Using Linear Programming in the Design and Analysis of Approximation Algorithms: Two Illustrative Problems

David B. Shmoys

Cornell University, Ithaca NY 14853, USA

Abstract. One of the foremost techniques in the design and analysis of approximation algorithms is to round the optimal solution to a linear programming relaxation in order to compute a near-optimal solution to the problem at hand. We shall survey recent work in this vein for two particular problems: the uncapacitated facility location problem and the problem of scheduling precedence-constrained jobs on one machine so as to minimize a weighted average of their completion times.

1 Introduction

One of the most successful techniques in the design and analysis of approximation algorithms for combinatorial optimization problems has been to first solve a relaxation of the problem, and then to round the optimal solution to the relaxation to obtain a near-optimal solution for the original problem. Although the relaxation used varies from problem to problem, linear programming relaxations have provided the basis for approximation algorithms for a wide variety of problems. Throughout this paper, we shall discuss approximation algorithms, where a *ρ-approximation algorithm* for an optimization problem is a polynomial-time algorithm that is guaranteed to find a feasible solution for the problem with objective function value within a factor of ρ of optimal.

In this brief survey, we shall discuss recent developments in the design of approximation algorithms for two specific problems, the uncapacitated facility location problem, and a rather basic single-machine scheduling problem. In focusing on just two problems, clearly we are omitting a great deal of important recent work on a wide cross-section of other problems, but the reader can obtain an accurate indication of the level of activity in this area by considering, for example, the other papers in this proceedings. For a more comprehensive review of the use of this approach, the reader is referred to the volume edited by Hochbaum [16].

We shall consider the following scheduling problem. There are n jobs to be scheduled on a single machine, where each job j has a specified weight w_j and processing time p_j, $j = 1, \ldots, n$, which we restrict to be positive integers. Furthermore, there is a partial order $\prec$ that specifies a precedence relation among the jobs; if $j \prec k$ then we must find a schedule in which job j completes its processing before job k is started. Each job must be processed without interruption, and the machine can process at most one job at a time. If we let C_j denote the completion time of job j, then we wish to minimize the average weighted completion time $\sum_{j=1}^{n} w_j C_j/n$, or equivalently, $\sum_{j=1}^{n} w_j C_j$. In the notation of Graham, Lawler, Lenstra, & Rinnooy Kan [11], the problem is denoted $1|prec|\sum w_j C_j$; it was shown to be $\mathcal{NP}$-hard by Lawler [21].

The first non-trivial approximation algorithm for $1|prec|\sum w_j C_j$ is due to Ravi, Agrawal, & Klein [33], who gave an $O(\lg n \lg W)$-approximation algorithm, where $W = \sum_j w_j$. A slightly improved performance guarantee of $O(\lg n \lg \lg W)$ follows from work of Even, Naor, Rao, & Schieber [9]. We shall present a series of results that give constant approximation algorithms for this problem, where the resulting algorithms are both simple to state, and simple to analyze.

We shall also consider the uncapacitated facility location problem. In this problem, there is a set of locations F at which we may build a facility (such as a warehouse), where the cost of building at location i is f_i, for each $i \in F$. There is a set D of client locations (such as stores) that require to be serviced by a facility, and if a client at location j is assigned to a facility at location i, a cost of c_{ij} is incurred. All of the data are assumed to be non-negative. The objective is to determine a set of locations at which to open facilities so as to minimize the total facility and assignment costs.

Building on results for the set covering problem (due to Johnson [19], Lovász [25], and Chvátal [7]), Hochbaum [15] showed that a simple greedy heuristic is an $O(\log n)$-approximation algorithm, where n denotes the total number of locations in the input. Lin & Vitter [24] gave an elegant filtering and rounding technique that yields an alternate $O(\log n)$-approximation algorithm for this problem. We shall focus on the metric case of this problem, in which distances between locations are given in some metric (and hence satisfy the triangle inequality), and the assignment costs c_{ij} are proportional to the distance between i and j, for each $i \in F$, $j \in D$. We shall present a series of results that give constant approximation algorithms for this problem, where, once again, the resulting algorithms are both simple to state, and (relatively) simple to analyze.

2 A simple scheduling problem

We shall present approximation algorithms for the problem of scheduling precedence-constrained jobs on a single machine so as to minimize the average weighted completion time, $1|prec|\sum w_j C_j$. Although we will primarily focus on this one scheduling model, the starting point for the work that we shall survey is an extremely simple, elegant result of Phillips, Stein, & Wein [29] for a related problem, in which the jobs are now independent (that is, there are no precedence constraints) but instead each job j has a specified release date r_j before which it may not begin processing, $j = 1, \ldots, n$; furthermore, they consider the unit-weight case, or in other words, $w_j = 1$, for each $j = 1, \ldots, n$. This problem is denoted $1|r_j|\sum C_j$ and was shown to be $\mathcal{NP}$-hard by Lenstra, Rinnooy Kan, & Brucker [22].

The algorithm of Phillips, Stein, & Wein [29] is based on a relaxation of the problem that can be solved in polynomial time. In this case, however, the relaxation is not a linear program, but instead one motivated in purely scheduling terms: rather than requiring that each job be processed without interruption, we allow *preemption*. That is, the processing of a job may be interrupted to process another (higher priority) job instead, and then the first job may be resumed without penalty. This problem, denoted $1|r_j, pmtn|\sum C_j$, can be solved (to optimality) by the following simple rule: schedule the jobs in time, and always process the job with the least remaining processing time (among those already released).

The approximation algorithm of Phillips, Stein, & Wein works as follows: solve the preemptive relaxation, and then schedule the jobs in the order in which they complete in the relaxed solution. It is remarkably straightforward to show that this is a 2-approximation algorithm. Suppose that the jobs happen to be indexed in the order in which they complete in the preemptive relaxation, and so are processed in the order $1, 2, \ldots, n$ in the heuristically computed non-preemptive schedule as well. If we consider the schedule produced by the approximation algorithm, then any idle time in the schedule ends at the release date of some job k (since that idle time is, in effect, caused by waiting for job k to be released). Consequently, for each job j, there is no idle time between $\max_{k=1,\ldots,j} r_k$ and the completion time of job j, C_j. This implies that

$$C_j \leq \max_{k=1,\ldots,j} r_k + \sum_{k=1}^{j} p_j.$$

Let $\overline{C}_j$ denote the completion time of job j in the optimal preemptive schedule; since each job k, $k = 1, \ldots, j$, has completed its processing in the optimal preemptive schedule by $\overline{C}_j$, it follows that

$$r_k \leq \overline{C}_k \leq \overline{C}_j, \quad \text{for each } k = 1, \ldots, j,$$

By the same reasoning, $\sum_{k=1}^{j} p_k \leq \overline{C}_j$. Hence, $C_j \leq 2\overline{C}_j$. Furthermore, the value of the schedule found, $\sum_{j=1}^{n} C_j$, is at most twice the preemptive optimum, and so is at most twice the value of the non-preemptive optimal schedule as well.

