Semidefinite Relaxations for Parallel Machine Scheduling*

Martin Skutella
Fachbereich Mathematik, MA 6–1
Technische Universität Berlin
Straße des 17. Juni 136
D–10623 Berlin, Germany
skutella@math.tu-berlin.de

Abstract

We consider the problem of scheduling unrelated parallel machines so as to minimize the total weighted completion time of jobs. Whereas the best previously known approximation algorithms for this problem are based on LP relaxations, we give a $\frac{3}{2}$–approximation algorithm that relies on a convex quadratic programming relaxation. For the special case of two machines we present a further improvement to a 1.2752*–approximation; we introduce a more sophisticated semidefinite programming relaxation and apply the random hyperplane technique introduced by Goemans and Williamson for the* MAXCUT *problem and its refined version of Feige and Goemans. To the best of our knowledge, this is the first time that convex and semidefinite programming techniques (apart from LPs) are used in the area of scheduling.*

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., [27, 43, 33, 20, 40, 39, 19, 4, 34, 29, 10, 6, 42, 41, 14, 38, 31, 13, 44].

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 programming relaxations which have 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 [11]. Grötschel, Lovász, and Schrijver [17] 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 [15] for the problems MAXCUT, MAXDICUT, and MAX2SAT.

In semidefinite programming, one minimizes a linear function subject to linear constraints and the additional constraint that certain matrices whose entries are affine linear functions in the variables of the program are positive semidefinite. Since the subset of all positive semidefinite matrices constitutes a cone in $\mathbb{R}^{n\times n}$, semidefinite programs form a special class of convex programs. Under some requirements on the feasible space, they can be solved within an additive error of ε in polynomial time for any given $\varepsilon > 0$, cf. [18]. Practical efficiency can be obtained through interior point methods, see, e. g., [1]. More information on semidefinite programming can for example be found in [47].

Goemans and Williamson [15] 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. They introduced the 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 [8] 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

*Copyright 1998 IEEE. Published in the Proceedings of FOCS'98, 8-11 November 1998 in Palo Alto, CA. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works, must be obtained from the IEEE. Contact: Manager, Copyrights and Permissions / IEEE Service Center / 445 Hoes Lane / P.O. Box 1331 / Piscataway, NJ 08855-1331, USA. Telephone: + Intl. 732-562-3966.

design of approximation algorithms can for instance be found in [22, 5, 9, 46].

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, see also [9, 46]. The reason is that up to now only lower bounds on the crucial probabilities involved have been attained, see [15, 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 prove 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.

We consider the following scheduling problem. We are given a set of n jobs $J = \{1,\dots,n\}$ and m parallel machines or processors. Each job j has a positive integral processing requirement p_{ij} which depends on the machine i job j will be processed on. Each job j must be processed for the respective amount of time on one of the m machines, and may be assigned to any of them. Every machine can process at most one job at a time. The completion time of job j in a schedule is denoted by C_j. We aim to minimize the total weighted completion time $\sum_{j\in J} w_j C_j$ where w_j denotes a given nonnegative integral weight of job j which is a measure for its importance. In the standard notation of Graham et al. [16], this problem reads $\mathrm{R}|\,|\sum w_j C_j$. It is shown to be NP-hard in [3, 26], even for a fixed number $m \geqslant 2$ of identical parallel machines. The special case of identical parallel machines, i. e., for each job j and all machines i we have $p_{ij} = p_j$, is denoted by $\mathrm{P}|\,|\sum w_j C_j$.

The corresponding single machine scheduling problem can be solved in polynomial time using Smith's Ratio Rule [45]: Sequence the jobs in order of nonincreasing ratios w_j/p_j. Therefore the only problem in constructing good schedules for instances of $\mathrm{R}|\,|\sum w_j C_j$ is to assign the jobs appropriately to machines. Afterwards one sequences all jobs that have been assigned to machine i in order of nonincreasing ratios w_j/p_{ij}. Thus, whenever we talk about an assignment of jobs to machines we associate a corresponding schedule. We will always assume that whenever $w_k/p_{ik} = w_j/p_{ij}$ for a pair of jobs j,k and both jobs are assigned to machine i the job with smaller index is scheduled first. For each machine $i = 1,\dots,m$ we define 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 $j < k$. For the special case of identical parallel machines we always assume that the jobs are indexed such that $1 \prec 2 \prec \cdots \prec n$.

