16

Convex Quadratic Relaxations in Scheduling

Jay Sethuraman
Columbia University

16.1 Introduction

Scheduling problems have been at the heart of several important developments in the fields of combinatorial optimization, approximation algorithms, and complexity theory. Over the last four decades, research on scheduling theory has resulted in new tools and techniques that have been useful in larger contexts. At the same time, advances in these areas have resulted in an improved understanding of scheduling problems. The theory of NP-completeness helped distinguish relatively easy scheduling problems from difficult ones. Attempts to find reasonable solutions to these difficult problems gave birth to the field of approximation algorithms, which focuses on finding efficient algorithms for NP-hard (or harder) problems with a provable performance guarantee. While the earliest approximation algorithms were combinatorial in nature, the last fifteen years have resulted in an improved understanding of the power of mathematical programming methods in designing approximation algorithms.

Linear programming (LP) has proved to be an invaluable tool in the design and analysis of approximation algorithms. The typical steps in designing an LP-based approximation algorithm are as follows: first, we formulate the problem under consideration as an integer programming problem; second, we solve the underlying linear programming relaxation of the problem to obtain an optimal fractional solution, whose cost is obviously a lower bound on the achievable optimal cost; we then "round" the fractional solution to a feasible integer solution in a systematic way; and finally, we bound the "error" incurred in this rounding process. LP-based methods have been used to design improved approximation algorithms for an impressive array of problems arising in diverse settings. Moreover, these techniques have resulted in a significantly better understanding of the problem itself, often leading to a better combinatorial approximation algorithm. The attractiveness of LP as a tool in designing (approximation) algorithms is due to several factors: LPs can be solved efficiently, both in theory and practice; they admit an elegant duality theory,

which is often the key to analyzing the proposed (approximation) algorithm; and they arise naturally as relaxations of many optimization problems.

Given the success of LP-based methods, it seems natural to ask if more general mathematical programming methods can be used to design approximation algorithms for difficult optimization problems. After all, convex programming problems admit efficient algorithms (in theory). Moreover, semidefinite programming methods were used by Lovász [1] to find the Shannon capacity of a graph and Grötschel et al. [2] to design a polynomial-time algorithm for finding a maximum independent set in a perfect graph. However, the power of convex (specifically, semidefinite) programming methods was illustrated by Goemans and Williamson [3] in their pioneering work on the MAXCUT problem. Since then, researchers have investigated the use of semidefinite programming in various problems such as graph coloring [4] and betweenness [5], and graph bisection [6].

Scope and Organization. Our brief survey chapter is on the role of convex relaxations in scheduling. Specifically, we consider the problem of scheduling unrelated machines with the objective of minimizing weighted completion time, and several natural variants of this problem. We show that this problem leads to a natural integer convex programming problem. Relaxing the integrality constraints leads to a convex programming problem, which can be solved efficiently. We describe a natural rounding algorithm that converts this "fractional" schedule into an "integral" one, and analyze its performance.

Credits. Since this chapter is of an expository nature, we strive to keep the bibliographic references in the text to a minimum. The first convex relaxations for scheduling unrelated machines in the absence of release dates was proposed and analyzed independently by Skutella [7] and Sethuraman and Squillante [8,9]; this is discussed in Section 16.2.1. Skutella [7] further presented improved results for the two machine case based on a semidefinite programming relaxation (not discussed in this article). In a subsequent paper, Skutella [12] considered the problem of scheduling unrelated machines in the presence of machine-dependent release dates: this is discussed in Section 16.3.1. The preemptive versions of these problems were treated by Skutella [10]; these ideas are discussed in Section 16.2.2 and Section 16.3.1 respectively. The basic problem with cardinality constraints was considered by Ageev and Sviridenko [11] and forms the subject of Section 16.4. The presentation here draws heavily from Skutella [12], which summarizes his work on the problem.

16.2 Scheduling Unrelated Machines

We consider various versions of the problem of scheduling a set of n jobs on m unrelated machines. Job j can be processed on machine i at or after its (machine-dependent) release date r_{ij}, and requires p_{ij} time units. We assume that the r_{ij} and p_{ij} are positive integers. A job can be processed by at most one machine at any time, and each machine can process at most one job at a time. If preemption is not allowed, any job once started must be run to completion; if preemption is allowed, jobs may be interrupted during their execution, and resumed later on any machine. The objective is to find a schedule that minimizes the sum of weighted completion times, where job j is assumed to have a non-negative weight w_j. Since this problem is NP-hard (except for some trivial special cases), our goal is to find a near-optimal schedule with a provable approximation guarantee. Following the standard classification scheme of Graham et al. [13], the problems we consider are $R|r_{ij}|\sum w_j C_j$ and $R|r_{ij}$, pmtn $|\sum w_j C_j$.

The parallel-machine scheduling problem just described involves two decisions: first, the "assignment" (or "routing") decision for each job; and second, the sequencing decision for each machine given the set of jobs it processes. This viewpoint naturally leads to a nonlinear programming formulation on binary variables, which can be reformulated as a convex integer program. Relaxations of this latter problem form the basis of the approximation algorithms discussed here.

The rest of this section is organized as follows: we begin with the problem of scheduling unrelated machines without release dates, the simplest (and chronologically the first) problem on which many of the key ideas are illustrated. We then discuss the case in which preemption is allowed.

16.2.1 $R \,||\, \sum w_j C_j$

Recall that we view the scheduling problem as a two-stage problem of first assigning jobs to machines optimally, and then optimally sequencing the jobs in each machine. This latter problem is a $1 \,||\, \sum w_j C_j$ scheduling problem, which can be solved optimally using Smith's rule [14]: the jobs assigned to machine i are scheduled in decreasing order of w_j/p_{ij}, with ties broken arbitrarily. For convenience, we assume that machine i has an ordering $<_i$ of all the jobs such that $j <_i k$ if $w_j/p_{ij} > w_k/p_{ik}$. (Thus, ties are resolved consistent with the ordering $<_i$.) This observation reduces the scheduling problem to the assignment problem, which we formulate using the indicator variables x_{ij}, representing whether or not job j is assigned to machine i. The nonlinear integer program we thus obtain is

$$
\begin{aligned}
IQP \qquad & \text{Min} \sum_j w_j \left\{ \sum_i x_{ij} \left(p_{ij} + \sum_{k:k<_i j} x_{ik} p_{ik} \right) \right\} \\
& \text{subject to:} \\
& \sum_i x_{ij} = 1, \quad \forall j \\
& x_{ij} \in \{0,1\}, \quad \forall i, j
\end{aligned}
$$

The two sets of constraints ensure that each job is assigned to exactly one machine. Thus, of the m terms in the expression within the braces in the objective function, exactly one — the one corresponding to the machine to which j is assigned — will be nonzero, and this nonzero expression is exactly the sum of job j's processing time and the processing times of all higher priority jobs running on the same machine as j. The correctness of the formulation IQP is immediate.