For $1|prec|\sum w_j C_j$, we shall rely on a number of linear programming relaxations, but the overall approach will be identical. We will solve the relaxation, and then use the relaxed solution to compute a (natural) ordering of the jobs that is feasible with respect to $\prec$; this is the schedule computed by the approximation algorithm. This is not the first scheduling problem for which this approach has been considered; for example, Munier & König [28] have given a very elegant approximation algorithm where the schedule (for a particular parallel machine scheduling problem with communication delays) is derived from an optimal solution to a linear programming relaxation.

We start by considering a very strong linear programming relaxation, the *non-preemptive time-indexed formulation*. In this formulation, which is due to Dyer & Wolsey [8], we use the variable x_{jt} to indicate whether job j completes processing at time t, $j = 1, \ldots, n$, $t = 1, \ldots, T$, where $T = \sum_{j=1}^{n} p_j$. Given these decision variables, it is easy to represent the objective function:

$$\text{Minimize } \sum_{j=1}^{n} w_j \sum_{t=1}^{T} t \cdot x_{jt}. \tag{1}$$

We can constrain the assignments of the decision variables as follows. Each job must complete at a unique point in time; hence,

$$\sum_{t=1}^{T} x_{jt} = 1, \quad j = 1, \ldots, n. \tag{2}$$

No job j can complete before p_j:

$$x_{jt} = 0, \quad \text{if } t < p_j. \tag{3}$$

The sum $\sum_{s=1}^{t} x_{js} = 1$ if and only if job j has been completed by time t; if $j \prec k$, we know that job j must complete at least p_k time units earlier than job k, and hence

$$\sum_{s=1}^{t} x_{js} \geq \sum_{s=1}^{t+p_k} x_{ks}, \quad \text{if } j \prec k, \; t = 1, \ldots, T - p_k. \tag{4}$$

Of course, the machine can process at most one job at each time t; job j is processed at time t if it completes at any time within the interval $[t, t + p_j - 1]$:

$$\sum_{j=1}^{n} \sum_{s=t}^{t+p_j-1} x_{js} \leq 1, \quad t = 1, \ldots, T. \tag{5}$$

If we wish to give an integer programming formulation of the problem, then we would require each variable to be either 0 or 1. We shall consider the linear programming relaxation, in which we require that $x_{jt} \geq 0$, $j = 1, \ldots, n$, $t = 1, \ldots, T$. For any feasible fractional solution x, we define $\overline{C}_j = \sum_{t=1}^{T} t \cdot x_{jt}$ to be the *fractional completion time* of job j, $j = 1, \ldots, n$. If x is an optimal solution to the linear relaxation, then $\sum_{j=1}^{n} w_j \overline{C}_j$ is a lower bound on the optimal value for the original problem.

For a given α, $0 \leq \alpha \leq 1$, and a job j, $j = 1, \ldots, n$, we focus on the earliest point in time that a cumulative α-fraction of job j has been slated to complete: let the *α-point* of job j be $t_j(\alpha) = \min\{t \; : \; \sum_{s=1}^{t} x_{js} \geq \alpha\}$. The notion of an α-point was also introduced in the work of Phillips, Stein, & Wein [29], in a slightly different context. Hall, Shmoys, & Wein [14] proposed the following algorithm for $1|prec|\sum w_j C_j$: schedule the jobs in non-decreasing order of their α-points. It is easy to see that the constraints (4) ensure that the schedule found satisfies the precedence constraints.

The *α-point algorithm* of Hall, Shmoys, & Wein can be analyzed as follows. Suppose that the jobs happen to be indexed in non-decreasing α-point order. Hence, each job j completes at time

$$C_j = \sum_{k=1}^{j} p_k. \tag{6}$$

For each job k, $k = 1, \ldots, j$, an α fraction of each job k is done by time $t_j(\alpha)$, and hence

$$\alpha \sum_{k=1}^{j} p_k \le t_j(\alpha). \tag{7}$$

Consider the fractional completion time $\overline{C}_j$; one can view the values x_{jt} as providing a weighted average of the corresponding values t. Since less than a $1-\alpha$ fraction of the weight can be placed on values more than $1/(1-\alpha)$ times the average, we see that

$$t_j(\alpha) \le \overline{C}_j/(1-\alpha). \tag{8}$$

By combining (6)–(8), we see that each job j completes at time

$$C_j \le \overline{C}_j/(\alpha(1-\alpha)).$$

Consequently, we see that the value of the solution found, $\sum_{j=1}^{n} w_j C_j$, is within a factor of $1/(\alpha-\alpha^2)$ of $\sum_{j=1}^{n} w_j \overline{C}_j$, which is a lower bound on the optimal value. If we set $\alpha = 1/2$ (to minimize $1/(\alpha-\alpha^2)$), we see that we have obtained a solution of value within a factor of 4 of the optimum.

But is setting $\alpha = 1/2$ the best thing to do? Goemans [10] observed that rather than choosing α once, to optimize the performance guarantee, it makes more sense to consider, for each input, which choice of α would deliver the best schedule for *that particular input.* (Chekuri, Motwani, Natarajan, & Stein [3] independently suggested an analogous improvement to the algorithm of Phillips, Stein, & Wein.) The performance of this best-α algorithm can be analyzed by considering the following randomized algorithm instead: set $\alpha = a$ by choosing at random within the interval (0,1) according to the probability density function $f(a) = 2a$. The same analysis given above implies that we can bound

$$E[C_j] \le \int_0^1 (t_j(a)/a) f(a) da = 2\int_0^1 t_j(a) da.$$

If we interpret this integral as the area under the curve defined by the function $t_j(a)$ as a ranges from 0 to 1, then it is easy to see that this integral is precisely $\overline{C}_j$. Thus, the randomized algorithm produces a solution that has expected value at most *twice* the optimal value. Furthermore, the algorithm that finds the value of α for which the α-point algorithm delivers the best solution, the *best-α algorithm*, is a deterministic algorithm guaranteed to find a solution with objective function value at most twice the optimal value.