The first approximation result for minimizing the total weighted completion time on unrelated parallel machines was given by Phillips, Stein, and Wein [32] and achieves performance ratio $O(\log^2 n)$. Hall, Schulz, Shmoys, and Wein [19] presented a $\frac{16}{3}$–approximation algorithm that is based on an LP relaxation in time-indexed variables and uses the rounding technique of Shmoys and Tardos for the generalized assignment problem [43]. This result was successively improved by Schulz and Skutella to performance ratios $2+\varepsilon$ and $\frac{3}{2}+\varepsilon$ [42, 41]. The $\left(\frac{3}{2}+\varepsilon\right)$–approximation algorithm is also the best known result if the number of machines is fixed to $m \geqslant 2$. However, for the special case of identical parallel machines Kawaguchi and Kyan [23] showed that list scheduling in order of nonincreasing ratios w_j/p_j is a 1.21–approximation algorithm. For any fixed number of identical parallel machines Sahni [37] gave a polynomial time approximation scheme which builds upon a general dynamic programming technique of Rothkopf [36] and Lawler and Moore [25]. It was observed by Woeginger [48] that the result of Sahni can be extended to a fixed number of uniform parallel machines which run at different but not job-dependent speeds.

In this paper we introduce a new technique. We start with a formulation of $\mathrm{R}|\,|\sum w_j C_j$ as an integer quadratic program in Section 2, consider its continuous relaxation, and discuss a simple randomized rounding heuristic. In Section 3 we describe how the objective function of the quadratic program can be convexified yielding a convex programming relaxation that can be solved in polynomial time. Together with randomized rounding this leads to an approximation algorithm with performance guarantee $\frac{3}{2}$ which is an ε–improvement on the previously best known result. In Section 4 we present a more sophisticated vector programming relaxation for the special case of two machines. We show that a modification of the random hyperplane approach of Goemans and Williamson leads to a 1.339–approximation algorithm for $\mathrm{R}2|\,|\sum w_j C_j$. A combination of this algorithm with the $\frac{3}{2}$–approximation leads to an improved performance guarantee 1.276. In Section 5 we discuss relations between MAXCUT and scheduling on identical parallel machines. In particular, we present an approximation preserving reduction of $\mathrm{P}m|\,|\sum w_j C_j$ to MAXkCUT where $k = m$.

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 $Z_R(a)$. In this extended abstract we omit some details and proofs for reasons of brevity. A more detailed version can be found in [44, Chapter 3].

2 Quadratic programming formulations

The problem of scheduling a set of jobs on unrelated parallel machines can be formulated as an integer quadratic program in assignment variables. We introduce for each machine $i = 1, \dots, m$ 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 [27] used the same variables to formulate the problem of minimizing the makespan of unrelated parallel machines as an integer linear program. However, minimizing the average weighted completion time forces quadratic terms and leads to the following program (IQP):

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

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 the 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 right hand side of (2).

We denote the 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 assigned independently at random 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 (1). We refer to this rounding technique as Algorithm RANDOMIZED ROUNDING. A similar algorithm, however based on a time-indexed LP relaxation, has been studied by Schulz and Skutella [41]. The idea of using randomized rounding in the study of approximation algorithms was introduced by Raghavan and Thompson [35], an overview can be found in [30].

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 whose proof is straightforward and therefore omitted for reasons of brevity.

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*

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

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 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

$$\Pr(j,k \mapsto i) = \Pr(j \mapsto i) \cdot \Pr(k \mapsto i) = \bar{a}_{ij} \cdot \bar{a}_{ik} \ .$$

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

Algorithm RANDOMIZED ROUNDING can easily be derandomized by the method of conditional probabilities. We refer to its derandomized version as 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 theorem:

Theorem 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 [2] used similar techniques to establish the integrality of several polyhedra.

It follows from Theorem 2.3 that it is still NP-hard to find an optimum solution to the quadratic program (QP). In the following two sections we consider relaxations of (IQP) that can be solved in polynomial time. The idea of the first relaxation is to convexify the objective function of (QP) in order to get a convex quadratic program. The second relaxation is a semidefinite programming relaxation of (IQP) for the special case of two machines which extends the ideas used in [15] and [8] for MAX2SAT and MAXDICUT.

3 A convex quadratic programming relaxation

The quadratic program (QP) can be rewritten as

$$\begin{aligned} \min \quad & c^T a + \tfrac{1}{2} a^T D a && (4) \\ \text{s. t.} \quad & \sum_{i=1}^{m} a_{ij} = 1 \qquad \text{for } j \in J \\ & a \geqslant 0 \end{aligned}$$

where $a \in \mathbb{R}^{mn}$ denotes the vector consisting of all variables a_{ij} lexicographically ordered with respect to the natural order $1, 2, \dots, m$ 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, $i = 1, \dots, m$, 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:

$$D_i = \begin{pmatrix} 0 & w_2p_1 & w_3p_1 & \cdots & w_np_1 \\ w_2p_1 & 0 & w_3p_2 & \cdots & w_np_2 \\ w_3p_1 & w_3p_2 & 0 & & w_np_3 \\ \vdots & \vdots & & \ddots & \vdots \\ w_np_1 & w_np_2 & w_np_3 & \cdots & 0 \end{pmatrix} \tag{5}$$

