Chapter 19
Randomized Rounding Without Solving the Linear Program

Neal E. Young*

Abstract

We introduce a new technique called oblivious rounding — a variant of randomized rounding that avoids the bottleneck of first solving the linear program. Avoiding this bottleneck yields more efficient algorithms and brings probabilistic methods to bear on a new class of problems. We give oblivious rounding algorithms that approximately solve general packing and covering problems, including a parallel algorithm to find sparse strategies for matrix games.

1 Introduction

Randomized Rounding: Randomized rounding [18] is a probabilistic method [20, 1] for the design of approximation algorithms. Typically, one formulates an NP-hard problem as an integer linear program, disregards the integrality constraints, solves the resulting linear program, and randomly rounds each coordinate of the solution up or down with probability depending on the fractional part. One shows that, with non-zero probability, the rounded solution approximates the optimal solution. This yields a randomized algorithm; in most cases it can be derandomized by the method of conditional probabilities [17]. The probabilistic analyses are often simple, relying on just a few basic techniques. Yet for many NP-hard problems, randomized rounding yields the best approximation known by any polynomial time algorithm [3].

Oblivious Rounding: Derandomized or not, a main drawback of randomized rounding algorithms has been that they first solve a linear program to find a solution to round. We show that this bottleneck can sometimes be avoided as follows: (1) show that randomly rounding an optimal solution (possibly to smaller-than-integer units) yields an approximate solution; (2) apply the method of conditional probabilities, finding pessimistic estimators [17] that are essentially *independent* of the optimal solution. The method of conditional probabilities is used not to derandomize per se, but to achieve the independence.

Generalized Packing and Covering: The resulting algorithms find the approximate solution without first computing the optimal solution. This allows randomized rounding to give simpler and more efficient algorithms and makes it applicable for integer *and* non-integer linear programming. To demonstrate this, we give approximation algorithms for general packing and covering problems corresponding to integer and non-integer linear programs of small *width*, including a parallel algorithm for finding sparse, near-optimal strategies for zero-sum games.

Packing and covering problems have been extensively studied (see §2). For example, Plotkin, Shmoys, and Tardos [16] approached these problems using Lagrangian-relaxation techniques directly. Their algorithms and ours share the following features: (1) they depend similarly on the width, (2) they are Lagrangian-relaxation algorithms, (3) they allow the packing or covering set to be given by a (possibly approximate) subroutine for optimizing over it, (4) they produce dual solutions that prove near-optimality, and (5) they can provide integer solutions comparable to those obtainable by randomized rounding. Our approach shows a strong connection between probabilistic techniques and Lagrangian relaxation. Our algorithms are also relatively simple, although they are not as effective for some problems of large width.

Flavor of Oblivious Rounding Algorithms: For the (integer) set cover problem, oblivious rounding yields the greedy set cover algorithm [10, 14]. For the *fractional* set cover problem, it yields an algorithm that repeatedly chooses a set whose elements have the largest net weight, where the weight of an element is initially 1 and is multiplied by $1-\epsilon$ each time a set containing it is chosen. To obtain the final cover, each set is assigned a weight proportional to the number of times it was chosen (this is similar in spirit to [4] and related works). For multicommodity flow, it yields algorithms that repeatedly augment flow along a shortest path, where the length of an edge is initially 1 and is multiplied by $1+\epsilon c/c(e)$ each time the edge is used ($c(e)$ is the capacity of the edge and c is the minimum edge capacity).

*AT&T Bell Labs, rm. 2D-145, 600 Mountain Ave., Murray Hill, NJ 07974. Part of this research was done while at School of ORIE, Cornell University, Ithaca NY 14853 and supported by Éva Tardos' NSF PYI grant DDM-9157199. E-mail: ney@research.att.com.

Proceedings of the Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, 170-178 (1995)

Problem Definitions: Let P be a convex set in $\mathbb{R}^n$ and let f be a linear function (not nec. homogenous) from P to $\mathbb{R}^m$. The *width* of P with respect to f is $\omega = \max_{j,x} f_j(x) - L$, where $L = \min_{j,x} f_j(x)$.

The *generalized packing problem* is to compute $\lambda^* = \min_{x \in P} \max_j f_j(x)$. The *packing problem* occurs when f is non-negative on P. The *covering problem* is to compute $\lambda^* = \max_{x \in P} \min_j f_j(x)$, assuming f is non-negative. (This is equivalent to the generalized packing problem with the restriction that f is non-positive.)

Our algorithms assume an *optimization oracle* for P and f — given non-negative $y \in R_m$, the oracle returns x and $f(x)$, where x minimizes $\sum_j y_j f_j(x)$. (This models, e.g., packing source-sink paths subject to edge constraints; in this case the oracle would compute a shortest path for given non-negative edge lengths.) For covering, the oracle must *maximize* the sum.

Quality of Solutions: Given the oracle, n, m, ω, L, and $\epsilon > 0$, our algorithms return ϵ-approximate solutions. For generalized packing, ϵ is the additive error with respect to λ^*. For packing and covering, the error is a factor of $1 \pm \epsilon$.

Complexity: Table 1 shows the number of iterations required and the complexity per iteration. In that caption, "explicitly given" means that $f(x) = Ax + b$, where A and b are, respectively, an explicitly given matrix and vector, while $P = \{x \in \mathbb{R}^n : x \geq 0; \sum x_i = 1\}$.

Granularity: The oracle is called once in each iteration of the algorithm; the algorithm returns the average of the solutions returned by the oracle. Thus, the granularity of the final solution is the granularity of the solutions returned by the oracle, divided by the number of iterations. For the abstract problems we consider, this can provide integer solutions comparable to those obtainable by other techniques.

Dual Solutions: Our algorithms maintain a dual solution, represented by a vector y, initially uniform. In each iteration, each y_j is multiplied by a factor depending on $f_j(x)$ where x is the solution returned by the oracle (e.g., for packing, y_j is multiplied by $1 + \epsilon f_j(x)/\omega$). The average over all iterations of the values of these dual solutions is ϵ-optimal with respect to the value of the final (primal) solution.

