Nearly-Linear Time Positive LP Solver with Faster Convergence Rate

Zeyuan Allen-Zhu
zeyuan@csail.mit.edu
MIT CSAIL

Lorenzo Orecchia
orecchia@mit.edu
MIT Math

June 14, 2015

Abstract

Positive linear programs (LP), also known as packing and covering linear programs, are an important class of problems that bridges computer science, operation research, and optimization. Efficient algorithms for solving such LPs have received significant attention in the past 20 years [LN93, PST95, BBR97, You01, Nem04, BI04, BBR04, Nes05, CE05, AK08, Nes08, AHK12, KY13, You14, AO15]. Unfortunately, all known nearly-linear time algorithms for producing $(1+\varepsilon)$-approximate solutions to positive LPs have a running time dependence that is at least proportional to ε^{-2}. This is also known as an $O(1/\sqrt{T})$ convergence rate and is particularly poor in many applications.

In this paper, we leverage insights from optimization theory to break this longstanding barrier. Our algorithms solve the packing LP in time $\widetilde{O}(N\varepsilon^{-1})$ and the covering LP in time $\widetilde{O}(N\varepsilon^{-1.5})$. At high level, they can be described as linear couplings of several first-order descent steps. This is the first application of our linear coupling technique (see [AO14]) to problems that are not amenable to blackbox applications known iterative algorithms in convex optimization. Our work also introduces a sequence of new techniques, including the stochastic and the non-symmetric execution of gradient truncation operations, which may be of independent interest.

1 Introduction

A generic packing linear program (LP) takes the form $\max\{c^T x : Ax \le b\}$ where $c \in \mathbb{R}^n_{\ge 0}$, $b \in \mathbb{R}^m_{\ge 0}$, and $A \in \mathbb{R}^{m\times n}_{\ge 0}$; similarly, a generic covering LP can be written as $\min\{b^T y : A^T y \ge c\}$, with the same requirements on A, b, and c. We denote by N the number of non-zero elements in matrix A. They are also known as positive LPs as originally studied by Luby and Nisan [LN93].

We assume without loss of generality that the LP is in its *standard form*: $b = \mathbb{1}$ and $c = \mathbb{1}$.

$$\text{Packing LP:} \qquad \max_{x\ge 0}\{\mathbb{1}^T x : Ax \le \mathbb{1}\} \ , \tag{1.1}$$

$$\text{Covering LP:} \qquad \min_{y\ge 0}\{\mathbb{1}^T y : A^T y \ge \mathbb{1}\} \ . \tag{1.2}$$

Since the two programs are dual to each other, we denote by OPT their shared optimal value. We say that x is a $(1-\varepsilon)$-approximation for the packing LP if $Ax \le \mathbb{1}$ and $\mathbb{1}^T x \ge (1-\varepsilon)\mathsf{OPT}$, and y a $(1+\varepsilon)$-approximation for the covering LP if $A^T y \ge \mathbb{1}$ and $\mathbb{1}^T y \le (1+\varepsilon)\mathsf{OPT}$.

Of course, it is possible to adopt the general Interior Point or Ellipsoid Methods to obtain approximate solvers with a $\log(1/\varepsilon)$ dependence on the number of iterations. However, the computational cost of such algorithms is typically very high, as each iteration requires the solution of a system of linear equations in $A^T A$. As a consequence, this approach is simply not suitable to the solution of large-scale problems. To address this issue, researchers have developed *iterative approximate* solvers that achieve a better dependence on the problem size (e.g., nearly-linear time N) at the cost of having a $\mathsf{poly}(1/\varepsilon)$ dependence on the approximation parameter ε.

Fast approximate packing and covering LP solvers have been widely used in approximation algorithms (e.g., MinSetCover [LN93], MaxSet, MaxDiCut, Max-k-CSP [Tre98], bipartite matching), probabilistic checkable proofs [Tre98], zero-sum matrix games [Nes05], scheduling [PST95], graph embedding [PST95], flow controls [BBR97, BBR04], auction mechanisms [ZN01], wireless sensor networks [BN00], and many other areas. In addition, techniques developed in this line of research have also inspired many other important results, most notably regarding fast algorithms for multi-commodity flow problems [PST95, Fle00, GK07, Mad10, AKR12].

Previous approximate solvers can be further divided into two classes (see Table 1).

Width-Dependent Solvers. These algorithms[1] require a running time that is at least N multiplied with $\rho \cdot \mathsf{OPT}$, where ρ is the largest entry, i.e. the *width*, of matrix A. Since $\mathsf{OPT} \ge 1/\rho$, this value $\rho \cdot \mathsf{OPT}$ is at least 1. However, since OPT can easily be as large as 1 or even more than n, this resulting running time is not polynomial, but only pseudo-polynomial. More precisely, packing and covering LPs can be solved in $O(\frac{N\rho^2\mathsf{OPT}^2\log m}{\varepsilon^2})$ time [PST95], or $O(\frac{N\rho\mathsf{OPT}\log m}{\varepsilon^2})$ time using negative-width techniques [AHK12]. These algorithms strongly rely on multiplicative weight updates and only require "oracle-access" to the matrix A.

When A is given explicitly like in this paper, the number of iterations can be reduced to $O(\frac{\rho\mathsf{OPT}\log m}{\varepsilon})$ by deploying more advanced optimization tools such as Nesterov's accelerated gradient method [Nes05], or Nemirovski's mirror prox method [Nem04]. Bienstock and Iyengar [BI04] have converted this dependence on $\rho\mathsf{OPT}$ into a more benign, yet linear dependence on n. More specifically, their running time is $O(\varepsilon^{-1}N\sqrt{Kn\log m})$ where K is the maximum number of non-zeros per row of A. This is $O(\varepsilon^{-1}Nn\sqrt{\log m})$ in the worst case. The results of [Nes08, CE05] have improved this convergence rate (for packing LP only) to $\widetilde{O}(\varepsilon^{-1}N\sqrt{n})$, but at a cost of enduring an $\widetilde{O}(Nn)$-time preprocessing stage.

[1]Note that most width-dependent solvers are studied under the minmax form of positive LPs, whose optimal value equals $1/\mathsf{OPT}$. Their approximation guarantees are often written in terms of *additive* error. We have translated their performances to multiplicative error for a fair comparison.

[2]d is the maximum number of constraints each variable is in; md may be larger than N.

Paper	Running Time	Width Independent?
[PST95]	$O(N \times \frac{\rho^2 \mathsf{OPT}^2 \log m}{\varepsilon^2})$	no
[AHK12]	$O(N \times \frac{\rho \mathsf{OPT} \log m}{\varepsilon^2})$	no
[Nes05, Nem04]	$O(N \times \frac{\rho \mathsf{OPT} \log m}{\varepsilon})$	no
[BI04]	$O(N \times \frac{\sqrt{Kn} \log m}{\varepsilon})$	no
[Nes08, CE05]: packing LP	$\widetilde{O}(N \times (n + \frac{\sqrt{n}}{\varepsilon}))$	no
[LN93, BBR97, You01, BBR04, AK08, AO15]	$O(N \times \frac{\log^2 N}{\varepsilon^3})$ at best	yes
[You01]	$O((md+N) \times \frac{\log N}{\varepsilon^2})$ 2	yes
[BBR97, BBR04]	$O(nm \times \frac{\log N}{\varepsilon^2})$	yes
[You14]	$O(N \times \frac{\log N}{\varepsilon^2})$	yes
[KY13]	$O(N + (n+m) \times \frac{\log N}{\varepsilon^2})$	yes
[this paper]: packing LP	$O(N \times \frac{\log N \log \varepsilon^{-1}}{\boldsymbol{\varepsilon}})$	yes
[this paper]: covering LP	$O(N \times \frac{\log N \log \varepsilon^{-1}}{\boldsymbol{\varepsilon}^{1.5}})$	yes

Table 1: Comparisons among iterative approximate solvers for packing and covering LPs.

Width-Independent Solvers. In this paper, we are interested in a second, more efficient class of methods, i.e. *width-independent*,[3] truly polynomial-time approximate solvers (see Table 1).

This line of research was initiated by a seminal paper of Luby and Nisan [LN93], who gave an algorithm running in $O\big(\frac{N \log^2 N}{\varepsilon^4}\big)$ time with no dependance on ρ. This is the first *nearly-linear-time* approximate solver for solving packing and covering LPs, and also the first to run in parallel in nearly-linear-work and polylogarithmic depth.

The parallel algorithm of Luby and Nisan was extended by a sequence of works [BBR97, You01, AK08, AO15]s. Most notably, the algorithm of the same authors of this paper [AO15] runs in $O(\frac{\log^2 N}{\varepsilon^3})$ iterations, each costing a matrix-vector multiplication operation that can be implemented in $O(N)$ total work and logarithmic depth.

The ideas of Luby and Nisan also led to sequential width-independent solvers for packing and covering LPs [You01, BBR04, You14, KY13]. Most notably, the algorithm of Koufogiannakis and Young [KY13] runs in time $O\big(N + \frac{\log N}{\varepsilon^2} \times (n+m)\big)$. Despite the amount of work in this area, the $O(1/\varepsilon^2)$ convergence rate has not been improved since 1997. On a separate note, Klein and Young [KY99] have shown that essentially any Dantzig-Wolfe type algorithm has to pay for a $O(1/\varepsilon^2)$ convergence rate. This lack of progress constitutes a significant limitation, as the ε^{-2}-dependence on the approximation parameter ε is particularly pour. This ε^{-2} dependence is also known as the $O(1/\sqrt{T})$ convergence rate in the optimization language, because the error decreases only at the rate $\varepsilon \propto 1/\sqrt{T}$.

1.1 Our Results

Packing LP Solver. We present an algorithm `PacLPSolver` that can be implemented to run in $O(\frac{\log(nm/\varepsilon)\log(1/\varepsilon)}{\varepsilon} N)$ total time. This gives the first nearly-linear time solver for packing LP whose

[3]Some of these solvers may still have a $\mathsf{polylog}(\rho)$ dependence. Since each occurrence of $\log(\rho)$ can typically be replaced with $\log(nm)$ after slightly modifying the instance matrix A, we have done so in Table 1 for a fair comparisons.

running time has an ε^{-1}-dependence; this running time is also known as the $\widetilde{O}(1/T)$ convergence rate in the optimization literature. No nearly-linear time algorithm has achieved any convergence rate that is faster than $O(1/\sqrt{T})$ before our work (see Table 1).

Interestingly, the maximum (weighted) bipartite matching is just one instance of a packing LP. Therefore, our algorithm yields an $\widetilde{O}(m\varepsilon^{-1})$ approximate algorithm and an $\widetilde{O}(m\sqrt{n})$ exact algorithm[4] that arise purely from optimization for bipartite matching, without the use of any dynamic trees. This matches the best known combinatorial algorithms for maximum weighted bipartite matching. Any further improvement over the dependence on ε^{-1} would result in a maximum matching algorithm that runs in time $m \cdot \widetilde{o}(\sqrt{n})$, which may require very significantly different ideas.

Our algorithms optimizes a relaxation of the original packing LP, where the hard constraint $Ax \leq 1$ is replaced by an exponential penalty function for violating the constraint. In other words, we reduce the problem of approximately solving packing LP into approximately minimizing some function $f_\mu(x)$ over the positive orthant $x \geq 0$ —see (2.3). This interpretation of the solution of packing and covering linear programs was recently suggested by the same authors of this paper [AO15]. However, the techniques in our previous work [AO15] only lead to very slow sequential solvers (see Table 1). Furthermore, to the best of our knowledge, our objective $f_\mu(x)$ cannot be turned into any class of smooth functions, and therefore traditional accelerated gradient methods such as [Nes83, Nes05] no longer apply. We thus need fundamentally new ideas.

Our proposed algorithm is an iterative first-order method, and has a flavor of "stochastic coordinate descent" (cf. [ST11, FR13]). Suppose that we are given point $x \geq 0$ at some iteration, and observe the gradient $\nabla f(x) \in [-1, \infty)^n$. Then, we randomly pick a coordinate $i \in [m]$, and focus only on the coordinate gradient $\nabla_i f(x) \in [-1, \infty)$. (In fact, we do not even need to compute $\nabla_\ell f(x)$ for $\ell \neq i$, thus ensuring that each iteration can be implemented very efficiently.)

We divide $\nabla_i f(x) = \eta + \xi$, where $\eta \in [0, \infty)$ is the large component, and $\xi \in [-1, 1]$ is the small (and truncated) component. This *gradient-truncation technique* was developed in our prior work [AO15], but has never been applied to coordinate gradient.

We perform essentially three coordinate descent steps.

- A *gradient (descent) step* with respect to η, guaranteeing a large decrement on the objective.
- A *mirror (descent) step* and a *gradient (descent) step*, both with respect to ξ.

Both gradient and mirror descent are well-known tools from optimization (see for instance [Nes04, BN13], and for starters, mirror descent is a generalization of multiplicative weight updates).[5] Motivated by the linear coupling technique developed in [AO14], we combine the analysis of the above three descent steps for a faster algorithm.

To push through the idea sketched above, we also develop two independent techniques. The *redundant-constraint technique* imposes an additional box constraint; it requires each x_i to be upper bounded by a carefully chosen constant c_i. While this constraint $x_i \leq c_i$ is provably redundant from the viewpoint of minimizing $f_\mu(x)$, it is surprisingly crucial for our linear coupling to work. Our *gradient-mirror scaling* technique restricts our attention to a special type of gradient step, which is always a constant factor of the mirror step. Our two techniques together play an important role in enabling the three descent steps mentioned above to be effectively coupled.

Covering LP Solver. Unlike our most relevant prior work [AO15], it is not clear how one can extract an (approximate) covering LP solution from the packing LP solver mentioned above. There

[4]It is not hard to turn an $\widetilde{O}(m\varepsilon^{-1})$ approximate algorithm into an $\widetilde{O}(m\sqrt{n})$ algorithm, see for instance [DP14].