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

$$D_i = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \;; \tag{6}$$

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., [24, 7]). 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 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:

$$\min \quad (1-\gamma) \cdot c^T a + \tfrac{1}{2} a^T \big(D + 2\gamma \cdot \operatorname{diag}(c)\big) a \tag{7}$$

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 $a^T \operatorname{diag}(c) a$ for arbitrary $a \in [0,1]^{mn}$; equality holds for $a \in \{0,1\}^{mn}$. Therefore the modified objective function (7) underestimates (4). Since we want to keep the gap as small as possible and since (7) is nonincreasing in γ for each fixed vector a, we are looking for a smallest possible value of γ such that $D + 2\gamma \cdot \operatorname{diag}(c)$ is positive semidefinite.

Lemma 3.1. *The function*

$$a \mapsto (1-\gamma) \cdot c^T a + \frac{1}{2} a^T \big(D + 2\gamma \cdot \operatorname{diag}(c)\big) a$$

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

Proof. In order to show that the positive semidefiniteness of $D + 2\gamma \cdot \operatorname{diag}(c)$ for arbitrary instances implies $\gamma \geqslant \frac{1}{2}$, we consider the example given in (6). Here, the diagonal entries of the i–th block of $D + 2\gamma \cdot \operatorname{diag}(c)$ 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 $D + \operatorname{diag}(c)$ is positive semidefinite for arbitrary instances. Using the same notation as in (5) the i–th block of $D + \operatorname{diag}(c)$ has the form

$$A = \begin{pmatrix} w_1p_1 & w_2p_1 & w_3p_1 & \cdots & w_np_1 \\ w_2p_1 & w_2p_2 & w_3p_2 & \cdots & w_np_2 \\ w_3p_1 & w_3p_2 & w_3p_3 & \cdots & w_np_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ w_np_1 & w_np_2 & w_np_3 & \cdots & w_np_n \end{pmatrix} . \tag{8}$$

An easy calculation shows that the matrix A is positive semidefinite since $w_1/p_1 \geqslant \cdots \geqslant w_n/p_n$.

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

Lemma 3.1 and the remarks above motivate the consideration of the following convex quadratic programming relaxation (CQP):

$$\begin{aligned} \min \quad & \tfrac{1}{2} c^T a + \tfrac{1}{2} a^T \big(D + \operatorname{diag}(c)\big) a \\ \text{s. t.} \quad & \sum_{i=1}^{m} a_{ij} = 1 \qquad \text{for } j \in J && (9) \\ & a \geqslant 0 && (10) \end{aligned}$$

As mentioned above (CQP) can be solved in polynomial time. If we consider the special case of identical parallel machines $P| \,|\sum w_j C_j$, we can directly give a Karush-Kuhn-Tucker point and therefore an optimum solution to (CQP). The following result can be proved by a simple symmetry argument. We omit the proof for reasons of brevity.

Lemma 3.2. *For instances of* $P| \,|\sum w_j C_j$ *the vector* $\bar{a}$ *with* $\bar{a}_{ij} = \frac{1}{m}$*, 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.*

Theorem 3.3.

a) Computing an optimum solution $\bar{a}$ *to* (CQP) *and using* RANDOMIZED ROUNDING *to construct a feasible schedule is a* 2*–approximation algorithm for* $R| \,|\sum w_j C_j$.

b) *Assigning each job independently and uniformly at random to one of the m machines is a $\left(\frac{3}{2}-\frac{1}{2m}\right)$–approximation algorithm for* $\mathrm{P}|\,|\sum w_jC_j$.

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 3.2. Theorem 2.1 yields

$$\begin{aligned}\mathrm{E}\Big(\sum_j w_jC_j\Big) &= Z_{CQP}(\bar{a})+\tfrac{1}{2}\big(c^T\bar{a}-\bar{a}^T\mathrm{diag}(c)\bar{a}\big)\\ &\leqslant 2\cdot Z_{CQP}(\bar{a}) \;=\; 2\cdot Z^*_{CQP}\ .\end{aligned}$$

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} 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 (9) imply $c^Ta=\sum_j w_jp_j\leqslant Z^*$ 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 $\mathrm{E}\big(\sum_j w_jC_j\big)\leqslant(\frac{3}{2}-\frac{1}{2m})\cdot Z^*$. □

It also follows from Theorem 3.3 that (CQP) is a 2–relaxation, i. e., $Z^*\leqslant 2\cdot Z^*_{CQP}$. This bound is tight even for the case of identical parallel machines. Consider an instance with one job of size and weight one such that the value Z^* of an optimum schedule is equal to one. By Lemma 3.2 we get $Z^*_{CQP}=\frac{m+1}{2m}$. Thus, if m goes to infinity the ratio Z^*/Z^*_{CQP} converges to two.