The formulation using assignment variables suggests a natural decomposition of the given parallel machine scheduling problem into a set of single machine scheduling problems. Therefore, it is convenient to rewrite the objective function of IQP from a "machine" point of view. Specifically, by interchanging the first two summations, the objective function can be rewritten as $\sum_i \Phi(i)$, where

$$
\Phi(i) \equiv \sum_j w_j x_{ij} \left(p_{ij} + \sum_{k:k<_i j} x_{ik} p_{ik} \right)
$$

is the cost incurred by machine i. Note that $\Phi(i)$ is a function of the n variables $x_{i1}, x_{i2}, \ldots, x_{in}$. To check if it is convex, we compute its Hessian $D(i) = (d_{jk})$, which is an $n \times n$ matrix. Relabel the jobs, if necessary, so that $1 <_i 2 <_i, \ldots, <_i n$. Then, d_{jk} is exactly the coeficient of the term $x_{ij}x_{ik}$ in the expression for $\Phi(i)$; this is $w_k p_{ij}$, if $j < k$, $w_j p_{ik}$, if $j > k$, but zero, if $j = k$. Thus,

$$
D(i) = \begin{bmatrix}
0 & w_2 p_{i1} & w_3 p_{i1} & \cdots & w_{n-1} p_{i1} & w_n p_{i1} \\
w_2 p_{i1} & 0 & w_3 p_{i2} & \cdots & w_{n-1} p_{i2} & w_n p_{i2} \\
w_3 p_{i1} & w_3 p_{i2} & 0 & \cdots & w_{n-1} p_{i3} & w_n p_{i3} \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
w_{n-1} p_{i1} & w_{n-1} p_{i2} & w_{n-1} p_{i3} & \cdots & 0 & w_n p_{in-1} \\
w_n p_{i1} & w_n p_{i2} & w_n p_{i3} & \cdots & w_n p_{in-1} & 0
\end{bmatrix} \tag{16.1}
$$

Moreover, letting $x(i) = (x_{i1}, x_{i2}, \ldots, x_{in})$ and $c(i) = (w_1 p_{i1}, w_2 p_{i2}, \ldots, w_n p_{in})$, we can rewrite $\Phi(i)$ as

$$
\Phi(i) = c(i)^T x(i) + \frac{1}{2} x(i)^T D(i) x(i) \tag{16.2}
$$

Clearly, $\Phi(i)$ is convex if and only if its Hessian $D(i)$, shown in Equation (16.1), is positive semidefinite. Recall that a matrix is positive semidefinite if and only if all its principal minors have nonnegative

determinants. As the submatrix

$$\begin{bmatrix} 0 & w_2 p_{i1} \\ w_2 p_{i1} & 0 \end{bmatrix}$$

has a negative determinant (when $w_2, p_{i1} > 0$), $D(i)$ is not necessarily positive semidefinite. Even though $D(i)$ fails to be positive semidefinite, its form begs for considering the related matrix

$$\hat{D}(i) = \begin{bmatrix} w_1 p_{i1} & w_2 p_{i1} & w_3 p_{i1} & \cdots & w_{n-1} p_{i1} & w_n p_{i1} \\ w_2 p_{i1} & w_2 p_{i2} & w_3 p_{i2} & \cdots & w_{n-1} p_{i2} & w_n p_{i2} \\ w_3 p_{i1} & w_3 p_{i2} & w_3 p_{i3} & \cdots & w_{n-1} p_{i3} & w_n p_{i3} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ w_{n-1} p_{i1} & w_{n-1} p_{i2} & w_{n-1} p_{i3} & \cdots & w_{n-1} p_{in-1} & w_n p_{in-1} \\ w_n p_{i1} & w_n p_{i2} & w_n p_{i3} & \cdots & w_n p_{in-1} & w_n p_{in} \end{bmatrix} \tag{16.3}$$

obtained by adding the terms that are "obviously" missing from the diagonal.

Lemma 16.1

The matrix $\hat{D}(i)$ is positive semidefinite.

Proof

We show that every principal submatrix has a nonnegative determinant. Consider the rth principal submatrix. Divide the (j,k)th entry by $p_{ij} p_{ik}$ to get the matrix

$$\begin{bmatrix} w_1/p_{i1} & w_2/p_{i2} & \cdots & w_r/p_{ir} \\ w_2/p_{i2} & w_2/p_{i2} & \cdots & w_r/p_{ir} \\ \vdots & \vdots & \ddots & \vdots \\ w_r/p_{ir} & w_r/p_{ir} & \cdots & w_r/p_{ir} \end{bmatrix}$$

Subtracting the second row from the first, the third row from the second, etc., we obtain a lower triangular matrix, with the jth diagonal entry being $w_j/p_{ij} - w_{j+1}/p_{i,j+1}$, which, by our labeling of the jobs, is nonnegative. The lemma follows by noting that all the operations transforming $\hat{D}(i)$ to the lower triangular matrix preserve the sign of the determinant. □

Lemma 16.1 motivates us to consider working with $\hat{D}(i)$ instead of $D(i)$, so we write $\Phi(i)$ as

$$\Phi(i) = c(i)^T x(i) + \frac{1}{2} x(i)^T \hat{D}(i) x(i) - \frac{1}{2} \sum_{j=1}^{n} w_j p_{ij} x_{ij}^2 \tag{16.4}$$

