Fractional covering with upper bounds on the variables: Solving LPs with negative entries

Naveen Garg and Rohit Khandekar*

Indian Institute of Technology Delhi
{naveen, rohitk}@cse.iitd.ernet.in

Abstract. We present a Lagrangian relaxation technique to solve a class of linear programs with negative coefficients in the objective function and the constraints. We apply this technique to solve (the dual of) covering linear programs with upper bounds on the variables: $\min\{c^\top x \mid Ax \geq b, x \leq u, x \geq 0\}$ where $c, u \in \mathbb{R}^m_+, b \in \mathbb{R}^n_+$ and $A \in \mathbb{R}^{n\times m}_+$ have non-negative entries. We obtain a *strictly* feasible, $(1+\epsilon)$-approximate solution by making $O(m\epsilon^{-2}\log m + \min\{n, \log\log C\})$ calls to an oracle that finds the *most-violated constraint*. Here C is the largest entry in c or u, m is the number of variables, and n is the number of covering constraints. Our algorithm follows naturally from the algorithm for the fractional packing problem and improves the previous best bound of $O(m\epsilon^{-2}\log(mC))$ given by Fleischer [1]. Also for a fixed ϵ, if the number of covering constraints is polynomial, our algorithm makes a number of oracle calls that is strongly polynomial.

1 Introduction

In last couple of decades, there has been extensive research on solving linear programs (LPs) efficiently, albeit approximately using Lagrangian relaxation. Starting from the work of Shahrokhi & Matula [2] on multicommodity flows, a series of papers including those of Plotkin et al. [3], Luby & Nisan [4], Grigoriadis & Khachiyan [5, 6], Young [7, 8], Garg & Könemann [9], and Fleischer [10] proposed several algorithms for packing and covering LPs. Refer to Bienstock [11] for a survey of this field. All of these algorithms rely crucially on the *non-negativity* of the coefficients in the objective function and the constraints.

In this paper we show how Lagrangian relaxation can be used to solve a class of LPs with negative coefficients in the objective function and the constraints. The basic idea behind this approach is as follows. Suppose we would like to solve an LP of the form

$$\begin{aligned} \max\ & c^\top x \\ \text{s.t.}\ & Ax \leq b \\ & x \geq 0 \end{aligned} \tag{1}$$

where the entries in A and c *may be negative*. Suppose also that there exists an optimum solution x^* to (1) such that $c^\top x^* \geq 0$ and $Ax^* \geq 0$. In such a case, we can add

* Supported in part by a fellowship from Infosys Technologies Ltd., Bangalore.

constraints $c^\top x \geq 0$ and $Ax \geq 0$ to (1) without affecting the optimum solution. The resulting linear program can then be thought of as a fractional packing problem

$$\max\{c^\top x \mid Ax \leq b, x \in P\}$$

where $P = \{x \geq 0 \mid c^\top x \geq 0, Ax \geq 0\}$. Such a packing problem can be solved using algorithms similar to Plotkin et al. [3], Grigoriadis & Khachiyan [12], Garg & Könemann [9], and Jansen & Zhang [13] provided one has an oracle to minimize linear functions over P.

1.1 Our contribution and previous work

We apply the technique mentioned above to (the dual of) a covering LP with upper bounds on the variables:

$$\begin{aligned} \min\ & c^\top x \\ \text{s.t.}\ \ & Ax \geq b \\ & x \leq u \\ & x \geq 0 \end{aligned} \tag{2}$$

where $c, u \in \mathbb{R}^m_+, b \in \mathbb{R}^n_+$ and $A \in \mathbb{R}^{n\times m}_+$ have non-negative entries. For a positive integer k, let $[k] = \{1, \ldots, k\}$. To solve (2), we assume that we have an oracle that given $x \in \mathbb{R}^m_+$, finds the *most-violated constraint*. Formally,

$$\text{given } x \in \mathbb{R}^m_+, \text{ the oracle finds } \min_{j\in[n]} \frac{A_j x}{b_j} \tag{3}$$

where A_j denotes the jth row of A. The dual of (2) has negative entries in the constraint matrix; we reduce it to a packing problem and prove the following theorem.

Theorem 1. *There exists an algorithm that given an error parameter $\omega \in (0, 1)$, either computes a* strictly *feasible, $\exp(\omega)$-approximation to (2), or proves that (2) is infeasible. The algorithm makes $O(m\omega^{-2}\log m + \min\{n, \log\log C\})$ calls to the oracle (3) where $C = \frac{\max_{i\in[m]} c_i u_i}{\min_{i:c_i u_i>0} c_i u_i}$.*

The LP (2) is a special case of the mixed packing and covering LP: $\min\{c^\top x \mid Ax \geq b, Px \leq u, x \geq 0\}$ where $c \in \mathbb{R}^m_+, b \in \mathbb{R}^n_+, u \in \mathbb{R}^{n'}_+, A \in \mathbb{R}^{n\times m}_+$ and $P \in \mathbb{R}^{n'\times m}_+$. The known algorithms (Young [8]) for mixed LP however compute only *approximately feasible* solutions in which either the packing or the covering constraints are violated by a factor of $(1 + \epsilon)$. Our algorithm, on the other hand, computes a *strictly* feasible solution. Fleischer [1] recently presented an algorithm to compute a $\exp(\omega)$-approximation to (2) by making $O(m\omega^{-2}\log(mC))$ calls[1] to the oracle (3). Theorem 1 thus improves her running time, especially when C is large or n is polynomial — for example, when the instance is explicitly given. We note that an approximate version of the oracle (3) that computes a row $j' \in [n]$ such that $A_{j'}x/b_{j'} \leq (1 + \epsilon)\min_{j\in[n]} A_j x/b_j$ is not

[1] Fleischer [1] defines C as $\max_{i\in[m]} c_i / \min_{i:c_i>0} c_i$. However, the running time of her algorithm depends, in fact, on C as defined in Theorem 1. Note that this value of C is invariant under the scaling of variables.

strong enough to distinguish between feasibility and infeasibility of the given instance. For example, if all covering constraints are approximately satisfied by $x = u$, i.e., $Au \geq b/(1+\epsilon)$, the algorithm may output $x = u$ as a solution even if it may not satisfy $Au \geq b$ implying that the instance is infeasible. Our algorithm follows naturally from the packing algorithm; and, in fact, can be thought of as an alternate description of Fleischer's algorithm [1].