Unfortunately, we cannot directly carry over the $\frac{3}{2}$–approximation to the setting of unrelated parallel machines. The reason is that c^Ta 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^Ta. The idea for an improved approximation result is to add this lower bound as a constraint to (CQP). In the context of LP based approximations, the second part of Theorem 3.3 and a similar idea has been given in [41]. It leads to the following strengthened relaxation (CQP'), which is a convex program and can therefore be solved through the ellipsoid algorithm within any additive error $\varepsilon>0$ in polynomial time, see [18].

$$\begin{aligned}\min\quad & Z_{CQP'}\\ \text{s. t.}\quad & \sum_{i=1}^m a_{ij}=1 \qquad \text{for } j\in J\\ & Z_{CQP'}\geqslant\tfrac{1}{2}c^Ta+\tfrac{1}{2}a^T\big(D+\mathrm{diag}(c)\big)a \qquad (11)\\ & Z_{CQP'}\geqslant c^Ta \qquad (12)\\ & a\geqslant 0\end{aligned}$$

We show that (CQP') is actually equivalent to a semidefinite program.

Lemma 3.4. *There exists a symmetric $mn\times mn$–matrix $M(Z,a)$ which can be computed in polynomial time and whose entries are linear functions in the variables Z and a_{ij}, such that*

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

if and only if $M(Z,a)$ is positive semidefinite. In particular, (CQP') can be interpreted as a semidefinite program.

Sketch of proof. As proposed in [47, Section 2] we define

$$M(Z,a)=\begin{pmatrix}E_{mn-1} & Ba\\ (Ba)^T & 2Z-c^Ta\end{pmatrix}$$

where E_{mn-1} denotes the $(mn-1)$–dimensional identity matrix and the matrix B is defined by the Cholesky decomposition $\big(D+\mathrm{diag}(c)\big)=B^TB$. An easy calculation yields the result. □

The next lemma follows from the proof of Theorem 3.3 and constraint (12).

Lemma 3.5. *Given a feasible solution $\bar{a}$ to (CQP') Algorithm* RANDOMIZED ROUNDING *computes a schedule whose expected value is bounded from above by $\frac{3}{2}\cdot Z_{CQP'}(\bar{a})$.*

Lemma 3.5 yields that (CQP') is a $\frac{3}{2}$–relaxation for $\mathrm{R}|\,|\sum w_jC_j$. We get a randomized approximation algorithm with expected performance guarantee $\frac{3}{2}+\varepsilon$ 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 3.6. *Computing a near optimal solution to (CQP') and using Algorithm* DERANDOMIZED ROUNDING *to get a feasible schedule is a $\frac{3}{2}$–approximation algorithm for* $\mathrm{R}|\,|\sum w_jC_j$.

Proof. We compute a feasible solution $\bar{a}$ to (CQP') satisfying $Z_{CQP'}(\bar{a})<Z^*_{CQP'}+\frac{1}{3}$. By Lemma 3.5 Algorithm DERANDOMIZED ROUNDING converts this solution into a feasible schedule whose value is bounded by $\frac{3}{2}\cdot Z_{CQP'}(\bar{a})<\frac{3}{2}\cdot Z^*_{CQP'}+\frac{1}{2}\leqslant\frac{3}{2}\cdot Z^*+\frac{1}{2}$. Since all the weights and processing times are integral, the same holds for Z^*. The value of our schedule can therefore be bounded by $\frac{3}{2}\cdot Z^*$. □

The performance ratios given in Lemma 3.5 and Theorem 3.6 are only tight if (12) 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 ratio (see Figure 2 below).

Corollary 3.7. *For any feasible solution $\bar{a}$ to (CQP') Algorithm* RANDOMIZED ROUNDING *computes a feasible schedule whose expected value is bounded from above by $\big(1+\frac{c^T\bar{a}}{2Z_{CQP'}(\bar{a})}\big)\cdot Z_{CQP'}(\bar{a})$.*

Approximation algorithms with the same or slightly worse performance ratios than those presented in Theorems 3.3 and 3.6 have already been given by Schulz and Skutella [42, 41]. In contrast to our approach here they used LP relaxations in time-indexed variables together with randomized rounding. However, the underlying intuition of our quadratic programs and their LPs is similar.

4 A semidefinite relaxation for two machines

In this section we consider the problem of scheduling two unrelated parallel machines which is usually denoted by $R2| |\sum w_jC_j$. To keep the notation as simple as possible we assume that the two machines are numbered 0 and -1 throughout this section.