[5]It is important to note here that we have generalized the notion of "gradient descent" to indicate any descent step that is guaranteed to decrease the objective. This is in contrast to mirror descent, which is a "dual approach" that does not necessarily decrease the objective at any iteration, but minimizes the so-called regularized regret.

are at least two main issues behind this difficulty. Firstly, the dual guarantee naturally arising from `PacLPSolver` is on the history of the *full* gradients $\nabla f(x_k)$, rather than the randomly selected coordinate gradients $\nabla_i f(x_k)$, over all iterations k. As we mentioned earlier, it is computationally heavy to compute full gradients. Secondly, even if the dual guarantee is on the coordinate gradients $\nabla_i f(x_k)$, it is not clear how one can compute them efficiently in only nearly-linear time.

We therefore are forced to design a new algorithm `CovLPSolver` that works directly for covering LP. On one hand, this new algorithm relies on similar idea that are present in `PacLPSolver`: the linear coupling of gradient and mirror steps and the gradient truncation. On the other hand, we need a different version of the redundant-constraint technique (over a simplex constraint), as well as a negative-width technique.

Our `CovLPSolver` can be implemented to run in $O(\frac{\log(nm/\varepsilon)\log(1/\varepsilon)}{\varepsilon^{1.5}}N)$ total time. This gives the first nearly-linear time solver for covering LP whose running time has a faster dependence than ε^{-2} (or equivalently, the first one whose convergence rate is faster than $\widetilde{O}(1/\sqrt{T})$).

Leveraging the Optimization Viewpoint. Our optimization approach to solving packing and covering LPs is yet another example on *designing algorithms based on insights from optimization*. Before our work, the updates on x for all sequential algorithms were combinatorial in flavor. For instance, the algorithm of Young [You01] updates each coordinate of x_i in a maximally aggressive way, so that one of the constraints becomes tight. The optimization interpretation behind our two algorithms allows us to use more general steps and facilitates the analysis of the algorithm.

1.2 Roadmap

We transfer the packing LP problem into an optimization question in Section 2, and provide our packing LP solver in Section 3. We sketch the main ideas needed for our covering LP solver in Section 4, and defer the technical details to the appendix. Note that our `PacLPSolver` and `CovLPSolver` are stated in an implicit optimization language, and their (efficient) implementation details will be addressed in Appendix F and Appendix G.

2 Relaxation of the Packing Linear Program

Recall that, for input matrix $A \in \mathbb{R}_{\geq 0}^{m\times n}$, the packing LP in its standard form is $\max_{x\geq 0}\{\mathbb{1}^T x : Ax \leq \mathbb{1}\}$. Let us denote by OPT the optimal value of this linear program, and x^* any optimal solution. We say that x is a $(1-\varepsilon)$-approximation for the packing LP if $Ax \leq \mathbb{1}$ and $\mathbb{1}^T x \geq (1-\varepsilon)\mathsf{OPT}$.

Throughout this paper, we use the indices $i \in [n]$ to denote the columns of A, and the indices $j \in [m]$ to denote the rows of A. We let $A_{:i}$ be the i-th column vector of A, and $A_{j:}$ the j-th row vector of A. Given any vector x, we denote by $\|x\|_A = \sqrt{\sum_{i\in[n]} x_i^2 \cdot \|A_{:i}\|_\infty}$ the A-norm of x.

By scaling the matrix A and the optimum value, we can assume without loss of generality that

$$\min_{i\in[n]}\{\|A_{:i}\|_\infty\} = 1 \ . \tag{2.1}$$

We can now restrict the range of values x and OPT can take. The following is proved in Appendix A.

Fact 2.1. *Define the bounding box* $\Delta \overset{\text{def}}{=} \{x \in \mathbb{R}^n : x_i \in \big[0, \frac{1}{\|A_{:i}\|_\infty}\big]\}$. *Under assumption (2.1), we have* $\mathsf{OPT} \in [1, n]$ *and* $\{x : x \geq 0 \wedge Ax \leq \mathbb{1}\} \subseteq \Delta$.

This bounding-box constraint allows us to optimize over a bounded set for x.

Smoothed Objective. We now introduce the smoothed objective $f_\mu(x)$ that we minimize over Δ in order to approximately solve the packing LP. This objective $f_\mu(x)$ turns each row of the non-smooth LP constraint $Ax \leq \mathbb{1}$ into an exponential penalty function so that we only need to require $x \in \Delta$ throughout the algorithm. More formally, the packing LP can be written as the following minimizaton problem by introducing the Lagrangian variable $y \in \mathbb{R}^m$:

$$\min_{x\in\Delta} \; -\mathbb{1}^T x \; + \; \max_{y\geq 0}\{y^T A x - \mathbb{1}^T y\} \; . \tag{2.2}$$

The problem can be now smoothened by introducing a strongly concave regularizer over $y \geq 0$.

This is regularizer is usually taken to be the entropy function over all possible $y \geq 0$ satisfying $\mathbb{1}^T y = 1$, which yields the width-independent solvers in for instance [Nes05] and [Nem04], and is closely related to that of the multiplicative weight update in [AHK12].

In this paper, we take this regularizer to be the generalized entropy $H(y) = -\sum_{j=1}^m y_j \log y_j + y_j$ over the first orthant $y \geq 0$, and minimize the following smoothened objective $f_\mu(x)$ over $x \in \Delta$:

$$f_\mu(x) \stackrel{\text{def}}{=} -\mathbb{1}^T x \; + \; \max_{y\geq 0}\{y^T A x - \mathbb{1}^T y + \boxed{\mu \cdot H(y)}\} \; . \tag{2.3}$$

Above, $\mu > 0$ is some smoothing parameter to be chosen later. By explicitly computing the maximization over $y \geq 0$, $f_\mu(x)$ can be rewritten as

Lemma 2.2. $f_\mu(x) = \mu \sum_{j=1}^m \exp^{\frac{1}{\mu}((Ax)_j - 1)} - \mathbb{1}^T x$.

We wish to study the *minimization* problem on $f_\mu(x)$ over $x \in \Delta$. Intuitively $f_\mu(x)$ captures the original packing LP (1.1) as follows. Firstly, since we want to maximize $\mathbb{1}^T x$, the negative term $-\mathbb{1}^T x$ shows up in $f_\mu(x)$. Secondly, if a packing constraint $j \in [m]$ is violated by ε, that is, $(Ax)_j \geq 1 + \varepsilon$, the exponential penalty in $f_\mu(x)$ introduces a penalty at least $\exp^{\varepsilon/\mu}$; this will be a large penalty if $\mu \leq O(\varepsilon/\log n)$. Notice that this smoothed objective also appeared in previous works [AO15], albeit without this smoothening interpretation and without the constraint $x \in \Delta$.

The regularization of Lemma 2.2 will give us both some smoothness properties for $f_\mu(x)$, discussed in Lemma 2.6, and a regularization error, as we are now solving an objective different from our original packing LP. This error is quantified in the following lemma for our choice of μ. This follows a similar treatment in a previous paper of the authors [AO15] and is proved in Appendix A.

Proposition 2.3. *Let $\mu = \frac{\varepsilon}{4\log(nm/\varepsilon)}$ and x^* be an optimal solution for the packing LP (1.1). Then:*

(a) $f_\mu(u^*) \leq -(1-\varepsilon)\mathsf{OPT}$ for $u^* \stackrel{\text{def}}{=} (1-\varepsilon/2)x^* \in \Delta$.

(b) $f_\mu(x) \geq -(1+\varepsilon)\mathsf{OPT}$ for every $x \in \Delta$.

(c) If $x \in \Delta$ satisfies $f_\mu(x) \leq -(1-O(\varepsilon))\mathsf{OPT}$, then $\frac{1}{1+\varepsilon}x$ is a $(1-O(\varepsilon))$-approximate solution to the packing LP.

In short, they together imply that the minimum of $f_\mu(x)$ is around $-\mathsf{OPT}$, and if one can approximately find the minimum of $f_\mu(x)$, up to a multiplicative error $1 \pm O(\varepsilon)$, this corresponds to a $(1-O(\varepsilon))$-approximate solution to the packing LP (1.1).

Remark 2.4. *We emphasize that our constraint $x_i \leq \frac{1}{\|A_{:i}\|_\infty}$ is essentially redundant from the viewpoint of minimizing $f_\mu(x)$: whenever $x \geq 0$ and $f_\mu(x) \leq 0$, one should automatically have $x_i \leq \frac{1+\varepsilon}{\|A_{:i}\|_\infty}$. However, this redundant constraint shall become very crucial at the point we analyze the mirror-descent component our algorithm; after all, mirror descent steps do not necessarily decrease the objective, and thus may not guarantee $f_\mu(x) \leq 0$.*

Smoothness properties. Thanks to the smoothing of Lemma 2.2 and the choice of regularizer, our objective $f_\mu(x)$ enjoys a number of good smoothness properties. First, it is differentiable and the gradient is easy to compute:

Fact 2.5. $\nabla f_\mu(x) = A^T p(x) - \mathbb{1}$ *where* $p_j(x) \stackrel{\text{def}}{=} \exp^{\frac{1}{\mu}((Ax)_j - 1)}$.

Second, $f_\mu(x)$ enjoys two kinds of coordinate-wise smoothness properties in different regimes. These will be extremely useful in applying gradient descent arguments in Section 3.2, and are the

Algorithm 1 $\texttt{PacLPSolver}(A, x^{\mathsf{start}}, \varepsilon)$

```
Input: $A \in \mathbb{R}_{\geq 0}^{m\times n}, x^{\mathsf{start}} \in \Delta, \varepsilon \in (0, 1/10]$.
Output: $x \in \Delta$.
 1: $\mu \leftarrow \frac{\varepsilon}{4\log(nm/\varepsilon)}$, $L \leftarrow \frac{4}{\mu}$, $\tau \leftarrow \frac{1}{3\cdot nL}$ and $\alpha_0 \leftarrow \frac{1}{nL}$.      ▷ parameters
 2: $T \leftarrow \lceil 3nL\log(1/\varepsilon)\rceil = O(n \cdot \frac{\log(nm/\varepsilon)\cdot\log(1/\varepsilon)}{\varepsilon})$.      ▷ number of iterations
 3: $\mathsf{x}_0 = \mathsf{y}_0 \leftarrow x^{\mathsf{start}}$, $\mathsf{z}_0 \leftarrow 0$.
 4: for $k \leftarrow 1$ to $T$ do
 5:     $\alpha_k \leftarrow \frac{1}{1-\tau}\,\alpha_{k-1}$
 6:     $\mathsf{x}_k \leftarrow \tau \mathsf{z}_{k-1} + (1-\tau)\mathsf{y}_{k-1}$.
 7:     Randomly select $i \in [n]$ uniformly at random.
 8:     Define the vector $\xi_k^{(i)}$ to be all-zero except at coordinate $i$, where it equals $\mathbb{T}^{\mathsf{p}}(\nabla_i f_\mu(\mathsf{x}_k))$.
 9:     $\mathsf{z}_k \leftarrow \mathsf{z}_k^{(i)} \stackrel{\text{def}}{=} \arg\min_{z\in\Delta}\big\{\frac{1}{2}\|z-\mathsf{z}_{k-1}\|_A^2 + \langle n\alpha_k\xi_k^{(i)}, z\rangle\big\}$.      ▷ See Proposition 3.6
10:     $\mathsf{y}_k \leftarrow \mathsf{y}_k^{(i)} \stackrel{\text{def}}{=} \mathsf{x}_k + \frac{1}{n\alpha_k L}(\mathsf{z}_k^{(i)} - \mathsf{z}_{k-1})$.
11: end for
12: return $\mathsf{y}_T$.
```

main motivation for us to adopt the $\|\cdot\|_A$ norm for our proposed algorithms. The proof appears in Appendix A and it is a simple manipulation of the Hessian.

Lemma 2.6. *Define the smoothness parameter* $L \stackrel{\text{def}}{=} \frac{4}{\mu}$. *Then, for every* $x \in \Delta$, *and every* $i \in [n]$*:*

(a) If $|\nabla_i f_\mu(x)| \leq 1$, *then for all* $\lambda \leq \frac{1}{L\|A_{:i}\|_\infty}$, *we have* $\big|\nabla_i f_\mu(x+\lambda\mathbf{e}_i) - \nabla_i f_\mu(x)\big| \leq L\|A_{:i}\|_\infty\cdot|\lambda|$.

(b) If $|\nabla_i f_\mu(x)| \geq 1$, *then for all* $\lambda \leq \frac{1}{L\|A_{:i}\|_\infty}$, *we have* $\nabla_i f_\mu(x+\lambda\mathbf{e}_i) \geq \Big(1 - \frac{\|A_{:i}\|_\infty L}{2}|\lambda|\Big)\nabla_i f_\mu(x)$.

The first property is the same as the traditional (coordinate) Lipschitz-smoothness property, i.e. the Lipschitz continuity of the (coordinate) gradient $\nabla_i f(x)$, but holds only conditionally and not for all $x \geq 0$. The second property is a salient characteristic of this work and requires the positivity of A. It can be seen as a formalization of the "multiplicative Lipschitz" property used in our previous work [AO15].

Initialization. Iterative methods require the choice of a good starting point. We have

Fact 2.7. *Defining* $x_i^{\mathsf{start}} \stackrel{\text{def}}{=} \frac{1-\varepsilon/2}{n\|A_{:i}\|_\infty}$ *for for each* $i \in [n]$, *we have* $x^{\mathsf{start}} \in \Delta$ *and* $f_\mu(x^{\mathsf{start}}) \leq -\frac{1-\varepsilon}{n}$.