Sparse Strategies for Zero-Sum Games: The explicitly given general packing problem generalizes the problem of finding near-optimal strategies for zero-sum matrix games: P is the set of mixed strategies for one player, $f_j(x)$ is the expected payoff if the player plays according to x and the opponent plays the pure strategy j, and λ^* is the value of the game. approximate solution is a mixed strategy x guaranteeing an expected payoff within an additive ϵ of optimal.

generalized packing:	$\left\lceil \frac{\omega^2 \ln(m)}{2\epsilon^2} \right\rceil$
packing:	$\left\lceil \frac{(1+\epsilon)\omega \ln(m)}{\lambda^* b(\epsilon)} \right\rceil$
covering:	$\left\lceil \frac{\omega \ln(m)}{\lambda^* b(-\epsilon)} \right\rceil$

$b(\epsilon) := (1+\epsilon)\ln(1+\epsilon) - \epsilon;$
$b(-\epsilon) > \frac{\epsilon^2}{2} > b(\epsilon) > \frac{2\epsilon^2}{4.2+\epsilon}.$

Table 1: Number of iterations. Each iteration requires $O(\log m)$ time and $O(m)$ operations (on an EREW-PRAM), plus one oracle call. For an explicitly given problem (no oracle), each iteration requires $O(\log nm)$ time and $O(nm)$ operations.

Each iteration chooses the best pure strategy given that the opponent plays the mixed strategy represented by y. The final solution returned is a mixed strategy that plays uniformly from $\lceil \frac{\omega^2 \ln m}{2\epsilon^2} \rceil$ pure strategies, one for each iteration. (The opponent has m pure strategies; ω is the minimum minus the maximum payoff.) The existence of such sparse, near-optimal strategies was shown probabilistically [2, 13]; our existence proof of the approximate solution for generalized packing is a generalization of the proof in [13].

2 Related Work

Plotkin, Shmoys, and Tardos [16] (generalizing a series of works on multicommodity flow [19, 11, 12]) gave approximation algorithms for general packing and covering problems similar to those we consider. For these abstract problems, their results are comparable to those in this paper, but for many problems their results are stronger. Most importantly, they give techniques for reducing the effective width of a linear program and techniques for problems (such as concurrent multicommodity flow) when the packing or covering set is a Cartesian product.

Luby and Nisan [15] give a parallel approximation algorithm for positive linear programming — the special cases of linear programming of the form $\max_x\{c \cdot x : Ax \geq b; x \geq 0\}$ (a packing problem), or the dual $\min_y\{b \cdot y : A^T y \leq c; y \geq 0\}$ (a covering problem), where A, b, and c have non-negative coefficients. Here A, b, and c are explicitly given.

Previous algorithms applicable to zero-sum games either required the solution of a linear program [8] or did not provide sparse strategies [5, 6, 15].

3 Introductory Example: Set Cover

To introduce oblivious rounding, we give a simple example. The set cover problem is the following: given a family of sets $\mathcal{F} = \{S_1, \ldots, S_m\}$, with each $S_i \subseteq \{1, 2, \ldots, n\}$, a *set cover* C is a sub-family such that every element $j = 1, \ldots, n$ is in some set in C. The problem is to find a cover C that is not much larger than C^*, a minimum-cardinality cover. We derive an algorithm that, without knowing C^*, emulates a random experiment that draws sets randomly from C^*. The algorithm finds a cover of size at most $\lceil |C^*| \ln n \rceil$.

3.1 Existence: Let $s = \lceil |C^*| \ln n \rceil$. Consider drawing s sets uniformly at random from C^*. What is the expected number of elements left uncovered? For any given element $x \in X$, the probability that it is not covered in a given round is at most $1 - 1/|C^*|$, because it is in at least one set in C^*. Thus the expected number of elements left uncovered is at most $n(1 - 1/|C^*|)^s < n \exp(-s/|C^*|) \leq 1$. Thus, with non-zero probability we obtain a cover C of size s.

3.2 Construction: The method of conditional probabilities naively applied to the above proof yields an algorithm that depends on C^*. We outline this next. Our ultimate goal is not derandomization per se, but an algorithm that does not require knowledge of C^*.

Consider an algorithm that chooses the sets sequentially, making each choice deterministically to do "as well" as the corresponding random choice would. Specifically, the algorithm chooses each set to minimize the expected number of elements that would remain uncovered *if* the remaining sets were chosen randomly from C^*. Letting C denote the collection of sets chosen so far, this expected number is

$$\Phi(C) = \sum_{j \notin \cup C} \left(\frac{\sum_{S_i \not\ni j} x_i^*}{|C^*|} \right)^{s - |C|} \tag{3.1}$$

(We use x_i^* to denote 1 if $S_i \in C^*$ and 0 otherwise; we use $\cup C$ to denote the union of sets in C.) Φ is called a *pessimistic estimator* [17], because (a) it is an upper bound on the conditional probability of failure (in this case, by Markov's inequality), (b) it is initially less than 1, and (c) each choice can be made without increasing it. (The latter property follows in this case because Φ is an expected value conditioned on the choices made so far.) These three properties imply the invariant that if the remaining $s - |C|$ sets were to be chosen randomly from C^*, the probability of failure would be less than one. Consequently, when $|C| = s$, C is a cover.