We start with a reformulation of (IQP) in variables $x_j \in \{1,-1\}$, for $j \in J$, and additional variables $x_0, x_{-1} \in \{1,-1\}$ with $x_{-1} = -x_0$; job j is being assigned to machine 0 if $x_j = x_0$ and to machine -1 otherwise. Therefore the assignment variables a_{ij} in (IQP) are replaced by $\frac{1+x_ix_j}{2}$ and the quadratic terms $a_{ij}a_{ik}$ by $\frac{x_jx_k+x_ix_j+x_ix_k+1}{4}$. Note that we have introduced the variable x_{-1} only to keep notation simple; it could as well be replaced by $-x_0$. 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.

$$\begin{aligned} \min \quad & \sum_j w_jC_j \\ \text{s. t.} \quad & C_j = \sum_{i=-1}^{0}\Big(\frac{1+v_iv_j}{2}\cdot p_{ij} \\ & \qquad + \sum_{k\prec_i j}\frac{v_jv_k+v_iv_j+v_iv_k+1}{4}\cdot p_{ik}\Big) \qquad (13) \\ & v_{-1}v_0 = -1 \\ & v_jv_j = 1 \qquad\qquad \text{for } j\in J\cup\{0,-1\} \end{aligned}$$

Here v_jv_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., [15]). This program can be strengthened by adding the constraints

$$v_jv_k + v_iv_j + v_iv_k + 1 \geqslant 0 \qquad (14)$$

for $i \in \{0,-1\}$ and $j,k \in J$. Constraints of the same form have been used by Feige and Goemans [8] to improve some of the approximation results of Goemans and Williamson [15].

It is one of the key insights of this paper that the vector program can be further strengthened with the quadratic constraint (11) from the convex quadratic program (CQP'). For a feasible solution v to (VP) we denote by $a = a(v)$ the corresponding solution to (CQP), i. e., $a_{ij} = \frac{1+v_iv_j}{2}$, and add the constraint

$$\begin{aligned} \sum_j w_jC_j &\geqslant \tfrac{1}{2}c^Ta + \tfrac{1}{2}a^T\big(D+\operatorname{diag}(c)\big)a \qquad (15) \\ &= \sum_j w_j \sum_{i=-1}^{0} a_{ij}\cdot\Big(\frac{1+a_{ij}}{2}p_{ij} + \sum_{k\prec_i j} a_{ik}\cdot p_{ik}\Big) \end{aligned}$$

to the vector program (VP). The resulting program with the additional constraints (14) and (15) is denoted by (SDP). Note that constraint (12) is included in (13) and (14).

By Lemma 3.4 and our remark above (SDP) can be interpreted as a semidefinite program in variables corresponding to the scalar products v_jv_k. For a feasible solution to (SDP) we consider the random hyperplane rounding of Goemans and Williamson:

Algorithm RANDOM HYPERPLANE

1) Draw a vector r randomly and uniformly distributed from the unit-sphere of $\mathbb{R}^{n+1}$.

2) For each job j, assign j to the machine i with $\operatorname{sgn}(v_jr) = \operatorname{sgn}(v_ir)$.

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 sets. 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.

To analyze (SDP) and Algorithm RANDOM HYPERPLANE we need the following lemma which is a restatement of [15, 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 4.1. *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 4.1 the expected value of the completion time of job j in the schedule computed by Algorithm RANDOM HYPERPLANE is given by

$$\begin{aligned} \mathrm{E}(C_j) = \sum_{i=-1}^{0}\Big(\Big(1-\frac{\alpha_{ij}}{\pi}\Big)\cdot p_{ij} \\ + \sum_{k\prec_i j}\Big(1-\frac{\alpha_{jk}+\alpha_{ij}+\alpha_{ik}}{2\pi}\Big)\cdot p_{ik}\Big). \qquad (16) \end{aligned}$$

We want to compare the expected value of the schedule computed by Algorithm Random Hyperplane to the value of the feasible solution to (SDP) we started with.

Lemma 4.2. *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*

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

and

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

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

$$\rho_1 \frac{3c^Ta}{4} + \rho_2\Big(Z - \frac{3c^Ta}{4}\Big) \leqslant \rho_2 \cdot Z \ .$$

Sketch of proof. Observe that the required bounds in Lemma 4.2 compare the given probabilities to the average of the corresponding coefficients in (13) and (15). It therefore follows from Lemma 2.2, linearity of expectations, and an easy calculation that the expected value of the computed schedule is bounded by $\max\{\rho_1,\rho_2\}\cdot Z = \rho_2 \cdot Z$. A preciser analysis yields the stronger bound given in the lemma. $\square$

Inspired by the work of Feige and Goemans [8] we give a rounding scheme that fulfills the conditions described in Lemma 4.2 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 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 1.