As $x_{ij} \in \{0,1\}, x_{ij}^2 = x_{ij}$, so $\sum_{j=1}^{n} w_j p_{ij} x_{ij}^2 = \sum_{j=1}^{n} w_j p_{ij} x_{ij} = c(i)^T x(i)$, yielding

$$\Phi(i) = \frac{1}{2}\left(c(i)^T x(i) + \frac{1}{2} x(i)^T \hat{D}(i) x(i) \right) \tag{16.5}$$

This discussion leads to a reformulation of the scheduling problem as the following convex integer programming problem:

$$(ICQP) \qquad \text{Min} \sum_i \frac{1}{2}(c(i)^T x(i) + x(i)^T \hat{D}(i) x(i))$$

subject to:

$$\sum_i x_{ij} = 1, \quad \forall j$$

$$x_{ij} \in \{0,1\}, \quad \forall i, j$$

We thus obtain two nonlinear integer programming formulations of the scheduling problem: the formulation IQP has a nonconvex objective, whereas the formulation ICQP has a convex objective. Of course, these integer programming problems are hard to solve, so it is natural to investigate their (fractional) relaxations.

First we consider the formulation (IQP), whose fractional relaxation is

$$\text{(QP)} \qquad \text{Min} \sum_j w_j \left\{ \sum_i x_{ij} \left(p_{ij} + \sum_{k:k<_i j} x_{ik} p_{ik} \right) \right\}$$

subject to:

$$\sum_i x_{ij} = 1, \quad \forall j$$
$$x_{ij} \geq 0, \quad \forall i, j$$

Let x^* be an optimal solution to this problem. Our goal is to now convert this fractional assignment to an integral one. Consider the (random) assignment in which job j is sent to machine i with probability x^*_{ij}, independent of all other jobs. Since $\sum_i x^*_{ij} = 1$, every job is assigned to some machine. Moreover, the expected completion time of job j is exactly

$$\sum_i x^*_{ij} \left(p_{ij} + \sum_{k:k<_i j} x^*_{ik} p_{ik} \right)$$

Therefore, the expected cost of this random schedule is exactly the cost of the fractional solution x^*. Standard techniques can be used to derandomize this scheme, and thus to find an optimal schedule. Thus, finding an optimal (fractional) solution to QP is no easier than solving the scheduling problem $R \,||\, \sum_j w_j C_j$. In other words, given an optimal (possibly fractional) solution to QP, it is easy to find an optimal integer solution; however, finding an optimal solution to QP is difficult.

We next turn to the formulation ICQP and its fractional relaxation

$$\text{(CQP)} \qquad \text{Min} \sum_i \frac{1}{2} [c(i)^T x(i) + x(i)^T \hat{D}(i) x(i)]$$

subject to:

$$\sum_i x_{ij} = 1, \quad \forall j$$
$$x_{ij} \geq 0, \quad \forall i, j$$

Recall that ICQP was obtained from IQP by judiciously replacing some of the x_{ij} by x^2_{ij}. This transformation is valid if $x_{ij} \in \{0, 1\}$, but not if $x_{ij} \in (0, 1)$. In the latter case, $x^2_{ij} < x_{ij}$, so the objective function of CQP underestimates the true optimal cost. In this sense, CQP is a true relaxation of the problem $R \,||\, \sum_j w_j C_j$. Fortunately, CQP is a simple complex quadratic programming problem subject to linear constraints, and so can be solved efficiently in polynomial time. Thus, the situation here is in sharp contrast to that we had earlier: the fractional relaxation CQP is easy to solve, but there could be fractional solutions with cost strictly below that of an optimal integer solution. The task now is to somehow recover a good schedule from an optimal solution to CQP. The randomized assignment scheme described earlier is a natural candidate, and we analyze this next.

Let x^* be an optimal solution to CQP, with $\Phi^*(i)$ representing the cost incurred by machine i in this relaxation. Consider the (random) schedule obtained by assigning, as before, job j to machine i with probability x^*_{ij}, independent of all other jobs. The expected cost incurred by machine i in the schedule thus obtained, $E[\Phi(i)]$, is

$$\sum_j w_j x^*_{ij} \left(p_{ij} + \sum_{k:k<_i j} x^*_{ik} p_{ik} \right)$$

which, by Equation (16.4), is

$$c(i)^T x^*(i) + \frac{1}{2} x^*(i)^T \hat{D}(i) x(i) - \frac{1}{2} \sum_{j=1}^{n} w_j p_{ij} (x_{ij}^*)^2$$

Simplifying this expression, we have

$$\begin{aligned} E[\Phi(i)] &\leq \frac{1}{2} c(i)^T x^*(i) + \frac{1}{2} x^*(i)^T \hat{D}(i) x(i) + \frac{1}{2} c(i)^T x^*(i) \\ &= \Phi^*(i) + \frac{1}{2} c(i)^T x^*(i) \qquad (16.6) \\ &\leq 2\Phi^*(i) \qquad (16.7) \end{aligned}$$

where expression (16.7) follows expression (16.6) because $c(i)^T x^*(i)/2$ is a lower bound on $\Phi^*(i)$.

We have thus shown that the expected cost incurred by any machine i is at most twice its cost in the (fractional) relaxation CQP. Can this be improved? One possibility is to use the

$$\Phi^*(i) \geq c(i)^T x^*(i)$$

instead of the weaker

$$\Phi^*(i) \geq \frac{1}{2} c(i)^T x^*(i)$$

After all, $\Phi^*(i)$ is a lower bound on the cost incurred by machine i, and it seems natural to expect it to be at least as much as the sum of the weighted *processing times* of the jobs assigned to it. This may not be true because we are dealing with a fractional relaxation, which allows a job to be split across multiple machines. For instance, consider a single job with $w = 1$ and $p_i = 1$ on two machines, the optimal schedule splits this job into two pieces, each piece processed by a different machine. It is easy to check that $\Phi^*(1) = \Phi^*(2) = 3/8$, whereas the total weighted processing time on each of the machines is $1/2 > 3/8$. Notice, however, that $\Phi(i)$ must be at least $c(i)^T x(i)$ for all (integral) feasible schedules of the original problem, so we could explicitly enforce this inequality as a constraint. We are thus led to the following stronger formulation of the problem $R \,||\, \sum w_j C_j$:

$$\begin{aligned} \text{(SCQP)} \qquad & \text{Min} \sum_i \Phi(i) \\ & \text{subject to:} \\ & \quad \Phi(i) \geq \frac{1}{2}(c(i)^T x(i) + x(i)^T \hat{D}(i) x(i)), \quad \forall i \\ & \quad \Phi(i) \geq c(i)^T x(i), \quad \forall i \\ & \quad \sum_i x_{ij} = 1, \quad \forall j \\ & \quad x_{ij} \geq 0, \quad \forall i, j \end{aligned}$$