Achieving Obliviousness: Because an uncovered element that occurs in several sets in C^* contributes less to Φ, the above algorithm depends on the number of times each element is covered by C^*. This is counter-intuitive, in that the only aspect of C^* used in the proof was $\sum_{S_i \not\ni j} x_i^*/|C^*| \leq 1 - 1/|C^*|$. Replacing each corresponding term in Φ yields

$$\tilde{\Phi}(C) = \sum_{j \notin \cup C} \left(1 - \frac{1}{|C^*|}\right)^{s - |C|}. \tag{3.2}$$

$\tilde{\Phi}$ is a pessimistic estimator. More importantly, among collections of sets of the same size, $\tilde{\Phi}$ is uniformly proportional to the number of uncovered elements in the set. Thus, the algorithm that uses $\tilde{\Phi}$ instead of Φ does not depend on C^*, it simply chooses each set to minimize the number of elements remaining uncovered. Nonetheless, it is guaranteed to keep up with the random experiment, finding a cover within $\lceil |C^*| \ln n \rceil$ steps. This is the greedy set cover algorithm, originally analyzed non-probabilistically by Johnson [10] and Lovász [14].

Versus fractional cover: If the cover C^* is a fractional cover, the analyses of both algorithms carry over directly to show a $\ln n$ performance guarantee.

What enables oblivious rounding? We call such algorithms *oblivious rounding* algorithms. What kinds of randomized rounding schemes admit them? The key property is that the proof bounds the probability of failure by the expected value of a sum of products and bounds the terms corresponding across products uniformly. To illustrate, here is the explicit proof that $\min_i \tilde{\Phi}(C \cup \{S_i\}) \leq \tilde{\Phi}(C)$:

$$\begin{aligned}
\tilde{\Phi}(C) &= \sum_{j \notin \cup C} \left(1 - \frac{1}{|C^*|}\right)^{s-|C|} \\
&\geq \sum_{j \notin \cup C} \left(\sum_{S_i \not\ni j} \frac{x_i^*}{|C^*|} \right) \cdot \left(1 - \frac{1}{|C^*|}\right)^{s-|C|-1} \\
&= \sum_i \frac{x_i}{|C^*|} \sum_{j \notin (\cup C) \cup S_i} \left(1 - \frac{1}{|C^*|}\right)^{s-|C|-1} \\
&= \sum_i \frac{x_i}{|C^*|} \tilde{\Phi}(C \cup \{S_i\}) \\
&\geq \min_i \tilde{\Phi}(C \cup \{S_i\}).
\end{aligned}$$

The first inequality is obtained by "undoing" one step of the substitution that yielded $\tilde{\Phi}$ from Φ. The standard argument then applies. We use this principle for each of our analyses.,

4 Algorithm for Generalized Packing

Fix an instance $(P, f, L, \omega, \epsilon)$ of the generalized packing problem. We consider randomly rounding an optimal solution to obtain an ϵ-approximate solution; we then derive the algorithm that finds such a solution.

4.1 Existence: Let λ^* and x^* be an optimal solution. Let S be a multiset obtained by repeatedly choosing random elements of P, where each random element is chosen from a distribution over P with n-dimensional mean x^*. Let $\bar{x}$ be the average of the points in S.

LEMMA 4.1. *The probability that $\bar{x}$ is not an ϵ-approximate solution is less than $m/\exp\left[\frac{2|S|\epsilon^2}{\omega^2}\right]$.*

Proof. Without loss of generality, assume $L = 0$ and $\omega = 1$. Otherwise take $f(x) \leftarrow \frac{f(x)-L}{\omega}$ and $\epsilon \leftarrow \epsilon/\omega$.

The convexity of P ensures that $\bar{x} \in P$. For each j, $f_j(\bar{x}) = \sum_{x\in S} f_j(x)/|S|$, which is the average of $|S|$ independent random variables in $[0,1]$. Since $\mathrm{E}[f_j(x)] = f_j(x^*) \leq \lambda^*$, by Hoeffding's bound [7], $\Pr[f_j(\bar{x}) \geq \lambda^* + \epsilon]$ is less than $1/\exp(2|S|\epsilon^2)$. Since j ranges from 1 to m, the result follows. □

4.2 Construction: As in the set cover example, our algorithm mimics the random experiment. Each round it adds an element to S to minimize a pessimistic estimator. This pessimistic estimator is implicit in the existence proof. To find it, we need the inequalities that prove (a simplified version of) Hoeffding's bound:

LEMMA 4.2. ([7]) *Let $X = \sum X_i/s$ be the average of s independent random variables in $[0,1]$, with $\mathrm{E}(X_i) \leq \mu_i$ and $\sum \mu_i = \mu$. Then $\Pr[X \geq \mu + s\epsilon] < 1/\exp(2s\epsilon^2)$.*

Proof. Let $\alpha = e^{4\epsilon} - 1$.

$$\begin{aligned}
&\Pr\left[\sum X_i \geq \mu + s\epsilon\right] \\
&= \Pr\left[\prod_i \frac{(1+\alpha)^{X_i}}{(1+\alpha)^{\mu_i+\epsilon}} \geq 1\right] \\
&\leq \mathrm{E}\left[\prod_i \frac{1+\alpha X_i}{(1+\alpha)^{\mu_i+\epsilon}}\right] \qquad (4.3) \\
&= \prod_i \frac{1+\alpha E(X_i)}{(1+\alpha)^{\mu_i+\epsilon}} \\
&\leq \prod_i \frac{1+\alpha\mu_i}{(1+\alpha)^{\mu_i+\epsilon}} \\
&< \prod_i 1/e^{2\epsilon^2}.
\end{aligned}$$

The second step follows from $(1+\alpha)^z \leq 1 + \alpha z$ for $0 \leq z \leq 1$ and Markov's inequality. The last step uses $1 + \alpha z < (1+\alpha)^{z+\epsilon}/e^{2\epsilon^2}$ for $\epsilon > 0$, $\alpha = e^{4\epsilon} - 1$, and $z \geq 0$. □

