Fast Approximation Algorithms for Fractional Packing and Covering Problems

Serge A. Plotkin[*] Stanford University
David B. Shmoys[†] Cornell University
Éva Tardos[‡] Cornell University

Abstract

This paper presents fast algorithms that find approximate solutions for a general class of problems, which we call fractional packing and covering problems. The only previously known algorithms for solving these problems are based on general linear programming techniques. The techniques developed in this paper greatly outperform the general methods in many applications, and are extensions of a method previously applied to find approximate solutions to multicommodity flow problems [16, 9, 12]. Our algorithm is a Lagrangean relaxation technique; an important aspect of our results is that we obtain a theoretical analysis of the running time of a Lagrangean relaxation-based algorithm.

We give several applications of our algorithms. The new approach yields several orders of magnitude of improvement over the best previously known running times for the scheduling of unrelated parallel machines in both the preemptive and the non-preemptive models, for the job shop problem, for the cutting-stock problem, and for the minimum-cost multicommodity flow problem.

1 Introduction

We consider the following general type of problem: given a convex set $P \subseteq \mathbf{R}^n$ and a set of m inequalities $Ax \leq b$, decide if there exists a point $x \in P$ that satisfies $Ax \leq b$. We assume that we have a fast subroutine to minimize a non-negative cost function over P. A fractional packing problem is the case when P is in the positive orthant and $A \geq 0$. The intuition for this name is that if P is a polytope, then each $x \in P$ can be written as a convex combination of vertices of P; we are attempting to fractionally pack (or combine) vertices of P subject to the "capacity" constraints $Ax \leq b$.

A wide variety of problems can be expressed as fractional packing problems. Consider the following formulation of deciding if the maximum flow from s to t in an m-edge graph G is at least f: let the polytope $P \subseteq \mathbf{R}^m$ be the convex combination of incidence vectors of (s,t)-paths (scaled so that each such vertex is itself a flow of value f); the edge capacity constraints are given by $Ax \leq b$; the required subroutine for P is a shortest-path algorithm. In words, we view the maximum flow problem as packing (s,t)-paths subject to capacity constraints. We can similarly formulate the multicommodity flow problem, by setting $P = P^1 \times \cdots \times P^k$ where P^ℓ is the polytope of all feasible flows of commodity ℓ, and $Ax \leq b$ describes the joint capacity constraints. In Section 4, we shall discuss the following further applications: the Held & Karp [7] lower bound for the traveling salesman problem (packing 1-trees subject to degree-2 constraints); scheduling unrelated parallel machines in both the preemptive and non-preemptive models, as well as scheduling job shops (packing jobs subject to machine load constraints); and the minimum-cost multicommodity flow problem (packing paths subject to capacity and budget constraints).

Fractional covering problems are another important case of this framework: in this case, P and A are as above, but the constraints are $Ax \geq b$, and we have a maximization routine for P. The following problem of paper manufacturing is an example of a covering problem: paper is produced in wide rolls, called *raws*, and then subdivided into several different widths of narrower ones, called *finals*; there is a specified demand for the number of rolls of each final width, and the aim is to cover that demand using as few raws as possible. This is a classic problem known in the OR literature as the *cutting-stock problem*, and both a natural integer programming formulation and its linear relaxation have been well studied [3, 4]. This linear relaxation is also

[*]Research supported by NSF Research Initiation Award CCR-900-8226, by U.S. Army Research Office Grant DAAL-03-91-G-0102, and by ONR Contract N00014-88-K-0166.

[†]Research partially supported by an NSF PYI award CCR-89-96272 with matching support from UPS, and Sun Microsystems, and by the National Science Foundation, the Air Force Office of Scientific Research, and the Office of Naval Research, through NSF grant DMS-8920550.

[‡]Research supported in part by a Packard Research Fellowship and by the National Science Foundation, the Air Force Office of Scientific Research, and the Office of Naval Research, through NSF grant DMS-8920550.

the key ingredient of the fully polynomial approximation scheme for the bin-packing problem that is due to Karmarkar & Karp [8].

In this paper we focus on obtaining approximate solutions for these problems. For an error parameter $\epsilon > 0$, a point $x \in P$ is an approximate solution for the packing problem if $Ax \leq (1+\epsilon)b$. A point $x \in P$ is an approximate solution for the covering problem if $Ax \geq (1-\epsilon)b$. The running time of our algorithms depends polynomially on ϵ^{-1}, and the *width* of the set P relative to $Ax \leq b$, as defined by $\rho = \max_i \max_{x \in P} a_i x / b_i$, where $a_i x \leq b_i$ is the ith row of $Ax \leq b$. Significantly, the running time does not depend explicitly on n, and hence it can be applied when n is exponentially large, assuming that there exists a polynomial subroutine to optimize cx over P, where $c = y^t A$ (and t denotes transpose). In many applications, our algorithm will have a more efficient randomized analog. The improved expected efficiency of this randomized algorithm can be obtained while maintaining the same worst-case running time as the deterministic version.

All of the problems in our framework are known to be solvable (without relaxing the right-hand-sides) by the ellipsoid method in polynomial time and, more efficiently, by the algorithm of Vaidya [17] in $O(mL)$ calls to an optimization subroutine for P plus $O(m\mathcal{M}(m)L)$ additional time; L and $\mathcal{M}(m)$ denote the binary size of the problem and the time needed to invert m by m matrices, respectively. In order to make a fair comparison between our results and Vaidya's algorithm, we stop his algorithm when it has obtained an approximate solution; in this way, the parameter L becomes, roughly speaking, at least the number of bits of accuracy required. Our algorithm outperforms Vaidya's algorithm if ϵ is large (*e.g.*, a constant), ρ is small relative to m, or the optimization subroutine for P is faster than matrix inversion. In many cases, ρ is not sufficiently small (even exponential), so in Section 3 we give techniques that often reduce ρ. Figure 1 summarizes the comparison of our algorithm to Vaidya's for our applications, giving the speedup over his algorithms [17, 18] when we assume that $\epsilon > 0$ is any constant, and ignore polylogarithmic factors. A function $f(n)$ is said to be $\Omega^*(g(n))$ if $\exists$ a constant c such that $f(n)\log^c n \geq \Omega(g(n))$; we define O^* analogously.

Our approach extends a method previously applied to find approximate solutions to multicommodity flow problems, first by Shahrokhi & Matula [16], and later by Klein, Plotkin, Stein & Tardos [9] and Leighton, Makedon, Plotkin, Stein, Tardos & Tragoudas [12]. Recently, extensions of this method to other applications were found independently by Grigoriadis & Khachiyan [6].