Of course, none of these algorithms are efficient; that is, it is not known how to implement them to run in polynomial time, due to the size of the linear programs that must be solved. Since the size of the linear program can be bounded by a polynomial in n and $T = \sum_j p_j$, the α-point algorithm can be shown to run in pseudo-polynomial time. It is often the case that a pseudo-polynomial algorithm for a problem can be adapted to run in polynomial time while losing an additional $1+\epsilon$ factor in accuracy, basically by using only a polynomial number of bits of accuracy in the input. However, in this case it is not clear how to use to these well-known techniques.

Instead, Hall, Shmoys, & Wein [14] proposed using a different, more compact, linear programming relaxation, called an *interval-indexed formulation*. (This type of formulation was subsequently used in another context in the journal version of these results [13].) The key idea behind these constructions is that the time horizon is subdivided into the intervals $[1,1]$, $(1,1+\epsilon]$, $(1+\epsilon,(1+\epsilon)^2]$, $((1+\epsilon)^2,(1+\epsilon)^3],\ldots$, where ϵ is an arbitrarily small positive constant; the linear program only specifies the interval in which a job is completed. Since all completion times within an interval are within a $(1+\epsilon)$ factor of each other, the relative scheduling within an interval will be of little consequence.

Given this basic idea, it is extremely straightforward to complete all of the details of this polynomial-sized formulation. The linear programming relaxation relies on the variables $x_{j\ell}$, which indicate whether job j completes within the ℓth interval. There are assignment constraints completely analogous to (2). The precedence constraints are enforced only to the extent that if $j \prec k$, then the interval in which j finishes is no later than the interval in which k finishes. To capture the load constraint, we merely require that the total length of jobs assigned to complete in the interval $((1+\epsilon)^{\ell-1},(1+\epsilon)^{\ell}]$ is at most $(1+\epsilon)^{\ell}$. The analogue of the α-point algorithm is as follows: for each job, compute its α-interval, and schedule the jobs in order of non-decreasing α-intervals, where the jobs assigned to the same interval are scheduled in any order that is consistent with the precedence relation. Thus, Hall, Shmoys, & Wein obtained, for any fixed $\epsilon > 0$, a $4+\epsilon$-approximation algorithm, and the best-α-point algorithm of Goemans can be adapted to yield a $2+\epsilon$-approximation algorithm.

As it turns out, it is even easier to obtain a 2-approximation algorithm for this problem by using other compact linear programming relaxations. Schulz [35] (and subsequently in its journal version [13]) showed how to improve the earlier work of Hall, Shmoys, & Wein by using a relaxation due to Wolsey [41] and Queyranne [31]. In this formulation, there is a variable C_j for each job j in $N = \{1,\ldots,n\}$:

$$\text{Minimize } \sum_{j=1}^{n} w_j C_j \tag{9}$$

subject to

$$\sum_{j\in S} p_j C_j \ge \sum_{(j,k)\in S\times S} p_j p_k, \qquad \text{for each } S \subseteq N, \tag{10}$$

$$C_k \ge C_j + p_k, \qquad \text{if } j \prec k. \tag{11}$$

If the jobs are independent, and hence there are neither precedence constraints nor constraints in (11), then Wolsey [41] and Queyranne [31] independently showed that this linear program provides an exact characterization of the problem $1||\sum w_j C_j$: extreme points of this linear program correspond to schedules. Of course, in the case in which there are precedence constraints, the situation is quite different, since otherwise $\mathcal{P}$ would be equal to $\mathcal{NP}$.

The most natural approximation algorithm for $1|prec|\sum w_j C_j$ based on this linear relaxation is as follows: solve the relaxation to obtain a solution $\overline{C}_j$, $j = 1, \ldots, n$, and schedule the jobs so that their LP values are in non-decreasing order. The analysis of this algorithm is also remarkably simple. Suppose that the jobs happen to be indexed so that $\overline{C}_1 \leq \cdots \leq \overline{C}_n$, and so they are scheduled by the algorithm in their index order as well. Once again, job j completes at time $C_j = \sum_{k=1}^{j} p_k$. If we consider the constraint (10) when $S = \{1, \ldots, j\}$, then we see that

$$\sum_{k=1}^{j} p_k \overline{C}_k \geq \sum_{(k,k') \in S \times S} p_k p_{k'} \geq (1/2)(\sum_{k=1}^{j} p_k)^2.$$

However, $\overline{C}_j(\sum_{k=1}^{j} p_k) \geq \sum_{k=1}^{j} p_k \overline{C}_k$. Hence $\overline{C}_j \geq (\sum_{k=1}^{j} p_k)/2$, or equivalently, $C_j \leq 2\overline{C}_j$. This proves that the value of the solution found is within a factor of 2 of optimal. However, it is not at all clear that this linear programming relaxation is sufficiently more compact than the time-indexed one, since it contains an *exponential* number of constraints. However, one can solve this linear program in polynomial time with the ellipsoid algorithm, since it is easy to devise a polynomial-time algorithm that determines whether a given fractional solution is feasible, or if not, returns a violated constraint (see Queyranne [31]). Hence, we have a 2-approximation algorithm.

Potts [30] has proposed yet another linear programming relaxation of the problem $1|prec|\sum w_j C_j$, which is called the *linear ordering formulation*. In this formulation, there are variables δ_{ij} that indicate whether or not job i is processed before job j:

$$\text{Minimize } \sum_{j=1}^{n} w_j C_j$$

subject to

$$\begin{aligned}
p_j + \textstyle\sum_{i=1}^{n} p_i \delta_{ij} &= C_j, & & j = 1, \ldots, n; \\
\delta_{ij} + \delta_{ji} &= 1, & & i, j = 1, \ldots, n,\ i < j; \\
\delta_{ij} + \delta_{jk} + \delta_{ki} &\leq 2, & & i, j, k = 1, \ldots, n,\ i < j < k \text{ or } i > j > k; \\
\delta_{ij} &= 1, & & i, j = 1, \ldots, n,\ i \prec j; \\
\delta_{ij} &\geq 0, & & i, j = 1, \ldots, n,\ i \neq j.
\end{aligned}$$

Schulz [35] has observed that for any feasible solution to this linear program, the C_j values are feasible for the linear program (9)–(11). Hence, if we solve the linear ordering formulation to obtain values $\overline{C}_j$, and then schedule the jobs so that these values are in non-decreasing order, then we obtain a more efficient

2-approximation algorithm (since any polynomial-time linear programming algorithm can be used to solve this LP with n^2 variables and $O(n^3)$ constraints).

