FORMULATIONS, RELAXATIONS, APPROXIMATIONS, AND GAPS IN THE WORLD OF SCHEDULING

Gerhard J. Woeginger
Department of Mathematics and Computer Science, TU Eindhoven
P.O. Box 513, 5600 MB Eindhoven, The Netherlands
g.j.woeginger@tue.nl

Abstract We discuss a number of polynomial time approximation results for scheduling problems. All presented results are based on the technique of rounding the optimal solution of an underlying linear programming relaxation. We analyse these relaxations, their integrality gaps, and the resulting approximation algorithms, and we derive matching worst-case instances.

Keywords: approximation algorithm, worst-case analysis, performance guarantee, linear programming relaxation, integrality gap, scheduling.

INTRODUCTION

Most real-world optimization problems are NP-hard, and most NP-hard problems are difficult to solve to optimality. We conclude: most real-world problems are difficult to solve to optimality. A standard way of working around this (rather pessimistic) conclusion is to forget about *exact optimality*, and to satisfy oneself instead with *near-optimal* or *approximate* solutions. This leads us into the area of approximation algorithms for combinatorial optimization problems.

A combinatorial optimization problem consists of a set $\mathcal{I}$ of instances, and a family $\mathcal{F}(I)$ of feasible solutions for every instance $I \in \mathcal{I}$. Every feasible solution $F \in \mathcal{F}(I)$ comes with a non-negative cost $c(F)$. In this paper, we will only consider minimisation problems, where the objective is to determine a feasible solution of minimum possible cost. An *approximation algorithm* is an algorithm that for every instance $I \in \mathcal{I}$ returns a near-optimal solution. If it manages to do this in polynomial time, then it is called a *polynomial time* approximation algorithm. An approximation algorithm for a minimisation problem is called a *ρ-approximation algorithm*, if it always returns a near-optimal

solution with cost at most a factor ρ above the optimal cost. Such a value $\rho \geq 1$ is called a *worst-case performance guarantee* of the algorithm.

Approximations through relaxations. One standard approach for designing polynomial time approximation algorithms for a (difficult, NP-hard) optimisation problem $\mathcal{P}$ is the following:

(S1) Relax some of the constraints of the hard problem $\mathcal{P}$ to get an easier problem $\mathcal{P}'$ (the so-called *relaxation*).

(S2) Compute (in polynomial time) an optimal solution S' for this easier relaxed problem $\mathcal{P}'$.

(S3) Translate (in polynomial time) the solution S' into an approximate solution S for the original problem $\mathcal{P}$.

(S4) Analyse the quality of solution S for $\mathcal{P}$ by comparing its cost to the cost of solution S' for $\mathcal{P}'$.

Let C^{Opt} denote the optimal cost of the original problem instance, let C^{Rlx} denote the optimal cost of the relaxed instance, and let C^{App} denote the cost of the translated approximate solution. To show that the sketched approach has a performance guarantee of ρ, one usually establishes the following chain of inequalities:

$$C^{Rlx} \leq C^{Opt} \leq C^{App} \leq \rho \cdot C^{Rlx} \leq \rho \cdot C^{Opt}. \tag{1}$$

The first and the last inequality in this chain are trivial, since problem $\mathcal{P}'$ results from problem $\mathcal{P}$ by relaxing constraints. The second inequality is also trivial, since the optimal solution is at least as good as some approximate solution. The third inequality contains the crucial step in the chain, and all the analysis work goes into proving that step. This third inequality relates the relaxed solution to the approximate solution; both solutions are polynomial time computable, and hence their combinatorics will be nice and well-behaved. Thus, the chain yields the desired relation $C^{App} \leq \rho \cdot C^{Opt}$ by analysing nice, polynomial time computable objects. The analysis avoids touching the original NP-hard problem whose combinatorics is messy and complicated and hard to grasp.

Worst-case gaps and integrality gaps. Of course, we would like to make the value of the parameter ρ as small as possible: The closer ρ is to 1, the better is the performance of the approximation algorithm. How can we argue that our worst-case analysis is complete? How can we argue that we have reached the smallest possible value for ρ? That is usually done by exhibiting a so-called *worst-case instance*, that is, an instance I that demonstrates a *worst-case gap* of ρ for the approximation algorithm:

$$C_I^{App} = \rho \cdot C_I^{Opt} \qquad \text{and} \qquad C_I^{Opt} = C_I^{Rlx} \tag{2}$$

Here the left-hand equation establishes the gap, and together with the chain (1) it yields the right-hand equation. The worst-case instance (2) illustrates that our analysis of the *combined* approach (S1)–(S3) is tight. Is this the end of the story? Not necessarily. We could possibly start with the same relaxation step (S1), then solve the relaxation with the same step (S2), and then come up with a completely new (and better) translation step. How can we argue that this is not possible? How can we argue that the value ρ is already the best performance guarantee that we possibly can get out of the considered relaxation? That is usually done by exhibiting an instance J that demonstrates an *integrality gap* of ρ between the original problem and the relaxation:

$$C_J^{Opt} \;=\; \rho \cdot C_J^{Rlx} \qquad \text{and} \qquad C_J^{Opt} \;=\; C_J^{App} \tag{3}$$

The equation on the left-hand side establishes the gap, and with (1) it yields the equation on the right-hand side. In particular, we have $C_J^{App} = \rho \cdot C_J^{Rlx}$. For instance J the third inequality in the chain (1) is tight, and there is no way of proving a better performance guarantee for an approximation algorithm built around the considered relaxation. We stress that such a better approximation algorithm around the relaxation might well exist, but we will never be able to *prove* that its performance guarantee is better than ρ within the framework described above.

For $\rho > 1$, the conditions in (2) and in (3) cannot be satisfied by the same instance, as they would be contradictory. Hence, a complete analysis of an approximation algorithm within our framework always must provide *two* separate bad instances, one for the worst-case gap and one for the integrality gap.