2 The covering LP with upper bounds on the variables

To simplify the presentation, we modify the given instance of (2) as follows. It is no loss of generality to assume that all entries of c, u, and b are positive. This is because if $c_i = 0$, we can reduce the problem by setting $x_i = u_i$ and replacing each b_j with $b_j - A_{ji}u_i$. If $u_i = 0$, then clearly $x_i = 0$ in any feasible solution and hence we can remove this variable. Similarly, if $b_j = 0$, then the jth covering constraint is trivially satisfied and can be removed. We can also assume that the objective function $c = \mathbf{1}$ is the vector of all ones; this can be achieved by working with new variables $x'_i = c_i x_i$ and replacing each A_{ji} with A_{ji}/c_i and replacing each u_i with $c_i u_i$. We finally assume that $\min_{i\in[m]} u_i = 1$. This is achieved by dividing each entry of u and b by $\min_{i\in[m]} u_i$. While doing this, it is not necessary to scale the vector c by $\min_{i\in[m]} u_i$, since that amounts to just scaling the optimum value. Observe that after these modifications, we have $\max_{i\in[m]} u_i = C$. Note that the oracle (3) for the modified instance can be obtained from the given oracle for the original instance; and a solution for the original instance can be obtained from a solution of the modified instance. Since b and c have only positive entries, the optimum value of (2) is *positive*.

Before solving this linear program, we make a call to oracle (3) with $x = u$ to find a row $j \in [n]$ that minimizes $A_j u/b_j$. If $A_j u < b_j$, then the jth covering constraint cannot be satisfied even if we set each variable x_i to its upper bound u_i. In this case, we output "infeasible". Therefore, in what follows, we assume that $Au \geq b$, i.e., the given instance is feasible.

2.1 Reduction to the packing problem

Consider the following dual LP of (2).

$$\begin{aligned} \max\ & b^\top y - u^\top z \\ \text{s.t.}\quad & A^\top y - z \leq \mathbf{1} \\ & y, z \geq 0 \end{aligned} \tag{4}$$

Although (4) has negative entries in the constraints and the objective function, the following lemma proves that it satisfies the conditions discussed in Section 1.

Lemma 1. *Any optimum solution (y^*, z^*) to (4) satisfies $b^\top y^* - u^\top z^* > 0$ and $A^\top y^* - z^* \geq 0$.*

Proof. Since the optimum value of (2) is positive, by LP duality, $b^\top y^* - u^\top z^*$ is also positive. Now the only constraints on z are $z_i \geq 0$ and $z_i \geq (A^\top y)_i - 1$. Once $y = y^*$ is fixed, in order to maximize the objective function, we would set z_i to the smallest possible value, i.e., $\max\{0, (A^\top y)_i - 1\}$ and we have $(A^\top y^*)_i - z^*_i \geq 0$.

Thus we can add constraints $b^\top y - u^\top z > 0$ and $A^\top y - z \geq 0$ to (4) without affecting its optimum value. The resulting LP is

$$\begin{aligned} \max\ & b^\top y - u^\top z \\ \text{s.t.}\quad & A^\top y - z \leq \mathbf{1} \\ & b^\top y - u^\top z > 0 \\ & A^\top y - z \geq 0 \\ & y, z \geq 0 \end{aligned} \tag{5}$$

We now define an instance of the packing problem. Define a polytope $P \subset \mathbb{R}^{n+m}_+$ and functions $f_i : P \to \mathbb{R}_+, i \in [m]$ as follows.

$$\begin{aligned} P &= \{(y,z) \geq 0 \mid A^\top y - z \geq 0, b^\top y - u^\top z = 1\}, \\ f_i(y,z) &= (A^\top y)_i - z_i \ \text{ for } i \in [m]. \end{aligned} \tag{6}$$

Now consider the packing problem

$$\min_{(y,z)\in P} \max_{i\in[m]} f_i(y,z). \tag{7}$$

It is easy to see that (5) has an optimum value λ if and only if (7) has an optimum value $1/\lambda$. Furthermore, an $\exp(\omega)$-approximate solution of (7) can be scaled to obtain an $\exp(\omega)$-approximate solution of (5). It is therefore sufficient to obtain an $\exp(\omega)$ approximation to (7).

3 The packing algorithm

For completeness, we outline an algorithm for the packing problem. Given a convex set $P \subseteq \mathbb{R}^n$ and m convex non-negative functions $f_i : P \to \mathbb{R}_+, i \in [m]$, the packing problem is to compute

$$\lambda^* = \min_{y\in P} \lambda(y) \quad \text{where} \quad \lambda(y) := \max_{i\in[m]} f_i(y).$$

We refer to this problem as PRIMAL. For a non-zero vector $x \in \mathbb{R}^m_+$, define $\underline{x} := x/\sum_{i=1}^m x_i$ as the normalized vector and define $d(x) := \min_{y\in P} \langle \underline{x}, f(y)\rangle$ where $\langle \underline{x}, f(y)\rangle = \sum_i \underline{x}_i f_i(y)$ denotes the inner product. It is easy to see that $d(x) \leq \lambda^* \leq \lambda(y)$ for any $x > 0$ and $y \in P$. We refer to the problem of finding $x > 0$ such that $d(x)$ is maximum as the DUAL. The duality theorem (see, e.g., Rockafellar [14], p.393) implies that

$$\min_{y\in P} \lambda(y) = \max_{x>0} d(x).$$

The algorithm assumes an oracle that given $x > 0$, computes $y \in P$ such that

$$\langle \underline{x}, f(y)\rangle \leq \exp(\epsilon) \cdot d(x) \tag{8}$$

for a constant $\epsilon \in (0,1)$.