Given an optimal solution to SCQP, we can again interpret the x_{ij}^* as assignment probabilities. Assigning job j to machine i with probability x_{ij}^*, and sequencing the set of jobs assigned to any particular machine i using Smith's rule, we obtain a (random) schedule in which the expected cost of machine i is within $3/2$ of its cost in the relaxation SCQP (see Expression (16.6) and Expression (16.7)). Since this is true for every machine, the total expected cost of the schedule obtained is at most $3/2$ times the cost of the relaxed problem SCQP. Thus, we have a randomized $3/2$-approximation algorithm, assuming we can find an optimal solution to SCQP.

The formulation SCQP is no longer a simple quadratic program, but is still a convex programming problem, for which a solution within an additive error of any $\epsilon > 0$ can be obtained in polynomial-time.

This, when combined with the randomized rounding scheme, gives us a randomized $3/2+\epsilon$ approximation algorithm. As we deal with integer data (the optimal cost is also an integer), choosing any $\epsilon < 1/3$ and derandomizing the rounding algorithm using standard techniques (see [11]) yields a schedule with cost at most 3/2 of the optimal cost. We thus have the following result, whose proof follows from our discussion so far.

Theorem 16.1

Given an instance of $R \mid\mid \sum w_j C_j$, let x^* be a near-optimal solution to SCQP within an additive error of 1/3. Consider the (random) assignment obtained by assigning job j to machine i with probability x^*_{ij}, and let each machine sequence its job according to Smith's rule. Let π be the resulting (random) schedule, Z^π its cost, and let Z^* be the optimal cost. Finally, let $\hat{\pi}$ be the schedule computed by derandomizing the random assignment. Then (a) $E[Z^\pi] \leq (3/2)Z^* + 1/3$; and (b) $E[Z^{\hat{\pi}}] \leq (3/2)Z^*$.

16.2.2 $R \mid \text{pmtn} \mid \sum w_j C_j$

The preemptive version of the problem of scheduling unrelated machines can be treated using similar techniques. However, we need to use a slightly different reasoning to get a valid relaxation. Recall that the contribution of job j to the cost of machine i in a nonpreemptive schedule in terms of the assignment variables is given by

$$w_j x_{ij} \left(p_{ij} + \sum_{k:k<_i j} x_{ik} p_{ik} \right)$$

This is zero if j is not assigned to i; otherwise, this is simply the weighted completion time of job j.

Consider the preemptive version of the same problem. It is easy to see that this expression is no longer correct. To derive a valid expression, we consider a simple charging scheme based on another interpretation of the objective function: assume that each unit of job j in the system incurs a cost of w_j per unit time. If x_{ij} is the portion of job j assigned to machine i, then, machine i will only be responsible for the cost incurred by these x_{ij} units of job j; the amount of job j at machine i is exactly x_{ij} in the time interval $[0, \sum_{k:k<_i j} x_{ik} p_{ik}]$, and decreases linearly to zero in the time interval $[\sum_{k:k<_i j} x_{ik} p_{ik}, x_{ij} p_{ij} + \sum_{k:k<_i j} x_{ik} p_{ik}]$. Thus, the cost incurred by the (portions of the) jobs in machine i, $\Phi(i)$, is

$$\Phi(i) \geq \sum_j w_j x_{ij} \left(\frac{x_{ij}}{2} p_{ij} + \sum_{k:k<_i j} x_{ik} p_{ik} \right)$$

This leads us to the following convex programming problem:

$$\begin{aligned}
\text{(SCPP)} \qquad & \text{Min} \sum_i \Phi(i) \\
& \text{subject to:} \\
& \Phi(i) \geq \frac{1}{2}(x(i)^T \hat{D}(i) x(i)), \quad \forall i \\
& \Phi(i) \geq c(i)^T x(i), \quad \forall i \\
& \sum_i x_{ij} = 1, \quad \forall j \\
& x_{ij} \geq 0, \quad \forall i, j
\end{aligned}$$

The only difference from the nonpreemptive case is that the terms $c(i)^T x(i)/2$ are missing from the first set of constraints. We can solve this problem (SCPP) to obtain a near optimal solution x^*. We can think of allocating job-fractions according to x^*, but we will then have the difficult task of making sure no job is run simultaneously on multiple machines. Since our approach is to reduce everything to dealing

with independent single-machine problems, we do not attempt to implement this fractional solution x^* directly. Rather, we resort to finding a nonpreemptive schedule $\hat{x}$ by applying randomized rounding on x^*. The schedule $\hat{x}$ has expected cost that is at most

$$\sum_i c(i)^T x(i) + \frac{1}{2}(x(i)^T \hat{D}(i) x(i)),$$

which is within twice the cost of SCPP. We thus have the following result.

Theorem 16.2

Given an instance of $R|\,\text{pmtn}\,|\sum w_j C_j$, *let* x^* *be a near-optimal solution to SCPP within an additive error of 1/3. Consider the (random) assignment obtained by assigning job* j *to machine* i *with probability* x^*_{ij}, *and let each machine sequence its job according to Smith's rule. Let* π *be the resulting (random) schedule,* Z^π *its cost, and let* Z^* *be the optimal cost. Finally, let* $\hat{\pi}$ *be the schedule computed by derandomizing the random assignment. Then (a)* $E[Z^\pi] \geq 2Z^* + 1/3$, *and (b)* $E[Z\hat{\pi}] \geq 2Z^*$.

In addition, we also get a bound on the *power of preemption.* Since we find a nonpreemptive schedule within a factor of 2 of the best possible preemptive schedule, we can also conclude that by allowing preemption we can reduce the sum of weighted completion times by at most a factor of 2.

