the set of vectors v_j, $j \in J$, and therefore the jobs into two subsets. In contrast to Algorithm Randomized Rounding jobs are no longer assigned independently to the machines, but the hyperplane induces a correlation between the random decisions for different jobs.

To analyze (SDP) and the schedule computed by Algorithm Random Hyperplane we need the following lemma which is a restatement of [Goemans and Williamson 1995, Lemma 3.2 and Lemma 7.3.1]. For given vectors v_j, v_k, $j, k \in J \cup \{0, -1\}$, we denote the enclosed angle by α_{jk}.

Lemma 5.1. *For $j, k \in J$, $i \in \{0, -1\}$, and for given unit vectors v_i, v_j, v_k, Algorithm* Random Hyperplane *yields the following probabilities:*

a) $\Pr[j \mapsto i] = 1 - \frac{\alpha_{ij}}{\pi}$.
b) $\Pr[j, k \mapsto i] = 1 - \frac{\alpha_{jk} + \alpha_{ij} + \alpha_{ik}}{2\pi}$.

As a result of Lemma 2.2 and Lemma 5.1 the expected value of the completion time of job j in the schedule computed by Algorithm Random Hyperplane is given by

$$\mathrm{E}[C_j] = \sum_{i=-1}^{0} \bigg(\Big(1 - \frac{\alpha_{ij}}{\pi}\Big) \cdot p_{ij} + \sum_{k \prec_i j} \Big(1 - \frac{\alpha_{jk} + \alpha_{ij} + \alpha_{ik}}{2\pi}\Big) \cdot p_{ik} \bigg) . \tag{33}$$

We want to bound the expected value of the schedule computed by Algorithm Random Hyperplane in terms of the feasible solution to (SDP) we started with. In view of (33) and the lower bounds on $Z_{SDP}(v)$ given in (28) and (32), we try to bound the probabilities given in Lemma 5.1 in terms of the corresponding coefficients in (28) and (32).

LEMMA 5.2. *Let v and $a = a(v)$ be a feasible solution to (SDP) with value Z and consider a random assignment of jobs to machines satisfying*

$$\Pr[j \mapsto i] \;\leqslant\; \frac{\rho_1}{2}\left(\frac{1+v_iv_j}{2} + \frac{1+a_{ij}}{2}\cdot a_{ij}\right) \tag{34}$$

and

$$\Pr[j,k \mapsto i] \;\leqslant\; \frac{\rho_2}{2}\left(\frac{v_jv_k + v_iv_j + v_iv_k + 1}{4} + a_{ij}a_{ik}\right) \tag{35}$$

for $i \in \{0,-1\}$, $j,k \in J$, and for certain parameters $1 \leqslant \rho_1 \leqslant \rho_2$. Then the expected value of the computed schedule is bounded from above by

$$\mathrm{E}\Big[\sum_j w_jC_j\Big] \;\leqslant\; \rho_1\cdot\frac{3\,c^T a(v)}{4} + \rho_2\cdot\Big(Z - \frac{3\,c^T a(v)}{4}\Big) \;\leqslant\; \rho_2\cdot Z \;\;.$$

PROOF. To simplify notation we denote the vector $a(v)$ by a and its entries by a_{ij}. The main idea of the proof is to bound the expected value of the schedule by a new lower bound which is the average of the two bounds on Z given in (28) and (32). Using linearity of expectations we can bound the expected value of the schedule by

$$\mathrm{E}\Big[\sum_j w_jC_j\Big] \;=\; \sum_j w_j \sum_{i=-1}^{0}\Big(\Pr(j\mapsto i)\cdot p_{ij} + \sum_{k\prec_i j}\Pr(j,k\mapsto i)\cdot p_{ik}\Big)$$

by Lemma 2.2,

$$\begin{aligned}
&\leqslant\; \sum_j w_j \sum_{i=-1}^{0}\Big(\frac{\rho_1}{2}\cdot\frac{1+v_iv_j}{2}\cdot p_{ij} + \sum_{k\prec_i j}\frac{\rho_2}{2}\cdot\frac{v_jv_k+v_iv_j+v_iv_k+1}{4}\cdot p_{ik}\Big)\\
&\quad+\sum_j w_j \sum_{i=-1}^{0}\Big(\frac{\rho_1}{2}\cdot\frac{1+a_{ij}}{2}\cdot a_{ij}\cdot p_{ij} + \sum_{k\prec_i j}\frac{\rho_2}{2}\cdot a_{ij}\cdot a_{ik}\cdot p_{ik}\Big)
\end{aligned}$$

by (34) and (35). Plugging in (28) and (31) and making use of $\rho_1 \leqslant \rho_2$ we can bound the last expression by

$$\begin{aligned}
&\leqslant\; \frac{\rho_1}{2}\cdot c^T a + \frac{\rho_2}{2}\cdot(Z - c^T a) + \frac{\rho_1}{2}\cdot\frac{c^T a}{2} + \frac{\rho_2}{2}\cdot\Big(Z - \frac{c^T a}{2}\Big)\\
&\leqslant\; \rho_1\cdot\frac{3\,c^T a}{4} + \rho_2\cdot\Big(Z - \frac{3\,c^T a}{4}\Big)\;\;.
\end{aligned}$$

The second bound in the lemma follows from $\rho_1 \leqslant \rho_2$. □

Inspired by the work of Feige and Goemans [Feige and Goemans 1995] we give a rounding scheme that fulfills the conditions described in Lemma 5.2 for $\rho_1 = 1.1847$ and $\rho_2 = 1.3388$. We apply Algorithm RANDOM HYPERPLANE to a set of modified vectors u_j, $j \in J$, which are constructed from the vectors v_j by taking advantage of the special role of v_0 and v_{-1}. For each job $j \in J$ the vectors v_0, v_j, and u_j are linearly dependent, i.e., u_j is coplanar with v_0 and v_j. Moreover, u_j lies on the same side of the hyperplane orthogonal to v_0 as v_j and its distance to this