Theorem 2. *The algorithm in Figure 1 computes $x^* > 0$ and $y^* \in P$ such that $\lambda(y^*) \leq \exp(3\epsilon) \cdot d(x^*)$. The duality then implies that x^* and y^* form the near-optimum solutions to the* DUAL *and* PRIMAL *problems respectively. The algorithm makes $O(m\epsilon^{-2}\log m)$ calls to the oracle (8).*

For the proof of Theorem 2, we refer to [15].

(1)	$x := \mathbf{1}$	{$\mathbf{1}$ is a vector of all ones}
(2)	Find $y \in P$ such that $\langle \underline{x}, f(y)\rangle \leq \exp(\epsilon) \cdot d(x)$	{oracle call}
(3)	$x^* := x, d^* = \langle \underline{x}, f(y)\rangle$	{current dual and its value}
(4)	$r := 0$	{initialize the round number}
(5)	Repeat	
(6)	$r := r + 1$	{round r begins}
(7)	Find $y_r \in P$ such that $\langle \underline{x}, f(y_r)\rangle \leq \exp(\epsilon) \cdot d(x)$	{oracle call}
(8)	$w_r := 1/\max_{i\in[m]} f_i(y_r)$	{pick y_r to an extent w_r}
(9)	For $i \in [m]$ do $x_i := x_i \cdot \exp(\epsilon w_r f_i(y_r))$	{update the dual variables}
(10)	If $\langle \underline{x}, f(y_r)\rangle > d^*$	
	then $x^* := x, d^* := \langle \underline{x}, f(y_r)\rangle$	{keep track of the best dual}
(11)	Until ($\max_{i\in[m]} x_i \geq m^{2/\epsilon}$)	{round r ends}
(12)	Output x^* and $y^* = (\sum_{s=1}^{r} w_s y_s)/(\sum_{s=1}^{r} w_s)$	

Fig. 1. The packing algorithm with exponential updates of the duals

4 Implementing the desired oracle

To solve (7) using the packing algorithm, we need an oracle that given a non-zero vector $x \in \mathbb{R}^m_+$, computes $(y, z) \in P$ that minimizes $\underline{x}^\top(A^\top y - z)$ within a factor of $\exp(\epsilon)$, where $\underline{x} = x/\sum_{i=1}^m x_i$. Since $\sum_{i=1}^m x_i$ is fixed, we need to approximate

$$R(x) := \min_{(y,z)\in P} x^\top(A^\top y - z). \tag{9}$$

Since $b^\top y - u^\top z = 1$ for $(y, z) \in P$, we have

$$R(x) = \min\left\{ \frac{x^\top(A^\top y - z)}{b^\top y - u^\top z} \mid y, z \geq 0, A^\top y - z \geq 0, b^\top y - u^\top z > 0 \right\}.$$

We now show how to find (y, z) that achieves an approximate value of $R(x)$.

For $x \in \mathbb{R}^m_+$ and $\alpha \geq 0$, define $(x|_\alpha) \in \mathbb{R}^m_+$ as the "truncated" vector with $(x|_\alpha)_i = \min\{x_i, \alpha u_i\}$ for $i \in [m]$. Define

$$r_\alpha(x) = \min_{j\in[n]} \frac{A_j(x|_\alpha)}{b_j}$$

where A_j is the jth row of A. Given x and α, we can compute $r_\alpha(x)$ by making one call to oracle (3). The proof of the following lemma is omitted due to lack of space.

Lemma 2. *For a fixed $x \in \mathbb{R}^m_+$, the function $r_\alpha(x)$ is a non-decreasing and concave function of α.*

Lemma 3. *For any $x \in \mathbb{R}^m_+$ with $x_i > 0$ for $i \in [m]$, we have $r_{R(x)}(x) = R(x)$.*

Proof. Let $y, z \geq 0$ be such that $A^\top y - z \geq 0, b^\top y - u^\top z > 0$ and

$$R(x) = \frac{x^\top(A^\top y - z)}{b^\top y - u^\top z}.$$

It is no loss of generality to assume that the above minimum is achieved for $\langle \mathbf{1}, y\rangle = 1$; this can be achieved by scaling y, z. Recall that $R(x)$ is the minimum possible value of this fraction.

Claim. Without loss of generality, we can assume that the vector z is of the form

$$z_i = \begin{cases} (A^\top y)_i, \text{ if } x_i > R(x) \cdot u_i; \\ 0, \qquad \text{otherwise.} \end{cases} \tag{10}$$

Proof. Note that the coefficient of z_i in the numerator is x_i, while its coefficient in the denominator is u_i. If $x_i/u_i \leq R(x)$, we can set $z_i = 0$, since in this case, by decreasing z_i, the value of the fraction does not increase. If on the other hand $x_i/u_i > R(x)$, the variable z_i must be set to its highest possible value, $(A^\top y)_i$, since otherwise by increasing z_i, we can decrease the value of the fraction. On increasing z_i, however, the value of the denominator $b^\top y - u^\top z$ may reach zero. But in this case, since $x_i/u_i > R(x)$, the numerator $x^\top(A^\top y - z)$ reaches zero "before" the denominator. Thus we would have $x^\top(A^\top y - z) = 0$ and $b^\top y - u^\top z > 0$ for some $y, z \geq 0$. Since $x_i > 0$ for $i \in [m]$, it must be that $A^\top y - z = 0$. This, in turn, implies that (4) is unbounded and (2) is infeasible yielding a contradiction. This proves the claim.