Chudak & Hochbaum [5] proposed a somewhat weaker linear programming relaxation, which also uses the variables δ_{ij}. In this relaxation, the constraints that enforce the transitivity of the ordering relaxation, $\delta_{ij} + \delta_{jk} + \delta_{ki} \leq 2$, are instead replaced with the constraints that $\delta_{ki} \leq \delta_{kj}$, whenever $i \prec j$, and k is different from both jobs i and j. Once again, a straightforward calculation shows that for any feasible solution to this weaker linear program, the C_j values are feasible for the constraints (10) and (11). Consequently, one also obtains a 2-approximation algorithm by first solving this weaker linear program, and then using the resulting C_j values to order the jobs. The advantage of using this formulation is as follows: Chudak & Hochbaum also observed that a result of Hochbaum, Meggido, Naor, & Tamir [17] can be applied to show that there always exists an optimal solution to this linear program that is *half-integral*, i.e., each variable δ_{ij} is either 0,1/2, or 1; furthermore, an optimal half-integral solution can be computed by a maximum flow computation. Thus, this approach yields a 2-approximation algorithm that does not require the solution of a linear program, but rather only a single maximum flow computation.

Chekuri & Motwani [2] and Margot, Queyranne, & Wang [27] independently devised another, more combinatorial 2-approximation algorithm for the problem $1|prec|\sum w_j C_j$. We shall say that a subset S of jobs is an *initial set* of the precedence relation $\prec$ if, for each job $k \in S$, each of its predecessors is also in S, or more formally,

$$(k \in S \text{ and } j \prec k) \Rightarrow j \in S.$$

For each subset of jobs $S \subseteq N$, let $\rho(S) = \sum_{j \in S} p_j / \sum_{j \in S} w_j$.

Suppose that we minimize $\rho(S)$ over all initial subsets to obtain a subset S^*. Chekuri & Motwani and Margot, Queyranne, & Wang proved a remarkable fact: if $S^* = N$, then *any* ordering of the jobs that is consistent with $\prec$ has objective function value within a factor of 2 of the optimum. The proof of this fact is amazingly simple. In each feasible schedule, each job j completes by time $\sum_{k \in N} p_k$, and so the cost of any solution is at most $(\sum_{k \in N} p_k)(\sum_{k \in N} w_k)$. So we need only show that the optimal value is at least $(\sum_{k \in N} p_k)(\sum_{k \in N} w_k)/2$. Suppose that the jobs happen to be indexed so that job j is the jth job to be scheduled in an optimal schedule. Then each set $\{1, \ldots, j\}$ is an initial set, and hence the completion time of job j,

$$C_j = \sum_{k=1}^{j} p_k \geq \rho(N) \sum_{k=1}^{j} w_k.$$

Consequently, we know that

$$\sum_{j=1}^{n} w_j C_j \geq \rho(N) \sum_{j=1}^{n} \sum_{k=1}^{j} w_j w_k \geq \rho(N)(\sum_{j=1}^{n} w_j)^2/2.$$

Recalling that $\rho(N) = \sum_{j=1}^{n} p_j / \sum_{j=1}^{n} w_j$, we see that we have obtained the desired lower bound on the optimal value.

Of course, there is no reason to believe that N is the initial set S for which $\rho(S)$ is minimized. Fortunately, if this is not the case, then we can rely on the following decomposition result of Sidney [37]: if S^* is the initial set S for which $\rho(S)$ is minimized, then there exists an optimal solution in which the jobs of S^* precede the jobs of $N - S^*$. This suggests the following recursive 2-approximation algorithm: find the set S^*, and schedule it first in any order consistent with the precedence relation $\prec$, and then recursively apply the algorithm to $N - S^*$, and concatenate the two schedules found. It is not hard to show that the initial set S^* can be found via a minimum cut (or equivalently, a maximum flow) computation.

For each of the results above, we have presented an algorithm and then showed that it delivers a solution whose objective function value is within some constant factor of the optimal value of a linear programming relaxation of the problem. Such a result not only shows that we have found a good algorithm, but also implies a guarantee for the quality of the lower bound provided by that linear program. For each of the linear programs concerned, one might ask whether these particular algorithms can be improved; that is, might it be possible to round the optimal fractional solutions in a more effective manner? Unfortunately, the answer to each of these questions is no. For the time-indexed formulation, Schulz & Skutella [34] have given instances for which the ratio between the integer and fractional optima is arbitrarily close to 2. For the linear ordering formulation, Chekuri & Motwani [2] have given a surprising construction based on expander graphs for which the ratio of the integer to fractional optimal values asymptotically approaches 2. Each of these results implies the analogous result for the linear program (9)–(11), but for this relaxation it is also relatively simple to construct examples directly. Of course, there might still be other relaxations that provide stronger lower bounds, and this is an extremely interesting direction for further research.

3 The uncapacitated facility location problem

The uncapacitated facility location problem is one of the most well-studied problems in the Operations Research literature, dating back to the work of Balinski [1], Kuehn & Hamburger [20], Manne [26], and Stollsteimer [38, 39] in the early 60's. We shall focus on one important special case of this problem, where the locations are embedded in some metric space, and the assignment costs c_{ij} are proportional to the distances between locations; we shall call this the *metric uncapacitated facility location problem.*

Although there is little work that has specifically focused on the metric case of this location problem, for many others, such as the k-center problem (see, e.g., [18]) and the k-median problem (see, e.g., [23]) this assumption is prevalent. In fact, the algorithms of Lin & Vitter [23] contained many of the seeds of the work that we shall present for the metric uncapacitated facility location problem.

Once again, all of the algorithms that we shall discuss will be based on rounding an optimal solution to a linear programming relaxation of the problem. For this problem, the most natural relaxation is as follows. There are two types of decision variables x_{ij} and y_i, for each $i \in F$, $j \in D$, where each variable y_i, $i \in F$, indicates whether or not a facility is built at location i, and each variable x_{ij} indicates whether or not the client at location j is assigned to a facility at location i, for each $i \in F$, $j \in D$:

$$\text{Minimize } \sum_{i \in F} f_i y_i + \sum_{i \in F} \sum_{j \in D} c_{ij} x_{ij} \tag{12}$$

subject to

$$\sum_{i \in F} x_{ij} = 1, \qquad \text{for each } j \in D, \tag{13}$$

$$x_{ij} \le y_i, \qquad \text{for each } i \in F,\ j \in D, \tag{14}$$

$$x_{ij} \ge 0, \qquad \text{for each } i \in F,\ j \in D. \tag{15}$$

Shmoys, Tardos, & Aardal [36] gave a simple algorithm to round an optimal solution to this linear program to an integer solution of cost at most $3/(1-e^3) \approx 3.16$ times as much. The algorithm relies on the filtering technique of Lin & Vitter [24]. We can interpret each fractional solution (x, y) as the following bipartite graph $G(x, y) = (F, D, E)$: the two sets of nodes are F and D, and there is an edge $(i, j) \in E$ exactly when $x_{ij} > 0$.