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, see [8].

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

$$\Pr(j \mapsto i) = \frac{1+\cos(\alpha_{ij})}{2} = \frac{1+v_iv_j}{2} = a_{ij}$$

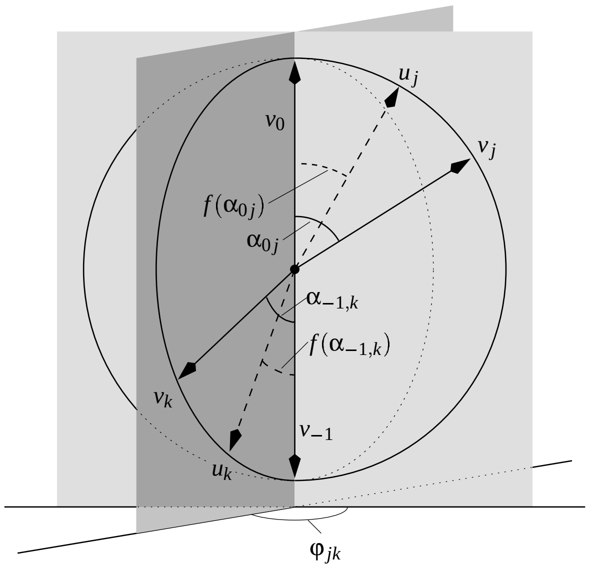

Figure 1. Modification according to the function f.

for each job j. In other words, the probability that a job is assigned to a machine is equal to the corresponding coefficient in (13). 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 (13) and (15).

Therefore we use a different function f_2 defined by $f_2(\theta) = f_1\big(\xi(\theta)\big)$ where $\xi(\theta) = \min\big\{\pi, \max\{0, \frac{\pi}{2} + 1.3662 \cdot (\theta - \frac{\pi}{2})\}\big\}$. This yields a rounding scheme with expected performance guarantee 1.3388. We have shown numerically that the conditions in Lemma 4.2 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 which are nearly tight. We should mention that both constraints (14) and (15) 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 omit details for reasons of brevity.

Theorem 4.3. *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 was shown in [28] 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 strongly believe that there exists a function similar to f_2 which yields an expected performance guarantee of $\frac{4}{3}$. 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 4.2 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 (13) and (15) rather than using their average as in Lemma 4.2; but this did not lead to any improvement.

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 4.4. *For an almost optimal solution to (SDP), either Algorithm* Randomized Rounding *or Algorithm* Random Hyperplane *(together with f_2) produces a schedule whose expected value is bounded by* $1.2752 \cdot Z^*$.

Sketch of Proof. Notice that by Corollary 3.7 and Lemma 4.2 the two rounding techniques do not behave bad for the same class of instances and solutions to (SDP), see Figure 2. $\square$

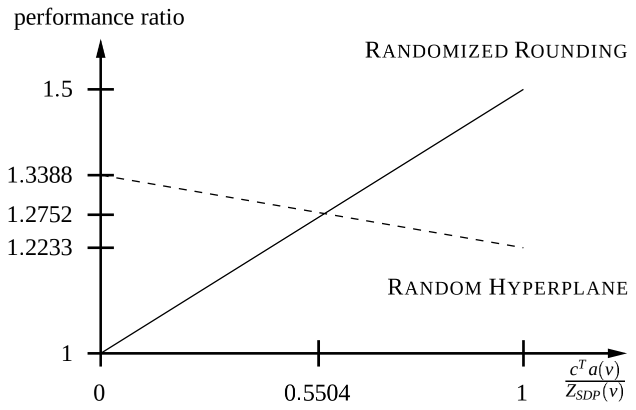

Figure 2. Comparison of Randomized Rounding and Random Hyperplane.

Observations of this type have already been used in other contexts to get improved approximation results. Theorem 4.4 also implies that (SDP) is a 1.2752–relaxation for $R2| |\sum w_j C_j$.

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 [15, 8, 9, 46]. 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 (15). We hope that this approach will also prove useful for other problems in combinatorial optimization.