16.3 Scheduling Unrelated Machines with Release Dates

We now turn to the problem of scheduling unrelated machines when the jobs have (possibly machine-dependent) release dates. Specifically, a job j can be processed on machine i only at or after time r_{ij} . As before, we first consider the nonpreemptive version followed by the preemptive one.

16.3.1 $R\,|\,r_{ij}\,|\,\sum w_j C_j$

Recall that our basic approach is to think of the scheduling problem as consisting of two subproblems: first, the routing or assignment of jobs to machines; and second, the sequencing of the jobs assigned to each machine. The latter is simply a set of m independent single machine scheduling problems, each of which can be solved to optimality by Smith's rule if all the jobs are released at the same time. We would of course like to pursue this approach for the problem with release dates. To do so, however, we need to overcome two difficulties: first, the single machine scheduling problem $1\,|\,r_j\,|\sum w_j C_j$ is itself NP-hard [15], so the single machine sequencing problem is itself nontrivial; and second, even if we did know the optimal sequencing rule for all subsets of jobs assigned to a given machine, the resulting cost, as a function of the assignment variables, may not be "convex" (or easily "convexifiable"). Thus an optimal solution to $1\,|\,r_j\,|\sum w_j C_j$ may actually not be helpful to our approach. However, we do know that scheduling according to Smith's rule results in a cost function that can be convexified easily. Therefore, we would like the sequencing rule to be "similar" to Smith's rule.

For the single machine scheduling problem with release dates, when is Smith's rule optimal? Fix a machine i, and suppose machine i has an ordering $<_i$ of all the jobs such that $j <_i k$ if $w_j/p_{ij} > w_k/p_{ik}$. Clearly, if the jobs are released in the order of their priority, Smith's rule is optimal, i.e., if $r_{ij} \leq r_{ik}$ for $j <_i k$, then scheduling j before k is optimal. Thus, if $j <_i k$, and both are assigned to machine i and are available for processing at any time t, then we can schedule j before k in an optimal schedule. Therefore, in any interval of time during which no job is released, we may assume that jobs are scheduled according to Smith's rule in an optimal schedule. This motivates us to work with intervals of time during which no job is released; these are called *slots*.

We work with a fixed machine i; without loss of generality, we assume that jobs are labeled so that $1 <_i 2 \cdots <_i n$. Let $\rho_{i,1} < \rho_{i,2} < \cdots < \rho_{i,q_i}$ be an ordering of the distinct release dates in the set $\{r_{i1}, r_{i2}, \ldots, r_{in}\}$. The interval $[\rho_{i,k}, \rho_{i,k+1}]$ will be called slot (i,k) or simply slot k if i is evident. Job j is eligible for slot (i,k) if and only if $r_{ij} \leq \rho_{i,k}$. By our discussion, jobs scheduled within a slot are sequenced according to Smith's rule. So it is tempting to treat the scheduling problem as one of allocating jobs to slots (instead of machines), with slot (i,k) "available for processing jobs only in the interval $[\rho_{i,k}, \rho_{i,k+1})$. If we do so, note that we need to allow "fractional" jobs to be assigned to slots; otherwise we implicitly impose an additional constraint, namely that no job straddles any slot, which is simply absent from the original problem. Because we eventually plan to relax the assignment constraints anyway, and because the overall effect of allowing this flexibility is to underestimate the cost, this does not pose a problem. Let $\Phi(i,k)$ be the cost incurred by slot k of machine i, and let $\Phi(i) = \sum_k \Phi(i,k)$ be the cost incurred by machine i. Clearly,

$$\Phi(i,k) = \sum_j w_j x_{ikj} \left(\rho_{i,k} + p_{ij} + \sum_{l:l<_i j} x_{iklpil} \right)$$

where x_{ikj} is the portion of job j assigned to slot (i,k). We are thus led to the following nonlinear programming problem:

$$\text{(QPR)} \qquad \text{Min} \sum_{i,k} \sum_j w_j x_{ikj} \left(\rho_{i,k} + p_{ij} + \sum_{l:l<_i j} x_{iklpil} \right)$$

subject to:

$$\sum_{i,k} x_{ikj} = 1, \quad \forall j$$

$$\sum_j x_{ikj} p_{ij} \leq \rho_{i,k+1} - \rho_{i,k}, \quad \forall i,k$$

$$x_{ikj} = 0, \quad \text{if } r_{ij} < \rho_{i,k}, \quad \forall j$$

$$x_{ikj} \geq 0, \quad \forall i,k,j$$

We note that given any feasible schedule to the original problem, we can set the assignment variable x_{ikj} to be the fraction of job j processed in slot (i,k). The cost of processing a job by a machine (i.e., its weighted completion time) is computed for these different pieces (processed in different slots) independently. This, and the additional flexibility that each slot has to reorder its assigned jobs, results in an underestimate of the actual weighted completion time of the discrete schedule. Thus, QPR is a relaxation of the problem $R \mid r_{ij} \mid \sum w_j C_j$.

As before, the objective function $\Phi(i,k)$ is not necessarily convex. However, the same trick can be used to make this a convex function: we replace

$$w_j p_{ij} x_{ikj}$$

in $\Phi(i,k)$ by

$$w_j p_{ij} \left(\frac{x_{ikj}}{2} + \frac{x_{ikj}^2}{2} \right)$$

which has the effect of further underestimating the actual objective value, causing it to remain a valid relaxation. This convex relaxation is formally stated below:

$$\text{(CQPR)} \qquad \text{Min} \sum_{i,k} \sum_{j} \left\{ w_j \rho_{i,k} x_{ikj} + w_j p_{ij} \frac{x_{ikj}}{2} + w_j x_{ikj} \left(\frac{p_{ij} x_{ikj}}{2} + \sum_{l:l <_i j} x_{ikl} p_{il} \right) \right\}$$

subject to:

$$\sum_{i,k} x_{ikj} = 1, \quad \forall j$$

$$\sum_{j} x_{ikj} p_{ij} \leq \rho_{i,k+1} - \rho_{i,k}, \quad \forall i,k$$

$$x_{ikj} = 0, \quad \text{if } r_{ij} < \rho_{i,k}, \quad \forall j$$

$$x_{ikj} \geq 0, \quad \forall i,k,j$$

An optimal solution x^* to CQPR can be found in polynomial time. As before, we interpret x^*_{ijk} as the probability with which job j is assigned to slot (i,k), thus we consider each job j and independently assign it to a slot (i,k) with probability x^*_{ikj}. Let T_{ik} be the set of jobs assigned to slot (i,k). A machine i processes its assigned jobs in the order of their slots, sequencing the jobs within a slot according to $<_i$. Thus machine i processes all the jobs in T_{i1} first, then the jobs in T_{i2}, etc. We next prove that the cost of this schedule is at most twice the optimal cost.

