Greedy Approximations and LP Duality

Kent Quanrud

April 23, 2021

1 Maximum coverage

In the maximum coverage problem, we have m elements $[m] = \{1, \dots, m\}$, and n sets $A_1, \dots, A_n \subseteq [m]$. We also have a cardinality constraint $k \in \mathbb{N}$. The goal is to select (at most) k sets $A_{e_1}, A_{e_2}, \dots, A_{e_k}$ that maximizes the size of the union

$$\left|A_{e_1} \cup A_{e_2} \cup \cdots \cup A_{e_k}\right|.$$

This problem is NP-Hard. We will describe the proof next class, when we discuss a closely related problem, set cover.

We will analyze the following *greedy algorithm* for maximum coverage.

> *Starting with an empty collection of sets. For k iterations, select the set that covers the most elements left uncovered by the current collection, and that set to the collection.*

Notation. To analyze the greedy algorithm, it is convenient to adopt the following notation. We identify a set A_e by its index e. We identify a collections of indices $S \subseteq [n]$ with the corresponding collection of sets $\{A_e : e \in S\}$. We define a function $f : 2^{[n]} \to \mathbb{R}_{\geq 0}$ by

$$f(S) \stackrel{\text{def}}{=} (\text{\# points covered by sets (w/ index) in } S) = \left|\bigcup_{e \in S} A_e\right|.$$

Now, for $i = 1, \dots, k$, let $e_i \in [n]$ be the index of the ith set selected by the greedy algorithm. Let $S_i = \{e_1, \dots, e_i\}$ be the first i selected indices. Thus $S_0 = \emptyset$, and S_k is the output of the greedy algorithm.

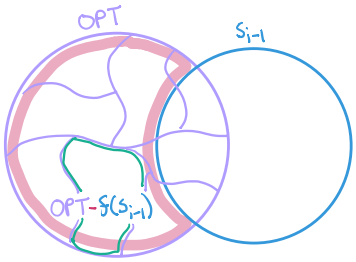
The key lemma. The key lemma analyzes the improvement we make in each iteration. We can interpret Lemma 1.1 below as saying that the greedy algorithm maintains the inequality

$$(\text{improvement}) \geq \frac{1}{k}(\text{room for improvement}),$$

on each iteration.

Lemma 1.1. *For $i = 1, \ldots, k$, $f(S_i) - f(S_{i-1}) \geq \frac{1}{k}(\mathrm{OPT} - f(S_{i-1}))$.*

Proof of Lemma 1.1. The proof is illustrated by the following diagram.



There are at least $\mathrm{OPT} - f(S_{i-1})$ elements covered by the optimal solution, but not covered by S_{i-1}. Meanwhile there are k sets in the optimal solution. In particular there is some set in the optimal solution that covers at least $(1/k)$th of the elements. Thus whatever set the greedy algorithm selects adds at least $(1/k)(\mathrm{OPT} - f(S_{i-1}))$ to the total coverage. ■

Overall analysis.

Theorem 1.2. $f(S_k) \geq \left(1 - e^{-1}\right) \mathrm{OPT}$.

Proof. We have shown that

$$\left(i\text{th improvement}\right) = f(S_i) - f(S_{i-1}) \geq \frac{1}{k}(\mathrm{OPT} - f(S_{i-1})) = \begin{pmatrix}\text{room for improvement} \\ \text{after } i-1 \text{ iterations}\end{pmatrix}.$$

Rearranging, we get

$$\mathrm{OPT} - f(S_i) \leq \left(1 - \frac{1}{k}\right)(\mathrm{OPT} - f(S_{i-1})).$$

That is,

$$\begin{pmatrix}\text{room for improvement} \\ \text{after } i \text{ iterations}\end{pmatrix} \leq \left(1 - \frac{1}{k}\right)\begin{pmatrix}\text{room for improvement} \\ \text{after } i-1 \text{ iterations}\end{pmatrix}.$$