An important theoretical aspect of our results is their connection to Lagrangean relaxation. The main idea of our algorithm is as follows. We maintain a point $x \in P$ that does not satisfy $Ax \leq b$, and repeatedly solve an optimization problem over P to find a direction in which the violation of the inequalities can be decreased. To do this, we define a "penalty" y on the rows of $Ax \leq b$. Rows with $a_i x$ large relative to b_i get a large penalty, other rows get smaller penalties. We relax the $Ax \leq b$ constraints, and instead solve the Lagrangean relaxed problem $\min(yA\tilde{x} : \tilde{x} \in P)$. The idea is that a large penalty tends to imply that the resulting point $\tilde{x}$ improves the corresponding inequality. We then set $x := (1-\sigma)x + \sigma\tilde{x}$, where σ is an suitably small number, and repeat. These ideas are often used to obtain empirically good algorithms for solving linear programs; however, unlike previous methods, we give a rule for adjusting the penalties for which a theoretical analysis proves a very favorable performance in many applications. Lagrangean relaxation has been recognized as an important tool for combinatorial optimization problems since the work of Held & Karp on the traveling salesman problem [7]; in our discussion of this application (in Section 4) we examine the relationship between our algorithm and the traditional approach.

As in [9], our algorithms can also be modified to generate integral approximate solutions and thus yield theorems relating the linear and fractional optima along the lines of Raghavan & Thompson [15] and give alternative deterministic algorithms to obtain the results of Raghavan [14]. The modified algorithm is, in some cases, more efficient than the original algorithm, due to the fact that it terminates as soon as it can no longer improve the current solution while maintaining integrality.

2 Fractional Packing and Covering Algorithms

The *fractional packing problem* is defined as follows:

> PACKING: $\exists ? x \in P$ such that $Ax \leq b$, where A is an $m \times n$ nonnegative matrix, $b \geq 0$, and P is a convex set in the positive orthant of $\mathbf{R}^n$.

We shall use a_i to denote the ith row of A and b_i to denote the ith coordinate of b. We assume that we are given an error parameter $\epsilon > 0$ and a fast subroutine to find $x \in P$ that minimizes an objective function cx, where $c = y^t A$, for any nonnegative y. A vector $x \in P$ such that $Ax \leq (1+\epsilon)b$ is an *approximate solution* to the PACKING problem. A *relaxed decision procedure* either finds an approximate solution or concludes that no (exact) solution exists.

Application	Deterministic Time	Speedup	Randomized Time	Speedup
Preemptive scheduling of N jobs on M machines	$O^*(MN^2)$	$\Omega^*(M^{1.5}N^{.5})$	$O^*(MN)$	$\Omega^*(M^{1.5}N^{1.5})$
Nonpreemptive scheduling of N jobs on M machines	$O^*(MN^2)$	$\Omega^*((N/M)^{1.5})$	$O^*(MN)$	$\Omega^*(N^{1.5}/M^{.5})$
Min-cost K-commodity flow in M-edge N-node graph	$O^*(K^2M^2)$	$\Omega^*(K^{.5}N^2/M^{1.5})$	$O^*(KM^2)$	$\Omega^*(K^{1.5}N^2/M^{1.5})$
Cutting-stock with M widths of finals	$O^*(M^2)$	$\Omega^*(\mathcal{M}(M)/M)$	$O^*(M^2)$	$\Omega^*(\mathcal{M}(M)/M)$
Job shop scheduling of N μ-operation jobs on M machines	$O^*(N^3\mu^3)$	$\Omega^*(N^2M^2\mu^2\sqrt{N})$	—	—

Figure 1: Summary of the performance of the described algorithms

Relaxed Optimality. Consider the following optimization version of the PACKING problem:

$$\min(\lambda : Ax \leq \lambda b \text{ and } x \in P);$$

let λ^* denote the optimal value. A point $x \in P$ is *ϵ-optimal* if it satisfies $Ax \leq (1+\epsilon)\lambda^* b$. Linear programming duality gives a characterization of the optimal solution for the optimization version. Let $y \geq 0$, $y \in \mathbf{R}^m$ denote a *dual solution*, and let $C(y)$ denote the minimum cost cx for any $x \in P$ with respect to costs $c = y^t A$. Let (x, λ) and y denote feasible primal and dual solutions, respectively. Consider the following inequalities:

$$\lambda y^t b \geq y^t A x \geq C(y). \quad (1)$$

Observe that both $y^t b$ and $C(y)$ are independent of x and λ. Hence, for any dual solution y, $C(y)/y^t b \leq \lambda^*$. The goal of our algorithm is to find either an approximate solution x, or feasible primal and dual solutions (x, λ) and y such that $\lambda > 1+\epsilon$ and $\lambda \leq (1+\epsilon)C(y)/y^t b$. In the latter case we can conclude that the current solution is ϵ-optimal and hence no (exact) solution to the PACKING problem exists. Linear programming duality implies that there exists an optimal dual solution y^* for which $C(y^*)/y^{*t} b = \lambda^*$. Hence, for optimal (x^*, λ^*) and y^*, all three terms in (1) are equal.

Consider an error parameter $\epsilon > 0$, a point $x \in P$ satisfying $Ax \leq \lambda b$, and a dual solution y. The following two relaxed optimality conditions can be used to detect if this solution is ϵ-optimal.

$(\mathcal{P}1)$ $(1-\epsilon)\lambda y^t b \leq y^t A x$

$(\mathcal{P}2)$ $y^t A x - C(y) \leq \epsilon(y^t A x + \lambda y^t b)$.

Lemma 2.1 If (x, λ) and y are feasible primal and dual solutions that satisfy the relaxed optimality conditions $\mathcal{P}1$ and $\mathcal{P}2$ and $\epsilon \leq 1/6$, then λ is 6ϵ-optimal.

The algorithm. The core of the algorithm is the procedure IMPROVE, which takes as input a point $x \in P$ and an error parameter $\epsilon \geq 0$. Given x, it computes λ_0, the minimal λ such that $Ax \leq \lambda b$ is currently satisfied.

IMPROVE(x, ϵ)
$\lambda_0 \leftarrow \max_i a_i x/b_i$; $\alpha \leftarrow 4\lambda_0^{-1}\epsilon^{-1}\ln(2m\epsilon^{-1})$; $\sigma \leftarrow \frac{\epsilon}{4\alpha\rho}$.
While $\max_i a_i x/b_i \geq \lambda_0/2$ and x and y do not satisfy $\mathcal{P}2$
  For each i: set $y_i \leftarrow \frac{1}{b_i} e^{\alpha a_i x/b_i}$.
  Find the min-cost point $\tilde{x} \in P$ for costs $c = y^t A$.
  Update $x \leftarrow (1-\sigma)x + \sigma\tilde{x}$.
Return x.

Figure 2: Procedure IMPROVE.

IMPROVE either produces a 6ϵ-optimal solution or reduces λ by a factor of 2. It uses a dual solution y defined as a function of x, where $y_i = \frac{1}{b_i} e^{\alpha a_i x/b_i}$. We will choose α so that the relaxed optimality condition $\mathcal{P}1$ is always satisfied throughout the execution of the algorithm. If at some point we have (x, λ) such that condition $\mathcal{P}2$ is also satisfied, then we claim that λ is sufficiently close to optimality, and IMPROVE terminates. Otherwise we find a point $\tilde{x} \in P$ that attains the minimum $C(y)$, and modify x by setting $x \leftarrow (1-\sigma)x + \sigma\tilde{x}$. Although a single update of x might increase λ, we will show that the sequence of such updates gradually reduces λ.