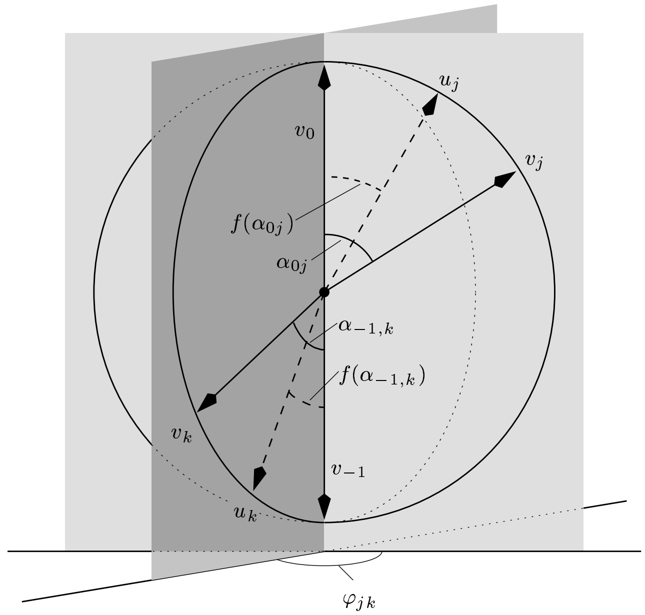

Fig. 2. Modification according to the function f.

hyperplane is increased compared to v_j. In other words, u_j is attained by moving v_j towards the nearer of the two points v_0 and v_{-1}, see Figure 2.

We describe this mapping of v_j to u_j by a function $f : [0, \pi] \to [0, \pi]$ where $f(\alpha_{ij})$ equals the angle formed by u_j and v_i for $i \in \{0, -1\}$. In particular, f has the property that $f(\pi - \theta) = \pi - f(\theta)$ such that both machines are treated in the same manner. In order to compute the probability $\Pr[j, k \mapsto i]$ for Algorithm Random Hyperplane based on the modified vectors u_j and u_k, we need to know the angle between u_j and u_k for two jobs $j, k \in J$. This angle is implicitly given by the cosine rule for spherical triangles. We denote the angle between the two planes defined by (v_0, v_j) and (v_0, v_k) by φ_{jk} (see Figure 2):

$$\begin{aligned}\cos(\alpha_{jk}) &= \cos(\alpha_{0j})\cos(\alpha_{0k}) + \cos(\varphi_{jk})\sin(\alpha_{0j})\sin(\alpha_{0k})\\ u_j u_k &= \cos\big(f(\alpha_{0j})\big)\cos\big(f(\alpha_{0k})\big) + \cos(\varphi_{jk})\sin\big(f(\alpha_{0j})\big)\sin\big(f(\alpha_{0k})\big)\ .\end{aligned}$$

The last expression is equal to the cosine of the angle between u_j and u_k; the first expression can be used to determine φ_{jk} for given vectors v_0, v_j, v_k.

If we use the function $f_1(\theta) := \frac{\pi}{2}\big(1 - \cos(\theta)\big)$ proposed by Feige and Goemans (see Figure 3), we get

$$\Pr[j \mapsto i] = \frac{1 + \cos(\alpha_{ij})}{2} = \frac{1 + v_i v_j}{2} = a_{ij}$$

for each job j. In other words, the probability that a job is assigned to a machine is equal to the corresponding coefficient in (28). On the other hand, the function f_1 does not yield a possibility to bound the probabilities $\Pr[j, k \mapsto i]$ for $j, k \in J$ in terms of the corresponding coefficients in (28) and (31). Consider the constellation of vectors given by $\varphi_{jk} = \frac{\pi}{2}$ and $\alpha_{0j} = \alpha_{0k}$. If α_{0j} and α_{0k} simultaneously go to

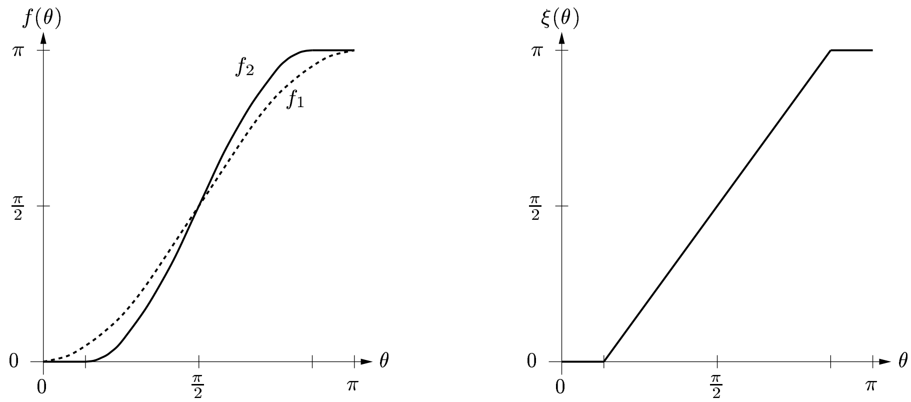

Fig. 3. Description of the functions f_1 and f_2.

π then $\Pr[j, k \mapsto i]$ as well as the right hand side of (35) go to zero; however, the order of convergence is higher for the right hand side.