We see that "room for improvement" decays exponentially. Unrolling the above, we have

$$\begin{aligned}
\mathrm{OPT} - f(S_0) &= \mathrm{OPT} \\
\mathrm{OPT} - f(S_1) &\leq \left(1 - \frac{1}{k}\right)\mathrm{OPT} \\
\mathrm{OPT} - f(S_2) &\leq \left(1 - \frac{1}{k}\right)^2 \mathrm{OPT} \\
\mathrm{OPT} - f(S_3) &\leq \left(1 - \frac{1}{k}\right)^3 \mathrm{OPT} \\
&\vdots
\end{aligned}$$

$$\begin{aligned}\mathrm{OPT} - f(S_k) &\leq \left(1 - \frac{1}{k}\right)^k \mathrm{OPT} \\ &\overset{(a)}{\leq} e^{-1}\,\mathrm{OPT}\,.\end{aligned}$$

The last step (a) comes from the inequality $1 + x \leq e^x$ for all x. The final inequality is the claim, up to rearrangement. ■

2 Linear programs and LP duality

Preliminaries. A matrix $A \in \mathbb{R}^{m \times n}$ is defined by coordinates $A_{i,j} \in \mathbb{R}$ for $i \in [m]$ and $j \in [n]$. A matrix A can be understood as a linear map $A : \mathbb{R}^n \to \mathbb{R}^m$ defined by

$$(Ax)_i = \sum_{j=1}^{n} A_{ij} x_j$$

for all $i \in [m]$. Conversely every linear map from $\mathbb{R}^n$ to $\mathbb{R}^m$ is defined by a matrix.

The **transpose** of a matrix $A \in \mathbb{R}^{m \times n}$ is the matrix $A^T \in \mathbb{R}^{n \times m}$ uniquely defined by[1]

$$\langle y, Ax \rangle = \left\langle A^T y, x \right\rangle$$

for all $x \in \mathbb{R}^n$ and $y \in \mathbb{R}^m$. The coordinates of the transpose $A^T \in \mathbb{R}^{n \times m}$ are given by

$$A^T_{ij} = A_{ji}$$

for all $i \in [n]$ and $j \in [m]$.

Linear programs. Linear programs are *constrained* continuous optimization problems where the goal is to (a) select a vector $x \in \mathbb{R}^n$ that (b) optimizes a linear objective subject to (c) linear equality and inequality constraints. That is, an optimization problem of the form

$$\begin{gathered}\min/\max \langle b, x \rangle = \sum_{j=1}^{n} b_j x_j \text{ over } x \in \mathbb{R}^n \\ \text{s.t. } A_1 x \leq c_1,\ A_2 x = c_2, \text{ and } A_3 x \geq c_3.\end{gathered}$$

where A_1, A_2, A_3 are matrices and b, c_1, c_2, c_3 are vectors. Linear programs are useful for modeling real problems where we seek continuous solutions. Next time we will also discuss their use in *relaxations* of integer problems, and how to turn the continuous solutions produced by a linear program into integral solutions, often for NP-Hard problems.

Packing LPs. A **packing LP** is a linear program of the form

$$\max \langle b, x \rangle \text{ over } x \in \mathbb{R}^n_{\geq 0} \text{ s.t.} Ax \leq c. \tag{P}$$

where $A \in \mathbb{R}^{m \times n}_{\geq 0}$, $b \in \mathbb{R}^n_{>0}$, and $c \in \mathbb{R}^m_{>0}$ all have nonnegative coefficients. We let $\textsc{Opt}(\mathsf{P})$ denote the optimum value of the LP $\textsc{Opt}(\mathsf{P})$.

[1] Recall that the inner product $\langle \cdot, \cdot \rangle$ in $\mathbb{R}^n$ is defined by $\langle x, y \rangle = \sum_{i=1}^n x_i y_i$.