3 Packing LP Solver

To describe our algorithm, we first make the following choice of thresholding function

Definition 3.1. *The thresholding function* $\mathbb{T}^{\mathsf{p}}\colon [-1,\infty) \to [-1,1]$ *is defined as follows*

$$\mathbb{T}^{\mathsf{p}}(v) \stackrel{\text{def}}{=} \begin{cases} v, & v \in [-1,1]; \\ 1, & v > 1. \end{cases}$$

Our algorithm $\texttt{PacLPSolver}$ starts with some initial vector $\mathsf{x}_0 = \mathsf{y}_0 = x^{\mathsf{start}}$ (introduced in Fact 2.7) and $\mathsf{z}_0 = 0$, and is divided into T iterations. In each iteration, we start by computing a weighted midpoint $\mathsf{x}_k \leftarrow \tau\mathsf{z}_{k-1} + (1-\tau)\mathsf{y}_{k-1}$ for some parameter $\tau \in (0,1)$, and then proceed to compute y_k and z_k as follows.

- Select $i \in [n]$ uniformly at random, and let $\xi_k^{(i)} = (0,\dots,0,\mathbb{T}^{\mathsf{p}}(v),0,\dots,0)$ be the vector that is only non-zero at coordinate i, where $v = \nabla_i f_\mu(\mathsf{x}_k) = \sum_{j=1}^m A_{j,i}\exp^{\frac{1}{\mu}((A\mathsf{x}_k)_j-1)} - 1 \in [-1,\infty)$.
- Perform a *mirror (descent) step* $\mathsf{z}_k \leftarrow \mathsf{z}_k^{(i)} \stackrel{\text{def}}{=} \arg\min_{z\in\Delta}\big\{\frac{1}{2}\|z-\mathsf{z}_{k-1}\|_A^2 + \langle n\alpha_k\xi_k^{(i)}, z\rangle\big\}$ for some parameter $\alpha_k \ll 1/n$ to be chosen later.

- Perform a *gradient (descent) step* $\mathsf{y}_k \leftarrow \mathsf{y}_k^{(i)} \overset{\text{def}}{=} \mathsf{x}_k + \frac{1}{n\alpha_k L}(\mathsf{z}_k^{(i)} - \mathsf{z}_{k-1})$.

Above, the reason that the the two steps on y_k and z_k are named after "gradient step" and "mirror step" will become clear in the follow-up sections. We use the superscript $^{(i)}$ on $\xi_i^{(i)}$, $\mathsf{y}_k^{(i)}$ and $\mathsf{z}_k^{(i)}$ to emphasize that the value depends on the choice of i. We have used generic parameters τ, α_k, T in the above description and their precise values are presented in Algorithm 1. [6]

For readers familiar with optimization tools, the above triple sequence $\{\mathsf{x}_k, \mathsf{y}_k, \mathsf{z}_k\}_k$ is reminiscent of Nesterov's accelerated gradient method [Nes05]. However, our algorithm is not an instance of any variant of the known accelerated gradient method. (This is so because, for instance, our objective $f_\mu(x)$ is not globally Lipschitz smooth.)

In fact, our algorithm `PacLPSolver` is strongly motivated by our linear-coupling technique introduced in [AO14], a technique that allows one to linearly combine gradient and mirror steps for a better performance. This linear coupling requires one to use a triple sequence $\{\mathsf{x}_k, \mathsf{y}_k, \mathsf{z}_k\}_k$.

We emphasize here that our iterates $\mathsf{x}_k, \mathsf{y}_k, z_k$ never leave the bounding box Δ:

Lemma 3.2. *We have* $\mathsf{x}_k, \mathsf{y}_k, \mathsf{z}_k \in \Delta$ *for all* $k = 0, 1, \dots, T$.

The proof of Lemma 3.2 is deferred to Appendix B, and crucially relies on the fact that our gradient and mirror steps are multiples of each other: $\mathsf{y}_k^{(i)} - \mathsf{x}_k = \frac{1}{n\alpha_k L}(\mathsf{z}_k^{(i)} - \mathsf{z}_{k-1})$. The key idea of this lemma was also known by Fercoq and Richtárik [FR13].

We shall also prove in Section F that

Lemma 3.3. *Each iteration of* `PacLPSolver` *can be implemented to run in expected* $O(N/n)$ *time.*

The key idea used in the implementation is to compute x_k and y_k only *implicitly.* For instance, explicitly maintaining x_k and computing $p(x_k)$ require $O(N)$ time per iteration, but representing x_k implicitly as a linear combination of two less-frequently-modified vectors reduces it to $O(N/n)$.

In this section, we shall prove the following theorem in three steps.

Theorem 3.4. `PacLPSolver`$(A, x^{\mathsf{start}}, \varepsilon)$ *outputs some* y_T *satisfying* $\mathbf{E}[f_\mu(\mathsf{y}_T)] \le -(1-3\varepsilon)\mathsf{OPT}$.

3.1 Step 1: Mirror Descent Guarantee

Since our update $\mathsf{z}_k^{(i)} = \arg\min_{z\in\Delta} \left\{\frac{1}{2}\|z - \mathsf{z}_{k-1}\|_A^2 + \langle n\alpha_k \xi_k^{(i)}, z\rangle\right\}$ —see Line 9 of `PacLPSolver`— is written in the form of a *mirror descent step* from optimization, the following inequality is a classical upper bound on the "regret" of mirror descent. Its proof can be found in Appendix B.

Lemma 3.5. $\langle n\alpha_k \xi_k^{(i)}, \mathsf{z}_{k-1} - u\rangle \le n^2\alpha_k^2 L \cdot \langle \xi_k^{(i)}, \mathsf{x}_k - \mathsf{y}_k^{(i)}\rangle + \frac{1}{2}\|\mathsf{z}_{k-1} - u\|_A^2 - \frac{1}{2}\|\mathsf{z}_k^{(i)} - u\|_A^2$.

Although defined in a variational way, it is perhaps beneficial to explicitly describe how to implement this mirror step. Its proof is straightforward but can be found in Appendix B.

Proposition 3.6. *If* $\mathsf{z}_{k-1} \in \Delta$, *the minimizer* $z = \arg\min_{z\in\Delta}\left\{\frac{1}{2}\|z - \mathsf{z}_{k-1}\|_A^2 + \langle \delta \mathbf{e}_i, z\rangle\right\}$ *for any scalar* $\delta \in \mathbb{R}$ *and basis vector* $\mathbf{e}_i$ *can be computed as follows:*

1. $z \leftarrow \mathsf{z}_{k-1}$.
2. $z_i \leftarrow z_i - \delta/\|A_{:i}\|_\infty$.
3. *If* $z_i < 0$, *then* $z_i \leftarrow 0$; *if* $z_i > 1/\|A_{:i}\|_\infty$, $z_i \leftarrow 1/\|A_{:i}\|_\infty$.
4. *Return* z.

As a simple corollary, we have the following fact

[6]We encourage the readers to ignore their specific values for now. Our specific choices of the parameters shall become clearer and natural at the end of this section, and be discussed whenever they are used.

Fact 3.7. *We have* $|\mathsf{z}^{(i)}_{k,i}-\mathsf{z}_{k-1,i}| \leq \frac{n\alpha_k|\xi^{(i)}_{k,i}|}{\|A_{:i}\|_\infty}$ *and* $|\mathsf{y}^{(i)}_{k,i}-\mathsf{x}_{k,i}| = \frac{1}{n\alpha_k L}|\mathsf{z}^{(i)}_{k,i}-\mathsf{z}_{k-1,i}| \leq \frac{|\xi^{(k)}_{k,i}|}{L\|A_{:i}\|_\infty} \leq \frac{1}{L\|A_{:i}\|_\infty}$.

3.2 Step 2: Gradient Descent Guarantee

We call our update rule $\mathsf{y}^{(i)}_k \leftarrow \mathsf{x}_k + \frac{1}{n\alpha_k L}(\mathsf{z}^{(i)}_k - \mathsf{z}_{k-1})$ a gradient descent step, because the following lemma guarantees $f_\mu(\mathsf{y}^{(i)}_k) \leq f_\mu(\mathsf{x}_k)$, that is, the objective only decreases; moreover, the objective decreases at least by $\frac{1}{2}\langle \nabla f_\mu(\mathsf{x}_k), \mathsf{x}_k - \mathsf{y}^{(i)}_k\rangle$.

Lemma 3.8. *We have* $f_\mu(\mathsf{x}_k) - f_\mu(\mathsf{y}^{(i)}_k) \geq \frac{1}{2}\langle \nabla f_\mu(\mathsf{x}_k), \mathsf{x}_k - \mathsf{y}^{(i)}_k\rangle$. *In particular, this implies* $f_\mu(\mathsf{x}_k) \geq f_\mu(\mathsf{y}^{(i)}_k)$ *because* $\nabla_i f_\mu(\mathsf{x}_k)$ *and* $\mathsf{x}_{k,i} - \mathsf{y}^{(i)}_{k,i}$ *have the same sign, while* $\mathsf{x}_{k,\ell} = \mathsf{y}^{(i)}_{k,\ell}$ *for* $\ell \neq i$.

Proof. Note that $\mathsf{y}^{(i)}_k = \mathsf{x}_k + \lambda \mathbf{e}_i$ for some step length λ such that $|\lambda| \leq \frac{1}{L\|A_{:i}\|_\infty}$ according to Fact 3.7. We first prove this lemma in the case of $\nabla_i f_\mu(\mathsf{x}_k) \in [-1,1]$ so that $\xi^{(i)}_{k,i} = \nabla_i f_\mu(\mathsf{x}_k)$.

$$\begin{aligned}
f_\mu(\mathsf{x}_k) - f_\mu(\mathsf{y}^{(i)}_k) &= f_\mu(\mathsf{x}_k) - f_\mu(\mathsf{x}_k + \lambda\mathbf{e}_i) = -\int_0^\lambda \Big(\nabla_i f_\mu(\mathsf{x}_k + \chi\mathbf{e}_i)\Big)d\chi \\
&\overset{①}{\geq} \int_0^\lambda \Big(-\nabla_i f_\mu(\mathsf{x}_k) - L\|A_{:i}\|_\infty \cdot |\chi|\Big)d\chi = -\nabla_i f_\mu(\mathsf{x}_k)\cdot|\lambda| - \frac{L\|A_{:i}\|_\infty}{2}\cdot\lambda^2 \\
&\overset{②}{\geq} -\nabla_i f_\mu(\mathsf{x}_k)\cdot|\lambda| - \frac{L\|A_{:i}\|_\infty}{2}\cdot|\lambda|\cdot\frac{|\xi^{(k)}_{k,i}|}{L\|A_{:i}\|_\infty} = -\frac{1}{2}\langle \nabla f_\mu(\mathsf{x}_k), \mathsf{y}^{(i)}_k - \mathsf{x}_k\rangle \ .
\end{aligned}$$

Above, ① uses Lemma 2.6.a, and ② uses Fact 3.7.

Next, we turn to the case of $\nabla_i f_\mu(\mathsf{x}_k) > 1$.

$$\begin{aligned}
f_\mu(\mathsf{x}_k) - f_\mu(\mathsf{y}^{(i)}_k) &= f_\mu(\mathsf{x}_k) - f_\mu(\mathsf{x}_k + \lambda\mathbf{e}_i) = -\int_0^\lambda \nabla_i f_\mu(\mathsf{x}_k + \chi\mathbf{e}_i)d\chi \\
&\overset{①}{\geq} \int_0^\lambda \Big(1 - \frac{\|A_{:i}\|_\infty L}{2}|\chi|\Big)\nabla_i f_\mu(x)d\chi \overset{②}{\geq} \int_0^\lambda \frac{1}{2}\nabla_i f_\mu(x)d\chi = \frac{1}{2}\langle \nabla f_\mu(\mathsf{x}_k), \mathsf{x}_k - \mathsf{y}^{(i)}_k\rangle \ .
\end{aligned}$$

Above, ① uses Lemma 2.6.b and ② uses $|\chi| \leq |\lambda| \leq \frac{1}{L\|A_{:i}\|_\infty}$. □

3.3 Step 3: Putting All Together

In the following, we denote by $\eta^{(i)}_k \in \mathbb{R}^n_{\geq 0}$ the vector that is only non-zero at coordinate i, and satisfies $\eta^{(i)}_{k,i} = \nabla_i f_\mu(\mathsf{x}_k) - \xi^{(i)}_{k,i} \in [0,\infty)$. In other words, the full gradient

$$\nabla f_\mu(\mathsf{x}_k) = \mathbf{E}_i[n\nabla_i f_\mu(\mathsf{x}_k)] = \mathbf{E}_i[n\eta^{(i)}_k + n\xi^{(i)}_k]$$

can be (in expectation) decomposed into the a large but non-negative component $\eta^{(i)}_k \in [0,\infty)^n$ and a small component $\xi^{(i)}_k \in [-1,1]^n$. Recall that $\eta^{(i)}_k$ is the part of the gradient that was truncated, and did not contribute to the mirror step (see Line 9 of `PacLPSolver`). Next, for any $u \in \Delta$, we can use a basic convexity argument and the mirror descent lemma to compute that

$$\begin{aligned}
&\alpha_k(f_\mu(\mathsf{x}_k) - f_\mu(u)) \leq \big\langle \alpha_k \nabla f_\mu(\mathsf{x}_k), \mathsf{x}_k - u\big\rangle \\
&= \big\langle \alpha_k \nabla f_\mu(\mathsf{x}_k), \mathsf{x}_k - \mathsf{z}_{k-1}\big\rangle + \big\langle \alpha_k \nabla f_\mu(\mathsf{x}_k), \mathsf{z}_{k-1} - u\big\rangle \\
&= \big\langle \alpha_k \nabla f_\mu(\mathsf{x}_k), \mathsf{x}_k - \mathsf{z}_{k-1}\big\rangle + \mathbf{E}_i\Big[\big\langle n\alpha_k\eta^{(i)}_k, \mathsf{z}_{k-1} - u\big\rangle + \big\langle n\alpha_k\xi^{(i)}_k, \mathsf{z}_{k-1} - u\big\rangle\Big] \\
&\overset{①}{=} \frac{(1-\tau)\alpha_k}{\tau}\big\langle \nabla f_\mu(\mathsf{x}_k), \mathsf{y}_{k-1} - \mathsf{x}_k\big\rangle + \mathbf{E}_i\Big[\big\langle n\alpha_k\eta^{(i)}_k, \mathsf{z}_{k-1} - u\big\rangle + \big\langle n\alpha_k\xi^{(i)}_k, \mathsf{z}_{k-1} - u\big\rangle\Big] \qquad (3.1) \\
&\overset{②}{\leq} \frac{(1-\tau)\alpha_k}{\tau}(f_\mu(\mathsf{y}_{k-1}) - f_\mu(\mathsf{x}_k))
\end{aligned}$$

$$+\mathbf{E}_i\Big[\boxed{\big\langle n\alpha_k\eta_k^{(i)}, \mathsf{z}_{k-1}-u\big\rangle + n^2\alpha_k^2 L\cdot\big\langle \xi_k^{(i)}, \mathsf{x}_k-\mathsf{y}_k^{(i)}\big\rangle} + \frac{1}{2}\|\mathsf{z}_{k-1}-u\|_A^2 - \frac{1}{2}\|\mathsf{z}_k^{(i)}-u\|_A^2\Big] \quad (3.2)$$

Above, ① is because $\mathsf{x}_k = \tau\mathsf{z}_{k-1} + (1-\tau)\mathsf{y}_{k-1}$, which implies that $\tau(\mathsf{x}_k - \mathsf{z}_{k-1}) = (1-\tau)(\mathsf{y}_{k-1} - \mathsf{x}_k)$. ② uses convexity and Lemma 3.5. This above computation is motivated by [AO14], and as we shall see below, it allows one to linearly couple gradient and mirror steps.