Therefore we use a different function f_2 defined by $f_2(\theta) = f_1\big(\xi(\theta)\big)$ where $\xi(\theta)$ is given by $\xi(\theta) = \min\big\{\pi, \max\{0, \frac{\pi}{2} + 1.3662 \cdot (\theta - \frac{\pi}{2})\}\big\}$ (see Figure 3). We have tested numerically that the conditions in Lemma 5.2 are fulfilled for $\rho_1 = 1.1847$ and $\rho_2 = 1.3388$ in this case. As proposed by Feige and Goemans this was done by discretizing the set of all possible angles between three vectors and testing for each triple the validity of the bounds for the given parameters ρ_1 and ρ_2. The parameter ρ_1 is nearly tight for the angle $\alpha_{0j} = 0.3864 \cdot \pi$, the same holds for ρ_2 and the angles $\alpha_{0j} = \alpha_{0k} = 0.751 \cdot \pi$ and $\varphi_{jk} = 0.554 \cdot \pi$.

CONJECTURE 5.3. *The rounding scheme induced by the function f_2 fulfills the conditions in Lemma 5.2 for $\rho_1 = 1.1847$ and $\rho_2 = 1.3388$.*

We should mention that both constraints (29) and (31) are crucial for our analysis. In the absence of one of these constraints one can construct constellations of vectors such that no constant worst case bounds ρ_1 and ρ_2 exist for our analysis.

We strongly believe that there exists a function similar to f_2 which yields an improved bound of $\rho_2 = \frac{4}{3}$. On the other hand we can show that this value is best possible for our kind of analysis. Consider the constellation $\alpha_{0j} = \alpha_{0k} = \frac{\pi}{2}$ and $\alpha_{jk} = 0$. The symmetry of f yields $f(\frac{\pi}{2}) = \frac{\pi}{2}$ such that $u_j = u_k = v_j = v_k$. Therefore $\Pr[j, k \mapsto 0] = \frac{1}{2}$ and the right hand side of the corresponding inequality in Lemma 5.2 is equal to $\frac{3}{8}\rho_2$. We have also tried to bound the probabilities by a different convex combination of the corresponding coefficients in (28) and (31) rather than using their average as in Lemma 5.2; but this did not lead to any improvement.

Unfortunately, it is far from being obvious how to give a reasonably simple proof for Conjecture 5.3. Similar problems have also been encountered by others who used variants of the random hyperplane technique that are based on a modification of the original vector solution, see, e.g., [Feige and Goemans 1995; Zwick 1999]. Of course, one could give a proof by partitioning the space of possible configurations of the three vectors v_i, v_j, v_k into small areas and prove the conjecture analytically for each area. However, we think that it is not worth to spend too much effort on this task. On the one hand, the question about the approximability of the scheduling problem under consideration has recently been settled (there is a PTAS even for the more general problem $\mathrm{R}m \,|\, r_j \,|\, \sum w_j C_j$, see [Afrati et al. 1999]); on the other hand, our computational 'proof' seems to give sufficient indication of the quality of the underlying semidefinite programming relaxation (SDP).

THEOREM 5.4. *If Conjecture 5.3 is valid, computing an almost optimal solution to* (SDP)*, modifying it according to* f_2*, and using Algorithm* RANDOM HYPERPLANE *to construct a feasible schedule yields a randomized approximation algorithm with expected performance guarantee* 1.3388*.*

It is shown in [Mahajan and Ramesh 1999] that Algorithm RANDOM HYPERPLANE can be derandomized. We get a deterministic version of our approximation algorithm if we make use of the derandomized version of Algorithm RANDOM HYPERPLANE.

We can also apply Algorithm RANDOMIZED ROUNDING to turn a feasible solution $a = a(v)$ to (SDP) into a provably good schedule. Although the worst case ratio of this algorithm is worse than the performance ratio of the rounding scheme based on Algorithm RANDOM HYPERPLANE, a combination of the two rounding techniques leads to a further improvement in the performance guarantee.

THEOREM 5.5. *Under the assumption of Conjecture 5.3, for an almost optimal solution* $a(v)$ *to* (SDP)*, either Algorithm* RANDOMIZED ROUNDING *or Algorithm* RANDOM HYPERPLANE *(together with* f_2*) produces a schedule whose expected value is bounded by* $1.2752 \cdot Z^*$*.*

PROOF. It follows from Corollary 2.11 that the expected value of the schedule computed by RANDOMIZED ROUNDING is within a factor $1 + \frac{x}{2}$ of the value of the solution $a(v)$, where x denotes the ratio of $c^T a(v)$ to the value of the solution $a(v)$.

On the other hand, by Lemma 5.2 and Conjecture 5.3, the value of the schedule computed by RANDOM HYPERPLANE is within a factor

$$1.1847 \cdot \tfrac{3}{4} \cdot x + 1.3388 \cdot (1 - \tfrac{3}{4} x)$$

of the value of the solution $a(v)$.

Thus, the result follows since the maximum of the function

$$x \;\mapsto\; \min\bigl\{1 + \tfrac{x}{2}, 1.1847 \cdot \tfrac{3}{4} \cdot x + 1.3388 \cdot (1 - \tfrac{3}{4} x)\bigr\}$$

over the interval $[0, 1]$ is strictly less than 1.2752, see Figure 4. □

Observations of this type have already been used in other contexts to get improved approximation results. Theorem 5.5 also implies that (SDP) is a 1.2752-relaxation for $\mathrm{R2} \,|\,|\, \sum w_j C_j$.