The proof of Lemma (4.1) bounds the probability of failure by a sum of probabilities, each of which is bounded by an expected value (4.3) in Hoeffding's proof. Thus (when $L = 0$ and $\omega = 1$), the proof bounds the probability of failure by the expected value of

$$\sum_j \prod_{x\in S} \frac{1+\alpha f_j(x)}{(1+\alpha)^{\lambda^*+\epsilon}},$$

the expectation of which is less than $m/\exp(2|S|\epsilon^2)$. The conditional expectation of the sum given $T \subseteq S$ is

$$\sum_j \left[\prod_{x\in T} \frac{1+\alpha f_j(x)}{(1+\alpha)^{\lambda^*+\epsilon}}\right] \cdot \left[\frac{1+\alpha f_j(x^*)]}{(1+\alpha)^{\lambda^*+\epsilon}}\right]^{s-|T|}$$

where s is the desired size of S. To obtain the pessimistic estimator for the algorithm, replace each $f_j(x^*)$ by the upper bound λ^*:

$$\sum_j \left[\prod_{x\in T} \frac{1+\alpha f_j(x)}{(1+\alpha)^{\lambda^*+\epsilon}}\right] \cdot \left[\frac{1+\alpha\lambda^*}{(1+\alpha)^{\lambda^*+\epsilon}}\right]^{s-|T|}$$

When s is large enough that $m/\exp(2|S|\epsilon^2) \leq 1$, this quantity is a pessimistic estimator: (a) it is an upper bound on the conditional probability of failure, (b) it is initially less than 1, and (c) some x can always be added to S without increasing it. Properties (a) and (b) follow from the choice of s and the inequalities in the proof of Hoeffding's lemma. Property (c) follows from the derivation, as explained for the set cover example. Among multisets of a given size, this pessimistic estimator is uniformly proportional to

$$\sum_j \prod_{x\in T} 1 + \alpha f_j(x).$$

Thus, to augment a given multiset T, the algorithm adds the element x minimizing $\sum_j y_j f_j(x)$, where $y_j = \prod_{x\in T} 1+\alpha f_j(x)$. This, accounting for the normalization $L = 0$ and $\omega = 1$, is the algorithm in Figure 1.

5 Packing and Covering Algorithms

We derive the packing algorithm analogously. Fix an instance (P, f, ω, ϵ) of the packing problem. Let λ^*, x^*, S and $\bar{x}$ be as for Lemma 4.1. Note that, for this problem, an ϵ-approximate solution is an $x \in P$ with $f(x) \leq (1+\epsilon)\lambda^*$.

5.1 Existence:

LEMMA 5.1. *The probability that $\bar{x}$ is not an ϵ-approximate solution is less than $m/\exp\left[\frac{|S|b(\epsilon)\lambda^*}{\omega}\right]$.*

FIND-GENERALIZED-PACKING$(P, f, L, \omega, \epsilon)$
1. $\epsilon \leftarrow \frac{\epsilon}{\omega}$; $\alpha \leftarrow e^{4\epsilon} - 1$; $S \leftarrow \{\}$; $s \leftarrow \frac{\ln m}{2\epsilon^2}$
2. $y_j \leftarrow 1 \quad (j = 1, \ldots, m)$
3. **repeat**
4. choose $x \in P$ to minimize $\sum_j y_j f_j(x)$
5. $S \leftarrow S \cup \{x\}$
6. $y_j \leftarrow y_j \cdot \left[1 + \alpha \frac{f_j(x) - L}{\omega}\right] \quad (j = 1, \ldots, m)$
7. **until** $|S| \geq s$
8. **return** $\frac{1}{|S|} \sum_{x \in S} x$

Figure 1: Algorithm for generalized packing

Proof. Without loss of generality, assume $\omega = 1$. Otherwise take $f(x) \leftarrow f(x)/\omega$ and $\lambda^* \leftarrow \lambda^*/\omega$.

The convexity of P ensures that $\bar{x} \in P$. For each j, $f_j(\bar{x}) = \sum_{x \in S} f_j(x)/|S|$, which is the average of $|S|$ independent random variables in $[0, 1]$, each with expectation $f_j(x^*) \leq \lambda^*$. By Raghavan's bound [17], $\Pr[f_j(\bar{x}) \geq (1+\epsilon)\lambda^*]$ is less than $1/\exp[|S|b(\epsilon)\lambda^*]$. Since j ranges from 1 to m, the result follows. □

5.2 Construction: Here is Raghavan's proof:

LEMMA 5.2. ([17]) *Let $X = \sum X_i/s$ be the average of independent random variables in $[0, 1]$ with $\mathrm{E}(X_i) \leq \mu_i$ and $\sum \mu_i = \mu > 0$. Then $\Pr[X \geq (1 + \epsilon)\mu] < 1/\exp[b(\epsilon)\mu]$.*

Proof.

$$\begin{aligned} & \Pr[X \geq (1+\epsilon)\mu] \\ &= \Pr\left[\prod_i \frac{(1+\epsilon)^{X_i}}{(1+\epsilon)^{(1+\epsilon)\mu_i}} \geq 1\right] \\ &\leq \mathrm{E}\left[\prod_i \frac{1+\epsilon X_i}{(1+\epsilon)^{(1+\epsilon)\mu_i}}\right] \qquad (5.4) \\ &= \prod_i \frac{1+\epsilon\,\mathrm{E}(X_i)}{(1+\epsilon)^{(1+\epsilon)\mu_i}} \\ &< \prod_i \frac{e^{\epsilon\mu_i}}{(1+\epsilon)^{(1+\epsilon)\mu_i}} \end{aligned}$$

The last line equals $1/\exp[b(\epsilon)\mu]$. The second step uses $(1+\alpha)^z \leq 1 + \alpha z$ for $0 \leq z \leq 1$ and Markov's inequality. The last uses $\mathrm{E}(X_i) \leq \mu_i$ and $1 + z \leq e^z$, which is strict if $z \neq 0$. □

Thus (assuming $\omega = 1$), the proof of Lemma 5.1 bounds the probability of failure by the expectation of

$$\sum_j \prod_{x \in S} \frac{1 + \epsilon f_j(x)}{(1+\epsilon)^{(1+\epsilon)\lambda^*}}$$