Now since $x^\top(A^\top y - z) = R(x) \cdot (b^\top y - u^\top z)$, we have $x^\top A^\top y - (x^\top - R(x) \cdot u^\top)z = R(x) \cdot b^\top y$.

Lemma 4. *If z satisfies $z_i = (A^\top y)_i$ if $x_i > \alpha \cdot u_i$, and 0 otherwise for some α, then we have $x^\top A^\top y - (x^\top - \alpha \cdot u^\top)z = (x|_\alpha)^\top A^\top y$.*

Proof. We show that $x_i(A^\top y)_i - (x_i - \alpha \cdot u_i)z_i = (x|_\alpha)_i (A^\top y)_i$ holds for each $i \in [m]$. If $x_i - \alpha \cdot u_i \leq 0$, we have $z_i = 0$ and $(x|_\alpha)_i = x_i$, hence the equality holds. On the other hand, if $x_i - \alpha \cdot u_i > 0$, we have $z_i = (A^\top y)_i$ and $x_i(A^\top y)_i - (x_i - \alpha \cdot u_i)z_i = (\alpha \cdot u_i)(A^\top y_i) = (x|_\alpha)_i (A^\top y)_i$. This proves Lemma 4.

Thus we have $(x|_{R(x)})^\top A^\top y = R(x) \cdot b^\top y$, i.e.,

$$R(x) = \frac{(x|_{R(x)})^\top A^\top y}{b^\top y}.$$

Further we can assume that y is of the form

$$\begin{aligned} y_j &= 1 \quad \text{for a row } j \in [n] \text{ such that } ((x|_{R(x)})^\top A^\top)_j / b_j \text{ is minimum,} \\ y_{j'} &= 0 \quad \text{for } j' \neq j. \end{aligned} \tag{11}$$

This is true because by replacing y with the vector as defined above, the value of the ratio does not increase. Therefore indeed we have $r_{R(x)}(x) = \min_j A_j(x|_{R(x)})/b_j = R(x)$ and the proof of Lemma 3 is complete.

Lemma 5. *For any $x \in \mathbb{R}^m_+$ with $x_i > 0$ for $i \in [m]$, we have $r_\alpha(x) < \alpha$ if and only if $\alpha > R(x)$.*

Proof. Since r_α is a concave function of α and $r_0(x) = 0, r_{R(x)} = R(x)$, we have $r_\alpha(x) \geq \alpha$ for $0 \leq \alpha \leq R(x)$. Therefore $r_\alpha(x) < \alpha$ implies $\alpha > R(x)$.

Now let $\alpha > R(x)$. Again using the concavity of r_α, we have $r_\alpha(x) \leq \alpha$. We now show that strict inequality holds. Let (y, z) be vectors as given in (11) and (10) that achieve the minimum ratio $R(x)$. Note that we have $(A^\top y)_i - z_i > 0$ for some $i \in [m]$;

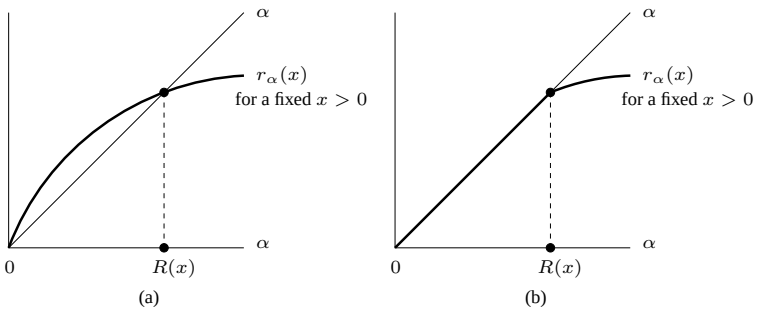

Fig. 2. The possible forms of the function r_α for a fixed $x > 0$. (a) case if $\min_{j\in[n]} A_j u/b_j > 1$, (b) case if $\min_{j\in[n]} A_j u/b_j = 1$.

otherwise the argument in the proof of Lemma 3 implies that (2) is infeasible. For such $i \in [m]$, we have $(A^\top y)_i > 0$ and $z_i = 0$. This implies $x_i \le R(x) \cdot u_i$ and $A_{ij} > 0$ for index $j \in [n]$ defined in (11). Note that $x_i \le R(x) \cdot u_i < \alpha \cdot u_i$ implies $(x|_\alpha)_i = (x|_{R(x)})_i = x_i > 0$. This combined with $A_{ij} > 0$ and $(x|_\alpha) \le \alpha/R(x) \cdot (x|_{R(x)})$ implies that $A_j(x|_\alpha)/b_j < \alpha/R(x) \cdot A_j(x|_{R(x)})/b_j = \alpha/R(x) \cdot R(x) = \alpha$. Since this holds for all $j \in [n]$ that attain the minimum in (11), we have $r_\alpha(x) < \alpha$. Thus Lemma 5 is proved.