First, we apply an α-*filtering algorithm* to convert the optimal fractional solution to a new one, $(\bar{x}, \bar{y})$, in which the cost c_{ij} associated with each edge in $G(\bar{x}, \bar{y})$ is relatively cheap. As in the algorithm based on the time-indexed formulation for the scheduling problem, we first define the notion of an α-*point*, $c_j(\alpha)$, for each location $j \in D$. Focus on a location $j \in D$, and let π be a permutation such that $c_{\pi(1)j} \le c_{\pi(2)j} \le \cdots \le c_{\pi(n)j}$. We then set $c_j(\alpha) = c_{\pi(i^*)j}$, where $i^* = \min\{i' : \sum_{i=1}^{i'} x_{\pi(i)j} \ge \alpha\}$. To construct $(\bar{x}, \bar{y})$, for each $(i, j) \in E(x, y)$ for which $c_{ij} > c_j(\alpha)$ we set $\bar{x}_{ij} = 0$, and then renormalize by setting each remaining $\bar{x}_{ij}$ equal to x_{ij}/α_j, where $\alpha_j = \sum_{(i,j) \in E:\ c_{ij} \le c_j(\alpha)} x_{ij}$. We also renormalize $\bar{y}_i = y_i/\alpha$. It is easy to check that $(\bar{x}, \bar{y})$ is a feasible solution to the linear program (12)–(15) with the further property that $\bar{x}_{ij} > 0 \Rightarrow c_{ij} \le c_j(\alpha)$. Motivated by this, given values g_j, $j \in D$, we shall call a solution g-*close* if $\bar{x}_{ij} > 0 \Rightarrow c_{ij} \le g_j$.

The central element of the rounding algorithm of Shmoys, Tardos, & Aardal is a polynomial-time algorithm that, given a g-close feasible solution $(\bar{x}, \bar{y})$ to (12)–(15), finds a $3g$-close integer solution $(\hat{x}, \hat{y})$ such that

$$\sum_{i \in F} f_i \hat{y}_i \le \sum_{i \in F} f_i \bar{y}_i.$$

The algorithm works as follows. It partitions the graph $G(\bar{x}, \bar{y}) = (F, D, E)$ into clusters, and then, for each cluster, opens one facility that must serve all clients in it. The clusters are constructed iteratively as follows. Among all clients that have not already been assigned to a cluster, let j' be the client j for which g_j is smallest. This cluster consists of j', all neighbors of j' in $G(\bar{x}, \bar{y})$, and all of their neighbors as well (that is, all nodes j such that there exists some i for which

(i, j) and (i, j') are both in E. Within this cluster, we open the cheapest facility i' and use it to serve all clients within this cluster.

We next show that this rounding algorithm has the two claimed properties. Each client j in the cluster is assigned to a facility i' for which there is a path in $G(\bar{x}, \bar{y})$ consisting of an edge connecting i' and j' (of cost at most $g_{j'}$), an edge connecting j' and some node i (of cost at most $g_{j'}$), and an edge connecting i and j (of cost at most g_j). Hence, by the triangle inequality, the cost of assigning j to i' is at most $2g_{j'} + g_j$. Since j was chosen as the remaining client with minimum g-value, it follows that $g_{j'} \leq g_j$, and so the cost of assigning j to i' is at most $3g_j$. In other words, the integer solution found is $3g$-close.

Consider the first cluster formed, and let j' be the node with minimum g-value used in forming it. We know that $\sum_{i:(i,j')\in E} \bar{x}_{ij'} = 1$. Since the minimum of a set of values is never more than a weighted average of them, the cost of the facility selected

$$f_{i'} \leq \sum_{i:(i,j')\in E} \bar{x}_{ij'} f_i \leq \sum_{i:(i,j')\in E} \bar{y}_i f_i,$$

where the last inequality follows from constraint (14). Observe that, throughout the execution of the algorithm, each location $j \in D$ that has not yet been assigned to some cluster, has the property that each of its neighbors i must also remain unassigned. Hence, for each cluster, the cost of its open facility is at most the cost that the fractional solution assigned to nodes in F within that cluster. Hence, in total,

$$\sum_{i\in F} f_i \hat{y}_i \leq \sum_{i\in F} f_i \bar{y}_i.$$

Thus, we have argued that the rounding algorithm of Shmoys, Tardos, & Aardal has the two key properties claimed above.

Suppose that we apply this rounding theorem to an α-filtered solution. What can we prove about the cost of the resulting integer solution? By the two properties proved above, we know that the cost of the solution is at most

$$\sum_{i\in F} f_i \hat{y}_i + \sum_{i\in F}\sum_{j\in D} c_{ij}\hat{x}_{ij} \leq \sum_{i\in F} f_i \bar{y}_i + \sum_{j\in D} 3c_j(\alpha) = \sum_{i\in F} f_i y_i/\alpha + 3\sum_{j\in D} c_j(\alpha).$$

However, exactly analogous to (8), we again know that at most a $(1-\alpha)$ fraction of the values in a weighted average can exceed $1/(1-\alpha)$ times the average, and hence

$$c_j(\alpha) \leq (\sum_{i\in D} c_{ij} x_{ij})/(1-\alpha).$$

Plugging this bound into the previous inequality, we see that the total cost of the solution found is at most

$$\max\{\frac{1}{\alpha}, \frac{3}{1-\alpha}\}(\sum_{i\in F} f_i y_i + \sum_{i\in F}\sum_{j\in D} c_{ij} x_{ij}).$$

If we set $\alpha = 1/4$, then we see that the total cost of the solution found is at most 4 times the cost of (x, y), and so by rounding an optimal solution to the linear relaxation, we obtain a 4-approximation algorithm.