FIND-PACKING-GIVEN-s $(P, f, \epsilon, \omega, s)$
1. $S \leftarrow \{\}$
2. $y_j \leftarrow 1 \quad (j = 1, \ldots, m)$
3. **repeat**
4. choose $x \in P$ to minimize $\sum_j y_j f_j(x)$
5. $S \leftarrow S \cup \{x\}$
6. $y_j \leftarrow y_j \cdot \left[1 + \epsilon \frac{f_j(x)}{\omega}\right] \quad (j = 1, \ldots, m)$
7. **until** $|S| \geq s$ **do**
8. **return** $\frac{1}{|S|} \sum_{x \in S} x$

Figure 2: Algorithm for packing, given s. To obtain covering algorithm, negate ϵ and change "minimize" to "maximize".

corresponding to (5.4). The expectation given $T \subseteq S$ is

$$\sum_j \left[\prod_{x \in T} \frac{1 + \epsilon f_j(x)}{(1+\epsilon)^{(1+\epsilon)\lambda^*}}\right] \cdot \left[\frac{1 + \epsilon f_j(x^*)}{(1+\epsilon)^{(1+\epsilon)\lambda^*}}\right]^{s - |T|},$$

where s is the desired size of S. When s is large enough that $m/\exp[|S|b(\epsilon)\lambda^*] \leq 1$, replacing $f_j(x^*)$ by λ^* gives a pessimistic estimator. Among multisets T of the same size, the pessimistic estimator is proportional to

$$\sum_j \prod_{x \in T} 1 + \epsilon f_j(x).$$

Thus, to augment a given multiset T, the algorithm adds the element x minimizing $\sum_j y_j f_j(x)$, where $y_j = \prod_{x \in T} 1 + \epsilon f_j(x)$. This, accounting for the normalization to the case $\omega = 1$, gives the algorithm in Figure 2. This algorithm assumes s is given. We remove this requirement in Section 6.

5.3 Covering Algorithm. The covering algorithm is described in Figure 2. Its derivation is analogous to that of the packing algorithm. Fix an instance (P, f, ω, ϵ) of the approximate covering problem. Let λ^*, x^*, S and $\bar{x}$ be as for Lemma 4.1. Note that for this problem, $\lambda^* = \min_j f_j(x^*)$ and an ϵ-approximate solution $x \in P$ satisfies $f(x) \geq (1 - \epsilon)\lambda^*$.

LEMMA 5.3. *The probability that $\bar{x}$ is not an ϵ-approximate solution is less than $m/\exp[|S|b(\epsilon)\lambda^*/\omega]$.*

We omit the proof, which is essentially the same as for packing, except it is based on the following variant of Raghavan's bound:

LEMMA 5.4. ([17]) *Let $X = \sum X_i/s$ be the average of independent random variables in $[0, 1]$ with $\mathrm{E}(X_i) \geq \mu_i$ and $\sum \mu_i = \mu > 0$. Then $\Pr[X \leq (1 - \epsilon)\mu] < 1/\exp[b(-\epsilon)\mu)$.*

We omit the derivation of the algorithm, noting only that the proof of Lemma 5.3 implicitly bounds the

probability of failure by the expectation of

$$\sum_j \prod_{x \in S} \frac{1 - \epsilon f_j(x)}{(1-\epsilon)^{(1-\epsilon)\lambda^*}}.$$

6 Dual Solutions

Our algorithms implicitly find good approximate solutions to the underlying dual linear programs. The argument that the algorithm "keeps up" with the random rounding of an unknown optimal solution implicitly uses a dual solution to bound the optimal at each iteration. The value of the solution generated by the algorithm thus converges not only to the value of the optimum, but also to the average of the values of these dual solutions. The basic principle in each case is similar to that for set cover, which we give first for illustration.

6.1 Set Cover Dual: The dual problem is to assign non-negative weights to the elements so that the net weight assigned to the elements in any set is at most one. The value of the dual solution is the net weight assigned.

At the start of a given iteration, suppose r elements remain uncovered, and let d denote the largest number in any set in $\mathcal{F}$. Then assigning each uncovered element a weight of $1/d$ yields a dual solution of value $v = r/d$.

During the course of the algorithm, let $\bar{v}$ denote the harmonic mean of the dual solutions corresponding to the iterations so far.

LEMMA 6.1. *The set cover algorithm maintains the invariant that the number of elements not covered by the current partial cover C is less than $n/\exp(|C|/\bar{v})$.*

The proof is essentially the same as the proof that $\tilde{\Phi}$ is a pessimistic estimator, except the values of the dual solutions take the place of $|C^*|$.

Proof. In an iteration where the dual solution has value r/d, the number of uncovered elements decreases from r to $r - d = r(1 - 1/v) < re^{-1/v}$. By induction on the iterations, the algorithm maintains the invariant that the number of uncovered elements is less than $n/\exp\left(\sum_\ell 1/v_\ell\right)$ where v_ℓ is the value of the dual solution corresponding to the ℓth iteration and ℓ ranges over the iterations so far. Note that $\bar{v} = |C|/\sum \frac{1}{v_\ell}$. □

Before the last iteration at least one element is left, so at that point $n/\exp((k-1)/\bar{v}) \geq 1$. Thus,

COROLLARY 6.1. *The harmonic mean of the values of the dual solutions over the first $k-1$ iterations is larger than $\frac{k-1}{\ln n}$, where k is the size of the final cover.*

The maximum value is at least the arithmetic mean, which is at least the harmonic mean, so at least one of these simple dual solutions has value above $\frac{k-1}{\ln n}$.

6.2 Generalized Packing Dual: The vector y maintained by the generalized packing algorithm represents a dual solution. At the start of a given iteration, the value of the dual solution associated with y is

$$\frac{\min_{x \in P} \sum_j y_j f_j(x)}{\sum_j y_j}. \tag{6.5}$$

(Since $y \geq 0$, a simple argument shows this is a lower bound on $\lambda^* = \min_{x \in P} \max_j f_j(x)$.)