Now from Lemmas 2,3, and 5, the function r_α must be of the form given in Figure 2. Fix an $x > 0$. Consider an $\alpha \in [0, \min_{i\in[m]} x_i/u_i]$. Note that $(x|_\alpha) = \alpha \cdot u$. Therefore we have $r_\alpha(x) = \alpha \cdot \min_{j\in[n]} A_j u/b_j$. Since (2) is feasible, we have $\min_{j\in[n]} A_j u/b_j \ge 1$. Now if $\min_{j\in[n]} A_j u/b_j > 1$, we have $r_\alpha(x) > \alpha$. In this case, from Lemmas 2,3, and 5, we get that $\alpha = 0$ and $\alpha = R(x)$ are the only values of α satisfying $r_\alpha(x) = \alpha$ and the function r_α has the form given in Figure 2 (a). On the other hand, if $\min_{j\in[n]} A_j u/b_j = 1$, we have $r_\alpha(x) = \alpha$. In this case, the function r_α has the form given in Figure 2 (b). In either case, $R(x)$ is the largest value of α such that $r_\alpha(x) = \alpha$.

4.1 Computing $R(\mathbf{1})$

In the packing algorithm (Figure 1), we initialize $x = \mathbf{1}$. In this section, we describe how to compute $R(\mathbf{1})$ and prove the following lemma.

Lemma 6. *The exact value of $R(\mathbf{1})$ can be computed by making $O(n + \log m)$ calls to the oracle (3). Furthermore, given $\epsilon \in (0, 1)$, an $\exp(\epsilon)$ approximation to $R(\mathbf{1})$ can be computed by making $O(\log\log C + \log(1/\epsilon))$ calls to the oracle (3).*

Lemma 7. $R(\mathbf{1}) \ge 1/\max_i u_i = 1/C$. *Further if* $R(\mathbf{1}) \ge 1/\min_i u_i = 1$, *then* $R(\mathbf{1}) = \min_{j\in[n]} A_j\mathbf{1}/b_j$.

Proof. We first argue that $R(u) \ge 1$. Recall that

$$R(u) = \min_{(y,z)} \frac{u^\top A^\top y - u^\top z}{b^\top y - u^\top z}$$

where the minimum is taken over $(y, z) \geq 0$ such that $u^\top A^\top y - u^\top z \geq 0$ and $b^\top y - u^\top z > 0$. Since (2) is feasible, we have $\min_{j\in[n]} A_j u/b_j \geq 1$. Hence to minimize the above ratio, one has to set z to zero and this gives $R(u) = \min_{j\in[n]} A_j u/b_j \geq 1$. Since $C \cdot \mathbf{1} \geq u$, we have $R(C \cdot \mathbf{1}) \geq R(u) \geq 1$. Therefore $R(\mathbf{1}) = R(C \cdot \mathbf{1})/C \geq 1/C$.

Now assume $R(\mathbf{1}) \geq 1$. Since $u \geq \mathbf{1}$, we have $\min\{1, R(\mathbf{1}) \cdot u_i\} = 1$ for each $i \in [m]$. Therefore $(\mathbf{1}|_{R(\mathbf{1})}) = \mathbf{1}$. From Lemma 3, we have $R(\mathbf{1}) = r_{R(\mathbf{1})}(\mathbf{1})$. Therefore, $R(\mathbf{1}) = \min_{j\in[n]} A_j \mathbf{1}/b_j$ as desired.

Computing $R(1)$ approximately Now we describe a simple procedure to compute an $\exp(\epsilon)$ approximation to $R(\mathbf{1})$ where $\epsilon \in (0, 1)$. We first compute $r_1(\mathbf{1})$. If $r_1(\mathbf{1}) \geq 1$, Lemma 5 implies that $R(\mathbf{1}) \geq 1$. In this case, Lemma 7 gives that $R(\mathbf{1}) = \min_{j\in[n]} A_j \mathbf{1}/b_j$ and we can compute this value by making one more oracle call.

On the other hand, if $r_1(\mathbf{1}) < 1$, then we have $1/C \leq R(\mathbf{1}) < 1$. In such a case, we have lower and upper bounds on $R(\mathbf{1})$ within a factor C. We do a bisection search as follows. We set $\alpha = \sqrt{1/C}$ and compute $r_\alpha(\mathbf{1})$. If $r_\alpha(\mathbf{1}) \geq \alpha$, Lemma 5 implies that $R(\mathbf{1}) \in [\alpha, 1]$. On the other hand, if $r_\alpha(\mathbf{1}) < \alpha$, we get that $R(\mathbf{1}) \in [1/C, \alpha]$. In either case, we reduce the ratio of upper and lower bounds from C to $\sqrt{C}$. By repeating this $O(\log\log C + \log(1/\epsilon))$ times, we can reduce the ratio of upper and lower bounds to $\exp(\epsilon)$. Thus we can compute α such that $\exp(-\epsilon) \cdot \alpha \leq R(\mathbf{1}) < \alpha$.

Computing $R(1)$ exactly We use the fact that $R(\mathbf{1})$ is the maximum value of α such that $r_\alpha(\mathbf{1}) = \alpha$ (see Figure 2) to compute the exact value of $R(\mathbf{1})$. Reorder the indices $i \in [m]$ such that $1 = u_1 \leq u_2 \leq \cdots \leq u_m = C$. We first compute $r_1(\mathbf{1})$. If $r_1(\mathbf{1}) \geq 1/u_1 = 1$, then Lemmas 5 and 7 imply that $R(\mathbf{1}) = \min_{j\in[n]} A_j \mathbf{1}/b_j$ and we can compute it exactly. We therefore assume that $1/u_m \leq R(\mathbf{1}) < 1/u_1$. Given $i \in [m]$, we can determine if $R(\mathbf{1}) \leq 1/u_i$. This is done by computing $r_\alpha(\mathbf{1})$ with $\alpha = 1/u_i$ and comparing its value with α (see Lemma 5). Therefore, by making $O(\log m)$ oracle calls, we can compute (unique) $k \in [m-1]$ such that $1/u_{k+1} \leq R(\mathbf{1}) < 1/u_k$. Now note that for $\alpha \in [1/u_{k+1}, 1/u_k]$, we have

$$(\mathbf{1}|_\alpha)_i = \min\{1, \alpha u_i\} = \begin{cases} \alpha u_i, & \text{for } i = 1, \ldots, k; \\ 1, & \text{for } i = k+1, \ldots, m. \end{cases}$$