Up to now, the combination of semidefinite relaxations like the one we are discussing here and the rounding technique of Algorithm RANDOM HYPERPLANE has only proved useful for approximations in the context of maximization problems, see, e. g., [Goemans and Williamson 1995; Feige and Goemans 1995; Frieze and Jerrum 1997]. In contrast to our considerations, in the analysis of these results one needs a good lower bound on the probabilities for the assignments in Algorithm RANDOM HYPERPLANE. However, it seems to be much harder to attain good upper bounds. Our main contribution to this problem is the additional quadratic cut (31). We

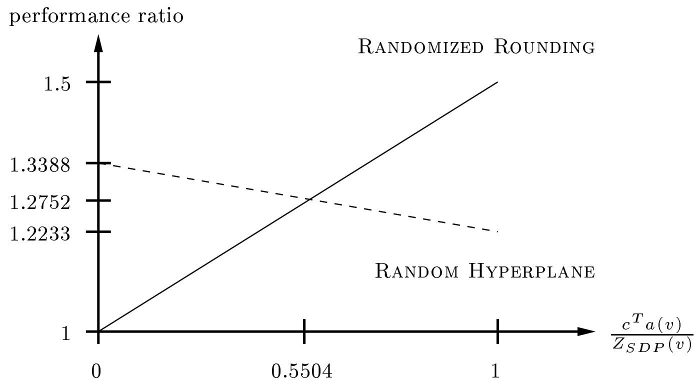

Fig. 4. Comparison of Randomized Rounding and Random Hyperplane.

hope that this approach will also prove useful for other problems in combinatorial optimization.

6. MAXCUT ALGORITHMS FOR IDENTICAL PARALLEL MACHINES

In this section we provide an approximation preserving reduction from the identical parallel machine scheduling problem $\mathrm{P}m \,|\, |\sum w_j C_j$ to the graph partitioning problem Max$k$Cut for $k = m$. In particular we show how any constant factor approximation algorithm for Max$k$Cut translates into an approximation algorithm for $\mathrm{P}m \,|\, |\sum w_j C_j$ with constant performance guarantee. This sheds new light on the random hyperplane algorithm from the last section for the special case of two identical parallel machines.

We associate with each instance of $\mathrm{P} \,|\, |\sum w_j C_j$ a complete undirected graph G_J on the set of vertices J together with weights on the set of edges E_J given by $c(jk) = w_j p_k$ for $j, k \in J$, $k \prec j$. Each partition of the set of vertices J of G_J into m subsets $J_1, \ldots, J_m$ can be interpreted as a machine assignment and corresponds to a feasible schedule. Moreover, the value of a schedule can be interpreted as the weight of the set E_{sch} formed by those edges in E_J with both endpoints in the same subset plus the constant term $\sum_j w_j p_j$. The remaining edges in $E_{\mathrm{cut}} := E_J \setminus E_{\mathrm{sch}}$ are contained in the induced m-cut. In particular we get

$$c(E_J) \;=\; \sum_j w_j C_j - \sum_j w_j p_j + c(E_{\mathrm{cut}}) \tag{36}$$

where C_j denotes the completion time of job j in the schedule corresponding to the partition of J. Since $\sum_j w_j p_j$ and $c(E_J)$ are constant, minimizing the average weighted completion time $\sum_j w_j C_j$ of the schedule is equivalent to maximizing the value $c(E_{\mathrm{cut}})$ of the induced m-cut. This reduction of $\mathrm{P}m \,|\, |\sum w_j C_j$ to Max$m$Cut is approximation preserving:

Theorem 6.1. *For any $\rho \leqslant 1$, a ρ-approximation algorithm for* Max$m$Cut *translates into an approximation algorithm for $\mathrm{P}m \,|\, |\sum w_j C_j$ with performance guarantee $\rho + m \cdot (1 - \rho)$.*

Proof. We use the lower bound Z^*_{CQP} on the value of an optimal schedule to get an upper bound on the weight Z^*_{cut} of a maximum m-cut. Lemma 2.5 yields

$$Z^* \;\geqslant\; Z^*_{CQP} \;=\; \frac{1}{m}\cdot c(E_J) + \Big(\frac{1}{2}+\frac{1}{2m}\Big)\sum_j w_j p_j \tag{37}$$

such that

$$Z^*_{\text{cut}} \;\leqslant\; \frac{m-1}{m}\cdot c(E_J) + \Big(\frac{1}{2}-\frac{1}{2m}\Big)\sum_j w_j p_j \tag{38}$$

by (36) and (37). Any m-cut in G_J whose weight is at least $\rho\cdot Z^*_{\text{cut}}$ therefore yields a schedule whose value can be bounded as follows:

$$\begin{aligned}\sum_j w_j C_j \;&\leqslant\; Z^* + (1-\rho)\cdot Z^*_{\text{cut}} && \text{by (36)}\\ &\leqslant\; Z^* + (1-\rho)\cdot(m-1)\cdot Z^* && \text{by (38), (37)}\\ &=\; \big(\rho + m\cdot(1-\rho)\big)\cdot Z^*\;.\end{aligned}$$

This completes the proof. □

While the problem $P|\,|\sum w_j C_j$ has a polynomial time approximation scheme [Skutella and Woeginger 2000] (even a fully polynomial time approximation scheme when the number of machines is constant [Sahni 1976]), it is shown in [Kann et al. 1997] that MAXmCUT cannot be approximated within $\rho < 1+\frac{1}{34m}$, unless P=NP. The best currently known approximation algorithms for MAXmCUT have performance ratio $1-\frac{1}{m}+o(\frac{1}{m})$ which yields $(2-\frac{1}{m})$-approximation algorithms for $P|\,|\sum_j w_j C_j$ by Theorem 6.1. It is interesting to mention and easy to see that assigning each vertex randomly to one of the m subsets is an approximation algorithm with performance guarantee $1-\frac{1}{m}$ for MAXmCUT. Moreover, this algorithm coincides with Algorithm RANDOMIZED ROUNDING based on the optimal solution to (CQP) given in Lemma 2.5 and therefore achieves performance ratio $\frac{3}{2}-\frac{1}{2m}$ for $P|\,|\sum_j w_j C_j$ by Theorem 2.6 b).