Overview of this paper. We will illustrate the approach (S1)–(S4) with three examples from scheduling theory presented in the following three sections. For each example we provide an integer programming formulation, a relaxation, an approximation algorithm, a worst-case analysis, and two gap instances. In the conclusions section we give some pointers to the literature, and we pose one open problem.

Throughout the paper, we use the standard three-field scheduling notation (see e.g. Graham *et al.*, 1979; Lawler *et al.*, 1993).

1. MAKESPAN ON PARALLEL MACHINES

As our first example we will study $P\,||\,C_{\max}$, the problem of minimising makespan on parallel identical machines. The input consists of m identical machines $M_1, \ldots, M_m$ together with n jobs $J_1, \ldots, J_n$ with processing times $p_1, \ldots, p_n$. Every job is available for processing at time 0. The goal is to schedule the jobs such that the maximum job completion time (the so-called *makespan* $C_{\max}$) is minimised. Problem $P\,||\,C_{\max}$ is known to be NP-hard in

the strong sense (Garey and Johnson, 1979). The goal of this section is to give a first, simple illustration for the topics discussed in this paper.

Exact formulation and relaxation. Consider the following integer programming formulation (4) of problem $P\,||\,C_{\max}$. For machine M_i and job J_j, the binary variable x_{ij} decides whether J_j is assigned to M_i (in which case $x_{ij} = 1$) or whether J_j is assigned to some other machine (in which case $x_{ij} = 0$). The continuous variables L_i describe the total load (i.e. the total job processing time) assigned to machine M_i. Finally, the continuous variable C denotes the makespan:

$$\begin{array}{lll} \min & C & \\ \text{s.t.} & \sum_{i=1}^{m} x_{ij} \;=\; 1 & \text{for } j = 1, \ldots, n \\ & \sum_{j=1}^{n} x_{ij} p_j \;=\; L_i & \text{for } i = 1, \ldots, m \\ & L_i \;\leq\; C & \text{for } i = 1, \ldots, m \\ & p_j \;\leq\; C & \text{for } j = 1, \ldots, n \\ & x_{ij} \in \{0,1\} & \text{for } i = 1, \ldots, m \text{ and } j = 1, \ldots, n \end{array} \tag{4}$$

The first family of constraints states that every job has to be assigned to precisely one machine. The second family of constraints connects the assigned jobs to the machine loads, and the third family of constraints connects the machines loads to the makespan. The fourth family of constraints requires that the makespan is at least as large as any job processing time.

Since $P\,||\,C_{\max}$ is an NP-hard problem, the equivalent integer programming formulation (4) will also be NP-hard to solve. Therefore, we relax this formulation to get something simpler: We replace the integrality constraints "$x_{ij} \in \{0,1\}$" by continuous constraints "$0 \leq x_{ij} \leq 1$". Now all the variables are continuous variables, and so this relaxation is a linear program that can be solved to optimality in polynomial time. From now on we will use $P = \sum_{j=1}^{n} p_j$ to denote the overall job processing time. Furthermore, we denote the optimal objective value of (4) by C^{Opt}, and the optimal objective value of the LP-relaxation by C^{LP}. Setting $x_{ij}^{LP} \equiv 1/m$ and $L_i^{LP} \equiv P/m$ gives us a feasible solution for the LP-relaxation. In fact, it can be verified that this feasible solution is an optimal LP-solution with objective value

$$C^{LP} \;=\; \max\left\{\frac{1}{m}P,\; \max_{j=1}^{m} p_j\right\} \tag{5}$$

The approximation algorithm. Now let us design a polynomial time approximation algorithm for $P\,||\,C_{\max}$ that is based on the above ideas.

Since C^{LP} is an under-estimation of the true optimal objective value C^{Opt}, we will multiply C^{LP} by an appropriate stretching factor $\beta := 2m/(m+1)$ and reserve a time interval of length $\beta \cdot C^{LP}$ on each machine:

- Turn every machine into a corresponding bin of capacity $\beta \cdot C^{LP}$.
- Pack the processing times $p_1, \ldots, p_n$ one by one into these m bins. Every processing time is packed into the first bin (the bin with smallest index) into which it will fit.

That is a very simple and fairly natural algorithm. In the case all n processing times can be fit into the bins, the makespan of the corresponding schedule is at most

$$C^{App} \;\leq\; \beta \cdot C^{LP} \;\leq\; \beta \cdot C^{Opt} \;=\; \frac{2m}{m+1} \cdot C^{Opt}$$

This would yield a worst-case performance guarantee of $\beta = 2m/(m+1)$. Hence, it remains to be shown that we indeed can pack all the jobs into the bins. Suppose for the sake of contradiction that this is not possible. Consider the first moment in time where some job, say job J_y with processing time p_y, does not fit into any bin. Let $B_1, \ldots, B_m$ be the contents of the bins at that moment, and let $B_x = \min_i B_i$ denote the smallest contents. Since p_y does not fit into any bin,

$$p_y + B_i \;>\; \beta \cdot C^{LP} \qquad \text{for } i = 1, \ldots, m \tag{6}$$

In particular, this implies that all bins are non-empty. Next, we claim that

$$B_i + B_x \;>\; \beta \cdot C^{LP} \qquad \text{for } i = 1, \ldots, m \text{ with } i \neq x \tag{7}$$

Indeed, if $i < x$ then every job in B_x was too big to fit into B_i, and if $i > x$ then every job in B_i was too big to fit into B_x. This proves (7). Adding up the m inequalities in (6) yields

$$m \cdot p_y + \sum_{i=1}^{m} B_i \;>\; m \cdot \beta \cdot C^{LP} \tag{8}$$

Since $p_y + \sum_{i=1}^{m} B_i \leq P$, the left hand side in (8) is bounded from above by $P + (m-1)p_y$, which by (5) in turn is bounded from above by $m \cdot C^{LP} + (m-1)p_y$. Plugging this upper bound into (8) and simplifying yields that

$$p_y \;>\; \frac{\beta}{2} \cdot C^{LP} \tag{9}$$

By adding the $m-1$ inequalities in (7) to the inequality (9), we get that

$$p_y + (m-2)B_x + \sum_{i=1}^{m} B_i \;>\; \left(m - \frac{1}{2}\right) \cdot \beta \cdot C^{LP} \tag{10}$$