Intuitively, the first (non-negative) term in the box of (3.2) is the loss introduced by the large gradient $\eta_k^{(i)}$. This part was truncated so did not contribute to the mirror step. The second (non-negative) term in the box is the loss introduced by mirror descent on the small gradient $\xi_k^{(i)}$.

Now comes an important observation. As shown by Lemma 3.9 below, the performance of the gradient step —that is, the objective decrease of $f_\mu(\mathsf{x}_k) - f_\mu(\mathsf{y}_k^{(i)})$— is at least proportional to the loss incurred in the box.

Lemma 3.9. $\big\langle n\alpha_k\eta_k^{(i)}, \mathsf{z}_{k-1}-u\big\rangle + n^2\alpha_k^2 L\cdot\big\langle \xi_k^{(i)}, \mathsf{x}_k-\mathsf{y}_k^{(i)}\big\rangle \leq 3n\alpha_k L\cdot\big(f_\mu(\mathsf{x}_k) - f_\mu(\mathsf{y}_k^{(i)})\big)$.

Since the proof of the above lemma is a careful case analysis and several simple applications of Lemma 3.8, we defer it to Appendix B. We make two important remarks.

- First, Lemma 3.9 is why we stated in the introduction that our `PacLPSolver` incorporates *two* gradient steps: one with respect to $\eta_k^{(i)}$ and one with respect to $\xi_k^{(i)}$. We have intentionally forced the two steps to be identical, in order to present our algorithm more cleanly.[7]
- Second, to properly upper bound $\langle n\alpha_k\eta_k^{(i)}, \mathsf{z}_{k-1}-u\rangle$, one needs to have some good upper bound the coordinates of z_{k-1}. This is exactly the place we need our redundant-constraint technique, which guarantees that each $\mathsf{z}_{k-1,i} \leq \frac{1}{\|A_{:i}\|_\infty}$.

Plugging the above lemma into (3.2), we have

$$\begin{aligned}
&\alpha_k(f_\mu(\mathsf{x}_k) - f_\mu(u)) \leq \langle \alpha_k\nabla f_\mu(\mathsf{x}_k), \mathsf{x}_k - u\rangle \\
&\overset{①}{\leq} \frac{(1-\tau)\alpha_k}{\tau}(f_\mu(\mathsf{y}_{k-1}) - f_\mu(\mathsf{x}_k)) + \mathbf{E}_i\Big[3n\alpha_k L\cdot(f_\mu(\mathsf{x}_k) - f_\mu(\mathsf{y}_k^{(i)})) + \frac{1}{2}\|\mathsf{z}_{k-1}-u\|_A^2 - \frac{1}{2}\|\mathsf{z}_k-u\|_A^2\Big] \\
&\overset{②}{\leq} \alpha_k f_\mu(\mathsf{x}_k) + \big(3n\alpha_k L - \alpha_k\big)f_\mu(\mathsf{y}_{k-1}) + \mathbf{E}_i\Big[-3n\alpha_k L\cdot f_\mu(\mathsf{y}_k^{(i)}) + \frac{1}{2}\|\mathsf{z}_{k-1}-u\|_A^2 - \frac{1}{2}\|\mathsf{z}_k-u\|_A^2\Big] \ . \quad (3.3)
\end{aligned}$$

Above, ① is because we have chosen α_k so that $n\alpha_k \leq n\alpha_T = \frac{1}{\varepsilon L} \leq \frac{1}{4}$; and ② is because we have chosen τ to satisfy $\frac{1}{\tau} = 3nL$.

Next, recall that we have picked α_k so that $(3nL-1)\alpha_k = 3nL\cdot\alpha_{k-1}$ in Algorithm 1. Telescoping (3.3) for $k = 1, \ldots, T$ and choosing $u^* = (1-\varepsilon/2)x^*$, we have

$$-\textstyle\sum_{k=1}^T \alpha_k f_\mu(u^*) \leq 3f_\mu(\mathsf{y}_0) - 3n\alpha_T L\cdot\mathbf{E}[f_\mu(\mathsf{y}_T)] + \|\mathsf{z}_0 - u^*\|_A^2 \leq -3n\alpha_T L\cdot\mathbf{E}[f_\mu(\mathsf{y}_T)] + \mathsf{OPT} \ .$$

Here, the second inequality is due to $f_\mu(\mathsf{y}_0) = f_\mu(x^{\mathsf{start}}) \leq 0$ from Fact 2.7, and the fact that

$$\|\mathsf{z}_0 - u^*\|_A^2 = \|u^*\|_A^2 = \textstyle\sum_{i=1}^n (u_i^*)^2\cdot\|A_{:i}\|_\infty \leq \sum_{i=1}^n (x_i^*)^2\cdot\|A_{:i}\|_\infty \leq \sum_{i=1}^n x_i^* = \mathsf{OPT} \ .$$

Finally, using the fact that $\sum_{k=1}^T \alpha_k = \alpha_T\cdot\sum_{k=0}^{T-1}\big(1-\frac{1}{3nL}\big)^k = 3n\alpha_T L\big(1-(1-\frac{1}{3nL})^T\big)$, we rearrange and obtain that

$$\mathbf{E}[f_\mu(\mathsf{y}_T)] \leq \frac{\sum_k \alpha_k}{3n\alpha_T L}f_\mu(u^*) + \frac{1}{3n\alpha_T L}\mathsf{OPT} = \big(1-(1-\frac{1}{3nL})^T\big)f_\mu(u^*) + \frac{1}{3n\alpha_T L}\mathsf{OPT} \ .$$

Choosing $T = \lceil 3nL\log(1/\varepsilon)\rceil$ so that $\frac{1}{n\alpha_T L} = (1-\frac{1}{3nL})^T \leq \varepsilon$. Combining this with the fact that $f_\mu(u^*) \leq -(1-\varepsilon)\mathsf{OPT} < 0$ (see Proposition 2.3.a), we obtain

$$\mathbf{E}[f_\mu(\mathsf{y}_T)] \leq (1-\varepsilon)f_\mu(u^*) + \varepsilon/3\cdot\mathsf{OPT} < -(1-3\varepsilon)\mathsf{OPT} \ .$$

Therefore, we have finished proving Theorem 3.4. □

[7]One can in fact separate the two gradient steps as $\mathsf{x}_k \to \mathsf{y}_k$ and $\mathsf{x}_k \to \mathsf{y}'_k$, but that will make the algorithm description only more involved.

It is now straightforward (but anyways proved in Appendix B) to use Markov inequality to turn the expected guarantee in Theorem 3.4 into a probabilistic one:

Corollary 3.10. *With probability at least* $9/10$, `PacLPSolver`$(A, x^{\mathsf{start}}, \varepsilon)$ *outputs a* $(1-O(\varepsilon))$ *approximate solution to the packing LP program. The expected running time is* $O(\frac{\log(nm/\varepsilon)\log(1/\varepsilon)}{\varepsilon}N)$.

4 Sketching the Main Ideas for Our Covering LP Solver

For the reasons stated in the introduction, we are forced to build a covering LP solver from scratch, rather than implicitly from `PacLPSolver`. We begin with a similar relaxation of the covering LP (1.2). That is, we show in Appendix C that it suffices to minimize

$$f_\mu(x) \stackrel{\text{def}}{=} \mu \textstyle\sum_{j=1}^m \exp^{\frac{1}{\mu}(1-(Ax)_j)} + \mathbb{1}^T x$$

over all $x \geq 0$. For technical reasons, this objective is much harder to work with than that of (2.3), because its gradient $\nabla f_\mu(x) \in (-\infty, 1]^n$ may be very negative. (This is why our prior work [AO15] intentionally avoided to solve covering LP directly.)

This time, we again pick a random coordinate $i \in [n]$ at each iteration, and then decompose $\nabla_i f(\mathsf{x}_k) = \xi + \eta$. Quite different from `PacLPSolver`, we define $\eta \in (-\infty, 0]$ to be the (negative) large gradient component, and $\xi \in [-\sqrt{\varepsilon}, 1]$ to be the small gradient component. Our main idea is to perform

- a gradient (descent) step with respect to η, and
- a mirror (descent) step with respect to ξ.

Note that we have intentionally truncated the gradient $\nabla_i f(\mathsf{x}_k)$ at (negative) $\sqrt{\varepsilon}$, rather than at 1 as in `PacLPSolver`. This is so because, as it is much harder to deal with negative gradients in the covering LP case, we cannot perform both a mirror and a gradient step anymore on the small component ξ, as it was in `PacLPSolver`; instead, we can only perform a single mirror step on ξ. If ξ were between -1 and 1, and even if η were always zero, classical theory of mirror descent (or multiplicative weight update) could only imply that the mirror step converges at a rate of $\propto \varepsilon^{-2}$. Instead, we discover that if we truncate the gradient to $\xi \in [-\sqrt{\varepsilon}, 1]$, a *negative-width technique* allows us to improve this convergence from ε^{-2} to $\varepsilon^{-1.5}$. This is the first time that this gradient truncation technique is performed non-symmetrically.[8]

Due to this weaker truncation at $-\sqrt{\varepsilon}$ instead of -1, our gradient step enjoys a convergence rate that is only $\propto \varepsilon^{-1.5}$, matching that of the mirror step. This is precisely why we truncate the gradient at $\sqrt{\varepsilon}$, as it provides the best truncation tradeoff between gradient and mirror descent.

It is perhaps worth mentioning that our gradient step is equipped with an novel analysis quite different from its classical counterpart in optimization theory. Traditionally, given convex function $g(x)$, the convergence analysis only uses the simple upper bound $g(x) - g(x^*) \leq \langle \nabla g(x), x - x^* \rangle$ on the objective distance to optimum. If $g(x) = e^{-x}$ is a univariate function, $x = -1$, and $x^* = -100$, this upper bound becomes $e^{-1} \approx e^{-1} - e^{-100} \leq e^{-1} \cdot 99$, which is too weak to be used. This is the place we need to use a *distance-adjustment technique*, which will effectively improve the distance estimation to the optimum.

The detailed description and the analysis of our `CovLPSolver` can be found in Appendix D.

[8]It is perhaps interesting to point out that this negative-width technique is already reused into a separate paper of ours on spectral sparsification [ALO14].

Appendix

A Missing Proofs for Section 2

Fact 2.1. *Define the bounding box* $\Delta \overset{\text{def}}{=} \{x \in \mathbb{R}^n : x_i \in \big[0, \frac{1}{\|A_{:i}\|_\infty}\big]\}$*. Under assumption (2.1), we have* $\mathsf{OPT} \in [1, n]$ *and* $\{x : x \geq 0 \wedge Ax \leq \mathbb{1}\} \subseteq \Delta$.

Proof. Suppose that i^* is the column that achieves the smallest infinite norm $\|A_{:i}\|_\infty$ over all columns. Letting x be such that $x_i = 1$ at $i = i^*$ and $x_i = 0$ elsewhere, we have obtained a feasible solution for the packing LP (1.1), owing to our choice of $\min_{i\in[n]}\{\|A_{:i}\|_\infty\} = 1$ in (2.1). This feasible x gives an objective $\mathbb{1}^T x = 1$, showing that $\mathsf{OPT} \geq 1$.

On the other hand, for any solution $x \in \mathbb{R}^n_{\geq 0}$ satisfying $Ax \leq \mathbb{1}$, we must have $x_i \leq \frac{1}{\|A_{:i}\|_\infty}$ for each i. Therefore, $\mathbb{1}^T x \leq \sum_i \frac{1}{\|A_{:i}\|_\infty} \leq n$, showing that $\mathsf{OPT} \leq n$.

The inclusion $\{x : x \geq 0 \wedge Ax \leq \mathbb{1}\} \subseteq \Delta$ is obvious, since if $x_i > \frac{1}{\|A_{:i}\|_\infty}$ for some i, that must violate the constraint $Ax \leq \mathbb{1}$. □

Proposition 2.3. *Let* $\mu = \frac{\varepsilon}{4\log(nm/\varepsilon)}$ *and* x^* *be an optimal solution for the packing LP (1.1). Then:*

(a) $f_\mu(u^*) \leq -(1-\varepsilon)\mathsf{OPT}$ *for* $u^* \overset{\text{def}}{=} (1-\varepsilon/2)x^* \in \Delta$.
(b) $f_\mu(x) \geq -(1+\varepsilon)\mathsf{OPT}$ *for every* $x \in \Delta$.
(c) If $x \in \Delta$ *satisfies* $f_\mu(x) \leq -(1-O(\varepsilon))\mathsf{OPT}$*, then* $\frac{1}{1+\varepsilon}x$ *is a* $(1-O(\varepsilon))$*-approximate solution to the packing LP.*

Proof.