If we consider the problem for the case $m=2$ we get performance guarantee 1.122 if we use the 0.878-approximation algorithm for MAXCUT by Goemans and Williamson. This result beats both the $\frac{5}{4}$-approximation in Theorem 2.6 and the 1.2752-approximation in Theorem 5.5. Notice that for the case of two identical parallel machines, (SDP) is a strengthened version of the semidefinite programming relaxation for the corresponding MAXCUT problem considered in [Goemans and Williamson 1995]. This leads to the following result.

COROLLARY 6.2. *Computing an almost optimal solution to* (SDP) *and applying Algorithm* RANDOM HYPERPLANE *to get a feasible schedule is a* 1.122*-approximation for* $P2|\,|\sum w_j C_j$.

This result has been further improved by Goemans (personal communication, September 1998) to performance guarantee 1.073 through a more sophisticated rounding technique based on the standard MAXCUT relaxation. Before choosing the random hyperplane, he modified the given solution to the vector program in the following way: Consider the positive semidefinite matrix consisting of all scalar products of vectors and take a convex combination with the identity matrix. The resulting matrix is still positive semidefinite and defines again a set of vectors. Notice that the identity matrix corresponds to a set of pairwise orthogonal vectors which are partitioned uniformly at random and independently by a random hyperplane. This algorithm coincides with the algorithms discussed in Theorem 2.6 b). The algorithm proposed by Goemans can therefore be interpreted as a combination of the original random hyperplane algorithm and this random assignment algorithm.

This approach recently also proved useful in other contexts. Zwick [Zwick 1999] used the same idea to give a slightly improved variant of the MaxCut algorithm of Goemans and Williamson; Ye [Ye 1999] applied this technique to improve the result of Frieze and Jerrum [Frieze and Jerrum 1997] for the MaxBisection problem.

7. NONAPPROXIMABILITY RESULTS

It has been shown by Hoogeveen, Schuurman, and Woeginger [Hoogeveen et al. 1998] that the problems $R|\,|\sum w_j C_j$ and $R\,|\,r_j\,|\sum C_j$ are MaxSNP-hard and therefore do not have a polynomial time approximation scheme, unless P=NP. Hoogeveen et al. construct an approximation preserving reduction from a MaxSNP-hard variant of a 3-dimensional matching problem to these scheduling problems. We provide an alternative and slightly simpler proof in this section by constructing reductions from the variant of Max3Sat where each variable occurs at most three times in the clauses. This variant is usually denoted by 3-Occurrence Max3Sat and is known to be MaxSNP-hard, see, e. g., [Ausiello et al. 1999, Corollary 8.15].

We first present an approximation preserving reduction from this satisfiability problem to $R\,|\,r_j\,|\sum C_j$. Given an instance I of 3-Occurrence Max3Sat with n variables and m clauses, we construct a corresponding scheduling instance $R(I)$ with $n+m$ jobs and $2n$ machines in the following way. For each variable we introduce one v-job (where 'v' stands for 'variable') and two machines corresponding to the variable — a 'true machine' and a 'false machine'; the v-job is released at time 0, its processing time is 4 on its two machines and infinity on the machines of other variables. Therefore, each feasible schedule S corresponds to an assignment $\textsc{sat}(S)$ of the values true and false to the variables of the given satisfiability instance: a variable is set to true if and only if the corresponding v-job is being processed on its 'true machine' in the schedule S.

Moreover, we introduce one c-job for each clause ('c' stands for 'clause'). The release date of a c-job is 3 and its processing time is 0, but it can only be processed on machines corresponding to variables of its clause: if a variable occurs in the nonnegated form in the clause, the c-job can be processed on the corresponding 'false machine'; if the variable occurs in the negated form in the clause, the c-job can be processed on the 'true machine'. The underlying intuition of this reduction is that for a given machine assignment of the v-jobs (i. e., a given truth assignment to the variables), a c-job can be started at time 3 without getting in conflict with a v-job if and only if the clause is satisfied. This intuition leads to the following lemma:

Lemma 7.1. *Let I be an instance of* 3-Occurrence Max3Sat *and $R(I)$ the corresponding instance of the scheduling problem constructed above.*

a) *Given a schedule S with value $\textsc{val}(S)$ for $R(I)$, the number of clauses satisfied by the corresponding truth assignment $\textsc{sat}(S)$ of I is $\#\big(\textsc{sat}(S)\big) \geqslant 4n+4m-\textsc{val}(S)$.*

b) $\mathrm{OPT}_{\mathrm{SCH}}\big(R(I)\big) = 4n+4m-\mathrm{OPT}_{\mathrm{SAT}}(I)$*, where* $\mathrm{OPT}_{\mathrm{SCH}}\big(R(I)\big)$ *and* $\mathrm{OPT}_{\mathrm{SAT}}(I)$ *denote the values of optimal solutions to* $R(I)$ *and* I*, respectively.*

PROOF. In order to prove part a), we modify the given schedule S as follows: We start each v-job at time 0 on the machine it has been assigned to in S, possibly delaying some c-jobs until time 4. Since each variable occurs in at most 3 clauses, there are at most 3 c-jobs on each machine; thus, we did not increase the value of the schedule. Now we start each c-job as early as possible, i. e., at time 3 if the corresponding clause is satisfied by the truth assignment $\mathrm{SAT}(S)$, and at time 4 otherwise. We denote the resulting schedule by S' and get

$$\mathrm{VAL}(S) \;\geqslant\; \mathrm{VAL}(S') \;=\; 4n + 4m - \#\big(\mathrm{SAT}(S)\big)$$

which yields a).