Lemma 2.2 If $\alpha \geq 2\lambda^{-1}\epsilon^{-1}\log(2m\epsilon^{-1})$, then any feasible solution (x, λ) and its corresponding dual solution y as defined above satisfy $\mathcal{P}1$.

At the beginning of IMPROVE (see Figure 2), α is set to $4\lambda_0^{-1}\epsilon^{-1}\log(2m\epsilon^{-1})$; hence, the relaxed optimality condition $\mathcal{P}1$ is satisfied throughout the execution of the procedure. The following lemma shows that moving the right amount towards a minimum-cost point $\tilde{x}$ results in a significant decrease in the potential function $\Phi = y^t b$.

Lemma 2.3 Consider a point $x \in P$ and an error parameter $\epsilon > 0$ such that x and its corresponding y have potential function value Φ and do not satisfy $\mathcal{P}2$. Let $\tilde{x} \in P$ attain the minimum $C(y)$. Assume that $\frac{\epsilon}{8\alpha\rho} \leq \sigma \leq \frac{\epsilon}{4\alpha\rho}$. Define a new solution by $(1-\sigma)x + \sigma\tilde{x}$, and let $\hat{y}$ and $\hat{\Phi}$ denote the new dual solution and potential function value. Then $\Phi - \hat{\Phi} = \Omega(\frac{\epsilon^2\lambda}{\rho}\Phi)$.

Proof: By the definition of ρ, $Ax \leq \rho b$ and $A\tilde{x} \leq \rho b$. This implies that $\alpha\sigma|a_i x - a_i\tilde{x}|/b_i \leq \epsilon/4 \leq 1/4$. Using the second order Taylor theorem we see that $|\delta| \leq \epsilon/4 \leq 1/4$ implies that for all x, $e^{x+\delta} \leq e^x + \delta e^x + \frac{e}{2}|\delta|e^x$. From this, we get that $\hat{y}_i \leq y_i + \alpha\sigma\frac{1}{b_i}(a_i\tilde{x} - a_i x)y_i + \epsilon\alpha\sigma\frac{1}{2b_i}(a_i\tilde{x} + a_i x)y_i$. Using this bound for $\hat{y}$, we get:

$$\begin{aligned}\Phi - \hat{\Phi} &\geq \alpha\sigma(y^t Ax - y^t A\tilde{x}) - \alpha\sigma\frac{\epsilon}{2}(y^t Ax + y^t A\tilde{x}) \\ &\geq \alpha\sigma(y^t Ax - y^t A\tilde{x}) - \alpha\sigma\epsilon y^t Ax.\end{aligned}$$

The fact that $\mathcal{P}2$ is not satisfied implies that the decrease in Φ is at least $\alpha\sigma\epsilon\lambda\Phi$, which is $\Omega(\frac{\epsilon^2\lambda}{\rho}\Phi)$ by the assumption about the magnitude of σ. ∎

Observe that during a single call to IMPROVE we have $e^{\alpha\lambda_0/2} \leq \Phi \leq me^{\alpha\lambda_0}$. If the initial solution is $O(\epsilon)$-optimal then we have the tighter bound, $e^{\alpha(1+O(\epsilon))^{-1}\lambda_0} \leq \Phi \leq me^{\alpha\lambda_0}$. This, together with the previous lemma, can be used to bound the number of iterations of IMPROVE.

Theorem 2.4 The procedure IMPROVE terminates in $O(\epsilon^{-3}\lambda_0^{-1}\rho\log(m\epsilon^{-1}))$ iterations. If the initial λ_0 is $O(\epsilon)$-optimal, then IMPROVE terminates in $O(\epsilon^{-2}\lambda_0^{-1}\rho\log(m\epsilon^{-1}))$ iterations.

To find an approximate solution with a given approximation parameter ϵ_0, we start by repeatedly applying IMPROVE with $\epsilon = 1/6$ until we find a solution that is either 6ϵ-optimal or has $\lambda \leq 1 + 6\epsilon$. As in [9, 12], we then use ϵ-scaling to reduce ϵ to ϵ_0.

The most time-consuming part of an iteration of IMPROVE is to find the minimum-cost vector $\tilde{x} \in P$. It is not hard to see that a subroutine that finds a point in P of cost no more than an $1 + \epsilon/2$ factor above the minimum can be used in the algorithm, and gives a bound on the number of iterations of the same order of magnitude. Observe that the costs used are exponential functions in x, which can take a long time to compute exactly. It is easy to show that it suffices to use an approximation to the vector y. However, in this abstract we shall not address the issue of the required precision.

Theorem 2.5 For $\epsilon > 0$, an approximate solution for the fractional packing problem can be found by an algorithm that makes $O(\epsilon^{-2}\rho\log(m\epsilon^{-1}))$ calls to a subroutine that computes a minimum-cost point in P.

Notice that the running time does not depend explicitly on n, the dimension of P. This makes it possible to apply the algorithm to problems defined with an exponential number of variables, assuming we have a polynomial-time subroutine to compute a point $x \in P$ of cost $C(y)$ given any positive y.

Randomized version. In some cases, the bound in Theorem 2.5 can be improved using randomization. This approach was introduced by Klein, Plotkin, Stein, & Tardos [9] in the context of multicommodity flow; we shall present other applications in Section 4.

Let us assume that the polytope P can be written as a product of polytopes of smaller dimension, *i.e.*, $P = P^1 \times \cdots \times P^k$. A point $x \in P$ can be written as $x = (x^1, \ldots, x^k)$ where $x^\ell \in P^\ell$ and the inequalities $Ax \leq b$ can be written as $\sum A^\ell x^\ell \leq b$. Let a_i^ℓ denote the ith row of A^ℓ. We can define ρ^ℓ for $\ell = 1, \ldots, k$ as $\rho^\ell = \max_i \max(a_i^\ell x^\ell / b_i : x^\ell \in P^\ell)$. Clearly $\rho \leq \sum_\ell \rho^\ell$. A subroutine to compute $C(y)$ for P consists of k subroutines, where the ℓth subroutine computes the analogous minimum over the polytope P^ℓ. Randomization speeds up the algorithm by roughly a factor of k in the following case: $\rho^\ell = \hat{\rho}\ \ \forall\ell$, the k subroutines run in the same amount of time, and our best estimate on ρ is $k\hat{\rho}$.

The idea of the more efficient algorithm is as follows. To find a minimum-cost point $\tilde{x}$ in P, we have to compute k minimum-cost points $\tilde{x}^\ell \in P^\ell$ for each ℓ. If we perturb the current solution using only $\tilde{x}^\ell$ for one particular polytope P^ℓ, then it is possible to make a significant reduction in the potential function, provided ℓ is chosen appropriately. The key to this version is the analog of Lemma 2.3; if the variable x^ℓ is selected uniformly and is updated using a larger $\sigma = \min\{1, \epsilon/(4\hat{\rho}\alpha)\}$, then this results in roughly the same expected improvement in the potential function. This particular method of randomizing is an extension of an idea that Goldberg [5] has used for the multicommodity flow problem, and was also independently discovered by Grigoriadis & Khachiyan [6].