Covering LPs. A **covering LP** is a linear program of the form

$$\min \langle c, y \rangle \text{ over } y \in \mathbb{R}^m_{\geq 0} \text{ s.t. } A^T y \geq b, \tag{C}$$

where[2] $A \in \mathbb{R}^{m \times n}_{\geq 0}$, $b \in \mathbb{R}^n_{>0}$, and $c \in \mathbb{R}^m_{>0}$. Abusing notation, we also let $\textsc{Opt}(\mathsf{C})$ denote the optimum value of the LP $\textsc{Opt}(\mathsf{C})$.

LP duality. LP duality is about the relationship between the linear programs $\textsc{Opt}(\mathsf{P})$ and $\textsc{Opt}(\mathsf{C})$, particular when the matrices and vectors A, b, c are the same for both problems.

Let $x \in \mathbb{R}^n_{\geq 0}$ be any feasible solution to (P) and let $y \in \mathbb{R}^n_{\geq 0}$ be any feasible solution to (C). We have

$$\langle b, x \rangle \overset{\text{(b)}}{\leq} \left\langle A^T y, x \right\rangle \overset{\text{(c)}}{=} \langle y, Ax \rangle \overset{\text{(d)}}{\leq} \langle y, c \rangle.$$

Here (b) is because $x \geq \mathbb{0}$ and $A^T y \geq b$. (c) is by definition of transpose. (d) is because $y \geq \mathbb{0}$ and $Ax \leq c$. Thus, for a packing problem (P) and a covering problem (C) linked by duality, we have

$$\textsc{Opt}(\mathsf{P}) \leq \textsc{Opt}(\mathsf{C}).$$

We have seen this argument before in several concrete instances: between (s, t)-flow and cuts, and between matchings and vertex covers. The max-flow min-cut theorem states that the maximum (s, t)-flow equals the minimum (s, t)-cut. On the other hand, the maximum cardinality matching does not generally equal the minimum vertex cover (e.g., in a triangle). We ask the same question for packing and covering LP's. When, if ever, is $\textsc{Opt}(\mathsf{P}) = \textsc{Opt}(\mathsf{C})$? The **LP duality theorem** (for packing and covering problems) states that in fact they are always equal.

Theorem 2.1 (LP Duality for packing and covering)**.** $\textsc{Opt}(\mathsf{P}) = \textsc{Opt}(\mathsf{C})$.

Our primary goal for the rest of the discussion is to prove Theorem 2.1. Along the way, we will also develop polynomial time approximation algorithms for (P).

3 An algorithmic proof of LP duality

3.1 Preliminaries

The partition function. We will use a convex potential function $\pi : \mathbb{R}^m \to \mathbb{R}$, alternatively called the **partition** or **soft-max** function, defined by

$$\pi(x) = \log\left(\sum_i e^{x_i}\right).$$

$\pi(x)$ has several nice properties which we now go over. The first property explains the name "soft-max".

[2]Of course, in (C), we could have written A instead of its transpose A^T, and swapped b and c, which would more closely resemble (P). It is convenient for the subsequent discussion on LP duality for A, b and c to have the same dimensions in (P) and (C).

Lemma 3.1. *For all $x \in \mathbb{R}^m$,*

$$\max_i x_i \leq \pi(x_i) \leq \max_i x_i + \log(m).$$

Proof. Suppose (without loss of generality) that $\max_i x_i = x_1$. Then

$$x_1 = \log(e^{x_1}) \leq \log\left(\sum_i e^{x_i}\right) \leq \log(me^{x_1}) = x_1 + \log(m),$$

as desired. ■

The second property, below, is about the derivative of $\pi(x)$ and how it is particularly easy to interpret.