The left-hand side in (10) is bounded from above by $P + (m-2)B_x$, which by (5) in turn is bounded from above by $m\,C^{LP} + (m-2)B_x$. Plugging this upper bound into (8) and simplifying yields that

$$B_x \;>\; \frac{\beta}{2} \cdot C^{LP} \tag{11}$$

Combining (9) with (11) leads to

$$P \;\geq\; p_y + \sum_{i=1}^{m} B_i \;>\; (m+1) \cdot \frac{\beta}{2} \cdot C^{LP} \;=\; m \cdot C^{LP} \tag{12}$$

Since this blatantly contradicts (5), we have arrived at the desired contradiction. To summarise, the described bin packing algorithm indeed has a worst-case performance guarantee of at most $\beta = 2m/(m+1)$.

Analysis of the two gaps. Is our worst-case bound of β for the bin packing algorithm best possible? Yes, it is; consider the instance that consists of m jobs of length $1/(m+1)$ followed by m jobs of length $m/(m+1)$. Then $C^{Opt} = 1$, whereas $C^{App} = \beta$. (This instance also illustrates that for stretching factors strictly less than β, the bin packing algorithm does not necessarily end up with a feasible solution.)

Gap 1.1 *There exist instances for $P\,|\,|\,C_{\max}$ for which the gap between the optimal makespan C^{Opt} and the makespan produced by the bin packing algorithm equals $\beta = 2m/(m+1)$.*

What about the integrality gap of the LP-relaxation? Consider the instance with $n = m+1$ jobs and processing times $p_j \equiv m/(m+1)$. Since the optimal solution must put some two of these $m+1$ jobs on the same machine, we have $C^{Opt} \geq 2m/(m+1)$. For the LP-relaxation, the formula in (5) yields $C^{LP} = 1$.

Gap 1.2 *The integrality gap of the linear programming relaxation for problem $P\,|\,|\,C_{\max}$ is $\beta = 2m/(m+1)$. For large m, the gap goes to 2.*

The results derived in this section for $P\,|\,|\,C_{\max}$ are fairly weak. The main motivation for stating them was to illustrate the basic relaxational approach. The literature contains much stronger approximation results for $P\,|\,|\,C_{\max}$. Already back in the 1960s, Graham (1966, 1969) investigated simple greedy algorithms with worst-case performance guarantees $4/3 - 1/(3m)$. In the 1980s, Hochbaum and Shmoys (1987) designed a *polynomial time approximation scheme* for $P\,|\,|\,C_{\max}$: For every $\varepsilon > 0$, they provide a polynomial time approximation algorithm with worst-case performance guarantee at most $1 + \varepsilon$.

The LP-relaxation of (4) essentially describes the *preemptive* version $P \,|\, pmtn \,|\, C_{\max}$ of makespan minimisation, and the formula in (5) states the optimal preemptive makespan. Woeginger (2000) discusses preemptive versus non-preemptive makespan for *uniform* machines (that is, machines that run at different speeds). Woeginger (2000) shows that the integrality gap between $Q \,||\, C_{\max}$ and $Q \,|\, pmtn \,|\, C_{\max}$ is at most $2 - 1/m$, and that this bound is the best possible. Lenstra *et al.* (1990) discuss similar relaxations for the problem $R \,||\, C_{\max}$ with unrelated machines. They establish an upper bound of 2 for the integrality gap of the preemptive relaxation.

2. MAKESPAN UNDER COMMUNICATION DELAYS

Communication delays (see Papadimitriou and Yannakakis, 1990; Veltman *et al.*, 1990) take the data transmission times between machines into account. We will look at one of the simplest problems in this area, where all jobs have unit processing times and where all the communication delays are one time-unit long. There are n unit-length jobs $J_1, \ldots, J_n$ that are precedence constrained in the following way: if there is a precedence constraint $J_i \rightarrow J_j$ between jobs J_i and J_j, then job J_j cannot be started before job J_i has been completed. Furthermore, if jobs J_i and J_j are processed on *different* machines, then it takes one time-unit to transmit the data generated by job J_i over to job J_j; in this case, J_j cannot start earlier than one time-unit after the completion time of J_i. The number of machines is not a bottleneck, and may be chosen by the scheduler. All jobs are available at time 0, and the objective is to minimise the makespan. This problem is denoted by $P\infty \,|\, prec, p_j{=}1, c{=}1 \,|\, C_{\max}$. (Picouleau, 1992) has proved its NP-hardness.

Exact formulation and relaxation. We introduce the notation $\textsc{Succ}(j) = \{i : \; J_j \rightarrow J_i\}$ and $\textsc{Pred}(j) = \{i : \; J_i \rightarrow J_j\}$ to encode the successor and predecessor sets of job J_j. Consider the following integer programming formulation (13) of problem $P\infty \,|\, prec, p_j{=}1, c{=}1 \,|\, C_{\max}$. The continuous variable C_j denotes the completion time of job J_j. For every pair of jobs $J_i \rightarrow J_j$, the binary variable x_{ij} decides whether there is a unit-time communication delay between J_i and J_j (in which case $x_{ij} = 1$), or whether there is no delay and J_j is run immediately after J_i on the same machine (in

which case $x_{ij} = 0$). The continuous variable C denotes the makespan:

$$\begin{array}{lll} \min & C & \\ \text{s.t.} & \sum_{i \in Pred(j)} x_{ij} \ \geq \ |\textsc{Pred}(j)| - 1 & \text{for } j = 1, \ldots, n \\ & \sum_{i \in Succ(j)} x_{ji} \ \geq \ |\textsc{Succ}(j)| - 1 & \text{for } j = 1, \ldots, n \\ & C_i + 1 + x_{ij} \ \leq \ C_j & \text{for } J_i \rightarrow J_j \\ & 1 \ \leq \ C_j \ \leq \ C & \text{for } j = 1, \ldots, n \\ & x_{ij} \in \{0, 1\} & \text{for } J_i \rightarrow J_j \end{array} \tag{13}$$