Theorem 2.6 For $\epsilon > 0$, an ϵ-optimal solution for the fractional packing problem defined with a polytope $P = P^1 \times \cdots \times P^k$ with $\rho^\ell \leq \hat{\rho}$ for every ℓ, can be found by a randomized algorithm that uses an expected $O(k(\epsilon^{-2}\hat{\rho} + 1)\log(m\epsilon^{-1}))$ calls to a subroutine computing minimum-cost solutions in a randomly selected P^ℓ.

Fractional Covering Problems. The *fractional covering problem* is defined as follows:

> COVERING: $\exists ?x \in P$ such that $Ax \geq b$, where $A \geq 0$ is an $m \times n$ matrix, $b > 0$, and P is a convex set in the positive orthant of $\mathbf{R}^n$.

We assume that we are given an error parameter $\epsilon > 0$, and a fast subroutine to maximize the objective function cx over P, where $c = y^t A$, for any nonnegative y. An *approximate solution* to the COVERING problem is a vector $x \in P$ such that $Ax \geq (1 - \epsilon)b$. With a method analogous to our relaxed decision procedure for the PACKING problem, we can prove the following

theorem.

Theorem 2.7 For $\epsilon > 0$, an approximate solution for the fractional covering problem can be found by an algorithm that uses $O(m + \rho \log^2 m + \epsilon^{-2}\rho \log(m\rho\epsilon^{-1}))$ calls to a subroutine maximizing a linear objective function over P.

If the polytope P is in the form of a product $P^1 \times \cdots \times P^k$, we can speed up the algorithm by roughly a factor of k by using randomization in a way analogous to the randomized version of the packing algorithm. The expected number of iterations of the randomized algorithm is $O(m+k\hat{\rho}\log^2 m+k\log\epsilon^{-1}+\epsilon^{-2}k\hat{\rho}\log(m\rho\epsilon^{-1}))$, roughly the same as the number of iterations of the deterministic version, but an iteration of the randomized version is faster because at each iteration the algorithm has to compute maximum value point only in one of the polytopes P_j rather than all of them.

The General Problem. Consider the class of problems in the following form:

> GENERAL: $\exists ? x \in P$ such that $Ax \leq b$, where A is an arbitrary $m \times n$ matrix, b is an arbitrary vector, and P is a convex set in $\mathbf{R}^n$.

We assume that we are given error parameter $\epsilon > 0$, a positive vector d, and a fast subroutine to minimize the objective function cx over P, where $c = y^t A$, for any nonnegative y. We say that a vector $x \in P$ is an *approximate solution* if $Ax \leq b + \epsilon d$. The running time of our relaxed decision procedure for this problem depends on the width ρ, which is defined in this case by $\rho = \max_i \max_{x \in P} \frac{|a_i x - b_i|}{d_i}$.

Theorem 2.8 For $\epsilon > 0$, an approximate solution for the problem can be found by an algorithm that uses $O(\rho^2\epsilon^{-2}\log(m\rho\epsilon^{-1}))$ calls to a subroutine computing a minimum-cost point in P.

This formulation of the problem is quite general. In the next section we show how to limit ρ for problems with simultaneous packing and covering constraints, *i.e.*, where the coefficients of each row of $Ax \leq b$ are either all positive (a packing constraint) or all negative (a covering constraint) and d is defined by $d_i = |b_i|$ for all i.

3 Decreasing the width ρ

The running times of our algorithms are proportional to the width ρ. In this section we present techniques that transform the original problem into an equivalent one, while reducing the width. Each of the techniques assumes the existence of a particular fast subroutine related to optimization over P; different subroutines might be available in different applications.

Relaxation of Integer Programming. Suppose that we are solving the packing problem defined by a convex set P and inequalities $Ax \leq b$ in order to compute a bound on an integer program by solving its linear relaxation. The assumptions that $A \geq 0$ and P is in the nonnegative orthant imply that any integer solution must satisfy $x_j = 0$ whenever $\exists i$ such that $a_{ij} > b_i$. Hence, instead of using P, we can use $\hat{P} = \{x : x \in P,\ x_j = 0 \text{ if } j \in J\}$, where $J = \{j : \exists i \text{ such that } a_{ij} > b_i\}$. The width $\hat{\rho}$ of $\hat{P}$ relative to $Ax \leq b$ is bounded by $\gamma = \max_{x \in P} \sum_j x_j$. For example, if the variables of the integer program are restricted to be 0 or 1, we get $\gamma \leq n$. When P is a direct product of convex sets, then so is $\hat{P}$, and hence the same technique can also be used to obtain a more efficient randomized version of the packing algorithm.

Decomposition for Packing Problems. Consider the packing problem defined by a polytope P and inequalities $Ax \leq b$. We introduce a decomposition technique that defines a related problem with decreased ρ by replacing P and A by an equivalent problem in the product form. One application of this technique is to solve the minimum-cost multicommodity flow problem, as described in the next section. We assume that there exists a fast subroutine for the following:

> Given a constant ν and a dual solution y, find a vertex $\tilde{x} \in P$ such that:
>
> $$A\tilde{x} \leq \nu b, \text{ and} \qquad (2)$$
> $$y^t A\tilde{x} = \min(y^t Ax : x \text{ a vertex of } P \text{ s.t. } Ax \leq \nu b).$$

Instead of finding a minimum-cost point in P that satisfies $Ax \leq \nu b$, the subroutine finds a minimum-cost *vertex* of P that satisfies the same condition. This subroutine, at first glance, might appear to directly solve the PACKING problem. However, observe that even if an instance is feasible, all vertices of P might violate $Ax \leq b$, and hence we cannot directly use subroutine (2) with $\nu = 1$ to solve the packing problem.

For simplicity of presentation we assume without loss of generality that P is the simplex: $P = \{x : \sum_j x_j = d,\ x \geq 0\}$ for some d. To convert a packing problem into this form we write each point $x \in P$ as a convex combination of the vertices of P; that is, if v_j, $j = 1, \ldots, s$, denote the vertices of P, then $x = \sum_j \xi_j v_j$, where $\sum_j \xi_j = 1$, and $\xi_j \geq 0$, $j = 1, \ldots, s$. If we let ξ_j, $j = 1, \ldots, s$, be the variables of the transformed problem, then this yields a problem in which the polytope is a simplex with $d = 1$, and there are possibly exponentially many variables; the coefficient of ξ_j in the ith packing constraint is $a_i v_j$. Note that essentially the

same optimization subroutine (2) suffices; the only difference is that the vertex $\tilde{x}$ must be (trivially) expressed in terms of the new variables.