Lemma 3.2. *For all x, $\pi'(x) \in \mathbb{R}^m$ is positive and its coordinates sum to 1. That is, $\pi'(x)$ describes a probability distribution over $[m]$.*

Proof. For each $i \in m$, we have

$$\pi'_i(x) = \frac{e^{x_i}}{\sum_{j=1}^m e^{x_j}}.$$

Note that $\pi'_i(x)$ is strictly positive, and the sum over i is 1. ■

The third and final property we discuss is similar to the "smoothness" property discussed last lecture. In a *multiplicative* sense, the following lemma says that a small change in x can only induce a small change in $\pi'(x)$.

Lemma 3.3. *Let $x, y \in \mathbb{R}^n$ and $\epsilon > 0$, and suppose $x_i \leq y_i \leq x_i + \epsilon$ for all coordinates i. Then for all i,*

$$e^{-\epsilon}\pi'_i(x) \leq \pi'_i(y) \leq e^{\epsilon}\pi'_i(x).$$

Proof. Recall that

$$\pi'_i(x) = \frac{e^{x_i}}{\sum_j e^{x_j}}.$$

If $x_i \leq y_i \leq x_i + \epsilon$, then

$$e^{x_i} \leq e^{y_i} \leq e^{\epsilon}e^{x_i}$$

for all i. The claim follows. ■

One can also show that $\pi(x)$ is convex in x, but we will not actually leverage convexity in our analysis.

Normalization. To simplify notation, we rescale (P) as follows. We first scale b to $\mathbb{1}$ (replacing each A_{ij} with A_{ij}/b_j). We also scale c to $\mathbb{1}$ (replacing each A_{ij} with A_{ij}/c_i). We are thus left with the following normalized packing problem,

$$\text{maximize } \langle \mathbb{1}, x \rangle \text{ over } x \in \mathbb{R}^n_{\geq 0} \text{ s.t. } Ax \leq \mathbb{1} \tag{$\mathsf{P}_{\mathbb{1}}$}$$

and the normalized covering problem,

$$\text{minimize } \langle \mathbb{1}, y \rangle \text{ over } y \in \mathbb{R}^m_{\geq 0} \text{ s.t. } A^T y \geq \mathbb{1}. \tag{$\mathsf{C}_{\mathbb{1}}$}$$

Note that the rescaling did not affect the optimum value of either LP: $\textsc{Opt}(\mathsf{P}) = \textsc{Opt}(\mathsf{P}_{\mathbb{1}})$, and $\textsc{Opt}(\mathsf{C}) = \textsc{Opt}((\mathsf{C}_{\mathbb{1}}))$. For ease of notation we will use $\textsc{Opt}(\mathsf{P})$ to refer to the optimum value of $(\mathsf{P}_{\mathbb{1}})$ and $\textsc{Opt}(\mathsf{C})$ to refer to the optimum value of $(\mathsf{C}_{\mathbb{1}})$.

3.2 Lagrangian relaxations

We let Δ^m denote the set of probability vectors over $[m]$; i.e.,

$$\Delta^m = \left\{ p \in \mathbb{R}^m_{\geq 0} : p_1 + \cdots + p_m = 1 \right\}.$$

For a probability vector $p \in \Delta^m$, consider the packing problem obtained by using p to collapse that packing constraints $Ax \leq \mathbb{1}$ into a single packing constraint $\langle p, Ax \rangle \leq 1$.

$$\text{maximize } \langle \mathbb{1}, x \rangle \text{ over } x \in \mathbb{R}^n_{\geq 0} \text{ s.t. } \langle p, Ax \rangle \leq 1. \tag{L}$$

For fixed p, we let $\textsc{Opt}(\mathsf{L})$ denote the objective value to (L).

Observe that $\langle p, Ax \rangle$ is a convex combination of packing constraints ($(Ax)_i \leq 1$ for all i) of $(\mathsf{P}_{\mathbb{1}})$. As such, (L) is a *relaxation* of $(\mathsf{P}_{\mathbb{1}})$ – every feasible solution to $(\mathsf{P}_{\mathbb{1}})$ is a feasible solution to (L). It is sometimes called a *Lagrangian relaxation* of $(\mathsf{P}_{\mathbb{1}})$. As a relaxation of $(\mathsf{P}_{\mathbb{1}})$, we automatically have

$$\textsc{Opt}(\mathsf{L}) \geq \textsc{Opt}(\mathsf{P}).$$

Interestingly, we can also relate $\textsc{Opt}(\mathsf{L})$ to $\textsc{Opt}(\mathsf{C})$, as follows.