Theorem 16.3

Let x^* be an optimal solution to CQPR. Assign jobs to slots independently, with job j assigned to slot k of machine i with probability x^*_{ikj}. Sequence the jobs at any machine in order of the slots, and sequence the jobs within a slot according to Smith's rule. The cost of the resulting schedule is at most twice the optimal cost.

Proof

Let $T_{i,k}$ be the set of jobs assigned to slot k of machine i. We shall show that the expected cost of the jobs in $T_{i,k}$ is at most $2\hat{\Phi}^*(i,k)$. Let $t_{i,k}$ be the epoch at which the first job from the set $T_{i,k}$ is processed. Let $\hat{x}_{ikj}$ be the indicator function of the resulting (random) schedule. Then, the expected cost of the jobs in $T_{i,k}$ is clearly

$$E\left[\sum_{j} w_j \hat{x}_{ikj} \left(t_{i,k} + p_{ij} + \sum_{l:l <_i j} p_{il} \hat{x}_{ikl} \right) \right]$$

which equals

$$\sum_{j} w_j E\left[\hat{x}_{ikj} t_{i,k} \right] + \sum_{j} w_j E\left[\hat{x}_{ikj} \left(p_{ij} + \sum_{l:l <_i j} p_{il} \hat{x}_{ikl} \right) \right] \tag{16.8}$$

Fix a job j. The term corresponding to j in the second expectation is simply

$$w_j x^*_{ikj} \left(p_{ij} + \sum_{l:l <_i j} p_{il} x^*_{ikl} \right)$$

Using an argument similar to the one in Section 16.2.1, this expression can be seen to be at most twice

$$w_j p_{ij} \frac{x^*_{ikj}}{2} + w_j x_{ikj} \left(\frac{p_{ij} x^*_{ikj}}{2} + \sum_{l:l <_i j} x^*_{ikl} p_{il} \right)$$

which is identical to the last two terms in the objective function of the relaxation CQPR. So to prove the theorem, it is enough to show that

$$E[\hat{x}_{ikj}t_{i,k}] \leq 2x^*_{ikj}\rho_{i,k}$$

for any job j and any slot (i,k). Since $\hat{x}_{ikj} \in \{0,1\}$

$$E[\hat{x}_{ikj}t_{i,k}] = \text{Prob}[\hat{x}_{ikj} = 1]E[t_{i,k}|\hat{x}_{ikj} = 1]$$

This latter expectation can be found by elementary reasoning. A job from $T_{i,k}$ will be processed as soon as at least one of the jobs in $T_{i,k}$ is released, and all of the jobs in the sets $T_{i,1}, T_{i,2}, \ldots, T_{i,k-1}$ are processed. Thus,

$$\begin{aligned}
E[t_{i,k} \mid \hat{x}_{ikj} = 1] &\leq E\left[\rho_{i,k} + \sum_{l=1}^{k-1}\sum_j \hat{x}_{ilj}p_{il}\right] \\
&= \rho_{i,k} + \sum_{l=1}^{k-1} E\left[\sum_j \hat{x}_{ilj}p_{il}\right] \\
&= \rho_{i,k} + \sum_{l=1}^{k-1}(\rho_{i,l+1} - \rho_{i,l}) \\
&\leq 2\rho_{i,k}
\end{aligned}$$

where the penultimate equation follows from a constraint of CQPR. Thus,

$$E[\hat{x}_{ikj}t_{i,k}] = \text{Prob}[\hat{x}_{ikj} = 1]E[t_{i,k} \mid \hat{x}_{ikj} = 1] \leq 2x^*_{ikj}\rho_{i,k}$$

and the proof is complete. □

16.3.2 $R \mid r_{ij}, \text{pmtn} \mid \sum w_j C_j$

By now all the key ideas are in place: the result here is mainly a matter of combining the ideas of Section 16.2.2 and Section 16.3.1, so we merely sketch the details.

As in $R \mid \text{pmtn} \mid \sum w_j C_j$, we first derive a lower bound on the weighted completion time of job j. The analogous result here (proof omitted) is

$$\sum_j w_j C_j \geq \sum_j w_j \sum_k x_{ikj}\left(\rho_{ik} + \frac{x_{ikj}}{2}p_{ij} + \sum_{l:l<_i j} x_{ikl}p_{il}\right)$$

The convex relaxation now becomes

$$\text{(CQPPR)} \qquad \text{Min} \sum_{i,k}\sum_j \left\{w_j\rho_{i,k}x_{ikj} + w_j x_{ikj}\left(\frac{p_{ij}x_{ikj}}{2} + \sum_{l:l<_i j} x_{ikl}p_{il}\right)\right\}$$

subject to:

$$\begin{aligned}
&\sum_{i,k} x_{ikj} = 1, \quad \forall j \\
&\sum_j x_{ikj}p_{ij} \leq \rho_{i,k+1} - \rho_{i,k}, \quad \forall i,k \\
&x_{ikj} = 0, \quad \text{if } r_{ij} < \rho_{i,k}, \ \forall j \\
&x_{ikj} \geq 0, \quad \forall i,k,j
\end{aligned}$$

Again, the main difference from the nonpreemptive version is the lack of the term $w_j p_{ij} x_{ikj}/2$ in the objective function of the relaxation. Using the same trick of Section 16.2.1, we can improve this relaxation by requiring $\sum_{i,k,j} w_j p_{ij} x_{ikj}$ to be a lower bound on the optimal cost. This strengthened formulation can be solved in polynomial time. Applying randomized rounding on an optimal solution x^* of CQPPR gives us a schedule whose cost is within a factor of 3 of the best possible, this also yields a bound on the power of preemption in this setting.