Notation: During the course of the algorithm, let $\bar{x}$ denote the current solution $\sum_{x \in S} x/|S|$ represented by S. Let $\bar{\lambda}$ denote $\max_j f_j(\bar{x})$. Let $\bar{v}$ denote the average of the values of the dual solutions for the previous iterations. Let $v(x, y)$ denote $\sum_j y_j f_j(x) / \sum_j y_j$.

LEMMA 6.2. *The generalized packing algorithm maintains the invariant*

$$(1+\alpha)^{|S|(\bar{\lambda}-L)/\omega} \leq (1+\alpha)^{|S|((\bar{v}-L)/\omega+\epsilon)} m/\exp(2|S|\epsilon^2).$$

Proof. WLOG, assume $L = 0$ and $\omega = 1$. We show that $\sum_j y_j$ is at least the left-hand side and at most the right-hand side. The first part follows from the same sequence of inequalities that was used in §4.2 to derive the (numerator of the) pessimistic estimator:

$$\begin{aligned}(1+\alpha)^{|S|\bar{\lambda}} &\leq \sum_j (1+\alpha)^{|S| f_j(\bar{x})} \\ &= \sum_j \prod_{x \in S} (1+\alpha)^{f_j(x)} \\ &\leq \sum_j \prod_{x \in S} 1 + \alpha f_j(x).\end{aligned}$$

Since $y_j = \prod_{x \in S} 1 + \alpha f_j(x)$, the first part follows.

For the second part, we first note the role of the dual solution in each iteration: given the current x and y, the iteration increases the quantity $\sum_j y_j$ by a factor of $1 + \alpha v(x, y)$. (This follows from inspection of the algorithm and the definition of $v(x, y)$.) Next we apply the sequence of inequalities that bounded the pessimistic estimator below 1 in §4.2: By the last inequality in Hoeffding's bound (Lemma 4.2), $1 + \alpha v(x, y) \leq (1+\alpha)^{v(x,y)+\epsilon}/\exp(2\epsilon^2)$. Let v_ℓ denote the value of $v(x, y)$ at the ℓth iteration (for $1 \leq \ell \leq |S|$). By induction on the iterations

$$\sum_j y_j \leq m(1+\alpha)^{\sum_\ell (v_\ell + \epsilon)}/\exp(2|S|\epsilon^2).$$

Since $|S|\bar{v} = \sum_\ell v_\ell$, this gives the result. □

COROLLARY 6.2. *After $\left\lceil \frac{\omega^2 \ln m}{2\epsilon^2} \right\rceil$ iterations of the generalized packing algorithm, $\bar{\lambda} \leq \bar{v} + \epsilon$. That is, the primal and average dual values differ by at most ϵ.*

FIND-PACKING(P, f, ϵ, ω)
1. $S \leftarrow \{\}$; $y_j \leftarrow 1 \quad (j = 1, \ldots, m)$
2. **repeat**
3. choose $x \in P$ to minimize $v = \sum_j y_j f_j(x)$
4. $S \leftarrow S \cup \{x\}$
5. $y_j \leftarrow y_j \cdot \left[1 + \epsilon \frac{f_j(x)}{\omega}\right] \quad (j = 1, \ldots, m)$
6. $V \leftarrow \max(V, v/\sum_j y_j)$
7. $F_j \leftarrow [(|S| - 1)F_j + f_j(x)]/|S| \quad (j = 1, \ldots, m)$
8. $\bar{\lambda} \leftarrow \max_j F_j$
9. **until** $\bar{\lambda} \leq (1 + \epsilon)V$
10. **return** $\sum_{x \in S} x/|S|$

Figure 3: Algorithm for packing. To obtain covering algorithm, negate ϵ's and change each "max" to "min", "minimize" to "maximize", and "$\leq$" to "$\geq$".

6.3 Packing Dual: The packing and covering algorithms also generate implicit dual solutions whose average values converge to the primal value. Let $\bar{\lambda}$ and $\bar{v}$ be defined as for the generalized packing dual.

LEMMA 6.3. *The packing algorithm maintains the invariant that*

$$(1 + \epsilon)^{|S|\bar{\lambda}/\omega} \leq me^{\epsilon|S|\bar{v}/\omega}.$$

We omit this and subsequent proofs in this section, since they are similar to that of Lemma 6.2.

COROLLARY 6.3. *After* $\left\lceil \frac{(1+\epsilon)\omega \ln m}{\lambda^* b(\epsilon)} \right\rceil$ *iterations of the packing algorithm,* $\bar{\lambda} \leq (1 + \epsilon)\bar{v}$. *That is, the primal and average dual values differ by at mosi a factor of* $1 + \epsilon$.

Our final packing algorithm detects convergence by comparing the primal value to the best dual value so far. The algorithm is shown in Figure 3. The algorithm maintains $f(\bar{x})$ (in the variable F) instead of $\bar{x}$.

6.4 Covering Dual:

LEMMA 6.4. *The covering algorithm maintains the invariant that*

$$(1 - \epsilon)^{|S|\bar{\lambda}/\omega} \leq me^{-\epsilon|S|\bar{v}/\omega}.$$

COROLLARY 6.4. *After* $\left\lceil \frac{\omega \ln m}{\lambda^* b(-\epsilon)} \right\rceil$ *iterations of the covering algorithm,* $\bar{\lambda} \geq (1 - \epsilon)\bar{v}$, *that is, the primal and average dual values differ by at most a factor of* $1 - \epsilon$.

The algorithm is described in Figure 3.

7 Using an Approximate Oracle

If the subroutine for computing $\min_{x \in P} \sum_j y_j f_j(x)$ returns only an *approximate* minimizer x, our algorithms still work well. The degree of approximation (absolute and/or relative) of the subroutine carries over into the performance guarantee of the algorithm. For covering, it can also affect the convergence rate (and therefore the granularity).