(a) We have $\mathbb{1}^T u^* = (1-\varepsilon/2)\mathsf{OPT}$ by the definition of OPT. Also, from the feasibility constraint $Ax^* \leq \mathbb{1}$ in the packing LP, we have $Au^* - \mathbb{1} \leq -\varepsilon/2 \cdot \mathbb{1}$, and can compute $f_\mu(u^*)$ as follows:

$$f_\mu(u^*) = \mu \sum_j \exp^{\frac{1}{\mu}((Au^*)_j - 1)} - \mathbb{1}^T u^* \leq \mu \sum_j \exp^{\frac{-\varepsilon/2}{\mu}} - (1-\varepsilon/2)\mathsf{OPT}$$
$$\leq \frac{\mu m}{(nm)^2} - (1-\varepsilon/2)\mathsf{OPT} \leq -(1-\varepsilon)\mathsf{OPT} \ .$$

(b) Suppose towards contradiction that $f_\mu(x) < -(1+\varepsilon)\mathsf{OPT}$. Since $f_\mu(x) > -\mathbb{1}^T x$, it must satisfy that $\mathbb{1}^T x > (1+\varepsilon)\mathsf{OPT}$. Suppose that $\mathbb{1}^T x = (1+v)\mathsf{OPT}$ for some $v > \varepsilon$. By the definition of OPT, we must have that $Ax < (1+v)\mathbb{1}$ is broken, and therefore there exists some $j \in [m]$ satisfying that $(Ax)_j \geq 1+v$. In such a case, the objective

$$f_\mu(x) \geq \mu \exp^{v/\mu} - (1+v)\mathsf{OPT} = \frac{\varepsilon}{4\log(nm)}\Big((\frac{nm}{\varepsilon})^4\Big)^{v/\varepsilon} - (1+v)\mathsf{OPT} \geq \Big(\big((\frac{nm}{\varepsilon}\big)^2)^{v/\varepsilon} - (1+v)\Big)\mathsf{OPT} > 0$$

giving a contradiction to the assumption that $f_\mu(x) < 0$.

(c) Suppose x satisfies $f_\mu(x) \leq -(1-O(\varepsilon))\mathsf{OPT} \leq 0$ and we first want to show $Ax \leq (1+\varepsilon)\mathbb{1}$. Let us assume that $v = \max_j((Ax)_j - 1) \geq 0$ because otherwise we will have $Ax \leq \mathbb{1}$. Under this definition, we have $Ax \leq (1+v)\mathbb{1}$ and therefore $\mathbb{1}^T x \leq (1+v)\mathsf{OPT}$ by the definition of OPT. We compute $f_\mu(x)$ as follows.

$$f_\mu(x) \geq \mu \exp^{\frac{v}{\mu}} - (1+v)\mathsf{OPT} \geq \mu\Big((\frac{nm}{\varepsilon})^4\Big)^{v/\varepsilon} - (1+v)n = \frac{\varepsilon}{4\log(nm)}\Big((\frac{nm}{\varepsilon})^4\Big)^{v/\varepsilon} - (1+v)n \ .$$

It is easy to see that the above quantity is positive whenever $v \geq \varepsilon$, and therefore, to satisfy $f_\mu(x) \leq 0$ we must have $v \leq \varepsilon$, which is equivalent to $Ax \leq (1+\varepsilon)\mathbb{1}$.

Next, because $-\mathbb{1}^T x \le f_\mu(x) \le -(1-O(\varepsilon))\mathsf{OPT}$, we know that x yields an objective $\mathbb{1}^T x \ge (1-O(\varepsilon))\mathsf{OPT}$. Letting $x' = \frac{1}{1+\varepsilon}x$, we both have that x' is feasible (i.e., $Ax' \le \mathbb{1}$), and x' has an objective $\mathbb{1}^T x'$ at least as large as $(1-O(\varepsilon))\mathsf{OPT}$. □

Lemma 2.6. *Define the smoothness parameter $L \stackrel{\text{def}}{=} \frac{4}{\mu}$. Then, for every $x \in \Delta$, and every $i \in [n]$:*

(a) If $|\nabla_i f_\mu(x)| \le 1$, then for all $\lambda \le \frac{1}{L\|A_{:i}\|_\infty}$

$$|\nabla_i f_\mu(x+\lambda \mathbf{e}_i) - \nabla_i f_\mu(x)| \le L\|A_{:i}\|_\infty \cdot |\lambda| \ .$$

(b) If $|\nabla_i f_\mu(x)| \ge 1$, then for all $\lambda \le \frac{1}{L\|A_{:i}\|_\infty}$:

$$\nabla_i f_\mu(x+\lambda \mathbf{e}_i) \ge \Big(1 - \frac{\|A_{:i}\|_\infty L}{2}|\lambda|\Big)\nabla_i f_\mu(x) \ .$$

Proof of Lemma 2.6. Using the fact that $\nabla_i f_\mu(x) > -1$ for all x, we have:

$$\Big|\log\frac{\nabla_i f_\mu(x+\lambda\mathbf{e}_i)+1}{\nabla_i f_\mu(x)+1}\Big| = \Big|\int_0^\lambda \frac{\nabla^2_{ii} f_\mu(x+\nu\mathbf{e}_i)}{\nabla_i f_\mu(x+\nu\mathbf{e}_i)+1}d\nu\Big| = \frac{1}{\mu}\Big|\int_0^\lambda \frac{(A^T\mathrm{diag}\{p(x+\nu\mathbf{e}_i)\}A)_{ii}}{(A^T p(x+\nu\mathbf{e}_i))_i}d\nu\Big|$$
$$\le \frac{\|A_{:i}\|_\infty}{\mu}|\lambda| = \frac{\|A_{:i}\|_\infty L}{4}|\lambda| \ .$$

The last equality holds as $L = \frac{4}{\mu}$. This immediately implies the following multiplicative bound:

$$e^{-\frac{\|A_{:i}\|_\infty L}{4}|\lambda|} \le \frac{\nabla_i f_\mu(x+\lambda\mathbf{e}_i)+1}{\nabla_i f_\mu(x)+1} \le e^{\frac{\|A_{:i}\|_\infty L}{4}|\lambda|}.$$

By our assumption on λ, we know that $\frac{\|A_{:i}\|_\infty L}{4}|\lambda| \le \frac{1}{4}$, so that we can use the approximation $x \le e^x - 1 \le 1.2x$ over $x \in [-\frac{1}{4}, \frac{1}{4}]$. This yields the simpler bound:

$$-\frac{\|A_{:i}\|_\infty L}{4}|\lambda| \le \frac{\nabla_i f_\mu(x+\lambda\mathbf{e}_i) - \nabla_i f_\mu(x)}{\nabla_i f_\mu(x)+1} \le 1.2\frac{\|A_{:i}\|_\infty L}{4}|\lambda|.$$

Now we are ready to prove the two points of the lemma.

(a) Assuming that $\nabla_i f_\mu(x) \in (-1, 1]$, we have:

$$\Big|\nabla_i f_\mu(x+\lambda\mathbf{e}_i) - \nabla_i f_\mu(x)\Big| \le 2.4 \cdot \frac{\|A_{:i}\|_\infty L}{4}|\lambda| \le \|A_{:i}\|_\infty L|\lambda| \ .$$

(b) Assuming $\nabla_i f_\mu(x) \ge 1$, we have

$$\nabla_i f_\mu(x+\lambda\mathbf{e}_i) \ge \nabla_i f_\mu(x) - \frac{\|A_{:i}\|_\infty L}{4}|\lambda|\Big(\nabla_i f_\mu(x)+1\Big) \ge \Big(1 - \frac{\|A_{:i}\|_\infty L}{2}|\lambda|\Big)\nabla_i f_\mu(x) \ . \quad \square$$

Fact 2.7. *Defining $x_i^{\mathsf{start}} \stackrel{\text{def}}{=} \frac{1-\varepsilon/2}{n\|A_{:i}\|_\infty}$ for for each $i \in [n]$, we have $x^{\mathsf{start}} \in \Delta$ and $f_\mu(x^{\mathsf{start}}) \le -\frac{1-\varepsilon}{n}$.*

Proof. Using the fact that $Ax^{\mathsf{start}} - \mathbb{1} \le -\varepsilon/2 \cdot \mathbb{1}$, we compute $f_\mu(x^{\mathsf{start}})$ as follows:

$$f_\mu(x^{\mathsf{start}}) = \mu\sum_j \exp^{\frac{1}{\mu}((Ax^{\mathsf{start}})_j - 1)} - \mathbb{1}^T x^{\mathsf{start}} \le \mu\sum_j \exp^{\frac{-\varepsilon/2}{\mu}} - \frac{1-\varepsilon/2}{n} \le \frac{\mu m}{(nm)^2} - \frac{1-\varepsilon/2}{n} \le -\frac{1-\varepsilon}{n} \ .$$

Above, we have used that $\mathbb{1}^T x^{\mathsf{start}} \ge x_i^{\mathsf{start}} = \frac{1-\varepsilon/2}{n}$, where i is the column such that $\|A_{:i}\|_\infty = 1$. □

B Missing Proofs for Section 3

Lemma 3.2. *We have* $\mathsf{x}_k, \mathsf{y}_k, \mathsf{z}_k \in \Delta$ *for all* $k = 0, 1, \dots, T$.

Proof. This is true at the beginning as $\mathsf{x}_0 = \mathsf{y}_0 = x^{\mathsf{start}} \in \Delta$ (see Fact 2.7) and $\mathsf{z}_0 = 0 \in \Delta$.

In fact, it suffices for us to show that for every $k \geq 0$, $\mathsf{y}_k = \sum_{l=0}^{k} \gamma_k^l \mathsf{z}_l$ for some scalers γ_k^l satisfying $\sum_l \gamma_k^l = 1$ and $\gamma_k^l \geq 0$ for each $l = 0, \dots, k$. If this is true, we can prove the lemma by induction: at each iteration k,

1. $\mathsf{x}_k = \tau \mathsf{z}_{k-1} + (1-\tau)\mathsf{y}_{k-1}$ must be in Δ because y_{k-1} and z_{k-1} are and $\tau \in [0,1]$,
2. z_k is in Δ by the definition that $\mathsf{z}_k = \arg\min_{z\in\Delta}\{\cdots\}$, and
3. y_k is also in Δ because $\mathsf{y}_k = \sum_{l=0}^{k} \gamma_k^l \mathsf{z}_l$ is a convex combination of the z_l's and Δ is convex.

For the rest of the proof, we only need to show that $\mathsf{y}_k = \sum_{l=0}^{k} \gamma_k^l \mathsf{z}_l$ for[9]

$$\gamma_k^l = \begin{cases} (1-\tau)\gamma_{k-1}^l, & l = 0, \dots, k-2; \\ \big(\frac{1}{n\alpha_{k-1}L} - \frac{1}{n\alpha_k L}\big) + \tau\big(1 - \frac{1}{n\alpha_{k-1}L}\big), & l = k-1; \\ \frac{1}{n\alpha_k L}, & l = k. \end{cases}$$

This is true at the base case because $\alpha_0 = \frac{1}{nL}$. It is also true at $k=1$ because $\mathsf{y}_1 = \mathsf{x}_1 + \frac{1}{n\alpha_1 L}(\mathsf{z}_1 - \mathsf{z}_0) = \frac{1}{n\alpha_1 L}\mathsf{z}_1 + \big(1 - \frac{1}{n\alpha_1 L}\big)\mathsf{z}_0$. For the general k, we have

$$\begin{aligned} \mathsf{y}_k &= \mathsf{x}_k + \frac{1}{n\alpha_k L}(\mathsf{z}_k - \mathsf{z}_{k-1}) \\ &= \tau \mathsf{z}_{k-1} + (1-\tau)\mathsf{y}_{k-1} + \frac{1}{n\alpha_k L}(\mathsf{z}_k - \mathsf{z}_{k-1}) \\ &= \tau \mathsf{z}_{k-1} + (1-\tau)\Big(\sum_{l=0}^{k-2}\gamma_{k-1}^l \mathsf{z}_l + \frac{1}{n\alpha_{k-1}L}\mathsf{z}_{k-1}\Big) + \frac{1}{n\alpha_k L}(\mathsf{z}_k - \mathsf{z}_{k-1}) \\ &= \Big(\sum_{l=0}^{k-2}(1-\tau)\gamma_{k-1}^l \mathsf{z}_l\Big) + \Big(\Big(\frac{1}{n\alpha_{k-1}L} - \frac{1}{n\alpha_k L}\Big) + \tau\Big(1 - \frac{1}{n\alpha_{k-1}L}\Big)\Big)\mathsf{z}_{k-1} + \frac{1}{n\alpha_k L}\mathsf{z}_k \ . \end{aligned}$$

Therefore, we obtain $\mathsf{y}_k = \sum_{l=0}^{k} \gamma_k^l \mathsf{z}_l$ as desired.

It is now easy to check that under our definition of α_k (which satisfies $\alpha_k \geq \alpha_{k-1}$ and $\alpha_k \geq \alpha_0 = \frac{1}{nL}$, we must have $\gamma_k^l \geq 0$ for all k and l. Also,

$$\begin{aligned} \sum_l \gamma_k^l &= \sum_{l=0}^{k-2}(1-\tau)\gamma_{k-1}^l + \Big(\Big(\frac{1}{n\alpha_{k-1}L} - \frac{1}{n\alpha_k L}\Big) + \tau\Big(1 - \frac{1}{n\alpha_{k-1}L}\Big)\Big) + \frac{1}{n\alpha_k L} \\ &= (1-\tau)\Big(1 - \frac{1}{n\alpha_{k-1}L}\Big) + \Big(\Big(\frac{1}{n\alpha_{k-1}L} - \frac{1}{n\alpha_k L}\Big) + \tau\Big(1 - \frac{1}{n\alpha_{k-1}L}\Big)\Big) + \frac{1}{n\alpha_k L} = 1 \ . \end{aligned}$$

□

Lemma 3.5. *When* $\mathsf{z}_k^{(i)} = \arg\min_{z\in\Delta}\big\{\frac{1}{2}\|z - \mathsf{z}_{k-1}\|_A^2 + \langle n\alpha_k \xi_k^{(i)}, z\rangle\big\}$, *we have*

$$\langle n\alpha_k \xi_k^{(i)}, \mathsf{z}_{k-1} - u\rangle \leq n^2\alpha_k^2 L \cdot \langle \xi_k^{(i)}, \mathsf{x}_k - \mathsf{y}_k^{(i)}\rangle + \frac{1}{2}\|\mathsf{z}_{k-1} - u\|_A^2 - \frac{1}{2}\|\mathsf{z}_k^{(i)} - u\|_A^2 \ .$$

Proof. Denoting by $V_a(b) = \frac{1}{2}\|b - a\|_A^2$ as a function of $b \in \Delta$ parameterized at $a \in \Delta$, we have that $\nabla_i V_a(b) = \|A_{:i}\|_\infty \cdot (a_i - b_i)$. In optimization theory, $V_a(b)$ is also known as the Bregman divergence of the $\|\cdot\|_A^2$ regularizer.