5 MaxCut algorithms for scheduling identical parallel machines

We associate with each instance of $P| |\sum w_j C_j$ 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 $J_1, \dots, J_m$ 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 $E_{\text{cut}} := E_J \setminus E_{\text{sch}}$ are contained in the induced m–cut. In particular we get

$$c(E_J) = \sum_j w_j C_j - \sum_j w_j p_j + c(E_{\text{cut}}) \qquad (17)$$

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 $c(E_{\text{cut}})$ of the induced m–cut. This reduction of $Pm| |\sum w_j C_j$ to Max$m$Cut is approximation preserving:

Theorem 5.1. *For any $\rho \leqslant 1$, a ρ–approximation algorithm for* Max$m$Cut *translates into an approximation algorithm for $Pm| |\sum w_j C_j$ 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 3.2 yields

$$Z^* \geqslant Z^*_{CQP} = \frac{1}{m} \cdot c(E_J) + \left(\frac{1}{2} + \frac{1}{2m}\right) \sum_j w_j p_j \qquad (18)$$

such that

$$Z^*_{\text{cut}} \leqslant \frac{m-1}{m} \cdot c(E_J) + \left(\frac{1}{2} - \frac{1}{2m}\right) \sum_j w_j p_j \qquad (19)$$

by (17) and (18). 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:

$$\begin{aligned}\sum_j w_j C_j &\leqslant Z^* + (1-\rho)\cdot Z^*_{\text{cut}} && \text{by (17)}\\ &\leqslant Z^* + (1-\rho)\cdot(m-1)\cdot Z^* && \text{by (19), (18)}\\ &= \big(\rho + m\cdot(1-\rho)\big)\cdot Z^* \ . \end{aligned}$$

□

Unfortunately, this result does not lead to an improved performance guarantee for $P||\sum w_jC_j$. The best currently known approximation algorithms for MAXmCUT have performance ratio $1-\frac{1}{m}+o\left(\frac{1}{m}\right)$ which leads to performance guarantee $2-\frac{1}{m}$ for $P||\sum_j w_jC_j$ by Theorem 5.1. 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 3.2 and therefore achieves performance ratio $\frac{3}{2}-\frac{1}{2m}$ for $P||\sum_j w_jC_j$ rather than $2-\frac{1}{m}$. It is shown in [21] that MAXmCUT cannot be approximated within $\rho < 1+\frac{1}{34m}$, unless P=NP.

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 3.3 and the 1.2752–approximation in Theorem 4.4. 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 [15]. This leads to the following result.

Corollary 5.2. *Computing an almost optimal solution to* (SDP) *and applying Algorithm* RANDOM HYPERPLANE *to get a feasible schedule is a* 1.122*–approximation for* $P2||\sum w_jC_j$.

This result has been further improved by Goemans [12] to performance guarantee 1.073 through a more sophisticated rounding technique.

Acknowledgments: I am grateful to Michel Goemans and Andreas Schulz for helpful comments on an earlier version of this paper.

References

[1] F. Alizadeh. Interior point methods in semidefinite programming with applications to combinatorial optimization. *SIAM Journal on Optimization*, 5:13 – 51, 1995.

[2] D. Bertsimas, C. Teo, and R. Vohra. On dependent randomized rounding algorithms. In W. H. Cunningham, S. T. McCormick, and M. Queyranne, editors, *Integer Programming and Combinatorial Optimization*, volume 1084 of *Lecture Notes in Computer Science*, pages 330 – 344. Springer, Berlin, 1996.

[3] J. L. Bruno, E. G. Coffman Jr., and R. Sethi. Scheduling independent tasks to reduce mean finishing time. *Communications of the Association for Computing Machinery*, 17:382 – 387, 1974.

[4] S. Chakrabarti, C. A. Phillips, A. S. Schulz, D. B. Shmoys, C. Stein, and J. Wein. Improved scheduling algorithms for minsum criteria. In F. Meyer auf der Heide and B. Monien, editors, *Automata, Languages and Programming*, volume 1099 of *Lecture Notes in Computer Science*, pages 646 – 657. Springer, Berlin, 1996.

[5] B. Chor and M. Sudan. A geometric approach to betweenness. In P. Spirakis, editor, *Algorithms – ESA '95*, volume 979 of *Lecture Notes in Computer Science*, pages 227 – 237. Springer, Berlin, 1995.

[6] F. A. Chudak and D. B. Shmoys. Approximation algorithms for precedence–constrained scheduling problems on parallel machines that run at different speeds. In *Proceedings of the 8th Annual ACM–SIAM Symposium on Discrete Algorithms*, pages 581 – 590, 1997.

[7] S. J. Chung and K. G. Murty. Polynomially bounded ellipsoid algorithms for convex quadratic programming. In O. L. Mangasarian, R. R. Meyer, and S. M. Robinson, editors, *Nonlinear Programming 4*, pages 439 – 485. Academic Press, 1981.

[8] U. Feige and M. X. Goemans. 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*, pages 182 – 189, 1995.

[9] A. Frieze and M. Jerrum. Improved approximation algorithms for MAX k-CUT and MAX BISECTION. *Algorithmica*, 18:67 – 81, 1997.

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

[11] M. X. Goemans. Semidefinite programming in combinatorial optimization. *Mathematical Programming*, 79:143 – 161, 1997.