We model the error by assuming that, given y, the oracle returns an x such that

$$\sum_j v(x, y) \leq (1 + \delta_1) \min_{x \in P} v(x, y) + \delta_2 \tag{7.6}$$

where $v(x, y) = \sum_j y_j f_j(x)/\sum_j y_j$, $\delta_1] \geq 0$ denotes the relative error and $\delta_2 \geq 0$ denotes the absolute error. We call this a *(δ_1, δ_2)-approximate* oracle. (For covering, the notion of approximation is defined analogously.)

In each iteration, y still represents a dual solution. Since x is only an approximate minimizer, the value of the dual solution is no longer $v(x, y)$, but it is at least $\frac{v(x,y) - \delta_2}{1 + \delta_1}$. Still using $\bar{v}$ to denote the average of the values of the dual solutions for the previous iterations, define $\tilde{v}$ to be the average of the corresponding $v(x, y)$'s. Lemmas 6.2, 6.3, and 6.4 go through directly provided "$\tilde{v}$" is substituted for "$\bar{v}$". From the (modified) lemmas, by the same reasoning that gives the corollaries to those lemmas, together with the fact that $\tilde{v} \leq (1 + \delta_1)\bar{v} + \delta_2$, we get the following propositions.

PROPOSITION 7.1. *Suppose the generalized packing algorithm uses a (δ_1, δ_2)-approximate oracle. After* $\left\lceil \frac{\omega^2 \ln m}{2\epsilon^2} \right\rceil$ *iterations,* $\bar{\lambda} \leq \tilde{v} + \epsilon \leq (1 + \delta_1)\bar{v} + \delta_2 + \epsilon$.

PROPOSITION 7.2. *Suppose the packing algorithm uses a (δ_1, δ_2)-approximate oracle. After* $\left\lceil \frac{(1+\epsilon)\omega \ln m}{\lambda^* b(\epsilon)} \right\rceil$ *iterations,* $\bar{\lambda} \leq (1 + \epsilon)\tilde{v} \leq (1 + \epsilon)(1 + \delta_1)\bar{v} + (1 + \epsilon)\delta_2$.

For covering, $\tilde{v} \geq (1 - \delta_1)\bar{v} - \delta_2$.

PROPOSITION 7.3. *Suppose the covering algorithm uses a (δ_1, δ_2)-approximate oracle. After*

$$\left\lceil \frac{\omega \ln m}{[(1 - \delta_1)\lambda^* - \delta_2]b(-\epsilon)} \right\rceil$$

iterations, $\bar{\lambda} \geq (1 - \epsilon)\tilde{v} \geq (1 - \epsilon)(1 - \delta_1)\bar{v} - (1 - \epsilon)\delta_2$.

These results hold for the algorithms without modification. In particular, V in the packing algorithm in Figure 3 equals the best $v(x, y)$ seen so far, which is at least $\tilde{v}$, so is guaranteed to be within a $1 + \epsilon$ factor of $\bar{\lambda}$ within the required number of rounds.

8 Integer Packing and Covering

The packing and covering algorithms in Figure 3, as they stand, do not allow explicit control over the granularity of the final solution. Because the number of iterations can be less than the upper bound, the algorithms only guarantee a lower bound on the granularity. Of course, the lower bound is the difficult part, so it is not surprising that exact control over the granularity can be obtained. In this section, we discuss briefly how to modify those algorithms to find, e.g., an integer solution.

For simplicity, we consider a particular case of integer packing. Fix an instance of the packing problem (P, f, ω, ϵ). Let λ^* and x^* be an optimal solution. In addition, let $V \subset P$ be the extreme points on the boundary of P (if P is a polytope, V is its vertex set). We assume that the oracle returns only elements of V. The *integer packing* problem is to compute a maximum cardinality multiset $S \subseteq V$ such that $\sum_{x \in S} f_j(x) \leq 1$.

Note that for any such S, $|S| \leq \lfloor 1/\lambda^* \rfloor$, because $f(\bar{x}) \leq 1/|S|$, where $\bar{x} = \sum_{x \in S} x/|S|$. An *$\epsilon$-approximate integer solution* is a set S such that $|S| \geq \lfloor 1/\lambda^* \rfloor$ and $\sum_{x \in S} f_j(x) \leq 1 + \epsilon$.

Let S be a multiset obtained by repeatedly choosing random elements of V, where each random element is chosen from a distribution on V with mean x^*. (Such distributions exist because P is the convex closure of V.)

LEMMA 8.1. *When* $|S| \leq 1/\lambda^*$,

$$\Pr\left[(\exists j) \sum_{x \in S} f_j(x) \geq 1 + \epsilon\right] < m/\exp[b(\epsilon)/\omega].$$

The proof is essentially the same as that of Lemma 5.1, except $1/|S|$ replaces λ^*.

A corollary to the lemma is that, provided $m/\exp[b(\epsilon)/\omega] \leq 1$, there exists an ϵ-approximate integer solution. The corresponding algorithm is the same as the basic packing algorithm, except the termination condition is different. The algorithm terminates when adding another element would cause $\sum_{x \in S} f_j(x) > 1 + \epsilon$ for some j. Because the algorithm keeps up with the random process, the resulting set has size at least $\lfloor 1/\lambda^* \rfloor$.

Complexity and Performance Guarantee: The algorithm is given in Figure 4. Note that $\lfloor 1/\lambda^* \rfloor \leq |S| \leq \lfloor (1+\epsilon)/\lambda^* \rfloor$, so the number of iterations in this case is at most $(1 + \epsilon)/\lambda^*$. For the condition $m/\exp[b(\epsilon)/\omega] \leq 1$, it suffices that, for instance, $\epsilon \geq 2\max(\omega \ln m, \sqrt{\omega \ln m})$.

Covering: The same techniques apply for integer covering. For covering, define an *ϵ-approximate integer solution* to be a set S such that $|S| \leq \lceil 1/\lambda^* \rceil$ and $\sum_{x \in S} f_j(x) \geq 1 - \epsilon$. (Many variations are possible.)