[9]We wish to point out that this proof coincides with a lemma from the accelerated coordinate descent theory of Fercoq and Richtárik [FR13]. Their paper is about optimizing an objective function that is Lipschitz smooth, and thus irrelevant to our work.

We deduce the following sequence of inequalities:

$$\begin{aligned}\big\langle n\alpha_k\xi_k^{(i)}, \mathsf{z}_{k-1}-u\big\rangle &= \big\langle n\alpha_k\xi_k^{(i)}, \mathsf{z}_{k-1}-\mathsf{z}_k^{(i)}\big\rangle + \big\langle n\alpha_k\xi_k^{(i)}, \mathsf{z}_k^{(i)}-u\big\rangle \\ &\overset{①}{\leq} \big\langle n\alpha_k\xi_k^{(i)}, \mathsf{z}_{k-1}-\mathsf{z}_k^{(i)}\big\rangle + \big\langle -\nabla V_{\mathsf{z}_{k-1}}(\mathsf{z}_k^{(i)}), \mathsf{z}_k^{(i)}-u\big\rangle \\ &\overset{②}{=} \big\langle n\alpha_k\xi_k^{(i)}, \mathsf{z}_{k-1}-\mathsf{z}_k^{(i)}\big\rangle - \frac{1}{2}\|\mathsf{z}_{k-1}-\mathsf{z}_k^{(i)}\|_A^2 + \frac{1}{2}\|\mathsf{z}_{k-1}-u\|_A^2 - \frac{1}{2}\|\mathsf{z}_k^{(i)}-u\|_A^2 \\ &\overset{③}{=} n^2\alpha_k^2 L\Big(\big\langle \xi_k^{(i)}, \mathsf{x}_k-\mathsf{y}_k\big\rangle - \frac{L}{2}\|\mathsf{x}_k-\mathsf{y}_k\|_A^2\Big) + \frac{1}{2}\|\mathsf{z}_{k-1}-u\|_A^2 - \frac{1}{2}\|\mathsf{z}_k^{(i)}-u\|_A^2 \\ &\leq n^2\alpha_k^2 L\cdot\big\langle \xi_k^{(i)}, \mathsf{x}_k-\mathsf{y}_k\big\rangle + \frac{1}{2}\|\mathsf{z}_{k-1}-u\|_A^2 - \frac{1}{2}\|\mathsf{z}_k^{(i)}-u\|_A^2 \ .\end{aligned}$$

Here, ① is due to the minimality of $\mathsf{z}_k^{(i)} = \arg\min_{z\in\Delta}\big\{V_{\mathsf{z}_{k-1}}(z) + \big\langle n\alpha_k\xi_k^{(i)}, z\big\rangle\big\}$, which implies that $\big\langle \nabla V_{\mathsf{z}_{k-1}}(\mathsf{z}_k^{(i)}) + n\alpha\xi_k^{(i)}, u-\mathsf{z}_k^{(i)}\big\rangle \geq 0$ for all $u\in\Delta$. Step ② is due to the "three-point equality" of Bregman divergence (cf. [CL06]), which can be checked for every coordinate $\ell\in[n]$ as follows:

$$\begin{aligned}-\nabla_\ell V_{\mathsf{z}_{k-1}}(\mathsf{z}_k^{(i)})\cdot(\mathsf{z}_{k,\ell}^{(i)}-u_\ell) &= \|A_{:i}\|_\infty(\mathsf{z}_{k-1,\ell}-\mathsf{z}_{k,\ell}^{(i)})\cdot(\mathsf{z}_{k,\ell}^{(i)}-u_\ell) \\ &= \|A_{:i}\|_\infty\Big(-\frac{1}{2}(\mathsf{z}_{k-1,\ell}-\mathsf{z}_{k,\ell}^{(i)})^2 + \frac{1}{2}(u_\ell-\mathsf{z}_{k-1,\ell})^2 - \frac{1}{2}(\mathsf{z}_{k,\ell}^{(i)}-u_\ell)^2\Big) \ .\end{aligned}$$

③ is by our choice of y_k which satisfies that $\mathsf{z}_{k-1}-\mathsf{z}_k^{(i)} = n\alpha_k L(\mathsf{x}_k-\mathsf{y}_k^{(i)})$. □

Proposition 3.6. *If $\mathsf{z}_{k-1}\in\Delta$, the minimizer $z=\arg\min_{z\in\Delta}\big\{\frac{1}{2}\|z-\mathsf{z}_{k-1}\|_A^2 + \langle\delta\mathbf{e}_i, z\rangle\big\}$ for any scalar $\delta\in\mathbb{R}$ and basis vector $\mathbf{e}_i$ can be computed as follows:*

1. *$z\leftarrow\mathsf{z}_{k-1}$.*
2. *$z_i\leftarrow z_i-\delta/\|A_{:i}\|_\infty$.*
3. *If $z_i<0$, then $z_i\leftarrow 0$; if $z_i>1/\|A_{:i}\|_\infty$, $z_i\leftarrow 1/\|A_{:i}\|_\infty$.*
4. *Return z.*

Proof of Proposition 3.6. Let us denote by z the returned value of the described procedure, and $g(u)\overset{\text{def}}{=}\frac{1}{2}\|u-\mathsf{z}_{k-1}\|_A^2+\langle\delta\mathbf{e}_i,u\rangle$. Since Δ is a convex body and $g(\cdot)$ is convex, to show $z=\arg\min_{z\in\Delta}\{g(z)\}$, it suffices for us to prove that for every $u\in\Delta$, $\langle\nabla g(z), u-z\rangle\geq 0$. Since the gradient $\nabla g(z)$ can be written explicitly, this is equivalent to

$$\delta(u_i-z_i)+\sum_{\ell=1}^n\|A_{:\ell}\|_\infty\cdot\big(z_\ell-\mathsf{z}_{k-1,\ell}\big)\cdot(u_\ell-z_\ell)\geq 0 \ .$$

However, since $z_\ell=\mathsf{z}_{k-1,\ell}$ for every $\ell\neq i$, this is equivalent to

$$\Big(\delta+\|A_{:i}\|_\infty\cdot\big(z_i-\mathsf{z}_{k-1,i}\big)\Big)\cdot(u_i-z_i)\geq 0 \ .$$

There are three possibilities here. If $z_i=\mathsf{z}_{k-1,i}-\delta/\|A_{:i}\|_\infty$ then the left-hand side is zero and we are done. Otherwise, if $z_i>\mathsf{z}_{k-1,i}-\delta/\|A_{:i}\|_\infty$, then it must satisfy that $z_i=0$; in such a case the left-hand side is the multiplication of two non-negatives, and therefore non-positive. If $z_i<\mathsf{z}_{k-1,i}-\delta/\|A_{:i}\|_\infty$, then it must satisfy that $z_i=1/\|A_{:i}\|_\infty$; in such a case the left-hand side is the multiplication of two non-positives, and therefore non-positive. □

Lemma 3.9. $\langle n\alpha_k\eta_k^{(i)}, \mathsf{z}_{k-1}-u\rangle + n^2\alpha_k^2L\cdot\langle\xi_k^{(i)},\mathsf{x}_k-\mathsf{y}_k^{(i)}\rangle\leq 3n\alpha_kL\cdot(f_\mu(\mathsf{x}_k)-f_\mu(\mathsf{y}_k^{(i)}))$.

Proof. Now there are three possibilities:

- If $\eta_k^{(i)}=0$, then we must have $\xi_{k,i}^{(i)}=\nabla_i f_\mu(\mathsf{x}_k)\in[-1,1]$, and Lemma 3.8 immediately implies

$$\langle n\alpha_k\eta_k^{(i)},\mathsf{z}_{k-1}-u\rangle+n^2\alpha_k^2L\cdot\langle\xi_k^{(i)},\mathsf{x}_k-\mathsf{y}_k^{(i)}\rangle = n^2\alpha_k^2L\cdot\langle\nabla f_\mu(\mathsf{x}_k),\mathsf{x}_k-\mathsf{y}_k^{(i)}\rangle\leq 2n^2\alpha_k^2L\cdot(f_\mu(\mathsf{x}_k)-f_\mu(\mathsf{y}_k^{(i)}))$$

- If $\eta_k^{(i)} > 0$ and $\mathsf{z}_{k,i}^{(i)} > 0$, then we precisely have $\mathsf{z}_{k,i}^{(i)} = \mathsf{z}_{k-1,i} - \frac{n\alpha_k}{\|A_{:i}\|_\infty}$ (see Proposition 3.6), and accordingly $\mathsf{y}_{k,i}^{(i)} = \mathsf{x}_{k,i} - \frac{1}{L\|A_{:i}\|_\infty} < \mathsf{x}_{k,i}$. In this case,

$$
\begin{aligned}
&\langle n\alpha_k \eta_k^{(i)}, \mathsf{z}_{k-1} - u\rangle + n^2\alpha_k^2 L \cdot \langle \xi_k^{(i)}, \mathsf{x}_k - \mathsf{y}_k^{(i)}\rangle \\
&\overset{①}{\leq} n\alpha_k \cdot \nabla f_\mu(\mathsf{x}_k) \cdot \frac{1}{\|A_{:i}\|_\infty} + n^2\alpha_k^2 L \cdot \langle \xi_k^{(i)}, \mathsf{x}_k - \mathsf{y}_k^{(i)}\rangle \\
&\overset{②}{<} n\alpha_k \cdot \nabla f_\mu(\mathsf{x}_k) \cdot \frac{1}{\|A_{:i}\|_\infty} + n^2\alpha_k^2 L \cdot \langle \nabla f_\mu(\mathsf{x}_k), \mathsf{x}_k - \mathsf{y}_k^{(i)}\rangle \\
&\overset{③}{=} n\alpha_k L \cdot \langle \nabla f_\mu(\mathsf{x}_k), \mathsf{x}_k - \mathsf{y}_k^{(i)}\rangle + n^2\alpha_k^2 L \cdot \langle \nabla f_\mu(\mathsf{x}_k), \mathsf{x}_k - \mathsf{y}_k^{(i)}\rangle \overset{④}{\leq} \big(2n\alpha_k L + 2n^2\alpha_k^2 L\big) \cdot (f_\mu(\mathsf{x}_k) - f_\mu(\mathsf{y}_k^{(i)})) \ .
\end{aligned}
$$

  Above, ① follows from the fact that $\mathsf{z}_{k-1} \in \Delta$ and therefore $\mathsf{z}_{k-1,i} \leq \frac{1}{\|A_{:i}\|_\infty}$ by the definition of Δ, and $u \geq 0$; ② follows from the fact that x_k and $\mathsf{y}_k^{(i)}$ are only different at coordinate i, and $\xi_{k,i}^{(i)} = 1 < \nabla_i f_\mu(\mathsf{x}_k)$ (since $\eta_{k,i}^{(i)} > 0$); ③ follows from the fact that $\mathsf{y}_k^{(i)} = \mathsf{x}_k - \frac{\mathbf{e}_i}{L\|A_{:i}\|_\infty}$; and ④ uses Lemma 3.8.

- If $\eta_k^{(i)} > 0$ and $\mathsf{z}_{k,i}^{(i)} = 0$, then we have

$$
\begin{aligned}
&\langle n\alpha_k \eta_k^{(i)}, \mathsf{z}_{k-1} - u\rangle + n^2\alpha_k^2 L \cdot \langle \xi_k^{(i)}, \mathsf{x}_k - \mathsf{y}_k^{(i)}\rangle \\
&\overset{①}{\leq} \big(n\alpha_k \nabla f_\mu(\mathsf{x}_k) \cdot \mathsf{z}_{k-1,i}\big) + n^2\alpha_k^2 L \cdot \langle \nabla f_\mu(\mathsf{x}_k), \mathsf{x}_k - \mathsf{y}_k^{(i)}\rangle \\
&\overset{②}{=} \langle n\alpha_k \nabla f_\mu(\mathsf{x}_k), \mathsf{z}_{k-1} - \mathsf{z}_k^{(i)}\rangle + n^2\alpha_k^2 L \cdot \langle \nabla f_\mu(\mathsf{x}_k), \mathsf{x}_k - \mathsf{y}_k^{(i)}\rangle \\
&\overset{③}{=} n^2\alpha_k^2 L \cdot \langle \nabla f_\mu(\mathsf{x}_k), \mathsf{x}_k - \mathsf{y}_k^{(i)}\rangle + n^2\alpha_k^2 L \cdot \langle \nabla f_\mu(\mathsf{x}_k), \mathsf{x}_k - \mathsf{y}_k^{(i)}\rangle \overset{④}{\leq} 4n^2\alpha_k^2 L \cdot (f_\mu(\mathsf{x}_k) - f_\mu(\mathsf{y}_k^{(i)})) \ .
\end{aligned}
$$

  Above, ① is because $u \geq 0$, $\nabla_i f_\mu(\mathsf{x}_k) = \eta_{k,i}^{(i)} + 1 > \eta_{k,i}^{(i)}$ and $\nabla_i f_\mu(\mathsf{x}_k) > \xi_{k,i}^{(i)}$; ② uses the assumption that $\mathsf{z}_{k,i}^{(i)} = 0$ and the fact that $\mathsf{z}_{k-1,\ell} = \mathsf{z}_{k,\ell}^{(i)}$ for every $\ell \neq i$; ③ is from our choice of y_k which satisfies that $\mathsf{z}_{k-1} - \mathsf{z}_k^{(i)} = n\alpha_k L(\mathsf{x}_k - \mathsf{y}_k^{(i)})$; and ④ uses Lemma 3.8.