We first show how to obtain an equivalent problem for which the width is roughly half of its original value ρ. We introduce two copies of each x_j: x_j^1 and x_j'. If we let $J = \{j : \exists i \text{ s.t. } a_{ij}d > 2mb_i\}$, then the new polytope is $P^1 \times P'$, where $P' = (1/2)P$ and $P^1 = \{x^1 : x^1 \in P',\ x_j^1 = 0 \text{ if } j \in J\}$. The new set of inequalities is $Ax^1 + Ax' \leq b$. Most importantly, the width of P' is $\rho' = \rho/2$, and the width of P^1 is $\rho^1 \leq m$.

Lemma 3.1 If the packing problem defined by P and $Ax \leq b$ has a solution, then so does the packing problem defined by $P^1 \times P'$ and $Ax^1 + Ax' \leq b$. An approximate solution to the latter problem can be used to find an approximate solution to the former.

Proof: We first note that if (x^1, x') is an approximate solution to the transformed problem, then $x = x^1 + x'$ is an approximate solution to the untransformed problem.

Now assume that we have a feasible solution x to the original problem. We claim that any such x corresponds to some solution of the transformed problem. Initially, set $x_j' = x_j$ and $x_j^1 = 0$ for all j. While $\sum_j x_j^1 < d/2$, find $j \notin J$ with $x_j' > 0$, and simultaneously increase x_j^1 and decrease x_j' by the same amount. Note that this maintains $Ax^1 + Ax' \leq b$. We claim that we can always continue if $x^1 \notin P^1$. Assume, for a contradiction, that for the current solution (x^1, x'), $j \in J$ whenever $x_j' > 0$, and $x^1 \notin P^1$. Hence, for each j such that $x_j' > 0$, we have a row $i(j)$ that contains a large coefficient in the jth column, $a_{i(j)j} > 2mb_{i(j)}/d$. Since $x^1 \notin P^1$, we know that $\sum_j x_j' > d/2$. By averaging over all rows, one can show that there exists a row which contradicts $Ax' \leq b$. ∎

If we recursively apply the above transformation to P', after $\log \rho$ applications we obtain an equivalent packing problem where its polytope is a product of $1 + \log \rho$ polytopes, and the width of each one of these subproblems is at most m. The minimization subroutine for the ℓth set is given by subroutine (2) with $\nu = 2^\ell m/d$.

Theorem 3.2 For $\epsilon > 0$, an approximate solution for the fractional packing problem can be found by a randomized algorithm that uses an expected $O(\epsilon^{-2} m \log \rho \log(m\epsilon^{-1}))$ calls to the subroutine (2), or by a deterministic algorithm that uses $O(\epsilon^{-2} m \log^2 \rho \log(m\epsilon^{-1}))$ calls.

Taking advantage of the fact that we are only interested in approximate solutions, we can improve the above theorem by replacing $\log \rho$ by $\log \epsilon^{-1}$.

In the minimum-cost multicommodity flow problem, the subroutine (2) will not be available. Instead, we will have the following relaxed subroutine for some parameters γ_i for $i = 1, \ldots, m$.

> Given a constant ν and a dual solution y, find a vertex $\tilde{x} \in P$ such that:
>
> $$\begin{aligned} a_i\tilde{x} &\leq \gamma_i \nu b_i \quad \forall i, \text{ and} \\ y^t A\tilde{x} &\leq \min(y^t Ax : x \text{ a vertex of } P \text{ with } Ax \leq \nu b). \end{aligned} \tag{3}$$

The subroutine (2) is a special case of (3) when $\gamma_i = 1$, $i = 1, \ldots, m$.

We can apply a transformation similar to the one described above, where the polytopes are defined using the restriction that $x_j^\ell = 0$ if $\exists i$ such that $a_{ij}d > 2^\ell(\Gamma/\gamma_i)b_i$. The packing algorithm for the transformed problem can easily be modified to use the subroutine (3). An analogous result can be proved if the original polytope P is in product form.

Theorem 3.3 For $\epsilon > 0$, an approximate solution for the fractional packing problem defined by $P = P^1 \times \cdots \times P^k$ can be found by a randomized algorithm using expected $O(\epsilon^{-2} k\Gamma \log \epsilon^{-1} \log(m\epsilon^{-1}))$ calls to the subroutine (3) for some P^ℓ, or by a deterministic algorithm using $O(\epsilon^{-2} k^2 \Gamma \log^2 \epsilon^{-1} \log(m\epsilon^{-1}))$ calls.

Decomposition for Covering Problems. We present a decomposition technique for the covering problem, which is analogous to the technique used for the packing problem. This technique will be used by our algorithm for the cutting-stock problem, as described in Section 4.

Consider the covering problem defined by the polytope P and the inequalities $Ax \geq b$. The subroutine that we use is as follows:

> Given a constant ν and a dual solution y, find a vertex $\tilde{x} \in P$ such that:
>
> $$\sum_{i \in I(\tilde{x})} y_i a_i \tilde{x} = \max\Big(\sum_{i \in I(x)} y_i a_i x : x \text{ a vertex of } P\Big), \tag{4}$$
>
> where $I(x) = \{i : a_i x \leq \nu b_i\}$.

For simplicity of presentation we shall again concentrate, without loss of generality, on the case when P is the simplex: $P = \{x : \sum_j x_j = d, x \geq 0\}$ for some d.

We first show how to obtain an equivalent problem for which the width is roughly half of its original value ρ. We introduce two copies of each x_j: x_j^1 and x_j'. The new polytope is $P^1 \times P'$, where $P' = P^1 = (1/2)P$. The new set of inequalities is $A^1x^1 + Ax' \geq b$ where a_{ij}^1, the (i,j)th coefficient of the matrix A^1 is a_{ij} if $a_{ij}d \leq 2mb_i$, and 0 otherwise. Note that the width of P' is $\rho' = \rho/2$, and the width of P^1 is $\rho^1 \leq m$.

Lemma 3.4 If the covering problem defined by P and $Ax \geq b$ has a solution, then so does the covering problem defined by $P^1 \times P'$ and $A^1x^1 + Ax' \geq b$. An approximate solution to the latter problem can be used to find an approximate solution to the former.

Proof: We first note that if (x^1, x') is an approximate solution to the transformed problem, then $x = x^1 + x'$ is an approximate solution to the untransformed problem.

Now assume that we have a feasible solution x to the original problem. We claim that any such x corresponds to some solution of the transformed problem. Initially, set $x'_j = x_j$ and $x^1_j = 0$ for all j. While $\sum_j x'_j > d/2$, find j with $x'_j > 0$ such that for every i with $a^1_i x^1 + a_i x' = b_i$, we have that $a^1_{ij} = a_{ij}$. Simultaneously increase x^1_j and decrease x'_j by the same amount, so that $A^1x^1 + Ax' \geq b$ is maintained. We claim that we can always continue if $x' \notin P'$. Let (x^1, x') be the current solution, and assume, for a contradiction, that for each $x'_j > 0$, we can select a row $i(j)$ such that $a^1_{i(j)j} \neq a_{i(j)j}$ and $a^1_{i(j)}x^1 + a_{i(j)}x' = b_{i(j)}$. By the definition of the matrix A^1, we have that $a_{i(j)j} > 2mb_{i(j)}/d$. Since $x' \notin P'$, we know that $\sum_j x'_j > d/2$. By averaging over the selected rows, one can show that one of these must be a strict inequality, which is a contradiction. ∎

If we recursively apply the above transformation to P', after $\log \rho$ applications we obtain an equivalent covering problem where its polytope is a product of $1+\log \rho$ polytopes, and the width of each one of these subproblems is at most m. The maximization subroutine for the ℓth set is given by subroutine (4) with $\nu = 2^\ell m/d$.