On the other hand, using the same ideas as above, an optimal truth assignment for I leads to a schedule of value $4n + 4m - \mathrm{OPT}_{\mathrm{SAT}}(I)$. Together with a) this completes the proof of b). ☐

We can now give a new proof for the following theorem of Hoogeveen, Schuurman, and Woeginger.

THEOREM 7.2 ([HOOGEVEEN ET AL. 1998]). *The parallel machine scheduling problem* $\mathrm{R}\,|\,r_j\,|\sum C_j$ *is* MAXSNP*-hard.*

PROOF. We show that the reduction given above is an L-reduction; for the definition of L-reductions we refer the reader to [Papadimitriou and Yannakakis 1991; Papadimitriou 1994]. Notice that the transformation R that maps an instance I of 3-OCCURRENCE MAX3SAT to an instance $R(I)$ of $\mathrm{R}\,|\,r_j\,|\sum C_j$ can be implemented to run in polynomial time. The same holds for the transformation SAT that maps a given schedule S to a truth assignment $\mathrm{SAT}(S)$.

Since the number of satisfied clauses in an optimal truth assignment is at least $m/2$ and since $n \leqslant 3m$, Lemma 7.1 b) yields

$$\mathrm{OPT}_{\mathrm{SCH}}\big(R(I)\big) \;\leqslant\; 12m + 4m \;\leqslant\; 32\mathrm{OPT}_{\mathrm{SAT}}(I)$$

and the first condition for an L-reduction is thus fulfilled. The second condition follows directly from Lemma 7.1 since for any schedule S we get

$$\mathrm{OPT}_{\mathrm{SAT}}(I) - \#\big(\mathrm{SAT}(S)\big) \;\leqslant\; \mathrm{VAL}(S) - \mathrm{OPT}_{\mathrm{SCH}}\big(R(I)\big)\;.$$

This completes the proof. ☐

For the problem $\mathrm{R}\,|\;|\sum w_j C_j$ we use a similar reduction. Each v-job has processing time and weight 1 and can again only be scheduled on its 'true' and 'false machine'. A c-job can be processed on the same machines as described above; its processing time and weight is set to a small positive constant ε. Thus, by Smith's ratio rule, any sequence of jobs is optimal on a machine. A similar proof as above shows that this reduction is in fact an L-reduction.

8. CONCLUSION

We have presented convex quadratic programming relaxations of strongly polynomial size which lead to simple and easy to analyze approximation algorithms for preemptive and nonpreemptive network scheduling. Although our approach and the presented results might be at first sight of mainly theoretical interest, we hope that nonlinear relaxations like the ones we discuss in this paper will also prove useful in solving real world scheduling problems in the near future. With the development of better algorithms that solve convex quadratic programs more efficiently in practice, the results obtained by using such relaxations might become comparable or even better than those based on linear programming relaxations with a huge number of time-indexed variables and constraints.

Precedence constraints between jobs play a particularly important role in most real world scheduling problems. Therefore it would be both of theoretical and of practical interest to incorporate those constraints into our convex quadratic programming relaxations.

As mentioned in the last section, the problems $\mathrm{R}\,|\,r_j\,|\sum C_j$ and $\mathrm{R}\,|\;|\sum w_jC_j$ cannot be approximated in polynomial time within arbitrarily good precision, unless P=NP. It is an interesting open problem to close the gap between this lower bound and the approximation results presented in this paper. For example, one approach is trying to obtain improved performance ratios by strengthening the convex quadratic relaxations discussed in this paper. Notice, however, that for the problem $\mathrm{R}\,|\;|\sum w_jC_j$ the convex hull of all assignments of jobs to machines is exactly given by constraints (5) and (6). Thus, in contrast to most integer linear programming problems, the task is not to find additional cuts bounding the convex hull of feasible solutions; to get improved relaxations based on the techniques described in this paper, we need to strengthen the convex objective function by adding constraints like (13). On the other hand, new ideas and techniques are needed in order to prove stronger lower bounds; since the approximability of many classical machine scheduling problems with min-sum objective has recently been settled (see, e. g., [Afrati et al. 1999]), this is one of the main challenges remaining for the scheduling problems under consideration.

For further open problems in the area of machine scheduling we refer to the very interesting recent paper [Schuurman and Woeginger 1999].

ACKNOWLEDGMENTS

The author is grateful to Michel Goemans, Andreas Schulz, and an anonymous referee for helpful comments.

REFERENCES

Afrati, F., Bampis, E., Chekuri, C., Karger, D., Kenyon, C., Khanna, S., Milis, I., Queyranne, M., Skutella, M., Stein, C., and Sviridenko, M. 1999. Approximation schemes for minimizing average weighted completion time with release dates. In *Proceedings of the 40th Annual IEEE Symposium on Foundations of Computer Science* (1999), pp. 32 – 43.

Ausiello, G., Crescenzi, P., Gambosi, G., Kann, V., Marchetti-Spaccamela, A., and Protasi, M. 1999. *Complexity and Approximation*. Springer, Berlin.

Awerbuch, B., Kutten, S., and Peleg, D. 1992. Competitive distributed job scheduling. In *Proceedings of the 24th Annual ACM Symposium on the Theory of Computing* (1992), pp. 571 – 581.

Bertsimas, D., Teo, C., and Vohra, R. 1996. On dependent randomized rounding algorithms. In W. H. Cunningham, S. T. McCormick, and M. Queyranne Eds., *Integer*

Programming and Combinatorial Optimization, Volume 1084 of *Lecture Notes in Computer Science*, pp. 330 – 344. Springer, Berlin.