16.4 Cardinality Constraints

In all of the models considered so far, we find a good fractional solution of an appropriate relaxation and then use randomized rounding to find a (random) integer solution whose expected cost is within a (small) constant factor of the optimal cost. In this section we consider similar scheduling problems except that we have additional constraints on the number of jobs any given machine can process. In the presence of such cardinality constraints, the randomized rounding algorithm may not even produce a *feasible* solution, so we have to rely on a different set of techniques. Our purpose here is to illustrate a deterministic rounding algorithm — called *pipage* rounding — that can be used in such situations. For concreteness we focus only on the problem $R \,||\, \sum w_j C_j$ with additional cardinality constraints. The model with release dates can be dealt with in a similar fashion.

Let n_i be the maximum number of jobs that machine i can process. Then the problem $R \,||\, \sum w_j C_j$ can be formulated as the following integer programming problem.

$$\text{(IQPC)} \qquad \text{Min} \sum_j w_j \left\{ \sum_i x_{ij} \left(p_{ij} + \sum_{k:k<_i j} x_{ik} p_{ik} \right) \right\}$$

subject to:

$$\sum_i x_{ij} = 1, \quad \forall j$$

$$\sum_j x_{ij} \leq n_i, \quad \forall i$$

$$x_{ij} \in \{0, 1\} \quad \forall i, j$$

Fix a machine i, and relabel the jobs so that $1 <_i 2 <_i \cdots <_i n$. Let $c(i)$ be the n-vector $(w_1 p_{i1}, w_2 p_{i2}, \ldots, w_n p_{in})$, and let $x(i)$ be the n-vector $(x_{i1}, x_{i2}, \ldots, x_{in})$. As before, let

$$\hat{D}(i) = \begin{bmatrix} w_1 p_{i1} & w_2 p_{i1} & w_3 p_{i1} & \cdots & w_{n-1} p_{i1} & w_n p_{i1} \\ w_2 p_{i1} & w_2 p_{i2} & w_3 p_{i2} & \cdots & w_{n-1} p_{i2} & w_n p_{i2} \\ w_3 p_{i1} & w_3 p_{i2} & w_3 p_{i3} & \cdots & w_{n-1} p_{i3} & w_n p_{i3} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ w_{n-1} p_{i1} & w_{n-1} p_{i2} & w_{n-1} p_{i3} & \cdots & w_{n-1} p_{in-1} & w_n p_{in-1} \\ w_n p_{i1} & w_n p_{i2} & w_n p_{i3} & \cdots & w_n p_{in-1} & w_n p_{in} \end{bmatrix}$$

The following convex programming problem is a relaxation of the problem of scheduling unrelated machines with cardinality constraints; if the $x_{ij} \in \{0, 1\}$, it is an exact formulation. (As discussed earlier in

the context of the problem $R\|\sum w_j C_j$, we add an additional set of constraints, which is redundant for the integer program but strengthens the relaxation.)

$$
\begin{aligned}
\text{(SCPC)} \qquad & \text{Min} \sum_i \Phi(i) \\
& \text{subject to:} \\
& \Phi(i) \geq \frac{1}{2}(c(i)^T x(i) + x(i)^T \hat{D}(i) x(i)), \quad \forall i \\
& \Phi(i) \geq c(i)^T x(i), \quad \forall i \\
& \sum_i x_{ij} = 1, \quad \forall j \\
& \sum_j x_{ij} \leq n_i, \quad \forall i \\
& x_{ij} \geq 0, \quad \forall i, j
\end{aligned}
$$

Suppose x^* is an optimal solution to SCPC. A simple randomized rounding argument, similar to the one used in Section 16.2.1, shows that the expected cost of the (random) schedule found is at most 3/2 of the cost of SCPC. However, the schedule found may violate the cardinality constraint at some machine. Moreover, there does not seem to be any simple way to satisfy the cardinality constraints if we resort to randomized rounding. As we discussed earlier, we address this problem by relying on a deterministic rounding algorithm called *pipage* rounding.

Pipage rounding is a deterministic rounding algorithm that is useful in handling cardinality constraints, budget constraints, etc. The main idea is to start with a fractional solution and to gradually and systematically reduce the number of fractional components, while always "improving" the objective value.

Recall that our proof of the 3/2-approximation algorithm for the problem $R \,||\, \sum w_j C_j$ proceeded by finding an optimal solution x^* to *SCQC* and showing that its cost, which is

$$
F(x^*) \equiv \sum_j w_j \left\{ \sum_i x^*_{ij} \left(p_{ij} + \sum_{k:k<_i j} x^*_{ik} p_{ik} \right) \right\}
$$

is within 3/2 of the optimal cost of SCQC. We shall now round this fractional solution deterministically to obtain a solution $\hat{x}$ with $F(\hat{x}) \leq F(x^*)$, so the cost of $\hat{x}$ will be within 3/2 of the optimal.

Consider the bipartite graph with m nodes — one for each machine — on one side, and n nodes — one for each job — on the other. Let the weight of edge (i, j) be x^*_{ij}. The assignment constraints imply that the degree of every job-node is exactly 1; the cardinality constraints imply that the sum of weights of the edges incident to machine-node i is at most n_i. Since our goal is to reduce the number of fractional components of x^*, we look at the subgraph defined by the fractional x^*_{ij} variables. We shall call this subgraph $G(x^*)$.