FIND-INTEGER-PACKING (P, f, ϵ, ω)
assumption: $m/\exp[b(\epsilon)/\omega] \leq 1$.
1. $S \leftarrow \{\}$; $y_j \leftarrow 1$, $F_j \leftarrow 0$ $(j = 1, \ldots, m)$
2. **repeat**
3. choose $x \in P$ to minimize $\sum_j y_j f_j(x)$
4. $F_j \leftarrow F_j + f_j(x)$ $(j = 1, \ldots, m)$
5. **if** $\max_j F_j > 1 + \epsilon$ **return** S
6. $S \leftarrow S \cup \{x\}$
7. $y_j \leftarrow y_j \cdot \left[1 + \epsilon \frac{f_j(x)}{\omega}\right]$ $(j = 1, \ldots, m)$

Figure 4: Algorithm for integer packing. To obtain covering algorithm, negate ϵ's and change "max" to "min", "minimize" to "maximize", and ">" to "<".

Let S be a random multiset as above.

LEMMA 8.2. *When* $|S| \geq 1/\lambda^*$,

$$\Pr\left[(\exists j) \sum_{x \in S} f_j(x) \leq 1 - \epsilon\right] < m/\exp[b(-\epsilon)/\omega].$$

The resulting algorithm is described in Figure 4. The number of iterations in this case is at most $\lceil 1/\lambda^* \rceil$. For the condition $m/\exp[b(-\epsilon)/\omega] \leq 1$, it suffices that $\epsilon \geq \sqrt{2\omega \ln m}$.

9 Conclusion

Partial derandomization: The point of oblivious rounding is not derandomization per se, but to achieve independence from the unknown aspects of the optimal solution. For some random rounding schemes, some of the parameters of the random process are known; these can be left in the algorithm. For instance, in concurrent multicommodity flow, the relative amount of flow of each commodity is known. A natural randomized rounding scheme is to choose a commodity with probability proportional to its (known) demand, and then to choose a flow path among paths for that commodity with probability proportional to its (unknown) weight in the optimal flow. Applying oblivious rounding to only the second random choice, gives a randomized algorithm in the style of [16].

Mixed bounds: Each of the random analyses in this paper employed a single type of probabilistic bound. This is not a limitation of the technique. Oblivous rounding can be applied to analyses using, e.g., sums of probabilities bounded by Raghavan's bounds, Hoeffding's bound, and Markov's inequality. This is relatively straightforward, if technically more tedious.

More general functions: Chernoff-type bounds exist for more general classes of functions than linear functions (e.g., Azuma's inequality [1]). A natural question is whether oblivious rounding can be applied to such bounds to optimize more general functions.

References

[1] Noga Alon and Joel H. Spencer. *The Probabilistic Method.* John Wiley and Sons, New York, 1992.

[2] Ingo Althöfer. On sparse approximations to randomized strategies and convex combinations. *Linear Algebra and its Applications*, 199, March 1994.

[3] Dimitris Bertsimas and Rakesh Vohra. Linear programming relaxations, approximation algorithms and randomization; a unified view of covering problems. Draft, January 1994.

[4] H. Brönnimann and Michael T. Goodrich. Almost optimal set covers in bounded VC-dimension. In *Proc. of the 10th Annual Symposium on Computational Geometry*, 1994.

[5] M. D. Grigoriadis and L. G. Kachiyan. *Approximate Solution of Matrix Games in Parallel*, pages 129–136. Elsevier Science Publishers B.V., 1992. Also available as TR-91-73 from DIMACS.

[6] M. D. Grigoriadis and L. G. Kachiyan. A sublinear-time randomized approximation algorithm for matrix games. Technical Report LCSR-TR-222, Rutgers University Computer Science Department, New Brunswick, NJ, April 1994.

[7] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *American Statistical Journal*, pages 13–30, March 1963.

[8] Thomas Hofmeister and Hanno Lefmann. Computing sparse approximations deterministically. Unpublished manuscript, Dortmund, Germany. hofmeist,lefmann@ls2.informatik.uni-dortmund.de, 1994.

[9] Joseph JáJá. *Introduction to Parallel Algorithms.* Addison-Wesley Publishing Company, Inc., 1992.

[10] David S. Johnson. Approximation algorithms for combinatorial problems. *Journal of Computer and System Sciences*, 9:256–278, 1974.

[11] P. Klein, S. Plotkin, C. Stein, and E. Tardos. Faster approximation algorithms for the unit capacity concurrent flow problem with applications to routing and finding sparse cuts. *SIAM Journal on Computing*, 23(3):466–487, June 1994.

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

[13] Richard J. Lipton and Neal E. Young. Simple strategies for large zero-sum games with applications to complexity theory. In *Proc. of the 26th Ann. ACM Symp. on Theory of Computing*, 1994. To appear.

[14] László Lovász. On the ratio of optimal integral and fractional covers. *Discrete Mathematics*, 13:383–390, 1975.

[15] Michael Luby and Noam Nisan. A parallel approximation algorithm for positive linear programming. In *Proc. of the 25th Ann. ACM Symp. on Theory of Computing*, pages 448–457, 1993.

[16] Serge Plotkin, David Shmoys, and Éva Tardos. Fast approximation algorithms for fractional packing and covering problems. In *Proc. of the 32nd IEEE Annual Symp. on Foundation of Computer Science*, pages 495–504, 1991.

[17] Prabhakar Raghavan. Probabilistic construction of deterministic algorithms approximating packing integer programs. *Journal of Computer and System Sciences*, 37(2):130–143, October 1988.

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

[19] F. Shahroki and D. W. Matula. The maximum concurrent flow problem. *Journal of the ACM*, 37:318–334, 1990.

[20] Joel H. Spencer. *Ten Lectures on the Probabilistic Method.* Society for Industrial and Applied Mathematics, 3600 University City Science Center, Philadelphia, PA 19104-2688, 1987.