CHAKRABARTI, S., PHILLIPS, C., SCHULZ, A. S., SHMOYS, D. B., STEIN, C., AND WEIN, J. 1996. Improved scheduling algorithms for minsum criteria. In F. MEYER AUF DER HEIDE AND B. MONIEN Eds., *Automata, Languages and Programming*, Volume 1099 of *Lecture Notes in Computer Science*, pp. 646 – 657. Springer, Berlin.

CHOR, B. AND SUDAN, M. 1998. A geometric approach to betweenness. *SIAM Journal on Discrete Mathematics 11*, 511 – 523.

CHUDAK, F. A. 1999. A min-sum 3/2-approximation algorithm for scheduling unrelated parallel machines. *Journal of Scheduling 2*, 73 – 77.

CHUDAK, F. A. AND SHMOYS, D. B. 1999. Approximation algorithms for precedence-constrained scheduling problems on parallel machines that run at different speeds. *Journal of Algorithms 30*, 323 – 343.

CHUNG, S. J. AND MURTY, K. G. 1981. Polynomially bounded ellipsoid algorithms for convex quadratic programming. In O. L. MANGASARIAN, R. R. MEYER, AND S. M. ROBINSON Eds., *Nonlinear Programming 4*, pp. 439 – 485. Academic Press.

DENG, X., LIU, H., LONG, J., AND XIAO, B. 1990. Deterministic load balancing in computer networks. In *Proceedings of the 2nd Annual IEEE Symposium on Parallel and Distributed Processing* (1990), pp. 50 – 57.

DYER, M. E. AND WOLSEY, L. A. 1990. Formulating the single machine sequencing problem with release dates as a mixed integer program. *Discrete Applied Mathematics 26*, 255 – 270.

ENGELS, D. W., KARGER, D. R., KOLLIOPOULOS, S. G., SENGUPTA, S., UMA, R. N., AND WEIN, J. 1998. Techniques for scheduling with rejection. In G. BILARDI, G. F. ITALIANO, A. PIETRACAPRINA, AND G. PUCCI Eds., *Proceedings of the 6th Annual European Symposium on Algorithms*, Volume 1461 of *Lecture Notes in Computer Science*, pp. 490 – 501. Springer, Berlin.

FEIGE, U. AND GOEMANS, M. X. 1995. Approximating the value of two prover proof systems, with applications to MAX 2SAT and MAX DICUT. In *Proceedings of the Third Israel Symposium on Theory of Computing and Systems* (1995), pp. 182 – 189.

FRIEZE, A. AND JERRUM, M. 1997. Improved approximation algorithms for MAX k-CUT and MAX BISECTION. *Algorithmica 18*, 67 – 81.

GOEMANS, M. X. 1997a. Improved approximation algorithms for scheduling with release dates. In *Proceedings of the 8th Annual ACM–SIAM Symposium on Discrete Algorithms* (1997), pp. 591 – 598.

GOEMANS, M. X. 1997b. Semidefinite programming in combinatorial optimization. *Mathematical Programming 79*, 143 – 161.

GOEMANS, M. X., QUEYRANNE, M., SCHULZ, A. S., SKUTELLA, M., AND WANG, Y. 1999. Single machine scheduling with release dates. Submitted.

GOEMANS, M. X., WEIN, J., AND WILLIAMSON, D. P. 2000. A 1.47-approximation algorithm for a preemptive single-machine scheduling problem. *Operations Research Letters 26*, 149 – 154.

GOEMANS, M. X. AND WILLIAMSON, D. P. 1995. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the Association for Computing Machinery 42*, 1115 – 1145.

GRAHAM, R. L., LAWLER, E. L., LENSTRA, J. K., AND RINNOOY KAN, A. H. G. 1979. Optimization and approximation in deterministic sequencing and scheduling: A survey. *Annals of Discrete Mathematics 5*, 287 – 326.

GRÖTSCHEL, M., LOVÁSZ, L., AND SCHRIJVER, A. 1981. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica 1*, 169 – 197. (Corrigendum: 4 (1984), 291 – 295).

GRÖTSCHEL, M., LOVÁSZ, L., AND SCHRIJVER, A. 1988. *Geometric Algorithms and Combinatorial Optimization*, Volume 2 of *Algorithms and Combinatorics*. Springer, Berlin.

HALL, L. A., SCHULZ, A. S., SHMOYS, D. B., AND WEIN, J. 1997. Scheduling to minimize average completion time: Off–line and on–line approximation algorithms. *Mathematics of*

Operations Research 22, 513 – 544.

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

Hoogeveen, H., Schuurman, P., and Woeginger, G. J. 1998. Non-approximability results for scheduling problems with minsum criteria. In R. E. Bixby, E. A. Boyd, and R. Z. Ríos-Mercado Eds., *Integer Programming and Combinatorial Optimization*, Volume 1412 of *Lecture Notes in Computer Science*, pp. 353 – 366. Springer, Berlin.

Kann, V., Khanna, S., and Lagergren, J. 1997. On the hardness of approximating MAX k–Cut and its dual. *Chicago Journal of Theoretical Computer Science 1997-2*.

Karger, D., Motwani, R., and Sudan, M. 1998. Approximate graph coloring by semidefinite programming. *Journal of the Association for Computing Machinery 45*, 246 – 265.

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

Kozlov, M. K., Tarasov, S. P., and Hačijan, L. G. 1979. Polynomial solvability of convex quadratic programming. *Soviet Mathematics Doklady 20*, 1108 – 1111.