Suppose $G(x^*)$ contains a cycle $C = (e_1, e_2, \ldots, e_{2l})$. Consider the sets $M_1 = \{e_1, e_3, \ldots, e_{2l-1}\}$, and $M_2 = \{e_2, e4, \ldots, e_{2l}\}$. Let

$$
\epsilon_1 = \min\{\min_{e \in M_1}(1 - x_e), \min_{e \in M_2} x_e)\}
$$

and

$$
\epsilon_2 = \min\{\min_{e \in M_1} x_e, \min_{e \in M_2}(1 - x_e)\}
$$

There are two natural ways to modify the weights on this cycle so as to reduce the number of fractional components and preserve the node degrees: increase the weights of the edges in M_1 by ϵ_1 and decrease the weights of the edges in M_2 by the same amount; or decrease the weights of the edges in M_1 by ϵ_2 and

increase the weights of the edges in M_2 by the same amount. Let $x^*_{\epsilon 1}$ and $x^*_{\epsilon 2}$ be the solutions obtained. We shall now argue that

$$\min\{F(x^*_{\epsilon_1}), F(x^*_{\epsilon_2})\} \leq F(x^*)$$

We do this by considering how $F(x^*)$ changes when the x variables are altered along a cycle as suggested earlier. Observe that the solutions we obtain — $x^*_{\epsilon_1}$ and $x^*_{\epsilon_2}$ — are essentially one dimensional modifications of x^*: we arrive at these by increasing $x^*_{e_1}$ by ϵ_1 and decreasing $x^*_{e_1}$ by ϵ_2, respectively. This motivates us to study $F(x^*_\epsilon)$ where $\epsilon \in [-\epsilon_2, \epsilon_1]$ is the amount added to $x^*_{e_1}$. Since the x variables change only along the cycle C, every machine-node in C will have the weights of one of its edges increased by ϵ and the weights of another of its edges decreased by ϵ; from the form of $F(x)$, it is clear that $F(\cdot)$ is a concave function of ϵ, so achieves its minimum at one of the extreme points of the feasible region. Therefore

$$\min\{F(x^*_{\epsilon_1}), F(x^*_{\epsilon_2})\} \leq F(x^*)$$

If $G(x^*)$ does not contain a cycle, it must be a collection of trees. So we pick any path connecting two nodes of degree 1, and consider the obvious two modifications along this path: we find ϵ_1 and ϵ_2 as before by looking at alternate edges along the path. The degrees of all intermediate nodes are preserved. Although the degrees of the endpoints are not preserved, the new degrees obtained in $x^*_{\epsilon_1}$ and $x^*_{\epsilon_2}$ satisfy both the assignment constraints and the cardinality constraints. The assignment constraints hold in these two new solutions because no job-node can have degree 1 (as every job is completely assigned), thus any path has to be from a machine-node to another machine-node. Since the n_i are integers, and since no edge's weight is increased beyond the next integer value, the cardinality constraints cannot be violated for any machine i in $x^*_{\epsilon_1}$ and $x^*_{\epsilon_2}$. It is clear again that $F(x^*_\epsilon)$ is concave in ϵ for $\epsilon \in [-\epsilon_2, \epsilon_1]$, so it will attain its minimum at one of the endpoints.

Thus, in either case we can "round" x^* into another feasible solution x^*_ϵ such that $F(x^*_\epsilon) \leq F(x^*)$ and x^*_ϵ has fewer fractional components than x^*. Continuing this process, we eventually reach an integer solution $\hat{x}$, which is the required near-optimal solution. (Clearly, the number of steps to reach this integer solution is bounded by the number of fractional components of x^*, which we know is small.)

An analogous argument can be shown to hold for the more general problem $R \mid r_{ij} \mid \sum w_j C_j$, the details are left to the interested reader.

16.5 Conclusions

In this chapter we saw how the problem of scheduling unrelated machines (and several variations) can be reduced to solving an assignment problem followed by a sequencing problem. This point-of-view resulted in a natural convex relaxation, which can be solved efficiently. We also saw how fractional solutions can be rounded to obtain integer solutions whose performance can be effectively bounded.

While the work discussed here appears to be very natural, such an approach was not pursued until 5 years ago. Perhaps these ideas can be pushed further, both to obtain improved guarantees and to broaden their applicability. As for the former, almost all of our analysis can be shown to be tight, so to obtain improved guarantees, we may have to obtain improved relaxations. As for the latter, while no significant advances have been made in this direction since the appearance of the original papers, we remain optimistic. It will be interesting to see new applications of such ideas, both in scheduling and, more broadly, in the field of approximation algorithms.

Acknowledgments

I thank Akshay-Kumar Katta, Joseph Leung, and Martin Skutella for their comments on an earlier version of this manuscript.

References

[1] L. Lovasz. On the shannon capacity of a graph. *IEEE Trans. Info. Theo.*, 25:1–7, 1979.

[2] M. Groetschel, L. Lovasz, and A. Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1:169–197, 1981.

[3] M. Goemans and D. P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. ACM*, 42:1115–1145, 1995.

[4] D. R. Karger, R. Motwani, and Madhu Sudan. Approximate graph coloring by semidefinite programming. In *IEEE Symposium on Foundations of Computer Science*, pp. 2–13, 1994.

[5] B. Chor and M. Sudan. A geometric approach to betweenness. *SIAM J. Disc. Math.*, 11:511–523, 1998.

[6] Y. Ye. A 0.699 approximation algorithm for Max-Bisection, *Mathematical Programming*, 90(1):101–111, 2001.

[7] M. Skutella. Semidefinite relaxations for parallel machine scheduling. In *IEEE Symp. Foundations of Computer Science*, pp. 472–481, 1998.

[8] J. Sethuraman and M. S. Squillante. Optimal scheduling of multiclass parallel machines. In *Proceding of 10th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 963–964, 1999.

[9] J. Sethuraman and M. S. Squillante. Optimal stochastic scheduling in multiclass parallel queues. In *ACM Sigmetrics Conference on Measurement and Modeling of Computer Systems*, pp. 93–102, 1999.

[10] M. Skutella. Convex quadratic programming relaxations for network scheduling problems. In *European Symposium on Algorithms*, pp. 127–138, 1999.

[11] A. Ageev and M. Sviridenko. Pipage rounding: a new method of constructing algorithms with proven performance guarantee. *J. Combinat. Opt.* (Unpublished)

[12] M. Skutella. Convex quadratic and semidefinite programming relaxations in scheduling. *J. ACM*, 48(2):206–242, 2001.

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

[14] W. E. Smith. Various optimizers for single-stage production. *Nav. Res. Log. Q.*, 3:59–66, 1956.

[15] J. K. Lenstra, A. H. G. Rinnooy Kan, and P. Brucker. Complexity of machine scheduling problems. *Ann. Disc. Math.*, 1:343–362, 1977.