Therefore for any row $j \in [n]$, we have that $A_j(\mathbf{1}|_\alpha)/b_j$ is a linear function of α in the range $\alpha \in [1/u_{k+1}, 1/u_k]$. In Figure 3, we have shown the linear functions corresponding to some rows j_0 and j_1. Recall that the function $r_\alpha(\mathbf{1})$ is the "lower envelope" of these functions and our objective is to compute $R(\mathbf{1})$ which is the largest value of α such that $r_\alpha(\mathbf{1}) = \alpha$. Thus $R(\mathbf{1})$ is the point at which the lower envelope intersects the line corresponding to the linear function α which is shown dotted in Figure 3.

We first make an oracle call to compute $j_0 = \arg\min_{j\in[n]} A_j(\mathbf{1}|_\alpha)/b_j$ where $\alpha = 1/u_k$. (Refer to Figure 3.) The solid line corresponds to the linear function $A_{j_0}(\mathbf{1}|_\alpha)/b_{j_0}$. We assume that the oracle also returns the most-violated constraint. Since we know the row A_{j_0}, we can compute α_{j_0} which is the value of α for which $A_{j_0}(\mathbf{1}|_\alpha)/b_{j_0} = \alpha$. As shown in the figure, α_{j_0} is the value of α at which the dotted and the solid lines intersect.

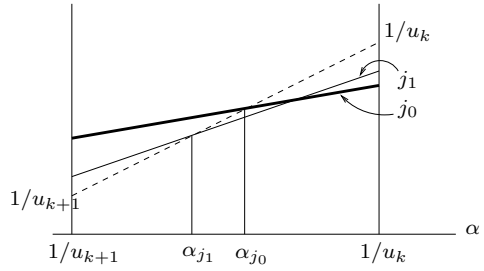

Fig. 3. Eliminating rows one-by-one to compute $R(\mathbf{1})$ exactly

We then make another oracle call to compute $j_1 = \arg\min_{j\in[n]} A_j(\mathbf{1}|_{\alpha_{j_0}})/b_j$. The light line in Figure 3 corresponds to the linear function $A_{j_1}(\mathbf{1}|_\alpha)/b_{j_1}$. If $j_1 = j_0$, then since the solid line goes below the dotted line for $\alpha > \alpha_{j_0}$, we know that α_{j_0} is the largest value of α such that $r_\alpha(\mathbf{1}) = \alpha$ and $R(\mathbf{1}) = \alpha_0$. On the other hand, if $j_1 \neq j_0$, we find α_{j_1} which is the value of α at which the dotted and the light lines intersect. We continue this to compute rows $j_2, j_3, \ldots$ till we find $j_{l+1} = j_l$ for some $l \geq 0$. Since $R(\mathbf{1})$ itself is a value of α that corresponds to the intersection of the dotted line and the line corresponding to the function $A_{j'}(\mathbf{1}|_\alpha)/b_{j'}$ for some row $j' \in [n]$, after $O(n)$ such iterations, we find $j_{l+1} = j_l = j'$ for some $l \geq 0$. Thus we can compute $R(\mathbf{1})$ by making $O(n + \log m)$ calls to the oracle (3).

Computing $(y, z) \in P$ that approximates $R(\mathbf{1})$ well Given an approximate or an exact value of $R(\mathbf{1})$, we still need to find $(y, z) \in P$ that achieves an approximation to $R(\mathbf{1})$.

Lemma 8. *Given $x \in \mathbb{R}^m_+$ such that $x_i > 0$ for $i \in [m]$ and α such that $\exp(-\epsilon) \cdot \alpha \leq R(x) < \alpha$, we can compute $(y, z) \in P$ such that $x^\top(A^\top y - z) \leq \exp(\epsilon) \cdot R(x)$ by making one call to the oracle (3).*

Proof. We use an argument similar to the one given in the proof of Lemma 3. We compute $j \in [n]$ such that $A_j(x|_\alpha)/b_j$ is minimum and set $y_j = 1$ and $y_{j'} = 0$ for $j' \neq j$. Therefore we have $(x|_\alpha)^\top A^\top y = r_\alpha(x) \cdot b^\top y$. We also set

$$z_i = \begin{cases} (A^\top y)_i, \text{ if } x_i > \alpha \cdot u_i; \\ 0, \qquad \text{otherwise.} \end{cases}$$

Now Lemma 4 implies that $x^\top A^\top y - (x^\top - \alpha \cdot u^\top)z = (x|_\alpha)^\top A^\top y$. Hence $x^\top A^\top y - (x^\top - \alpha \cdot u^\top)z = r_\alpha(x) \cdot b^\top y$. This, in turn, implies $x^\top(A^\top y - z) = r_\alpha(x) \cdot b^\top y - \alpha \cdot u^\top z < \alpha(b^\top y - u^\top z)$. The strict inequality follows from $r_\alpha(x) < \alpha$ (Lemma 5) and $b^\top y > 0$. Therefore we have $b^\top y - u^\top z > 0$ and

$$\frac{x^\top(A^\top y - z)}{b^\top y - u^\top z} < \alpha \leq \exp(\epsilon) \cdot R(x).$$

Thus the proof is complete.