Lemma 3.4. *For all $p \in \Delta^m$, $\textsc{Opt}(\mathsf{L}) \geq \textsc{Opt}(\mathsf{C})$.*

Proof. Let $\lambda > \textsc{Opt}(\mathsf{L})$. We claim that λp is a feasible solution to $(\mathsf{C}_{\mathbb{1}})$. This claim implies that $\lambda \geq \textsc{Opt}(\mathsf{C})$ for all $\lambda > \textsc{Opt}(\mathsf{L})$. Taking the limit as $\lambda \downarrow \textsc{Opt}(\mathsf{L})$, we obtain $\textsc{Opt}(\mathsf{L}) \geq \textsc{Opt}(\mathsf{C})$, desired.

Now, fix $i \in [m]$. Let $x = \lambda e_i$, where e_i is the vector that is 0 everywhere except the ith coordinate is 1. Since $\langle \mathbb{1}, x \rangle = \lambda > \textsc{Opt}(\mathsf{L})$, we must have $\langle p, Ax \rangle > 1$. Expanding out $\langle A, p \rangle x$, we have

$$\langle p, Ax \rangle = \lambda (A^T p)_i = \left(A^T (\lambda p) \right)_i > 1.$$

Thus $\left(A^T (\lambda p) \right)_i > 1$ for all i. That is, λp is a feasible solution to $(\mathsf{C}_{\mathbb{1}})$, hence $\langle \mathbb{1}, \lambda p \rangle = \lambda > \textsc{Opt}(\mathsf{C})$. ∎

3.3 Finally, the algorithm

We design an algorithm for $(\mathsf{P}_{\mathbb{1}})$ based on the following potential function that uses π. Let $\epsilon > 0$ be given, and let $\eta = \log(m)/\epsilon$. Consider the function $f : \mathbb{R}^n_{\geq 0} \to \mathbb{R}$ defined by

$$f(x) \stackrel{\text{def}}{=} \frac{1}{\eta}\pi(\eta Ax). \tag{1}$$

We think of f is a continuous approximation to $\max_i (Ax)_i$, as follows.

Lemma 3.5. *For all x,*

$$\max_i (Ax)_i \leq f(x) \leq \max_i (Ax)_i + \epsilon.$$

Proof. By Lemma 3.1, we have

$$\max_i \eta(Ax)_i \leq \eta f(x) \leq \max_i \eta(Ax)_i + \log(m).$$

Dividing by η gives the claimed inequality. ■

Note that the derivative of f,

$$f'(x) = A^T \pi'(\eta Ax) \in \mathbb{R}^n_{\geq 0},$$

can be interpreted as a convex combination of the original packing constraints.

Theorem 3.6. *Let $\epsilon > 0$ and consider an LP of the form $(\mathsf{P}_{\mathbb{1}})$. Suppose that, given any distribution $p \in \Delta_m$, one can compute a $(1-\epsilon)$-approximation to the Lagrangian relaxation*

$$\textit{maximimize } \langle \mathbb{1}, x \rangle \textit{ s.t. } \langle p, Ax \rangle \leq 1. \tag{L}$$

Then in $O\big(m \log(n)/\epsilon^2\big)$ iterations, one can compute a point $x \in \mathbb{R}^n_{\geq 0}$ such that

$$\langle \mathbb{1}, x \rangle \geq (1-\epsilon)\textsc{Opt}(\mathsf{C}) \text{ and } Ax \leq \mathbb{1}.$$

Therefore, $\textsc{Opt}(\mathsf{P}) \geq (1-\epsilon)\textsc{Opt}(\mathsf{C})$ for all $\epsilon > 0$ sufficiently small, hence $\textsc{Opt}(\mathsf{P}) = \textsc{Opt}(\mathsf{C})$.