Consider some fixed job J_j that completes at time C_j on machine M_z in some fixed feasible schedule. All predecessors of J_j that are processed on machines $\neq M_z$ must complete at time $C_j - 2$ or earlier. And all predecessors of J_j that are processed on machine M_z (with the exception of the last predecessor) must also complete at time $C_j - 2$ or earlier. Hence, at most one of the communication delays x_{ij} with $i \in \textsc{Pred}(j)$ can be 0, and all the others can be set to 1. All this is expressed by the first family of constraints. The second family of constraints states a symmetric condition for the successor sets. The third family of constraints connects the job completion times to the communication delays, and the fourth family of constraints connects the job completion times to the makespan.

Next, we will relax the integer programming formulation by replacing the integrality constraints "$x_{ij} \in \{0, 1\}$" by continuous constraints "$0 \leq x_{ij} \leq 1$". We get the corresponding LP-relaxation that can be solved in polynomial time. We denote an optimal solution of this LP by x_{ij}^{LP}, C_j^{LP}, and C^{LP}.

The approximation algorithm. Now let us translate the LP-solution into a "nearly" feasible IP-solution. The trouble-makers in the LP-solution are the delay variables x_{ij}^{LP} that need not be integral, whereas we would like them to be binary. A simple way of resolving this problem is threshold rounding: if $x_{ij}^{LP} < 1/2$, then we define a *rounded* variable $\tilde{x}_{ij} = 0$, and if $x_{ij}^{LP} \geq 1/2$, then we define a rounded variable $\tilde{x}_{ij} = 1$. Then we run through the jobs and fix the job completion times; the jobs are handled in topological order (that is, we handle a job only *after* all of its predecessors already have been handled). For a job J_j without predecessors, we define a *rounded* job completion time $\tilde{C}_j = 1$. For a job with predecessors, we define a rounded completion time

$$\tilde{C}_j \ = \ \max\{\tilde{C}_i + 1 + \tilde{x}_{ij} : \ i \in \textsc{Pred}(j)\} \tag{14}$$

That is, we place every job at the earliest possible moment in time without violating the precedence constraints and communication delays. Finally, we compute the *rounded* makespan $\tilde{C} = \max_j\{\tilde{C}_j\}$. Altogether, this gives us a *rounded* solution for (13).

Let us verify that the rounded values $\tilde{x}_{ij}$, $\tilde{C}_j$, and $\tilde{C}$ constitute a feasible solution of (13): if they violate a constraint from the first family of constraints, then $\tilde{x}_{ij} = \tilde{x}_{kj} = 0$ must hold for some $i, k \in \textsc{Pred}(j)$. But this would mean $x^{LP}_{ij} < 1/2$ and $x^{LP}_{kj} < 1/2$, whereas for all other $\ell \in \textsc{Pred}(j)$ we have $x^{LP}_{\ell j} \leq 1$. Consequently,

$$\sum_{i \in Pred(j)} x^{LP}_{ij} \;<\; \frac{1}{2} + \frac{1}{2} + (|\textsc{Pred}(j)| - 2) \cdot 1 \;=\; |\textsc{Pred}(j)| - 1$$

and $x^{LP}_{ij}, C^{LP}_j, C^{LP}$ would not be feasible for the LP-relaxation; a contradiction. A symmetric argument shows that the rounded solution also satisfies the second family of constraints. The third and fourth family are satisfied by the way we have computed the values $\tilde{C}_j$ and $\tilde{C}$. Hence, the rounded solution is indeed feasible for (13). What about its quality? Let us first observe that

$$1 + \tilde{x}_{ij} \;\leq\; \frac{4}{3}\,(1 + x^{LP}_{ij}) \qquad\qquad \text{for all } J_i \to J_j \tag{15}$$

First, consider the case $\tilde{x}_{ij} = 0$. Then in the inequality (15) the left-hand side equals 1, the right-hand side is at least $4/3$, and the statement clearly is true. Secondly, consider the case $\tilde{x}_{ij} = 1$. Then the inequality (15) is equivalent to $x^{LP}_{ij} \geq 1/2$, and that was precisely the condition for setting $\tilde{x}_{ij} = 1$. These two cases establish the correctness of (15). Next, let us analyse the rounded job completion times. We claim that

$$\tilde{C}_j \;\leq\; \frac{4}{3}\,C^{LP}_j \qquad\qquad \text{for all } j = 1, \ldots, n \tag{16}$$

This statement clearly holds true for all jobs without predecessors, since they satisfy $\tilde{C}_j = C^{LP}_j = 1$. For the remaining jobs, we use an inductive argument along the topological ordering of the jobs:

$$\begin{aligned}
\tilde{C}_j \;&=\; \max\left\{\tilde{C}_i + 1 + \tilde{x}_{ij} :\; i \in \textsc{Pred}(j)\right\} \\
&\leq\; \max\left\{\frac{4}{3}\,C^{LP}_i + \frac{4}{3}\,(1 + x^{LP}_{ij}) :\; i \in \textsc{Pred}(j)\right\} \\
&=\; \frac{4}{3}\,\max\left\{C^{LP}_i + (1 + x^{LP}_{ij}) :\; i \in \textsc{Pred}(j)\right\} \\
&\leq\; \frac{4}{3}\,C^{LP}_j
\end{aligned}$$

Here the first equation comes from (14); the first inequality follows from (15) and (16); the second inequality follows from the third family of constraints in (13). Finally, since all rounded job completion times are at most a factor of

$4/3$ above the job completion times in the LP-solution, the rounded makespan satisfies

$$\tilde{C} \;\leq\; \frac{4}{3}\, C^{LP} \;\leq\; \frac{4}{3}\, C^{Opt}$$

Hence, the described approximation algorithm has a worst-case performance guarantee of at most $4/3$. This result is due to (Munier and König, 1997).