Once again, we may apply the idea of Goemans [10]; it is foolish to set α once, rather than choosing the best α for each input. Once again, we will analyze this best-α algorithm by analyzing a randomized algorithm instead. Let $0 < \beta < 1$ be a parameter to be fixed later. We shall set $\alpha = a$, where a is selected uniformly at random within the interval $[\beta, 1]$. Once again, we shall rely on the fact that

$$\int_0^1 c_j(a)da = \sum_{i=1}^n c_{ij}x_{ij}.$$

The expected cost of the solution found can be upper bounded by

$$\begin{aligned}
E[\frac{1}{a}\sum_{i\in F} f_i y_i + 3\sum_{j\in D} c_j(a)] &= E[\frac{1}{a}]\sum_{i\in F} f_i y_i + 3\sum_{j\in D} E[c_j(a)] \\
&= (\int_\beta^1 \frac{1}{1-\beta}\frac{1}{a}da)\sum_{i\in F} f_i y_i + 3\sum_{j\in D}(\int_\beta^1 \frac{1}{1-\beta}c_j(a)da) \\
&\leq \frac{\ln(1/\beta)}{1-\beta}\sum_{i\in F} f_i y_i + \frac{3}{1-\beta}\sum_{j\in D}\int_0^1 c_j(a)da \\
&= \frac{\ln(1/\beta)}{1-\beta}\sum_{i\in F} f_i y_i + \frac{3}{1-\beta}\sum_{j\in D}\sum_{i\in F} c_{ij}x_{ij}.
\end{aligned}$$

If we set $\beta = 1/e^3$, then we have obtained the claimed $\frac{3}{1-e^3}$-approximation algorithm.

Guha & Khuller [12] proposed the following improvement to the algorithm of Shmoys, Tardos, & Aardal. A natural way in which to compute a better solution is to perform a post-processing phase in which one iteratively checks if an additional facility can be opened to reduce the overall cost, and if so, greedily opens the facility that most reduces the total cost. Furthermore, Guha & Khuller also proposed the following strengthening of the linear programming relaxation. If one knew the cost ϕ incurred to build facilities in the optimal solution, one could add the constraint that $\sum_{i\in F} f_i y_i \leq \phi$. Since we don't know this value, we can instead guess this value by setting ϕ equal to $(1+\epsilon)^k$, for each $k = 1, \ldots, \log_{1+\epsilon}\sum_{i\in F} f_i$, where ϵ is an arbitrarily small positive constant. There are only a polynomial number of settings for ϕ that must be considered, and so, in effect, we may assume that we know the correct ϕ to an arbitrary number of digits of accuracy. By adding the post-processing phase to the result of applying the rounding algorithm to the strengthened relaxation, Guha & Khuller obtain a 2.408-approximation algorithm. Guha & Khuller [12] and Sviridenko [40] independently showed that this problem is MAXSNP-hard, and hence there exists some constant $\rho > 1$ for which no ρ-approximation algorithm exists, unless $\mathcal{P} = \mathcal{NP}$. Guha & Khuller also showed a much stronger result, that no approximation algorithm can have performance guarantee better than 1.463 (unless $\mathcal{NP} \subseteq DTIME(n^{O(\log\log n)})$).

Chudak & Shmoys, independently, obtained a more modest improvement, a 3-approximation algorithm, which relies only on the original linear programming relaxation. The first essential idea in their improvement was the observation that the filtering step is, in some sense, completely unnecessary for the performance of the algorithm. This was based on a simple property of the optimal solution to the linear programming relaxation. Consider the dual to the linear program (12)–(15):

$$\text{Maximize} \sum_{j\in D} v_j \tag{16}$$

subject to

$$\begin{aligned} \sum_{j\in D} w_{ij} &\le f_i, && \text{for each } i \in F, \\ v_j - w_{ij} &\le c_{ij}, && \text{for each } i \in F,\ j \in D, \\ w_{ij} &\ge 0 && \text{for each } i \in F,\ j \in D. \end{aligned}$$

This dual can be motivated in the following way. Suppose that we wish to obtain a lower bound for our input to the uncapacitated facility location problem. If we reset all fixed costs f_i to 0, and solve this input, then clearly we get a (horrible) lower bound: each client $j \in D$ gets assigned to its closest facility at a cost of $\min_{i\in F} c_{ij}$. Now suppose we do something a bit less extreme. Each location $i \in F$ decides on a given cost-sharing of its fixed cost f_i. Each location $j \in D$ is allocated a share w_{ij} of the fixed cost; if j is assigned to an open facility at i, then it must pay an additional fee of w_{ij} (for a total of $c_{ij} + w_{ij}$), but the explicit fixed cost of i is once again reduced to 0. Of course, we insist that each $w_{ij} \ge 0$, and $\sum_{j\in D} w_{ij} \le f_i$ for each $i \in F$. But this is still an easy input to solve: each $j \in D$ incurs a cost $v_j = \min_{i\in F}(c_{ij} + w_{ij})$, and the lower bound is $\sum_{j\in D} v_j$. Of course, we want to allocate the shares so as to maximize this lower bound, and this maximization problem is precisely the LP dual.

Consider a pair of primal and dual optimal solutions: (x, y) and (v, w). Complementary slackness implies that if $x_{ij} > 0$, then the corresponding dual constraint is satisfied with equality. That is, $v_j - w_{ij} = c_{ij}$, and since $w_{ij} \ge 0$, we see that $c_{ij} \le v_j$; in other words, (x, y) is already v-close. Hence, if we apply the rounding algorithm of Shmoys, Tardos, & Aardal (without filtering first, and so $g_j = v_j$), we find a solution of cost at most

$$\sum_{i\in F} f_i y_i + \sum_{j\in D} 3v_j = \sum_{i\in F} f_i y_i + 3\Big(\sum_{i\in F} f_i y_i + \sum_{i\in F}\sum_{j\in D} c_{ij}x_{ij}\Big) \le 4\Big(\sum_{i\in F} f_i y_i + \sum_{i\in F}\sum_{j\in D} c_{ij}x_{ij}\Big),$$

where the first equality follows from the fact that the optimal solutions to the primal and the dual linear programs have equal objective function values.

The second key idea in the improvement of Chudak & Shmoys was the use of randomized rounding in the facility selection step. Randomized rounding is an elegant technique introduced by Raghavan & Thompson [32], in which a feasible solution to a linear programming relaxation of a 0–1 integer program is rounded to an integer solution by interpreting the fractions as probabilities, and setting each variable to 1 with the corresponding probability. Sviridenko [40] proposed a simple randomized rounding approximation algorithm for the special case of the metric uncapacitated facility location problem in which each $c_{ij} \in \{1, 2\}$. In the deterministic algorithm presented above, the cheapest facility in each cluster was opened. Instead, if the cluster is "centered" at j', one can open facility i with probability $x_{ij'}$. This does not really change the previous analysis, since the expected cost of the facilities selected is at most $\sum_{i \in F} f_i y_i$, and the bound on the assignment costs was independent of the choice of the facility opened in each cluster.