Using the above lemma, we can compute $(y, z) \in P$ that achieves an $\exp(\epsilon)$ approximation to $R(\mathbf{1})$.

5 Algorithm for solving (7)

In the packing algorithm (Figure 1), we initialize $x = \mathbf{1}$. We set $\epsilon = \omega/4$ where $\omega \in (0,1)$ is the given error parameter. As explained in the previous section, we compute an α such that $\exp(-\epsilon) \cdot \alpha \leq R(x) < \alpha$. Using Lemma 8, we compute $(y,z) \in P$ that forms a $\exp(\epsilon)$ approximation to the oracle call and update the "dual variables" x (step (9) in Figure 1). During the algorithm, the values of x_i for $i \in [m]$ never decrease. Therefore, the value of

$$R(x) = \min_{(y,z)\in P} x^\top (A^\top y - z)$$

also never decreases. We repeat this till we find that $R(x) \geq \alpha$; this can detected by checking if $r_\alpha(x) \geq \alpha$ (Lemma 5). In such a case, we update the value of α as $\alpha := \exp(\epsilon) \cdot \alpha$ and repeat.[2]

Let α_0 be the initial value of α. Note that we have $R(\mathbf{1}) \in [\alpha_0 \exp(-\epsilon), \alpha_0)$. We say that the algorithm is in phase $p \geq 0$ if the current value of α is $\alpha_0 \exp(p\epsilon)$.

Bounding the number of oracle calls We designate an oracle call in which we do not find the desired $(y,z) \in P$ as "unfruitful". After each unfruitful oracle call, we update the value of α and go into the next phase. Hence the number of unfruitful calls to (3) is at most the number of phases in the algorithm. The following lemma proves a bound on the number of phases in the algorithm.

Lemma 9. *The algorithm has $O(\epsilon^{-2} \log m)$ phases.*

Proof. The algorithm initializes $x = \mathbf{1}$ and terminates when for some $i \in [m]$, we have $x_i \geq m^{2/\epsilon}$ (step (11) in Figure 1). Thus the maximum value α can take is $\alpha_0 \cdot \exp(\epsilon) m^{2/\epsilon}$. Since the value of α increases by $\exp(\epsilon)$ in each phase, the number of phases in the algorithm is $O(\epsilon^{-2} \log m)$.

Thus the number of unfruitful oracle calls is $O(\epsilon^{-2} \log m)$. Call an oracle call in which we find $(y,z) \in P$ satisfying the desired inequality as "fruitful". Clearly the number of fruitful oracle calls is equal to the number of oracle calls in the packing algorithm and from Theorem 2, this is $O(m\epsilon^{-2} \log m)$. In the pre-processing, to compute an approximate value of $R(\mathbf{1})$, we make $O(\min\{n + \log m, \log\log C + \log(1/\epsilon)\})$ oracle calls. Thus the total number of calls to the oracle (3) in the algorithm is $O(m\epsilon^{-2} \log m + \min\{n + \log m, \log\log C + \log(1/\epsilon)\}) = O(m\omega^{-2} \log m + \min\{n, \log\log C\})$.

Proving feasibility and near-optimality of the solutions Now from Theorem 2, the algorithm outputs $\tilde{x} \geq 0$ and $(\tilde{y}, \tilde{z}) \in P$ such that

$$\lambda(\tilde{y}, \tilde{z}) \leq \exp(3\epsilon) \cdot d(\tilde{x}) \tag{12}$$

where

$$\lambda(\tilde{y}, \tilde{z}) = \max_{i\in[m]} \left((A^\top \tilde{y})_i - \tilde{z}_i \right) \qquad \text{and} \qquad d(\tilde{x}) = \min_{(y,z)\in P} \langle \underline{\tilde{x}}, f(y,z) \rangle \tag{13}$$

[2] This technique of updating α in multiples of $\exp(\epsilon)$ was first used by Fleischer [10] to reduce the running time of the maximum multicommodity flow algorithm.

where $\underline{\tilde{x}} = \tilde{x}/\sum_{i=1}^{m} \tilde{x}_i$. Recall that the algorithm (Figure 1) outputs the best dual solution found in any round and hence

$$\tilde{x} = x^r \text{ where } r = \arg \max_{1 \le r \le N} d(x^r)$$

where x^r denotes the value of x at the end of round r. Let p be the phase in which the algorithm computes the best dual solution $\tilde{x}$. Let $\hat{x}$ be the value of the dual variables x at the end of the phase $p-1$. Let $\alpha_{p-1} = \alpha_0 \exp((p-1)\epsilon)$ be the value of α in phase $p-1$.

Lemma 10. *The vectors $x^* = (\hat{x}|_{\alpha_{p-1}})/\alpha_{p-1}$ and $(y^*, z^*) = (\tilde{y}, \tilde{z})/\lambda(\tilde{y}, \tilde{z})$ form feasible solutions to the primal linear program (2) and the dual linear program (4) respectively. Furthermore,*

$$\mathbf{1}^\top x^* \le \exp(\omega) \cdot \left(b^\top y^* - u^\top z^* \right).$$

Thus x^* and (y^*, z^*) form near-optimum solutions to (2) and (4) respectively.

Proof. Since $(\hat{x}|_{\alpha_{p-1}}) \le \alpha_{p-1} u$, we have $x^* \le u$. Since the oracle call with $x = \hat{x}$ was unfruitful at the end of phase $p-1$, we have $r_{\alpha_{p-1}}(\hat{x}) \ge \alpha_{p-1}$. Therefore $\min_{j \in [n]} A_j(\hat{x}|_{\alpha_{p-1}})/b_j = r_{\alpha_{p-1}}(\hat{x}) \ge \alpha_{p-1}$ and hence $\min_{j \in [n]} A_j x^*/b_j \ge 1$. Thus x^* forms a feasible solution to (2). From the definition of $\lambda(\tilde{y}, \tilde{z})$, it is clear that $A^\top y^* - z^* \le \mathbf{1}$ so that (y^*, z^*) forms a feasible solution to (4).