Analysis of the two gaps. Now let us find a worst-case instance for the approximation algorithm: we use $n = 3k+1$ jobs that we call $J_1, \ldots, J_{k+1}$, and $J'_1, \ldots, J'_k$, and $J''_1, \ldots, J''_k$. For $j = 1, \ldots, k$ we introduce the precedence constraints $J_j \rightarrow J'_j \rightarrow J_{j+1}$ and $J_j \rightarrow J''_j \rightarrow J_{j+1}$.

- An optimal solution sequences all jobs on a single machine such that job J_j completes at time $3j-2$, job J'_j completes at time $3j-1$, and job J''_j completes at time $3j$. This yields a makespan of $3k+1$.
- An optimal LP-solution may set all delay variables $x_{ij}^{LP} \equiv 1/2$. Then the completion time of J_j becomes $3j-2$, the completion times of J'_j and J''_j become $3j - \frac{1}{2}$, and the optimal objective value becomes $C^{LP} = 3k+1$.
- In the rounded solution, all delay variables are $\tilde{x}_{ij} \equiv 1$. Job J_j completes at time $4j-3$, and jobs J'_j and J''_j complete at time $4j-2$. Hence, the rounded makespan is $\tilde{C} = 4k+1$.

Gap 2.1 *There exist instances for* $P\infty \,|\, prec, p_j{=}1, c{=}1 \,|\, C_{\max}$ *for which the gap between the optimal makespan and the makespan produced by the rounding algorithm comes arbitrarily close to* $4/3$.

And what about the integrality gap of the LP-relaxation for problem $P\infty \,|\, prec,\, p_j{=}1, c{=}1 \,|\, C_{\max}$? Consider an instance with $2^k - 1$ jobs, where the precedence constraints form a complete binary out-tree of height $k-1$. That is, every job (except the leaves) has exactly two immediate successors, and every job (except the root) has exactly one immediate predecessor.

- Consider a non-leaf job J_j that completes at time C_j in the optimal solution. Only one of its two immediate successors can be processed on the same machine during the time slot $[C_j; C_j + 1]$, whereas the other immediate successor must complete at time $C_j + 2$. This yields that the optimal makespan equals $2k-1$.
- Now consider an LP-solution in which all delay variables are $x_{ij}^{LP} \equiv 1/2$. Then jobs at distance d from the root complete at time $1 + \frac{3}{2}d$. Therefore, $C^{LP} = \frac{1}{2}(3k-1)$.

Gap 2.2 *For problem* $P\infty \,|\, prec, p_j{=}1, c{=}1 \,|\, C_{\max}$*, the integrality gap of the LP-relaxation is* $4/3$*.*

The approximation result described in this section is the strongest known result for $P\infty \,|\, prec, p_j{=}1, c{=}1 \,|\, C_{\max}$ (Munier and König, 1997). It is an outstanding open problem to decide whether there exists a better polynomial time approximation algorithm, with worst-case performance guarantee strictly less than $4/3$. The literature contains a remarkable *negative* result in this direction: (Hoogeveen *et al.*, 1994) have shown that the existence of a polynomial time approximation algorithm with worst-case performance guarantee strictly less than $7/6$ would imply $P = NP$.

3. PREEMPTIVE MAKESPAN UNDER JOB REJECTIONS

In this section, we consider an environment with n jobs $J_1, \ldots, J_n$ and with m *unrelated* parallel machines $M_1, \ldots, M_m$. Job J_j has a processing time p_{ij} on machine M_i, and moreover job J_j has a positive *rejection penalty* f_j. All jobs are available at time 0. Preemption of the jobs is allowed (that is, a job may be arbitrarily interrupted and resumed later on an arbitrary machine). A job may be processed on at most one machine at a time. For each job J_j, it must be decided whether to accept or to reject it. The accepted jobs are to be scheduled on the m machines. For the accepted jobs, we pay the makespan of this schedule. For the rejected jobs, we pay their rejection penalties. In other words, the objective value is the preemptive makespan of the accepted jobs plus the total penalty of the rejected jobs. This scheduling problem is denoted by $R \,|\, pmtn \,|\, \text{Rej} + C_{\max}$; it is NP-hard in the strong sense (Hoogeveen *et al.*, 2003).

Exact formulation and relaxation. Once again, we will start by stating an integer programming formulation. For job J_j, the binary variable y_j decides whether J_j is rejected ($y_j = 0$) or accepted ($y_j = 1$). The continuous variables x_{ij} describe which percentage of job J_j should be processed on machine M_i. The continuous variable C denotes the optimal preemptive

makespan for the accepted jobs:

$$\begin{array}{lll} \min & C + \sum_{j=1}^{n} (1 - y_j) f_j & \\ \text{s.t.} & \sum_{j=1}^{n} x_{ij} p_{ij} \leq C & \text{for } i = 1, \ldots, m \\ & \sum_{i=1}^{m} x_{ij} p_{ij} \leq C & \text{for } j = 1, \ldots, n \\ & \sum_{i=1}^{m} x_{ij} = y_j & \text{for } j = 1, \ldots, n \\ & x_{ij} \geq 0 & \text{for } i = 1, \ldots, m \text{ and } j = 1, \ldots, n \\ & y_j \in \{0, 1\} & \text{for } j = 1, \ldots, n \end{array} \tag{17}$$

The first family of constraints states that for every machine the total assigned processing time is at most C. The second family of constraints states that the total processing time of any accepted job cannot exceed C. The third family of constraints connects the binary decision variables y_j to the continuous variables x_{ij}: if a job is accepted ($y_j = 1$), then the percentages x_{ij} should add up to $1 = y_j$; if a job is rejected ($y_j = 0$), then the percentages x_{ij} should add up to $0 = y_j$.