Theorem 3.5 For $\epsilon > 0$, an approximate solution for the fractional covering problem can be found by a randomized algorithm that uses expected $O(m \log \rho \log^2 m + \epsilon^{-2} m \log \rho \log(m\epsilon^{-1}))$ calls to the subroutine (4), or by a deterministic algorithm that uses $O(m \log^2 \rho \log^2 m + \epsilon^{-2} m \log^2 \rho \log(m\epsilon^{-1}))$ calls.

As in Theorem 3.3, we can derive an analogous result that uses a similarly less demanding subroutine (though the $\log \epsilon$ terms in that result are replaced by $\log \rho$).

Decomposition for Simultaneous Packing and Covering. It is not hard to see that a combination of the techniques used to derive Theorems 3.2 and 3.5 and their extensions can be used to obtain analogous versions of Theorem 2.8 for problems with simultaneous packing and covering constraints.

4 Applications

In this section, we will show how to apply the techniques presented in the previous three sections to a variety of linear programs related to packing and covering problems. For an optimization problem, an ϵ-approximation algorithm delivers a solution of value within a factor of $(1 + \epsilon)$ of optimal. Although we will focus on ϵ-approximation algorithms for fixed ϵ, this is only to simplify the discussion of running times. In each of the applications except for the Held-Karp bound, we obtain a significant speedup over previously known algorithms; the reader is referred to Figure 1 for these comparisons.

Scheduling unrelated parallel machines: with and without preemption. Suppose that there are N jobs and M machines, and each job must be scheduled on exactly one of the machines. For simplicity, assume that $N \geq M$. Job j takes p_{ij} time units when processed by machine i. The *length* of a schedule is the maximum total processing time assigned to run on a machine; the objective is to minimize the schedule length. This problem, often denoted $R||C_{\max}$, is NP-complete; Lenstra, Shmoys, & Tardos [13] gave a 1-approximation algorithm for it, based on a relaxed decision procedure with $\epsilon = 1$. If there exists a schedule of length T, then the following LP has a feasible solution:

$$\sum_{j=1}^{N} p_{ij}x_{ij} \leq T, \quad i = 1, \ldots, M; \tag{5}$$

$$\sum_{i=1}^{M} x_{ij} = 1, \quad j = 1, \ldots, N; \tag{6}$$

$$x_{ij} = 0 \text{ if } p_{ij} > T, \tag{7}$$

$$x_{ij} \geq 0 \text{ if } p_{ij} \leq T, \quad i = 1, \ldots, M, \quad j = 1, \ldots, N. \tag{8}$$

Any vertex of this polytope can be rounded to a schedule of length $2T$ in $O(M)$ time.

To apply Theorem 2.5, we let P be defined by the constraints (6–8). It is easy to see that $\rho \leq N$: for any $x \in P$, $x_{ij} > 0$ implies that $p_{ij} \leq T$, and so $\sum_{j=1}^{N} p_{ij}x_{ij}/T \leq N$, $\forall i$. In each iteration, job j has modified processing times $y_i p_{ij}$, and we construct the schedule in which each job is assigned to the machine for which this time is smallest; this takes $O(MN)$ time. For any fixed $\epsilon > 0$, this implies that a fractional solution $\bar{x}$ of length $(1+\epsilon)T$ can be found in $O(MN^2 \log M)$ time, if one of length T exists. In order to apply the rounding procedure, $\bar{x}$ must be converted to a vertex of the polytope. If $K = |\{\bar{x}_{ij} : \bar{x}_{ij} > 0\}|$, then this can be done in $O(KM)$ time. Each iteration of IMPROVE produces at most N new positive variables, and hence $K = O(N^2 \log M)$.

Note that $P = P^1 \times \cdots \times P^N$, where $P^j = \{x^j : \sum_{i=1}^m x_{ij} = 1; x_{ij} = 0 \text{ if } p_{ij} > T, x_{ij} \geq 0 \text{ if } p_{ij} \leq T, \forall i\}$ and $\rho^j \leq 1$, $\forall j$. To optimize over P^j, we find the machine on which job j has minimum modified processing time $y_i p_{ij}$. Applying Theorem 2.6, we get a randomized algorithm that takes $O(N \log M)$ iterations, but now each iteration takes $O(M)$ time and $K = O(N \log M)$. Using bisection search to find the best length T, we get the following theorem.

Theorem 4.1 For any fixed $r > 1$, there is a deterministic r-approximation algorithm for $R||C_{\max}$ and a randomized analog that run in $O(MN^2 \log^2 M)$ and $O(MN \log^2 M)$ time, respectively.

Van de Velde [19] has obtained good results using an ascent algorithm for this LP, where each iteration invokes the same subroutine, but uses a much simpler rule for updating y.

In a related model, we consider schedules with preemptions: a job may be started on one machine, interrupted, and then later continued on another. Lawler & Labetoulle [11] showed that an optimal preemptive schedule for this problem, $R|pmtn|C_{\max}$, can be found by solving the following linear program: minimize T subject to a modified version of the previous LP where (7–8) are replaced by $x \geq 0$ and $\sum_i p_{ij} x_{ij} \leq T$, $j = 1, \ldots, N$ (which imply that each job is processed for no more than T units of time). We perform a bisection search to find the optimal T, and in each iteration we let P be defined by (6) and the constraints just added. As above, P can be expressed as $P^1 \times \cdots \times P^N$, and optimizing over each P^ℓ corresponds to solving a fractional multiple-choice knapsack problem, which can be solved in linear time [1]. Given the fractional solution, it is necessary to convert it into a schedule, preferably with few preemptions, and this can be done using an edge-coloring algorithm on a related bipartite multigraph.

Theorem 4.2 For any constant $\epsilon > 0$, there are deterministic and randomized ϵ-approximation algorithms for $R|pmtn|C_{\max}$ that run in $O(MN^2 \log^2 M)$ time and $O(MN(\log^2 M + \log N))$ time, respectively.

Job shop scheduling. In the job shop scheduling problem, there are N jobs to be scheduled on a collection of M machines; each job j consists of a specified sequence of operations, $O_{1j}, O_{2j}, \ldots, O_{\mu j}$, where O_{ij} must be processed on a particular machine m_{ij} for p_{ij} time units without interruption; the operations of each job must be processed in the given order, and each machine can process at most one operation at a time; the aim is to schedule the jobs so as to minimize the time by which all jobs are completed. Shmoys, Stein, & Wein give an $O(\log^2(M + \mu))$-approximation algorithm for this problem that uses the randomized rounding technique of Raghavan & Thompson [15] and its deterministic analogue due to Raghavan [14].