Proof. For simplicity, we will directly show how to obtain a $(1 - c\epsilon)$-approximation for some constant $c > 0$. A $(1-\epsilon)$-approximation is obtained by decreasing ϵ appropriately.

We will analyze a discrete algorithm that iteratively solves different Lagrangian relaxations of $(\mathsf{P}_{\mathbb{1}})$. It is convenient for the analysis to also take a continuous point of view. Let $t \in [0, 1]$ be a variable that represents time; the algorithm starts at $t = 0$ and terminates at $t = 1$. Each iteration k will increase t by a discrete amount of time $\delta_k > 0$. For $k = 1, 2, \ldots,$ we let $t_k = \sum_{\ell < k} \delta_k$ denote the time t at the beginning of the kth iteration.

We build a solution x incrementally over time. To this end, we let $x(t)$ denote the value of x at time t. Thus $x(t_k)$ denotes the value of x at the beginning of the kth iteration. We start from an empty solution, $x(t_1) = x(0) = \mathbb{0}$. Each iteration k, we will define a fixed rate of change $x'(t)$ for all $t \in [t_k, t_{k+1})$. Thus we have $x'(t) = \int_0^t x'(s)\, ds$ for all t.

Fix an iteration k. Consider the optimization problem,

$$\text{maximize } \langle \mathbb{1}, z \rangle \text{ s.t. } \langle f'(x(t_k)), z \rangle \leq 1.$$

Expanding out $f'(x(t_k)) = A^T \pi'(\eta A x(t_k))$ and transposing, *the above problem is the same as the following Lagrangian relaxation of* $(\mathsf{P}_{\mathbb{1}})$*:*

$$\text{maximize } \langle \mathbb{1}, z \rangle \text{ s.t. } \langle \pi'(\eta A x(t_k)), A x(t_k) \rangle \leq 1.$$

Here we recall that $\pi'(\eta A x(t_k)) \in \Delta^m$, by Lemma 3.2. Now, by Lemma 3.4, this relaxation has optimum value at least $\textsc{Opt}(\mathsf{C})$.

We approximately solve the Lagrangian relaxation and obtain a point z_k such that

$$\langle \mathbb{1}, z_k \rangle \geq (1-\epsilon)\textsc{Opt}(\mathsf{C}) \text{ and } \langle \pi'(\eta A x(t_k)), A z_k \rangle \leq 1.$$

Let δ_k large as possible such that $t_k + \delta_k \leq 1$ and

$$\delta_k \eta A z_k \leq \epsilon \mathbb{1}.$$

We set $t_{k+1} = t_k + \delta_k$. For all $t \in [t_k, t_k)$, we update $x(t)$ at a rate of

$$x'(t) = z_k.$$

Claim 1. For all $t \in [0,1]$, $\frac{d}{dt}\langle \mathbb{1}, x(t) \rangle \geq (1-\epsilon)\textsc{Opt}(\mathsf{C})$.

At any time t, we have $x'(t) = z_k$ for some k, hence $\frac{d}{dt}\langle \mathbb{1}, x(t) \rangle = \langle \mathbb{1}, x'(t) \rangle \geq (1-\epsilon)\textsc{Opt}(\mathsf{C})$.