As soon as we have fixed all the values x_{ij}, the remaining makespan minimisation problem is essentially makespan minimisation in a preemptive open shop. It is well known (Gonzalez and Sahni, 1976; Lawler *et al.*, 1993) that for a preemptive open shop, the smallest value C fulfilling the first and second family of constraints in (17) yields the optimal preemptive makespan. To summarise, the integer program (17) is a complete and correct description of the problem $R \,|\, pmtn \,|\, \text{Rej} + C_{\max}$.

We define the LP-relaxation of the integer programming formulation (17) by replacing the integrality constraints "$y_j \in \{0, 1\}$" by continuous constraints "$0 \leq y_j \leq 1$". This LP-relaxation can be solved in polynomial time, and we denote an optimal solution by x_{ij}^{LP}, y_j^{LP}, and C^{LP}.

The approximation algorithm. Now we want to translate the LP-solution into a reasonable feasible solution for (17). We mainly have to take care of the decision variables y_j; however, this time we also must pay attention to the continuous variables x_{ij}, since their values depend on the values y_j via the third family of constraints in (17).

We *randomly* choose a threshold α from the uniform distribution over the interval $[1/\mathrm{e}, 1]$; here as usual $\mathrm{e} \approx 2.71828$ denotes the base of the natural logarithm. If $y_j^{LP} \leq \alpha$, then we define a rounded decision variable $\tilde{y}_j := 0$, and otherwise we define $\tilde{y}_j := 1$. Jobs J_j with $\tilde{y}_j = 0$ are rejected in the rounded solution, and we set all their variables $\tilde{x}_{ij} = 0$. Jobs J_j with $\tilde{y}_j = 1$ are accepted in the rounded solution; we set all their variables $\tilde{x}_{ij} := x_{ij}^{LP} / y_j^{LP}$.

Finally, we define the rounded makespan by

$$\tilde{C} \;:=\; \max\left\{ \max_{1\le i\le m} \sum_{j=1}^{n} \tilde{x}_{ij} p_{ij},\;\; \max_{1\le j\le n} \sum_{i=1}^{m} \tilde{x}_{ij} p_{ij} \right\} \tag{18}$$

It can be verified that the rounded solution $\tilde{x}_{ij}$, $\tilde{y}_j$, and $\tilde{C}$ constitutes a feasible solution of (17): all variables $\tilde{y}_j$ are binary. For j with $\tilde{y}_j = 0$, the variables $\tilde{x}_{ij}$ add up to 0. For j with $\tilde{y}_j = 1$, the variables $\tilde{x}_{ij}$ add up to $\sum_i x_{ij}^{LP}/y_j^{LP} = 1$. Finally, in (18) the value of $\tilde{C}$ is fixed in order to fulfil the first and the second family of constraints.

Now let us analyse the quality of this rounded solution. For any fixed value of α, the rounded variable $\tilde{x}_{ij}$ is at most a factor of $1/\alpha$ above x_{ij}^{LP}. Hence, by linearity also $\tilde{C}$ is at most a factor of $1/\alpha$ above C^{LP}. Then the expected multiplicative increase in the makespan is at most a factor of

$$\frac{\mathrm{e}}{\mathrm{e}-1} \int_{1/\mathrm{e}}^{1} 1/\alpha \,\mathrm{d}\alpha \;=\; \frac{\mathrm{e}}{\mathrm{e}-1}$$

In the LP-solution, the contribution of job J_j to the total rejection penalty is $(1 - y_j^{LP}) f_j$. The expected contribution of J_j to the rejection penalty in the rounded solution is

$$\begin{aligned} f_j \cdot \text{Prob}[y_j^{LP} \le \alpha] \;&=\; f_j \int_{\max\{1/\mathrm{e}, y_j^{LP}\}}^{1} \frac{\mathrm{e}}{\mathrm{e}-1} \mathrm{d}\alpha \\ &\le\; f_j \int_{y_j^{LP}}^{1} \frac{\mathrm{e}}{\mathrm{e}-1} \mathrm{d}\alpha \\ &=\; \frac{\mathrm{e}}{\mathrm{e}-1} \cdot (1 - y_j^{LP}) f_j \end{aligned}$$

All in all, the expected objective value for the rounded solution is at most a factor of $\mathrm{e}/(\mathrm{e}-1) \approx 1.58$ above the optimal objective value of the LP-relaxation. Hence, our procedure yields a *randomised* polynomial time approximation algorithm with a worst-case performance guarantee of $\mathrm{e}/(\mathrm{e}-1)$.

How can we turn this randomised algorithm into a deterministic algorithm? Well, the only critical values for the threshold parameter α are the values y_j^{LP} with $j = 1, \ldots, n$. All other values of α will yield the same solution as for one of these critical values. Hence, it is straightforward to derandomise the algorithm in polynomial time: we compute the n rounded solutions that correspond to these n critical values, and we select the solution with smallest objective value.

Analysis of the two gaps. Our next goal is to give a worst-case instance for the above approximation algorithm for $R \,|\, pmtn \,|\, \text{Rej} + C_{\max}$. The

instance is based on an integer parameter q. There are $m = (q+1)^q - q^q$ machines M_j that are indexed by $j = q^q + 1, \ldots, (q+1)^q$. For every machine M_j, there are two corresponding jobs J_j and J'_j that have infinite processing requirements on all other machines M_i with $i \neq j$; this implies that these two jobs either have to be rejected or have to be processed on M_j. The processing time of job J_j on M_j is $j - q^q$, and its rejection penalty is $(j - q^q)/j$. The processing time of job J'_j on M_j is q^q, and its rejection penalty is q^q/j. Note that the overall processing time of J_j and J'_j is j, and that their overall penalty is 1.