Combining the three cases above, and using the fact that $f_\mu(\mathsf{x}_k) - f_\mu(\mathsf{y}_k^{(i)}) \geq 0$, we conclude that

$$
\begin{aligned}
\langle n\alpha_k \eta_k^{(i)}, \mathsf{z}_{k-1} - u\rangle + n^2\alpha_k^2 L \cdot \langle \xi_k^{(i)}, \mathsf{x}_k - \mathsf{y}_k^{(i)}\rangle &\leq (2n\alpha_k L + 4n^2\alpha_k^2 L) \cdot (f_\mu(\mathsf{x}_k) - f_\mu(\mathsf{y}_k^{(i)})) \\
&\leq 3n\alpha_k L \cdot (f_\mu(\mathsf{x}_k) - f_\mu(\mathsf{y}_k^{(i)})) \ .
\end{aligned}
$$

Above, the last inequality uses our choice of α_k (see Algorithm 1). □

Corollary 3.10. *With probability at least* $9/10$, `PacLPSolver`$(A, x^{\mathsf{start}}, \varepsilon)$ *outputs a* $(1 - O(\varepsilon))$ *approximate solution to the packing LP program. The expected running time is* $O(\frac{\log(nm/\varepsilon)\log(1/\varepsilon)}{\varepsilon} N)$.

Proof. Since for every $x \in \Delta$ it satisfies $f_\mu(x) \geq -(1+\varepsilon)\mathsf{OPT}$ according to Proposition 2.3.b, we obtain that $f_\mu(y_T) + (1+\varepsilon)\mathsf{OPT}$ is a random variable that is non-negative, whose expectation $\mathbf{E}[f_\mu(y_T) + (1+\varepsilon)\mathsf{OPT}] \leq 4\varepsilon$. By Markov bound, with at least probability $9/10$, we obtain some y_T satisfying $f_\mu(y_T) \leq -(1 - O(\varepsilon))\mathsf{OPT}$, which yields some $(1 - O(\varepsilon))$ approximate solution according to Proposition 2.3.c.

The running time follows from our efficient implementation in Section G. □

C Relaxation of the Covering Linear Program

Recall that, for input matrix $A \in \mathbb{R}_{\geq 0}^{m \times n}$, the covering LP in its standard form is

$$\text{Covering LP:} \qquad \min_{x \geq 0} \{ \mathbb{1}^T x \, : \, Ax \geq \mathbb{1} \} \enspace .$$

Let us denote by OPT the optimal value to this linear program, and by x^* any optimal solution of the covering LP (1.2). We say that x is a $(1+\varepsilon)$-approximation for the covering LP if $Ax \geq \mathbb{1}$ and $\mathbb{1}^T x \leq (1+\varepsilon)\mathsf{OPT}$. In our covering LP solver, we assume that some 2-approximate solution $x^\sharp$ is given to the algorithm, and $\mathbb{1}^T x^\sharp = \mathsf{OPT}'$ for some $\mathsf{OPT}' \in [\mathsf{OPT}, 2\mathsf{OPT}]$.[10]

Again, we use the indices $i \in [n]$ for the columns of A, and the indices $j \in [m]$ for the rows of A. We denote by $A_{:i}$ the i-th column vector of A, and $A_{j:}$ the j-th row vector of A. We can assume without loss of generality that[11]

$$\min_{j \in [m]} \{ \|A_{j:}\|_\infty \} = 1 \enspace . \tag{C.1}$$

We now introduce the smoothed objective $f_\mu(x)$ that we are going to minimize in order to approximately solve the covering LP. We skip the details regarding how it arises from a relaxation using the generalized entropy regularizer, because it is essentially a repetition of Section 2.

This smoothed objective turns each row of the LP constraint $Ax \geq \mathbb{1}$ into an exponential penalty function so that we only need to require $x \geq 0$ throughout the algorithm.

Definition C.1. *Letting parameter* $\mu \stackrel{\text{def}}{=} \frac{\varepsilon}{4\log(nm/\varepsilon)}$, *we define the smoothed objective* $f_\mu(x)$ *as*

$$f_\mu(x) \stackrel{\text{def}}{=} \mu \sum_{j=1}^{m} \exp^{\frac{1}{\mu}(1-(Ax)_j)} + \mathbb{1}^T x$$

over the simplex $x \in \Delta \stackrel{\text{def}}{=} \{x \in \mathbb{R}^n \, : \, x_i \geq 0 \, \wedge \, \mathbb{1}^T x \leq 2\mathsf{OPT}'\}$.

We wish to study the *minimization* problem on $f_\mu(x)$, subject to the constraint that each coordinate x_i is non-negative and the coordinates sum up to at most $2\mathsf{OPT}'$. The intuition that this smoothed objective $f_\mu(x)$ captures the original covering LP (1.2) is similar to that of the packing LP one. Note that our constraint $\mathbb{1}^T x \leq 2\mathsf{OPT}'$ is of course redundant; it will play some other important role in our algorithm.

We begin with several simple but important properties about OPT and $f_\mu(x)$. In short, they together imply that the minimum of $f_\mu(x)$ is around OPT, and if one can approximately find the minimum of $f_\mu(x)$ (up to an error $O(\varepsilon\mathsf{OPT})$), this corresponds to a $(1+O(\varepsilon))$-approximate solution to the covering LP (1.2).

Proposition C.2.

(a) $\mathsf{OPT} \in [1, m]$.
(b) $f_\mu(u^*) \leq (1+\varepsilon)\mathsf{OPT}$ *for* $u^* \stackrel{\text{def}}{=} (1+\varepsilon/2)x^* \in \Delta$.
(c) $f_\mu(x) \geq (1-\varepsilon)\mathsf{OPT}$ *for every* $x \geq 0$.
(d) Letting $x^{\mathsf{start}} = (1+\varepsilon/2) \cdot x^\sharp + (\frac{1}{n}, \dots, \frac{1}{n})$, *we have* $\mathbb{1}^T x^{\mathsf{start}} \leq 2\mathsf{OPT}'$ *and* $f_\mu(x^{\mathsf{start}}) \leq 4\mathsf{OPT}$.
(e) For any $x \geq 0$ *satisfying* $f_\mu(x) \leq 2\mathsf{OPT}$, *we must have* $Ax \geq (1-\varepsilon)\mathbb{1}$.
(f) If $x \geq 0$ *satisfies* $f_\mu(x) \leq (1+O(\varepsilon))\mathsf{OPT}$, *then* $\frac{1}{1-\varepsilon}x$ *is a* $(1+O(\varepsilon))$*-approximate solution to the covering LP.*

[10]This can be obtained via for instance the covering LP solver from Young [You14], whose running time is $O(N \log N)$. It can be relaxed to any constant approximation rather than 2-approximation.

[11]We can do so because first of all, we can assume $\min_{j \in [n]}\{\|A_{j:}\|_\infty\} > 0$ since otherwise the covering LP is infeasible. Next, we can scale A down by a factor of $\min_{j \in [n]}\{\|A_{j:}\|_\infty\}$; this also scales down the optimal value OPT and solution x^* by this same factor.

(g) The gradient of $f_\mu(x)$ can be written as

$$\nabla f_\mu(x) = \mathbb{1} - A^T p(x) \quad \textit{where} \quad p_j(x) \stackrel{\text{def}}{=} \exp^{\frac{1}{\mu}(1-(Ax)_j)} \tag{C.2}$$

Proof.

(a) Suppose that j^* is the row that achieves the smallest infinite norm $\|A_{j:}\|_\infty$ over all rows. Then, for any solution $x \in \mathbb{R}^n_{\geq 0}$ satisfying $\langle A_{:j^*}, x\rangle \geq 1$, we must have $\mathbb{1}^T x \geq 1/\|A_{:j^*}\|_\infty = 1$.

On the other hand, we can construct a feasible solution x as follows. Initialize $x = 0$, and then for each row j, let us find the coordinate i that maximizes the value of A_{ij} among all columns i. Then, we increase x_i by $1/A_{ij} = 1/\|A_{j:}\|_\infty$. After we have exhausted all the m rows, we arrive at some $x \geq 0$ satisfying $Ax \geq \mathbb{1}$ as well as $\mathbb{1}^T x = \sum_j 1/\|A_{j:}\|_\infty \leq m$.

(b) We have $\mathbb{1}^T u^* = (1+\varepsilon/2)\mathsf{OPT}$ by the definition of OPT. Also, from the feasibility constraint $Ax^* \geq \mathbb{1}$ in the covering LP, we have $Au^* - \mathbb{1} \geq \varepsilon/2 \cdot \mathbb{1}$, and can compute $f_\mu(u^*)$ as follows:

$$\begin{aligned} f_\mu(u^*) = \mu \sum_j \exp^{\frac{1}{\mu}(1-(Au^*)_j)} + \mathbb{1}^T u^* &\leq \mu \sum_j \exp^{\frac{-\varepsilon/2}{\mu}} + (1+\varepsilon/2)\mathsf{OPT} \\ &\leq \frac{\mu m}{(nm)^2} + (1+\varepsilon/2)\mathsf{OPT} \leq (1+\varepsilon)\mathsf{OPT} \ . \end{aligned}$$

(c) Suppose towards contradiction that $f_\mu(x) < (1-\varepsilon)\mathsf{OPT}$. Since $f_\mu(x) < \mathsf{OPT} \leq m$, we must have that for every $j \in [m]$, it satisfies that $\exp^{\frac{1}{\mu}(1-(Ax)_j)} \leq f_\mu(x)/\mu \leq m/\mu$. This further implies $(Ax)_j \geq 1-\varepsilon$ by the definition of μ. In other words, $Ax \geq (1-\varepsilon)\mathbb{1}$. By the definition of OPT, we must then have $\mathbb{1}^T x \geq (1-\varepsilon)\mathsf{OPT}$, finishing the proof that $f_\mu(x) \geq \mathbb{1}^T x \geq (1-\varepsilon)\mathsf{OPT}$, giving a contradiction.

(d) Using the fact that $Ax^{\mathsf{start}} - \mathbb{1} \geq (1+\varepsilon/2)Ax^\sharp - \mathbb{1} \geq \varepsilon/2 \cdot \mathbb{1}$, we compute $f_\mu(x^{\mathsf{start}})$ as follows:

$$f_\mu(x^{\mathsf{start}}) = \mu \sum_j \exp^{\frac{1}{\mu}(1-(Ax^{\mathsf{start}})_j)} + \mathbb{1}^T x^{\mathsf{start}} \leq \mu \sum_j \exp^{\frac{-\varepsilon/2}{\mu}} + 2\mathsf{OPT} + 1 \leq \frac{\mu m}{(nm)^2} + 3\mathsf{OPT} < 4\mathsf{OPT} \ .$$

Also, we have $\mathbb{1}^T x^{\mathsf{start}} \leq (1+\varepsilon/2)\mathsf{OPT}' + 1 \leq 2\mathsf{OPT}'$.

(e) To show $Ax \geq (1-\varepsilon)\mathbb{1}$, we can assume that $v = \max_j(1-(Ax)_j) > \varepsilon$ because otherwise we are done. Under this definition, we have

$$f_\mu(x) \geq \mu \exp^{\frac{v}{\mu}} = \mu\Big((\frac{nm}{\varepsilon})^4 \Big)^{v/\varepsilon} \geq \frac{\varepsilon}{4\log(nm)} (\frac{nm}{\varepsilon})^4 \gg 2\mathsf{OPT} \ ,$$

contradicting to our assumption that $f_\mu(x) \leq 2\mathsf{OPT}$. Therefore, we must have $v \leq \varepsilon$, that is, $Ax \geq (1-\varepsilon)\mathbb{1}$.

(f) For any x satisfying $f_\mu(x) \leq (1+O(\varepsilon))\mathsf{OPT} \leq 2\mathsf{OPT}$, owing to Proposition C.2.e, we first have that x is approximately feasible, i.e., $Ax \geq (1-\varepsilon)\mathbb{1}$. Next, because $\mathbb{1}^T x \leq f_\mu(x) \leq (1+O(\varepsilon))\mathsf{OPT}$, we know that x yields an objective $\mathbb{1}^T x \leq (1+O(\varepsilon))\mathsf{OPT}$. Letting $x' = \frac{1}{1-\varepsilon}x$, we both have that x' is feasible (i.e., $Ax' \geq \mathbb{1}$), and x' has an objective $\mathbb{1}^T x'$ at most $(1+O(\varepsilon))\mathsf{OPT}$.