Claim 2. For all $t \in [0,1]$*, we have*

$$\frac{d}{dt} f(x(t)) \leq e^{\epsilon}.$$

Fix t. Then $t \in [t_k, t_{k+1})$ for some iteration k. We have

$$\eta A x(t_k) \leq \eta A x(t) \leq \eta A x(t_{k+1}) = \eta A x(t_k) + \delta_k \eta A z_k \overset{\text{(a)}}{\leq} \eta A x(t_k) + \epsilon \mathbb{1}$$

by (a) choice of δ_k. By Lemma 3.3, the inequalities above imply that $f'(x(t)) \leq e^{\epsilon} f'(x(t_k))$. Thus

$$\frac{d}{dt} f(x(t)) = \langle f'(x(t)), z_k \rangle \leq e^{\epsilon} \langle f'(x(t_k)), z_k \rangle \leq e^{\epsilon},$$

as desired.

We have now analyze the rate of change of both the objective, $\langle \mathbb{1}, x(t) \rangle$, and the surrogate function for the constraints, $f(x(t))$, over time t. These translate to the following bounds for $x(1)$ by elementary calculus.

Claim 3. $\langle \mathbb{1}, x(1) \rangle \geq \textsc{Opt}(\mathsf{C})$.

We have

$$\langle \mathbb{1}, x(1)\rangle = \langle \mathbb{1}, x(1) - x(0)\rangle \overset{\text{(b)}}{=} \int_0^1 \frac{d}{dt}\langle \mathbb{1}, x(t)\rangle \, dt \overset{\text{(c)}}{\geq} \int_0^1 (1-\epsilon)\textsc{Opt}(\mathsf{C}) \, dt = (1-\epsilon)\textsc{Opt}(\mathsf{C}).$$

Here (b) is the fundamental theorem of calculus. (c) is by Claim 1.

Claim 4. $A(x(1)) \leq (e^\epsilon + \epsilon)\mathbb{1}$.

We have

$$\begin{aligned}\max_i A(x(1)) - \epsilon &\overset{\text{(d)}}{\leq} f(x(1)) - f(x(0)) = \int_0^1 \frac{d}{dt} f(x(t)) \, dt \\ &= \int_0^1 \langle f'(x(t)), x(t)\rangle \, dt \leq \int_0^1 e^\epsilon \, dt = e^\epsilon.\end{aligned}$$

(d) is by Lemma 3.5. (e) is the fundamental theorem of calculus. (f) is the chan rule. (g) is by Claim 2. Rearranging, we have

$$\max_i (A(x(1)))_i \leq e^\epsilon + \epsilon.$$

We also $e^\epsilon \leq 1 + 2\epsilon$ for $\epsilon > 0$ sufficiently small, which gives the claim.

Thus at time $t = 1$, $x(1)$ satisfies $\langle \mathbb{1}, x\rangle \geq (1-\epsilon)\textsc{Opt}(\mathsf{C})$ and $Ax \leq (1+3\epsilon)\mathbb{1}$. This means that

$$\left(\frac{1}{1+3\epsilon}\right) x(\mathbb{1})$$

is a feasible solution to $(\mathsf{P}_{\mathbb{1}})$, with objective value

$$\left\langle \mathbb{1}, \left(\frac{1}{1+3\epsilon}\right) x(\mathbb{1}) \right\rangle \geq \frac{1-\epsilon}{1+3\epsilon}\textsc{Opt}(\mathsf{C}).$$

Finally, for $\epsilon > 0$ sufficiently small, we have

$$\frac{1-\epsilon}{1+3\epsilon} \geq 1 - 5\epsilon.$$

We have now show that the algorithm produces an $(1-\epsilon)$-approximation (up to scaling) to $(\mathsf{P}_{\mathbb{1}})$. It remains to show that the number of iterations is at most $O\left(m \log(m)/\epsilon^2\right)$. Recall that each iteration k, we choose a step size δ_k such that

$$\eta(Az_k)_i = \epsilon$$

for some coordinate $i \in [m]$. That is, each iteration k, we increase $(Ax(t_k))_i$ by ϵ/η for some coordinate i. At the end, we have $(Ax(1))_i \leq 1 + 3\epsilon$ for all i. It follows that we can have at most $O(m/(\epsilon/\eta)) = O\left(m \log(m)/\epsilon^2\right)$ iterations. ■