Lawler, E. L., Lenstra, J. K., Rinnooy Kan, A. H. G., and Shmoys, D. B. 1993. Sequencing and scheduling: Algorithms and complexity. In S. C. Graves, A. H. G. Rinnooy Kan, and P. H. Zipkin Eds., *Logistics of Production and Inventory*, Volume 4 of *Handbooks in Operations Research and Management Science*, Chapter 9, pp. 445 – 522. North–Holland, Amsterdam, The Netherlands.

Lenstra, J. K., Rinnooy Kan, A. H. G., and Brucker, P. 1977. Complexity of machine scheduling problems. *Annals of Discrete Mathematics 1*, 343 – 362.

Lenstra, J. K., Shmoys, D. B., and Tardos, E. 1990. Approximation algorithms for scheduling unrelated parallel machines. *Mathematical Programming 46*, 259 – 271.

Mahajan, S. and Ramesh, H. 1999. Derandomizing approximation algorithms based on semidefinite programming. *SIAM Journal on Computing 28*, 1641 – 1663.

Möhring, R. H., Schäffter, M. W., and Schulz, A. S. 1996. Scheduling jobs with communication delays: Using infeasible solutions for approximation. In J. Diaz and M. Serna Eds., *Algorithms – ESA '96*, Volume 1136 of *Lecture Notes in Computer Science*, pp. 76 – 90. Springer, Berlin.

Motwani, R., Naor, J., and Raghavan, P. 1996. Randomized approximation algorithms in combinatorial optimization. In D. S. Hochbaum Ed., *Approximation algorithms for NP-hard problems*, Chapter 11, pp. 447 – 481. Thomson.

Motwani, R. and Raghavan, P. 1995. *Randomized Algorithms*. Cambridge University Press.

Munier, A., Queyranne, M., and Schulz, A. S. 1998. Approximation bounds for a general class of precedence constrained parallel machine scheduling problems. In R. E. Bixby, E. A. Boyd, and R. Z. Ríos-Mercado Eds., *Integer Programming and Combinatorial Optimization*, Volume 1412 of *Lecture Notes in Computer Science*, pp. 367 – 382. Springer, Berlin.

Papadimitriou, C. H. 1994. *Computational Complexity*. Addison-Wesley.

Papadimitriou, C. H. and Yannakakis, M. 1990. Towards an architecture-independent analysis of parallel algorithms. *SIAM Journal on Computing 19*, 322 – 328.

Papadimitriou, C. H. and Yannakakis, M. 1991. Optimization, approximation, and complexity classes. *Journal of Computer and System Sciences 43*, 425 – 440.

Phillips, C., Schulz, A. S., Shmoys, D. B., Stein, C., and Wein, J. 1998. Improved bounds on relaxations of a parallel machine scheduling problem. *Journal of Combinatorial Optimization 1*, 413 – 426.

Phillips, C., Stein, C., and Wein, J. 1997. Task scheduling in networks. *SIAM Journal on Discrete Mathematics 10*, 573 – 598.

Phillips, C., Stein, C., and Wein, J. 1998. Minimizing average completion time in the presence of release dates. *Mathematical Programming 82*, 199 – 223.

RAGHAVAN, P. AND THOMPSON, C. D. 1987. Randomized rounding: A technique for provably good algorithms and algorithmic proofs. *Combinatorica 7*, 365 – 374.

SAHNI, S. 1976. Algorithms for scheduling independent tasks. *Journal of the Association for Computing Machinery 23*, 116 – 127.

SAVELSBERGH, M. W. P., UMA, R. N., AND WEIN, J. M. 1998. An experimental study of LP–based approximation algorithms for scheduling problems. In *Proceedings of the 9th Annual ACM–SIAM Symposium on Discrete Algorithms* (1998), pp. 453 – 462.

SCHULZ, A. S. AND SKUTELLA, M. 1999a. The power of α-points in preemptive single machine scheduling. Submitted.

SCHULZ, A. S. AND SKUTELLA, M. 1999b. Scheduling unrelated machines by randomized rounding. Submitted. Different parts appeared in proceedings of ESA'97 and RANDOM'97.

SCHUURMAN, P. AND WOEGINGER, G. J. 1999. Polynomial time approximation algorithms for machine scheduling: ten open problems. *Journal of Scheduling 2*, 203 – 213.

SETHURAMAN, J. AND SQUILLANTE, M. S. 1999. Optimal scheduling of multiclass parallel machines. In *Proceedings of the 10th Annual ACM–SIAM Symposium on Discrete Algorithms* (1999), pp. 963 – 964.

SHMOYS, D. B. AND TARDOS, E. 1993. An approximation algorithm for the generalized assignment problem. *Mathematical Programming 62*, 461 – 474.

SKUTELLA, M. 1998. *Approximation and Randomization in Scheduling.* Ph. D. thesis, Technische Universität Berlin, Germany.

SKUTELLA, M. AND WOEGINGER, G. J. 2000. A PTAS for minimizing the total weighted completion time on identical parallel machines. *Mathematics of Operations Research 25*, 63 – 75.

SMITH, W. E. 1956. Various optimizers for single–stage production. *Naval Research and Logistics Quarterly 3*, 59 – 66.

SVIRIDENKO, M. 1999. New applications of the pipage rounding technique. Talk given at the workshop on Approximation Algorithms for Hard Problems in Combinatorial Optimization, The Fields Institute, Toronto, September 26 – October 1.

YE, Y. 1999. A .699-approximation algorithm for Max-Bisection. Manuscript.

ZWICK, U. 1999. Outward rotations: a tool for rounding solutions of semidefinite programming relaxations, with applications to MAX CUT and other problems. In *Proceedings of the 31st Annual ACM Symposium on Theory of Computing* (1999), pp. 679 – 687.