One possible feasible solution accepts all the jobs J'_j and rejects all the jobs J_j. The resulting makespan is $C = q^q$, and the resulting objective value equals

$$q^q + \sum_{j=q^q+1}^{(q+1)^q} (j - q^q)\,\frac{1}{j} \;=\; (q+1)^q - q^q \sum_{j=q^q+1}^{(q+1)^q} \frac{1}{j} \tag{19}$$

It can be verified that this in fact is the optimal objective value. Next, assume that the approximation algorithm starts from the following feasible solution for the LP-relaxation: for $j = q^q + 1, \ldots, (q+1)^q$ the two jobs J_j and J'_j both get an acceptance value $y_j^{LP} = q^q/j$. Then on every machine M_j, the overall accepted processing time is $(j - q^q)y_j^{LP} + q^q\, y_j^{LP} = q^q$. The penalty of job J_j plus the penalty of job J'_j equals $1 - y_j^{LP} = (j - q^q)/j$. Hence, the objective value of this LP-solution equals the value in (19).

Consider the rounding step for some fixed threshold α. Since the values $y_j^{LP} = q^q/j$ are decreasing in j, there exists some index k such that for $j \leq k$ the values $y_j^{LP} = q^q/j$ all are rounded up to 1 (and the corresponding jobs are accepted), whereas for $j \geq k+1$ the values $y_j^{LP} = q^q/j$ all are rounded down to 0 (and the corresponding jobs are rejected). Then the makespan becomes k (the load on machine M_k), the total rejection penalty is $(q+1)^q - k$, and the objective value is $(q+1)^q$. Thus, the objective value in the rounded solution always equals $(q+1)^q$, and does not depend on α or k.

The ratio between the optimal objective value in (19) and the approximate objective value of $(q+1)^q$ equals

$$1 - \left(\frac{q}{q+1}\right)^q \sum_{j=q^q+1}^{(q+1)^q} \frac{1}{j} \tag{20}$$

We estimate the sum in (20) by

$$\int_{q^q+1}^{(q+1)^q+1} \frac{1}{z}\,\mathrm{d}z \;\leq\; \sum_{j=q^q+1}^{(q+1)^q} \frac{1}{j} \;\leq\; \int_{q^q}^{(q+1)^q} \frac{1}{z}\,\mathrm{d}z$$

Since the integrals on the left- and on the right-hand side both converge to 1 when q goes to infinity, the same holds true for the sum in between. Hence, the

ratio in (20) behaves roughly like $((q+1)^q - q^q)/(q+1)^q$. For large q, this ratio tends to $(\mathrm{e}-1)/\mathrm{e}$.

Gap 3.1 *There exist instances for* $R \,|\, pmtn \,|\, \mathrm{Rej} + C_{\max}$ *for which the gap between the optimal makespan and the makespan produced by the rounding algorithm comes arbitrarily close to* $\mathrm{e}/(\mathrm{e}-1)$.

Our final goal in this section is to get the matching lower bound of $\mathrm{e}/(\mathrm{e}-1)$ for the integrality gap of the LP-relaxation for $R \,|\, pmtn \,|\, \mathrm{Rej} + C_{\max}$. We use a slight modification of the instance constructed above. Again, we use an integer parameter q, and again there are $m = (q+1)^q - q^q$ machines M_j that are indexed by $j = q^q + 1, \ldots, (q+1)^q$. For every machine M_j, there is one corresponding job J_j. The processing requirements of J_j are $p_{jj} = j$, and $p_{ij} = \infty$ for $i \neq j$. The rejection penalty is uniformly $f_j \equiv 1$.

Consider a "reasonable" feasible schedule with makespan T: then the jobs on machines M_j with $j \leq T$ will be accepted, and the jobs on machines M_j with $j > T$ will be rejected. This yields a total rejection penalty of $(q+1)^q - T$, and an optimal objective value of $(q+1)^q$. Next, consider the following feasible solution for the LP-relaxation: we set $y_j^{LP} = q^q/j$ for $j = q^q+1, \ldots, (q+1)^q$, and we set $C^{LP} = q^q$. The objective value of this solution is equal to

$$q^q + \sum_{j=q^q+1}^{(q+1)^q} (1 - y_j^{LP}) \quad = \quad (q+1)^q - q^q \sum_{j=q^q+1}^{(q+1)^q} \frac{1}{j}$$

Since this LP-value is equal to the value in (19), and since the optimal value is equal to $(q+1)^q$, the ratio of these two values equals the ratio in (20). The arguments around (20) show that as q becomes large, this ratio tends to $(\mathrm{e}-1)/\mathrm{e}$.

Gap 3.2 *For problem* $R \,|\, pmtn \,|\, \mathrm{Rej} + C_{\max}$*, the integrality gap of the LP-relaxation is* $\mathrm{e}/(\mathrm{e}-1)$.

The results presented in this section are essentially due to Hoogeveen *et al.* (2003). It would be nice to get a polynomial time approximation algorithm with a worst-case performance guarantee better than $\mathrm{e}/(\mathrm{e}-1)$. Hoogeveen *et al.* (2003) also show that problem $R \,|\, pmtn \,|\, \mathrm{Rej} + C_{\max}$ is APX-hard; this implies that unless $P = NP$, this problem cannot possess a polynomial time approximation scheme.

4. CONCLUDING REMARKS

We have discussed approximations through relaxations for three scheduling problems. For all three problems, we have presented a complete worst-case

analysis of a polynomial time approximation algorithm. For all three problems, we gave instances that illustrate the worst-case behaviour of the approximation algorithm (the worst-case gap), and we gave instances that illustrate the integrality gap of the linear programming relaxation. For all three problems, the worst-case gap and the integrality gap coincided. The scheduling literature contains many other results that fit into this framework.

- For $1 \,|\, \textit{prec} \,|\, \sum w_j C_j$, the problem of minimising the total weighted job completion time on a single machine under precedence constraints, the literature contains three different LP-relaxations: one by Potts (1980), one by Dyer and Wolsey (1990), and one by Chudak and Hochbaum (1999). Hall *et al.* (1997) and Chudak and Hochbaum (1999) show that these LP-relaxations have integrality gaps of at most 2. Chekuri and Motwani (1999) design instances that yield matching lower bounds of 2.