[12] M. X. Goemans, June 1998. Personal communication.

[13] M. X. Goemans, M. Queyranne, A. S. Schulz, M. Skutella, and Y. Wang. Single machine scheduling with release dates, 1998. In preparation.

[14] M. X. Goemans, J. Wein, and D. P. Williamson. A 1.47–approximation algorithm for a preemptive single-machine scheduling problem. Manuscript, 1997.

[15] M. X. Goemans and D. P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the Association for Computing Machinery*, 42:1115 – 1145, 1995.

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

[17] M. Grötschel, L. Lovász, and A. Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1:169 – 197, 1981. (Corrigendum: **4** (1984), 291 – 295).

[18] M. Grötschel, L. Lovász, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization*, volume 2 of *Algorithms and Combinatorics*. Springer, Berlin, 1988.

[19] L. A. Hall, A. S. Schulz, D. B. Shmoys, and J. Wein. Scheduling to minimize average completion time: Off–line and on–line approximation algorithms. *Mathematics of Operations Research*, 22:513 – 544, 1997.

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

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

[22] D. Karger, R. Motwani, and M. Sudan. Approximate graph coloring by semidefinite programming. In *Proceedings of the 35th Annual IEEE Symposium on Foundations of Computer Science*, pages 2 – 13, 1994.

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

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

[25] E. L. Lawler and J. M. Moore. A functional equation and its application to resource allocation and sequencing problems. *Management Science*, 16:77 – 84, 1969.

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

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

[28] S. Mahajan and H. Ramesh. Derandomizing semidefinite programming based approximation algorithms. In *Proceedings of the 36th Annual IEEE Symposium on Foundations of Computer Science*, pages 162 – 169, 1995.

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

[30] R. Motwani, J. Naor, and P. Raghavan. Randomized approximation algorithms in combinatorial optimization. In D. S. Hochbaum, editor, *Approximation algorithms for NP-hard problems*, chapter 11, pages 447 – 481. Thomson, 1996.

[31] A. Munier, M. Queyranne, and A. S. Schulz. 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, editors, *Integer Programming and Combinatorial Optimization*, volume 1412 of *Lecture Notes in Computer Science*, pages 367 – 382. Springer, Berlin, 1998.

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

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

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

[35] P. Raghavan and C. D. Thompson. Randomized rounding: A technique for provably good algorithms and algorithmic proofs. *Combinatorica*, 7:365 – 374, 1987.

[36] M. H. Rothkopf. Scheduling independent tasks on parallel processors. *Management Science*, 12:437 – 447, 1966.

[37] S. Sahni. Algorithms for scheduling independent tasks. *Journal of the Association for Computing Machinery*, 23:116 – 127, 1976.

[38] M. W. P. Savelsbergh, R. N. Uma, and J. M. Wein. An experimental study of LP–based approximation algorithms for scheduling problems. In *Proceedings of the 9th Annual ACM–SIAM Symposium on Discrete Algorithms*, pages 453 – 462, 1998.

[39] A. S. Schulz. *Polytopes and Scheduling*. PhD thesis, Technical University of Berlin, Germany, 1996.

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

[41] A. S. Schulz and M. Skutella. Random–based scheduling: New approximations and LP lower bounds. In J. Rolim, editor, *Randomization and Approximation Techniques in Computer Science*, volume 1269 of *Lecture Notes in Computer Science*, pages 119 – 133. Springer, Berlin, 1997.

[42] A. S. Schulz and M. Skutella. Scheduling–LPs bear probabilities: Randomized approximations for min–sum criteria. In R. Burkard and G. J. Woeginger, editors, *Algorithms – ESA '97*, volume 1284 of *Lecture Notes in Computer Science*, pages 416 – 429. Springer, Berlin, 1997.

[43] D. B. Shmoys and E. Tardos. An approximation algorithm for the generalized assignment problem. *Mathematical Programming*, 62:461 – 474, 1993.

[44] M. Skutella. *Approximation and Randomization in Scheduling*. PhD thesis, Technical University of Berlin, Germany, 1998.

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

[46] A. Srivastav and K. Wolf. Finding dense subgraphs with semidefinite programming. In K. Jansen and J. Rolim, editors, *Approximation Algorithms for Combinatorial Optimization, Proceedings of APPROX'98*, volume 1444 of *Lecture Notes in Computer Science*, pages 181 – 191. Springer, Berlin, 1998.

[47] L. Vandenberghe and S. Boyd. Semidefinite programming. *SIAM Review*, 38:49 – 95, 1996.

[48] G. J. Woeginger. When does a dynamic programming formulation guarantee the existence of an FPTAS ? Report Woe–27, Institut für Mathematik B, Technical University of Graz, Austria, April 1998.