Now let $\alpha_p = \alpha_0 \exp(p\epsilon)$ be the value of α in phase p. We have

$$\begin{aligned}
\mathbf{1}^\top x^* &= \frac{\sum_{i=1}^{m} (\hat{x}|_{\alpha_{p-1}})_i}{\alpha_{p-1}} \\
&\le \frac{\sum_{i=1}^{m} \tilde{x}_i}{\alpha_{p-1}} \quad \text{(since } x \text{ does not decrease, we have } \tilde{x} \ge (\hat{x}|_{\alpha_{p-1}})\text{)} \\
&= \exp(\epsilon) \cdot \frac{\sum_{i=1}^{m} \tilde{x}_i}{\alpha_p} \quad \text{(since } \alpha_p = \exp(\epsilon) \cdot \alpha_{p-1}\text{)}
\end{aligned}$$

From the definitions (9) and (13) of $R(x)$ and $d(x)$ respectively, we have $R(x) = d(x) \cdot \sum_{i=1}^{m} x_i$. Since the oracle call with $x = \tilde{x}$ was fruitful in phase p, we have $\alpha_p \ge R(\tilde{x}) = d(\tilde{x}) \cdot \sum_{i=1}^{m} \tilde{x}_i$. Therefore

$$\begin{aligned}
\mathbf{1}^\top x^* &\le \exp(\epsilon) \cdot \frac{1}{d(\tilde{x})} \\
&\le \exp(4\epsilon) \cdot \frac{1}{\lambda(\tilde{y}, \tilde{z})} \quad \text{(from (12))} \\
&= \exp(\omega) \cdot \frac{b^\top \tilde{y} - u^\top \tilde{z}}{\lambda(\tilde{y}, \tilde{z})} \quad \text{(since } (\tilde{y}, \tilde{z}) \in P \text{ and } \omega = 4\epsilon\text{)} \\
&= \exp(\omega) \cdot \left(b^\top y^* - u^\top z^* \right).
\end{aligned}$$

Therefore the proof is complete.

Note that we can modify the algorithm to keep track of the values of the dual variables at the end of the previous round and thus be able to compute $\hat{x}, \alpha_{p-1}$, and hence x^*.

Thus the proof of Theorem 1 is complete.

6 Conclusion

We presented a technique to handle negative entries in the objective function and the constraints via Lagrangian relaxation. We applied this technique to the fractional covering problem with upper bounds on the variables. We reduced the dual of this problem to the standard packing framework and derived an algorithm for this problem. However, the technique seems to be very limited in its applicability since it needs an optimum solution with non-negative values for the objective function and the constraints. The negative entries can also be handled by working with perturbed functions as done in [15]. However, there is still no fairly general and satisfactory way of taking care of negative entries.

References

1. Fleischer, L.: A fast approximation scheme for fractional covering problems with variable upper bounds. In: Proceedings, ACM-SIAM Symposium on Discrete Algorithms. (2004) 994–1003
2. Shahrokhi, F., Matula, D.: The maximum concurrent flow problem. J. ACM **37** (1990) 318–334
3. Plotkin, S., Shmoys, D., Tardos, E.: Fast approximation algorithms for fractional packing and covering problems. Math. Oper. Res. **20** (1995) 257–301
4. Luby, M., Nisan, N.: A parallel approximation algorithm for positive linear programming. In: Proceedings, ACM Symposium on Theory of Computing. (1993) 448–457
5. Grigoriadis, M., Khachiyan, L.: Fast approximation schemes for convex programs with many blocks and coupling constraints. SIAM J. Optimization **4** (1994) 86–107
6. Grigoriadis, M., Khachiyan, L.: Approximate minimum-cost multicommodity flows in $\tilde{O}(\epsilon^{-2}knm)$ time. Math. Programming **75** (1996) 477–482
7. Young, N.: Randomized rounding without solving the linear program. In: Proceedings, ACM-SIAM Symposium on Discrete Algorithms. (1995) 170–178
8. Young, N.: Sequential and parallel algorithms for mixed packing and covering. In: Proceedings, IEEE Symposium on Foundations of Computer Science. (2001) 538–546
9. Garg, N., Könemann, J.: Faster and simpler algorithms for multicommodity flow and other fractional packing problems. In: Proceedings, IEEE Symposium on Foundations of Computer Science. (1998) 300–309
10. Fleischer, L.: Approximating fractional multicommodity flow independent of the number of commodities. SIAM J. Discrete Math. **13** (2000) 505–520
11. Bienstock, D.: Potential Function Methods for Approximately Solving Linear Programming Problems: Theory and Practice. Kluwer Academic Publishers, Boston (2002) An early version also appears as CORE Lecture Series Monograph, Core, UCL, Belgium (2001) (download from www.core.ucl.ac.be).
12. Grigoriadis, M., Khachiyan, L.: Coordination complexity of parallel price-directive decomposition. Math. Oper. Res. **21** (1994) 321–340
13. Jansen, K., Zhang, H.: Approximation algorithms for general packing problems with modified logarithmic potential function. In: Theoret. Comput. Sci. (2002) 255–266
14. Rockafellar, T.: Convex Analysis. Princeton University Press, Princeton, NJ (1970)
15. Khandekar, R.: Lagrangian relaxation based algorithms for convex programming problems. PhD thesis, Indian Institute of Technology Delhi (2004) Available at http://www.cse.iitd.ernet.in/∼rohitk.