The overwhelming computational bottleneck of the deterministic algorithm is the solution of a fractional packing problem that is used to compute an initial delay for each of the jobs so as to minimize the maximum number of jobs that are simultaneously assigned to the same machine when delayed for that amount of time, and then run continuously until completion. In contrast, the delays are simply randomly selected according to a simple probability distribution in the faster variant. Using our algorithm, this packing problem can be solved more efficiently to yield the following theorem.

Theorem 4.3 A job shop schedule with maximum completion time that is a factor $O(\log^2(M+\mu))$ more than optimal can be found deterministically in $O(\tilde{N}^3 \mu^3 \log(M + \mu))$ time, where $\tilde{N} = \min\{N, M^2\mu^3\}$.

The Held-Karp bound for the TSP with triangle inequality. One of the most useful ways to obtain a lower bound on the length of the optimum tour for the traveling salesman problem (TSP) was proposed by Held & Karp [7], and is based on the idea of Lagrangean relaxation. We shall assume that an instance of the TSP is given by a symmetric $N \times N$ cost matrix $C = (c_{ij})$ that satisfies the triangle inequality, *i.e.*, $c_{ij} + c_{jk} \geq c_{ik}, \forall i, j, k$, and has optimum tour length $TSP(C)$. A 1-tree consists of 2 edges incident to node 1, and a spanning tree on $\{2, \ldots, N\}$. Since every tour is a 1-tree, the cost of the minimum 1-tree is at most $TSP(C)$; furthermore, it can be computed by a minimum spanning tree computation. Each node i is then given a price p_i and reduced costs $\bar{c}_{ij} = c_{ij} + p_i + p_j$ are formed; if $\bar{c}_T$ is the cost of a minimum 1-tree with respect to the reduced costs, then $\bar{c}_T - 2\sum_i p_i \leq TSP(C)$. The Held-Karp bound is attained by choosing p to maximize this lower bound. Held & Karp gave a subgradient optimization method to find such a p by iteratively computing the minimum 1-tree T, and then adjusting p by taking a step proportional to $d_i - 2$, where d_i is the degree of node i in T.

It is possible to formulate this bound as an LP with a constraint for each possible 1-tree, $T_1, T_2, \ldots, T_s$. Its dual is as follows, where c_j is the total cost of T_j (using C), and d_{ij} is the degree of node i in T_j: minimize $\sum_j c_j x_j$ subject to

$$\sum_{j=1}^{s} d_{ij} x_j \leq 2, \quad i = 1, \ldots, N, \tag{9}$$

$$\sum_{j=1}^{s} x_j = 1, \quad x_j \geq 0, \ j = 1, \ldots, s. \tag{10}$$

We apply Theorem 2.5 by using a bisection search for the minimum feasible cost K, so that P is defined by (10) and $Ax \leq b$ is equal to (9) and $\sum_{j=1}^{s} c_j x_j \leq K$.

The triangle inequality implies that $\rho \leq N$. To minimize a linear objective over P, we choose the 1-tree with minimum objective coefficient. If $y \in \mathbf{R}^N$ and $z \in \mathbf{R}$ denote the dual variables for $Ax \leq b$, then the objective coefficient of x_j is $c_j z + \sum_{i=1}^{n} d_{ij} y_j$. This implies that the minimum 1-tree found in this iteration is precisely the 1-tree found by minimizing with respect to the reduced costs $\bar{c}$ with $p = y/z$, which was used in each iteration of the Held-Karp subgradient optimization method. Of course, we use a rather different rule to compute the new vector p for the next iteration.

The bisection search produces a value $\tilde{K}$ and a solution $\tilde{x} \in P$ that satisfies $A\tilde{x} \leq (1+\epsilon)b$ with $K = \tilde{K}$, whereas for $K \leq \tilde{K}/(1+\epsilon)$, $\nexists x \in P$ of cost K that satisfies (9). However, this does *not* imply that $\tilde{K}$ is within a factor of $(1+\epsilon)$ of the optimum Held-Karp bound; it is possible that any $x \in P$ that satisfies (9) has a much larger cost. We *can* convert $\tilde{x}$ into such a solution by carefully perturbing it, but this increases its cost by a factor of $(1+\epsilon N)$. If $\epsilon = O(\epsilon_0/N)$, we obtain the Held-Karp bound within a factor of $1+\epsilon_0$. Unfortunately, this implies that the algorithm might run for $O(N^3 \log N)$ iterations, where each iteration takes $O(N^2)$ time. In contrast, Vaidya's algorithm [17] takes $O(N\mathcal{M}(N)\log N)$ time.

The cutting-stock problem. In the cutting-stock problem, we wish to subdivide a minimum number of raws of width W, in order to satisfy a demand d_i for finals of width w_i, $i = 1, \ldots, M$. This can be formulated as an integer program with a variable x_j for each feasible pattern for subdividing a single raw; that is, a pattern is an vector $b^t \in \mathbf{N}^M$, such that $\sum_i b_i w_i \leq W$, and $b_i \leq d_i$, $i = 1, \ldots, M$. Let $(a_{1j}, \ldots, a_{Mj})^t$, $j = 1, \ldots, N$, be a list of all patterns. Then we wish to minimize $\sum_j x_j$ subject to

$$\sum_{j=1}^{N} a_{ij} x_j \geq d_i, \quad i = 1, \ldots, M, \tag{11}$$

and $x_j \geq 0$, integer, $j = 1, \ldots, N$. Although we want an integer solution, the linear relaxation of this formulation has been extremely useful in practical applications; furthermore, there are applications in which patterns may be used fractionally [2].

Using a bisection search for the minimum number of raws, we try to find $x \in P = \{x_j : \sum_{j=1}^{N} x_j = r,\ x_j \geq 0,\ j = 1, \ldots, N\}$ that satisfies (11). Since ρ can only be bounded by r, we will use the decomposition result, and so we need subroutine (4) for this application. Consider a vertex x of P, where $x_k = r$ and $x_j = 0$, $j \neq k$. The *profit* of this kth pattern is $\sum((y_i r)a_{ik} : i \text{ such that } a_{ik} \leq \nu d_i/r)$; each final of width w_i that is used in this pattern has a profit of $y_i r$, unless more than $\nu d_i/r$ finals of width w_i are used, in which case none of those finals has any profit. For each pattern a, any vector b such that $b \leq a$ is also a pattern, and so we can find the optimal vertex of P among those patterns j for which $a_{ij} \leq \nu d_i/r$, $i = 1, \ldots, M$. Hence, subroutine (4) is equivalent to solving the following knapsack problem: there are M types of pieces, such that type i has weight w_i and has profit $y_i r$ and the total knapsack has capacity W; at most $\nu d_i/r$ pieces of type i can be used, and we wish to fill our knapsack as profitably as possible. Although this is NP-hard, recall that an $\epsilon/2$-approximation algorithm would suffice for our purposes, and so we can use an algorithm due to Lawler [10] that runs in $O(M\epsilon^{-2})$ time. Applying Theorem 3.5, we obtain the following result.