The final idea used to obtain the improved performance guarantee is as follows: rather than select the next center by finding the remaining client for which v_j is minimum (since $g_j = v_j$ in the version without filtering), select the client for which $v_j + \sum_{i \in F} c_{ij} x_{ij}$ is minimum. This enters into the analysis in the following way. For each client j in the cluster "centered" at j', its assignment cost is bounded by the cost of an edge (i, j) (of cost at most v_j), an edge (i, j') (of cost at most $v_{j'}$), and the edge (i', j'). The last of these costs is a random variable, and so we can focus on its expected value. Since j' chooses to open each facility i with probability $x_{ij'}$, the expected cost of the edge (i', j') is exactly $\sum_{i \in F} c_{ij'} x_{ij'}$. Thus, the expected cost of assigning j to i' is at most $v_j + v_{j'} + \sum_{i \in F} c_{ij'} x_{ij'}$. By our modified selection rule, this expectation is at most $2v_j + \sum_{i \in F} c_{ij} x_{ij}$, and hence the expected total cost of the solution is at most

$$\sum_{j \in D} 2v_j + \sum_{j \in D} \sum_{i \in F} c_{ij} x_{ij} + \sum_{i \in F} f_i y_i,$$

which is exactly equal to three times the optimal value of the linear programming relaxation.

The analogous deterministic algorithm is quite natural. Before, we merely chose the cheapest facility in each cluster. However, by choosing a facility, we also affect the assignment cost of each client in that cluster. Thus, if choose the facility that minimizes the *total cost for that cluster*, then we achieve a deterministic 3-approximation algorithm.

However, this is not the best possible analysis of this randomized algorithm. Subsequently, Chudak [4] and Chudak & Shmoys [6] have improved this bound to show that (essentially) this randomized algorithm leads to a $(1 + 2/e)$-approximation algorithm. We shall modify the algorithm in the following way. For each location $i \in F$, there is some probability p_i with which it has been opened by this algorithm. (For most locations, it is equal to some value $x_{ij'}$ when facility location i belongs to a cluster "centered" at j', but some locations i might not belong to any cluster.) In the modified algorithm, we also have independent events that open each facility i with probability $y_i - p_i$. In fact, we can simplify some of this discussion by making the following further assumption about the optimal solution (x, y) to the linear program (12)–(15): for each $x_{ij} > 0$, it follows that $x_{ij} = y_i$. We shall say that such a solution is *complete.* This assumption can be made without loss of generality, since it is not hard to show that for any input, there is an equivalent input for which the optimal fractional solution is complete.

For the algorithms above, we have indicated that each client is assigned to the facility that has been opened in its cluster. In fact, there is no need to make this assumption about the assignments, since we may simply assign each client to its cheapest open facility. Given this, the key insight to the improved analysis is as follows. Consider some client j (which is not the center of its cluster). We have shown that its assignment cost is at most $3v_j$ (for the 4-approximation algorithm, and a somewhat better bound for the 3-approximation algorithm). However, the randomized algorithm might very well open one of j's neighbors in $G(x, y)$. In that case, clearly we can obtain a much better bound on the assignment cost incurred for client j. In fact, one can show that the probability that a facility has been opened at least one of j's neighbors is at least $(1 - 1/e)$, and this is the basic insight that leads to the improved analysis.

Although the complete analysis of this algorithm is beyond the scope of this survey, we will outline its main ideas. The improvement in the bound is solely due to the fact that we can bound the expected assignment cost for each client j by $\sum_{i \in F} c_{ij}x_{ij} + (2/e)v_j$. In fact, we will only sketch the proof that this expectation is at most $\sum_{i \in F} c_{ij}x_{ij} + (3/e)v_j$, and will use as a starting point, the original clustering algorithm in which the next client selected is the one for which v_j is smallest (rather than the modified one in which selection was based on $v_j + \sum_{i \in F} c_{ij}x_{ij}$).

Suppose that the neighbors of client j in $G(x, y)$ happen to be nodes $1, \ldots, d$, where $c_{1j} \leq \cdots \leq c_{dj}$. Thus, $\sum_{i=1}^{d} x_{ij} = \sum_{i=1}^{d} y_i = 1$. We can bound the expected assignment cost for j, by considering nodes $i = 1, \ldots, d$ in turn, assigning j to the first of these that has been opened, and if none of these facilities have been opened, then assigning j to the "back-up" facility i' that has surely been opened in its cluster. If opening neighboring facilities $i = 1, \ldots, d$ were independent events, then a simple upper bound on the expected assignment cost for j is

$$y_1c_{1j} + (1 - y_1)y_2c_{2j} + \cdots + (1 - y_1)\cdots(1 - y_{d-1})y_dc_{dj} + (1 - y_1)\cdots(1 - y_d)3v_j,$$

which is clearly at most $\sum_{i=1}^{d} c_{ij}y_i + 3v_j \prod_{i=1}^{d}(1 - y_i)$. The Taylor series expansion of e^{-r} implies that $1 - r \leq e^{-r}$. Using this fact, and the assumption that the optimal LP solution (x, y) is complete, we see that the expected assignment cost for j is at most $\sum_{i \in F} c_{ij}x_{ij} + (3/e)v_j$.

However, opening the neighboring facilities $i = 1, \ldots, d$ are not independent events: for instance, if two of these neighbors are in the same cluster, then only one of them can be opened. The next question is: can the conditioning between these events be harmful? Fortunately, the answer is no, and it is fairly intuitive to see why this is the case. If it happens that none of the first k neighbors of j have not been opened, this only makes it more likely that the next cheapest facility is, in fact, open. A precise analysis of this situation can be given, and so one can prove that the expected assignment cost for j is at most $\sum_{i \in F} c_{ij}x_{ij} + (3/e)v_j$ (without relying on unsupportable assuptions).

These randomized approximation algorithms can each be derandomized, by a straightforward application of the method of conditional probabilities. Thus, if we return to the selection rule in which the next cluster is "centered" at the remaining client j for which $v_j + \sum_{i \in F} c_{ij} x_{ij}$ is minimized, then this derandomization leads to a $(1 + 2/e)$-approximation algorithm.

For the uncapacitated facility location problem, the natural questions for further research are even more tantalizing than for the scheduling problem discussed in the previous section. It is not known that the analysis of the algorithm of Chudak & Shmoys is tight (and in fact, we suspect that it is not tight). Guha & Khuller [12] have given an input for which the ratio between the optimal integer and fractional optima is at least 1.463, but this still leaves some room between that and the upper bound of $1 + 2/e \approx 1.736$ implied by the last algorithm. Furthermore, there are well-known ways to construct stronger linear programming relaxations for this problem, and it would be very interesting to use them to prove stronger performance guarantees.