- Chekuri *et al.* (2001) investigate the preemptive relaxation for $1 \,|\, r_j \,|\, \sum w_j C_j$, the problem of minimising the total weighted job completion time on a single machine under job release dates. They show that the integrality gap of the preemptive relaxation is at most $\mathrm{e}/(\mathrm{e}-1)$. Torng and Uthaisombut (1999) complete the analysis by providing matching lower bound instances.

- Kellerer *et al.* (1996) discuss $1 \,|\, r_j \,|\, \sum F_j$, the problem of minimising the total flow time of the jobs on a single machine under job release dates. By studying the preemptive relaxation, they derive a polynomial time approximation algorithm with worst-case performance guarantee $\mathrm{O}(\sqrt{n})$; here n denotes the number of jobs. Up to constant factors, the analysis is tight. Moreover, Kellerer *et al.* (1996) exhibit instances for which the integrality gap of the preemptive relaxation is $\Omega(\sqrt{n})$.

A difficult open problem is to analyse the preemptive relaxation for the open shop problem $O \,|\,| \, C_{\max}$. In an m-machine open shop, every job J_j consists of m operations $O_{1j}, \ldots, O_{mj}$ with processing times $p_{1j}, \ldots, p_{mj}$ that have to have processed on machines $M_1, \ldots, M_m$. The processing order of the m operations $O_{1j}, \ldots, O_{mj}$ is *not* prespecified; it may differ for different jobs, and it may be decided and fixed by the scheduler. At any moment in time, at most one operation from any job can be processed. The goal is to minimise the makespan. The preemptive version of the open shop is denoted by $O \,|\, \textit{pmtn} \,|\, C_{\max}$. Whereas the non-preemptive problem $O \,|\,| \, C_{\max}$ is strongly NP-hard (Williamson *et al.*, 1997), the preemptive version $O \,|\, \textit{pmtn} \,|\, C_{\max}$ can be solved in polynomial time (Gonzalez and Sahni, 1976).

A folklore conjecture states that the integrality gap of the preemptive relaxation for $O \,|\,| \, C_{\max}$ should be at most $3/2$. Settling this conjecture would mean a major breakthrough in the area. Here is an instance that demonstrates

that the integrality gap is at least $3/2$: consider an open shop with m machines and $n = m + 1$ jobs. For $j = 1, \ldots, m$ the job J_j consists of the operation O_{jj} with processing time $p_{jj} = m - 1$ on machine M_j, and with processing times 0 on the other $m - 1$ machines. The job J_{m+1} has m operations all with processing time 1. Then the optimal preemptive makespan is m, and the optimal non-preemptive makespan is $\lceil m/2 \rceil + m - 1$. As m becomes large, the ratio between non-preemptive and preemptive makespan tends to $3/2$.

Acknowledgments

I thank Martin Skutella for showing me the worst-case instance for preemptive makespan under job rejections.

References

Chekuri, C. and Motwani, R. (1999). Precedence constrained scheduling to minimize sum of weighted completion times on a single machine. *Discrete Applied Mathematics*, **98**:29–38.

Chekuri, C., Motwani, R., Natarajan, B. and Stein, C. (2001). Approximation techniques for average completion time scheduling. *SIAM Journal on Computing*, **31**:146–166.

Chudak, F. and Hochbaum, D. S. (1999). A half-integral linear programming relaxation for scheduling precedence-constrained jobs on a single machine. *Operations Research Letters*, **25**:199–204.

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

Garey, M. R. and Johnson, D. S. (1979). *Computers and Intractability*. Freeman, New York.

Gonzalez, T. and Sahni, S. (1976). Open shop scheduling to minimize finish time. *Journal of the ACM*, **23**:665–679.

Graham, R. L. (1966). Bounds for certain multiprocessing anomalies. *Bell System Technical Journal*, **45**:1563–1581.

Graham, R. L. (1969). Bounds on multiprocessor timing anomalies. *SIAM Journal on Applied Mathematics*, **17**:416–429.

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

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

Hochbaum, D. S. and Shmoys, D. B. (1987). Using dual approximation algorithms for scheduling problems: Theoretical and practical results. *Journal of the ACM*, **34**:144–162.

Hoogeveen, J. A., Lenstra, J. K. and Veltman, B. (1994). Three, four, five, six, or the complexity of scheduling with communication delays. *Operations Research Letters*, **16**:129–137.

Hoogeveen, H., Skutella, M. and Woeginger, G. J. (2003). Preemptive scheduling with rejection. *Mathematical Programming*, **94**:361–374.

Kellerer, H., Tautenhahn, T. and Woeginger, G. J. (1996). Approximability and nonapproximability results for minimizing total flow time on a single machine. In *Proceedings of the 28th Annual ACM Symposium on the Theory of Computing (STOC'96)*, pp. 418–426.

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

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

Munier, A. and König, J. C. (1997). A heuristic for a scheduling problem with communication delays. *Operations Research*, **45**:145–147.

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

Picouleau, C. (1992). Etude des problèmes d'optimisation dans les systèmes distribués. *Thèse*, l'Université Paris 6.

Potts, C. N. (1980). An algorithm for the single machine sequencing problem with precedence constraints. *Mathematical Programming Study*, **13**:78–87.

Torng, E. and Uthaisombut, P. (1999). A tight lower bound for the best-alpha algorithm. *Information Processing Letters*, **71**:17–22.

Veltman, B., Lageweg, B. J. and Lenstra. J. K. (1990). Multiprocessor scheduling with communication delays. *Parallel Computing*, **16**:173–182.

Williamson, D. P., Hall, L. A., Hoogeveen, J. A., Hurkens, C. A. J., Lenstra, J. K., Sevastianov, S. V. and Shmoys, D. B. (1997). Short shop schedules. *Operations Research*, **45**:288–294.

Woeginger, G. J. (2000). A comment on scheduling on uniform machines under chain-type precedence constraints. *Operations Research Letters*, **26**:107–109.