Theorem 4.4 For any fixed $\epsilon > 0$, there is a deterministic ϵ-approximation algorithm for the fractional cutting-stock problem that runs in $O(M^2 \log^3 D \log M)$ time, and a randomized analog that runs in $O(M^2 \log^2 D \log M)$ time, where $D = \sum_i d_i$.

It is not hard to see that the running times can be improved by replacing the $\log D$ by $\log r^*$, where r^* denotes optimal value of the cutting-stock linear program.

The integer version of the cutting-stock problem is equivalent to the bin-packing problem, which is usually stated in terms of pieces of specified sizes that are to be packed into the minimum number of bins, where the bins are all of some given capacity. Karmarkar & Karp [8] gave a fully polynomial approximation scheme for the bin-packing problem which uses an algorithm (based on the ellipsoid method) for the fractional cutting-stock problem. Given a bin-packing instance, their algorithm first constructs a refined instance that has a small number of distinct piece sizes, and only relatively large pieces. It approximately solves the linear program for this instance to obtain a vertex x, which is converted to the integer solution $\lceil x \rceil$. The additional bins introduced by this rounding is at most the number of non-zeros in x; since x is a vertex, this is at most the number of piece sizes. We can obtain a much faster version of the algorithm. Since finding a vertex solution appears to require too much time, we can instead rely directly on the fact that our algorithm produces solutions with few non-zeros. As a consequence, the error bound is somewhat inferior; the additive term in the performance guarantee of the original scheme is $O(\epsilon^{-2})$, and this, in fact, can be improved to $O(\epsilon^{-1}\log(\epsilon^{-1}))$.

Theorem 4.5 There is a fully polynomial approximation scheme for the bin-packing problem that, for an instance with N pieces and optimum value r^*, delivers a solution

that uses $(1+\epsilon)r^* + O(\epsilon^{-3}\log^4\epsilon^{-1})$ bins in $O(N\log N + \epsilon^{-6}\log^3\epsilon^{-1}\log^3 N)$ time.

Minimum-cost multicommodity flow. In the minimum-cost multicommodity flow problem, we generalize the usual minimum-cost flow problems as follows: given an N-node, M-edge graph G with non-negative costs and capacities assigned to the edges, along with demands d_j and source-sink pairs (s_j, t_j), $j = 1, \ldots, K$, that specify the commodities to be shipped, we wish to find flows that satisfy the demands and have total flow on each edge at most its capacity, so as to minimize the total cost.

To apply our relaxed decision procedure, we define P by the flow conservation and demand constraints and let $Ax \leq b$ be given by the capacity constraints and a budget constraint on the allowed cost for the current iteration of a bisection search. We apply the decomposition method of Theorem 3.3, by setting each $\gamma_i = 1$, except for the budget constraint, for which $\gamma_i = N$. The subroutine (3) runs in $O(M + N\log N)$ time using a shortest-path routine. In the following result, note that an ϵ-optimal flow may exceed the optimum cost and the capacity constraints by a $(1+\epsilon)$ factor.

Theorem 4.6 For any fixed $\epsilon > 0$, there exists a deterministic algorithm for the min-cost multicommodity flow problem that finds an ϵ-optimal flow and runs in $O(K^2 M\log M\ (M + N\log N)\log K)$ time, and a randomized analog that runs in $O(KM\log M\ (M + N\log N)\log K)$ time.

Acknowledgments

We are grateful to Andrew Goldberg and Cliff Stein for helpful discussions.

References

[1] M. E. Dyer. An $O(n)$ algorithm for the multiple-choice knapsack linear program. *Mathematical Programming*, 29:57-63, 1984.

[2] K. Eisemann. The trim problem. *Management Science*, 3:279–284, 1957.

[3] P. C. Gilmore and R. E. Gomory. A linear programming approach to the cutting-stock problem. *Operations Research*, 9:839–859, 1961.

[4] P. C. Gilmore and R. E. Gomory. A linear programming approach to the cutting-stock problem – Part II. *Operations Research*, 11:863–888, 1963.

[5] A. V. Goldberg. A natural randomization strategy for multicommodity flow and related algorithms. Unpublished manuscript, 1991.

[6] M. D. Grigoriadis and L. .G. Khachiyan. Fast approximation schemes for convex programs with many blocks and coupling constraints. Technical Report DCS-TR-273, Rutgers University, New Brunswick, NJ, 1991.

[7] M. Held and R. M. Karp. The traveling-salesman problem and minimum cost spanning trees. *Operations Research*, 18:1138–1162, 1970.

[8] N. Karmarkar and R. M. Karp. An efficient approximation scheme for the one-dimensional bin-packing problem. In *Proceedings of the 23rd Annual IEEE Symposium on Foundations of Computer Science*, pages 206–213, 1982.

[9] P. Klein, S. A. Plotkin, C. Stein, and É. Tardos. Faster approximation algorithms for the unit capacity concurrent flow problem with applications to routing and finding sparse cuts. Technical Report 961, School of Operations Research and Industrial Engineering, Cornell University, 1991. A preliminary version of this paper appeared in *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, pages 310–321, 1990.

[10] E. L. Lawler. Fast approximation algorithms for knapsack problems. In *Proceedings of the 18th Annual IEEE Symposium on Foundations of Computer Science*, pages 206–213, 1977.

[11] E. L. Lawler and J. Labetoulle. On preemptive scheduling on unrelated parallel processors by linear programming. *J. Assoc. Comput. Mach.*, 25:612–619, 1978.

[12] T. Leighton, F. Makedon, S. Plotkin, C. Stein, É. Tardos, and S. Tragoudas. Fast approximation algorithms for multicommodity flow problems. In *Proceedings of the 23rd Annual ACM Symposium on the Theory of Computing*, pages 101–111, 1991.

[13] J. K. Lenstra, D. B. Shmoys, and É. Tardos. Approximation algorithms for scheduling unrelated parallel machines. *Mathematical Programming, A*, 24:259–272, 1990.

[14] P. Raghavan. Probabilistic construction of deterministic algorithms: approximating packing integer programs. *J. Comput. System Sciences*, 37:130–143, 1988.

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

[16] F. Shahrokhi and D. W. Matula. The maximum concurrent flow problem. *J. Assoc. Comput. Mach.*, 37:318–334, 1990.

[17] P. M. Vaidya. A new algorithm for minimizing convex functions over convex sets. In *Proceedings of the 30th Annual IEEE Symposium on Foundations of Computer Science*, pages 338–343, 1989.

[18] P. M. Vaidya. Speeding up linear programming using fast matrix multiplication. In *Proceedings of the 30th Annual IEEE Symposium on Foundations of Computer Science*, pages 332–337, 1989.

[19] S. L. van de Velde. Machine scheduling and Lagrangian relaxation. Doctoral thesis, Centre for Mathematics and Computer Science, Amsterdam, 1991.