References

1. M. L. Balinksi. On finding integer solutions to linear programs. In *Proceedings of the IBM Scientific Computing Symposium on Combinatorial Problems*, pages 225–248. IBM, 1966.
2. C. Chekuri and R. Motwani. Precedence constrained scheduling to minimize weighted completion time on a single machine. Unpublished manuscript, 1997.
3. C. Chekuri, R. Motwani, B. Natarajan, and C. Stein. Approximation techniques for average completion time scheduling. *Proceedings of the Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 609–618, 1997.
4. F. A. Chudak. Improved approximation algorithms for uncapacitated facility location. In: *Proceedings of the 6th Integer Programming and Combinatorial Optimization Conference (IPCO)*, 1998, to appear.
5. F. A. Chudak and D. S. Hochbaum. A half-integral linear programming relaxation for scheduling precedence-constrained jobs on a single machine. Unpublished manuscript, 1997.
6. F. A. Chudak and D. B Shmoys. Improved approximation algorithms for the uncapacitated facility location problem. Unpublished manuscript, 1997.
7. V. Chvátal. A greedy heuristic for the set covering problem. *Math. Oper. Res.*, 4:233–235, 1979.
8. M. E. Dyer and L. A. Wolsey. Formulating the single machine sequencing problem with release dates as a mixed integer program. *Discrete Appl. Math.*, 26:255–270, 1990.
9. G. Even, J. Naor, S. Rao, and B. Schieber. Divide-and-conquer approximation algorithms via spreading metrics. In *Proceedings of the 36th Annual IEEE Symposium on Foundations of Computer Science*, pages 62–71, 1995.
10. M. X. Goemans. Personal communication, June, 1996.
11. 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. *Ann. Discrete Math.*, 5:287–326, 1979.
12. S. Guha and S. Khuller. Greedy strikes back: Improved facility location algorithms. In *Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 649–657, 1998.

13. L. A. Hall, A. S. Schulz, D. B. Shmoys, and J. Wein. Scheduling to minimize the average completion time: on-line and off-line approximation algorithms. *Math. Oper. Res.*, 22:513–544, 1997.
14. L. A. Hall, D. B. Shmoys, and J. Wein. Scheduling to minimize the average completion time: on-line and off-line algorithms. In *Proceedings of the 7th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 142–151, 1996.
15. D. S. Hochbaum. Heuristics for the fixed cost median problem. *Math. Programming*, 22:148–162, 1982.
16. D. S. Hochbaum, editor. *Approximation algorithms for NP-hard problems*, Boston, MA, 1997. PWS.
17. D. S. Hochbaum, N. Megiddo, J. Naor, and A. Tamir. Tight bounds and 2-approximation algorithms for integer programs with two variables per inequality. *Math. Programming*, 62:69–83, 1993.
18. D. S. Hochbaum and D. B. Shmoys. A best possible approximation algorithm for the k-center problem. *Math. Oper. Res.*, 10:180–184, 1985.
19. D. S. Johnson. Approximation algorithms for combinatorial problems. *J. Comput. System Sci.*, 9:256–278, 1974.
20. A. A. Kuehn and M. J. Hamburger. A heuristic program for locating warehouses. *Management Sci.*, 9:643–666, 1963.
21. E. L. Lawler. *Combinatorial Optimization: Networks and Matroids*. Holt, Rinehart, and Winston, New York, 1976.
22. J. K. Lenstra, A. H. G. Rinnooy Kan, and P. Brucker. Complexity of machine scheduling problems. *Ann. Discrete Math.*, 1:343–362, 1977.
23. J.-H. Lin and J. S. Vitter. Approximation algorithms for geometric median problems. *Inform. Proc. Lett.*, 44:245–249, 1992.
24. J.-H. Lin and J. S. Vitter. ϵ-approximations with minimum packing constraint violation. In *Proceedings of the 24th Annual ACM Symposium on Theory of Computing*, pages 771–782, 1992.
25. L. Lovász. On the ratio of optimal integral and fractional covers. *Discrete Math.*, 13:383–390, 1975.
26. A. S. Manne. Plant location under economies-of-scale-decentralization and computation. *Management Sci.*, 11:213–235, 1964.
27. F. Margot, M. Queyranne, and Y. Wang. Decompositions, network flows and a precedence constrained single machine scheduling problem. Unpublished manuscript, December, 1996.
28. A. Munier and J. C. König. A heuristic for a scheduling problem with communication delays. *Oper. Res.*, 45:145–147, 1997.
29. C. A. Phillips, C. Stein, and J. Wein. Minimizing average completion time in the presence of release dates. *Math. Programming B*, 1998. To appear.
30. C. N. Potts. An algorithm for the single machine sequencing problem with precedence constraints. *Math. Programming Stud.*, 13:78–87, 1980.
31. M. Queyranne. Structure of a simple scheduling polyhedron. *Math. Programming*, 58:263–285, 1993.
32. P. Raghavan and C. D. Thompson. Randomized rounding: a technique for provably good algorithms and algorithmic proofs. *Combinatorica*, 7:365–374, 1987.
33. R. Ravi, A. Agrawal, and P. Klein. Ordering problems approximated: single-processor scheduling and interval graph completion. In *Proceedings of the 18th International Colloquium on Automata, Languages, and Processing, Lecture Notes in Computer Science 510*, pages 751–762, 1991.
34. A. S. Schulz and M. Skutella. Personal communication, 1997.

35. A. S. Schulz. *Scheduling and Polytopes.* PhD thesis, Technical University of Berlin, 1996.
36. D. B. Shmoys, É. Tardos, and K. I. Aardal. Approximation algorithms for facility location problems. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, pages 265–274, 1997.
37. J. B. Sidney. Decomposition algorithms for single-machine sequencing with precedence and deferral costs. *Oper. Res.*, pages 283–298, 1975.
38. J. F. Stollsteimer. *The effect of technical change and output expansion on the optimum number, size and location of pear marketing facilities in a California pear producing region.* PhD thesis, University of California at Berkeley, Berkeley, California, 1961.
39. J. F. Stollsteimer. A working model for plant numbers and locations. *J. Farm Econom.*, 45:631–645, 1963.
40. M. Sviridenko. Personal communication, July, 1997.
41. L. A. Wolsey. Mixed integer programming formulations for production planning and scheduling problems. Invited talk at the 12th International Symposium on Mathematical Programming, MIT, Cambridge, August, 1985.

This article was processed using the LaTeX macro package with LLNCS style