(g) Straightforward by some simple computation. □

Algorithm 2 `CovLPSolver`$(A, x^{\mathsf{start}}, \varepsilon)$

```
Input: $A \in \mathbb{R}_{\geq 0}^{m\times n}, x^{\mathsf{start}} \in \Delta, \varepsilon \in (0, 1/10]$.
Output: $x \in \Delta$.
 1: $\mu \leftarrow \frac{\varepsilon}{4\log(nm/\varepsilon)}$, $\beta \leftarrow \sqrt{\varepsilon}$, $\tau \leftarrow \frac{\mu\beta}{12n}$.    ▷ parameters
 2: $T \leftarrow \lceil \frac{1}{\tau}\log(1/\varepsilon) \rceil = O(\frac{\log(nm/\varepsilon)\log(1/\varepsilon)}{\varepsilon^{1.5}} n)$.    ▷ number of iterations
 3: $\alpha_0 \leftarrow (1-\tau)^T \frac{\varepsilon}{12n\beta}$ and $\gamma \leftarrow \frac{\varepsilon}{6\beta}$.    ▷ so that $\alpha_T = \frac{\varepsilon}{12n\beta}$ and $\gamma = 2\alpha_T n$
 4: $\mathsf{x}_0 = \mathsf{y}_0 = \mathsf{z}_0 \leftarrow x^{\mathsf{start}}$.
 5: for $k \leftarrow 1$ to $T$ do
 6:     $\alpha_k \leftarrow \frac{1}{1-\tau}\alpha_{k-1}$.
 7:     $\mathsf{x}_k \leftarrow \tau \mathsf{z}_{k-1} + (1-\tau)\mathsf{y}_{k-1}$.
 8:     Randomly select $i$ uniformly at random from $[n]$.
 9:     Define $\xi_k^{(i)}$ to be a vector that is only non-zero at coordinate $i$, and equals to $\mathbb{T}^{\mathsf{c}}(\nabla_i f_\mu(\mathsf{x}_k))$.
            ▷ recall from (C.2) that $\nabla_i f_\mu(\mathsf{x}_k) = 1 - \sum_{j=1}^m A_{j,i} \exp^{\frac{1}{\mu}(1-(A\mathsf{x}_k)_j)}$
            ▷ recall from Definition D.1 that $\mathbb{T}^{\mathsf{c}}(v) \stackrel{\mathrm{def}}{=} \begin{cases} v, & v \in [-\beta, 1]; \\ -\beta, & v < -\beta. \end{cases}$
10:     $\mathsf{z}_k \leftarrow \mathsf{z}_k^{(i)} \stackrel{\mathrm{def}}{=} \arg\min_{z\in\Delta} \{ V_{\mathsf{z}_{k-1}}(z) + \langle (1+\gamma) n\alpha_k \xi_k^{(i)}, z\rangle \}$.    ▷ See Proposition D.9
11:     if $\nabla_i f_\mu(\mathsf{x}_k) < -\beta$ then
12:         Denote by $\pi$ the permutation that sorts the entries of $A_{:i}$ into $A_{\pi(1),i} \leq \cdots \leq A_{\pi(m),i}$ .
13:         Pick $j^* \in [m]$ such that $\sum_{j<j^*} A_{\pi(j),i} \cdot p_{\pi(j)}(\mathsf{x}_k) < 1+\beta$ but $\sum_{j\leq j^*} A_{\pi(j),i} \cdot p_{\pi(j)}(\mathsf{x}_k) \geq 1+\beta$.
                ▷ Such a $j^* \in [m]$ must exist because $\sum_{j=1}^m A_{ji} \cdot p_j(\mathsf{x}_k) \geq 1+\beta$.
14:         $\mathsf{y}_k \leftarrow \mathsf{y}_k^{(i)} \stackrel{\mathrm{def}}{=} \mathsf{x}_k + \delta \cdot \mathbf{e}_i$ where $\delta = \frac{\mu\beta}{2A_{\pi(j^*),i}}$.
15:     else
16:         $\mathsf{y}_k \leftarrow \mathsf{y}_k^{(i)} \stackrel{\mathrm{def}}{=} \mathsf{x}_k$.
17:     end if
18: end for
19: return $\mathsf{y}_T$.
```

D Covering LP Solver

To describe our covering LP solver we make the following choice of the thresholding function. Recall in the packing LP case, we have truncated each coordinate gradient from $[-1, \infty)$ to $[-1, 1]$. For this covering LP case, we truncate each such gradient from $(-\infty, 1]$ to $[-\beta, 1]$, for some parameter $\beta \stackrel{\mathrm{def}}{=} \sqrt{\varepsilon}$. The reason for this choice of $\beta = \sqrt{\varepsilon}$ shall become clear in later sections; at high level, $\sqrt{\varepsilon}$ is the best tradeoff between gradient and mirror descent.

Definition D.1. *The thresholding function* $\mathbb{T}^{\mathsf{c}} \colon (-\infty, 1] \to [-\beta, 1]$ *is defined as follows*

$$\mathbb{T}^{\mathsf{c}}(v) \stackrel{\mathrm{def}}{=} \begin{cases} v, & v \in [-\beta, 1]; \\ -\beta, & v < -\beta. \end{cases}$$

Our algorithm `CovLPSolver` starts with the initial vector $\mathsf{x}_0 = \mathsf{y}_0 = \mathsf{z}_0 = x^{\mathsf{start}}$ introduced in Proposition C.2.d, and is divided into T iterations. In each iteration, we start by computing a weighted midpoint $\mathsf{x}_k \leftarrow \tau \mathsf{z}_{k-1} + (1-\tau)\mathsf{y}_{k-1}$ for some parameter $\tau \in (0, 1)$, and then proceed to compute y_k and z_k as follows.

- Select $i \in [n]$ uniformly at random, and let $\xi_k^{(i)} = (0, \ldots, 0, \mathbb{T}^{\mathsf{p}}(v), 0, \ldots, 0)$ be the vector that is only non-zero at coordinate i, where $v = \nabla_i f_\mu(\mathsf{x}_k) = 1 - \sum_{j=1}^m A_{j,i} \exp^{\frac{1}{\mu}(1-(A\mathsf{x}_k)_j)} \in (-\infty, 1]$.

- Perform a *mirror (descent) step* $\mathsf{z}_k \leftarrow \mathsf{z}_k^{(i)} \stackrel{\text{def}}{=} \arg\min_{z \in \Delta} \big\{ V_{\mathsf{z}_{k-1}}(z) + \langle (1+\gamma) n \alpha_k \xi_k^{(i)}, z \rangle \big\}$ for some parameters $\gamma \ll 1$ and $\alpha_k \ll 1/n$, where $V_x(y) = \sum_{i=1}^n y_i \log \frac{y_i}{x_i} + x_i - y_i$ is the so-called Bregman divergence of the generalized entropy function (see Proposition D.9 below).
- Perform a *gradient (descent) step* $\mathsf{y}_k \leftarrow \mathsf{y}_k^{(i)} \stackrel{\text{def}}{=} \mathsf{x}_k + \delta \mathbf{e}_i$ for some value δ that is zero if $\nabla_i f_\mu(x_k) < -\beta$, and strictly positive otherwise. The precise definition of δ can be found in the pseudocode described in Algorithm 2.

Above, the reason that the the two steps on y_k and z_k are named after "gradient step" and "mirror step" will become clear in the follow-up sections. We use the superscript ${}^{(i)}$ on $\xi_i^{(i)}$, $\mathsf{y}_k^{(i)}$ and $\mathsf{z}_k^{(i)}$ to emphasize that the value depends on the choice of i. We have used generic parameters τ, α_k, T in the above description and their precise values are presented in Algorithm 2.[12]

Since the x^{start} satisfies $\mathbb{1}^T x^{\mathsf{start}} \leq 2\mathsf{OPT}'$ by Proposition C.2.d, we have $\mathsf{z}_0 = x^{\mathsf{start}} \in \Delta$. Also, the mirror descent step ensures that $\mathsf{z}_{k,i} > 0$ for all rounds k and coordinates i, as well as $\mathsf{z}_k \in \Delta$ for all rounds k. However, we note that x_k and y_k may not necessarily lie inside Δ, but will always stay non-negative. We summarize these properties as follows:

$$\forall k \in \{0, 1, \dots, T\}, \qquad \mathsf{x}_k, \mathsf{y}_k \geq 0, \quad \mathsf{z}_k > 0, \quad \mathsf{z}_k \in \Delta \ .$$

We shall also prove in Section G that

Lemma D.2. *Each iteration of* `CovLPSolver` *can be implemented to run in expected $O(N/n)$ time.*

The key idea is similar to that of the efficient implementation of `PacLPSolver`, that is to implementation the updates implicitly.

In this section, we prove the following theorem in five steps.

Theorem D.3. `CovLPSolver`$(A, x^{\mathsf{start}}, \varepsilon)$ *outputs some* y_T *satisfying* $\mathbf{E}[f_\mu(\mathsf{y}_T)] \leq (1+9\varepsilon)\mathsf{OPT}$.

D.1 Step 1: Distance Adjustment

Classically, using the convexity argument one can obtain $f_\mu(\mathsf{x}_k) - f_\mu(u) \leq \langle \nabla f_\mu(\mathsf{x}_k), \mathsf{x}_k - u \rangle$ for every $u \in \Delta$. In particular, if u is the optimal point, the right hand side is a simple upper bound on the objective distance from the current point $f_\mu(\mathsf{x}_k)$ to the optimum. This simple upper bound is essentially used by all the convergence analyses for first-order methods.

In this section, we strengthen this upper bound in the special case of $u = u^* \stackrel{\text{def}}{=} (1+\varepsilon/2)x^*$.

Define $\widetilde{A}$ be the *adjusted matrix* of A described as follows.

Definition D.4 (Adjusted matrix $\widetilde{A}$)**.** *For each row $j \in [m]$, if $(Au^*)_j \leq 2$ then we keep this row and define $\widetilde{A}_{j:} \stackrel{\text{def}}{=} A_{j:}$. Otherwise, —that is, if $(Au^*)_j > 2$— we define $\widetilde{A}_{j:} \stackrel{\text{def}}{=} \frac{2}{(Au^*)_j} \cdot A_{j:}$ to be the same j-th row $A_{j:}$, but scaled down by a factor of $\frac{2}{(Au^*)_j}$. It is clear from this definition that*

$$A_{ji} \geq \widetilde{A}_{ji} \text{ for all } i \in [n] \text{ and } j \in [m], \text{ while } (1+\varepsilon)\mathbb{1} \leq \widetilde{A}u^* \leq 2\mathbb{1}.$$

We now strengthen the classical bound $f_\mu(\mathsf{x}_k) - f_\mu(u) \leq \langle \nabla f_\mu(\mathsf{x}_k), \mathsf{x}_k - u \rangle$ as follows.

Lemma D.5 (Distance Adjustment)**.**

$$\begin{aligned} f_\mu(\mathsf{x}_k) - f_\mu(u^*) &\leq \langle \mathbb{1} - A^T p(\mathsf{x}_k), \mathsf{x}_k - u^* \rangle + \langle \widetilde{A}^T p(\mathsf{x}_k) - A^T p(\mathsf{x}_k), u^* \rangle + \varepsilon\mathsf{OPT} \\ &= \langle \nabla f_\mu(\mathsf{x}_k), \mathsf{x}_k - u^* \rangle + \langle \widetilde{A}^T p(\mathsf{x}_k) - A^T p(\mathsf{x}_k), u^* \rangle + \varepsilon\mathsf{OPT} \end{aligned}$$

[12]We encourage the readers to ignore their specific values for now. Our specific choices of the parameters shall become clearer and natural at the end of this section, and be discussed whenever they are used.

At high level, ignoring the negligible term $\varepsilon\mathsf{OPT}$ on the right hand side, the above upper bound strengthens the classical bound due to the extra term of $\langle \widetilde{A}^T p(\mathsf{x}_k) - A^T p(\mathsf{x}_k), u^* \rangle$. This extra term is always non-positive since $\widetilde{A} \leq A$ coordinate-wisely, but may be very negative in certain cases.

The intuition behind the proof is to realize that the convexity inequality $e^b - e^a \leq \langle e^b, b - a \rangle$ on the exponential function becomes far from tight when $a \ll 0$. For instance, when $b = 2$ and $a = -10$, we have $e^2 - e^{-10} \leq 12e^2$; when $b = 2$ and $a = -100$, we only get $e^2 - e^{-100} \leq 102e^2$. Although $e^{-100} \approx e^{-10}$, the two upper bounds are off from each other by a factor of 10. Therefore, when necessary, we can 'elevate' a to some higher value in order to obtain a tighter upper bound. We defer the detailed proof to Appendix E.

D.2 Step 2: Gradient Truncation

Let us separate the indices $i \in [n]$ into large and small ones.

Definition D.6. *We make the following definitions.*

- *Let* $\xi_k \in [-\beta, 1]^n$ *be the* truncated gradient *so that* $\xi_{k,i} = \mathbb{T}^{\mathsf{c}}(\nabla_i f_\mu(x_k))$ *for each* $i \in [n]$.
- *Let* $B_k \stackrel{\text{def}}{=} \{i \in [n] \,:\, \xi_{k,i} \neq \nabla_i f_\mu(\mathsf{x}_k)\}$ *be the set of* large indices.
- *Let* $\eta_k \in (-\infty, 0]^n$ *be the* large gradient *so that* $\nabla f_\mu(\mathsf{x}_k) = \xi_k + \eta_k$. *It is clear that*
$$\eta_{k,i} = 0 \text{ for every } i \notin B, \text{ and } \eta_{k,i} = (1+\beta) - (A^T p(\mathsf{x}_k))_i \text{ for every } i \in B.$$
- *Let* $\widetilde{\eta}_k \in (-\infty, \infty)^n$ *be the* adjusted large gradient *so that*
$$\widetilde{\eta}_{k,i} = 0 \text{ for every } i \notin B, \text{ and } \widetilde{\eta}_{k,i} = (1+\beta) - (\widetilde{A}^T p(\mathsf{x}_k))_i \text{ for every } i \in B.$$

For the rest of this section, we denote by $\eta_k^{(i)} = (0, \dots, 0, \eta_{k,i}, 0, \dots, 0)$*, the vector that is zero at all coordinates other than i, and equals to* $\eta_{k,i}$ *at location i. We similarly define* $\xi_k^{(i)}$ *as well as* $\widetilde{\eta}_k^{(i)}$.

We next state the following key lemma that is very analogous to (3.1) from packing LP. Note that if one uses $\eta_k^{(i)}$ instead of $\widetilde{\eta}_k^{(i)}$, the proof becomes identical to that of (3.1). The reason that we can use $\widetilde{\eta}_k^{(i)}$ rather than $\eta_k^{(i)}$ —thus giving a stricter upper bound— is precisely due to the distance adjustment introduced in Lemma D.5.

Lemma D.7.
$$f_\mu(\mathsf{x}_k) - f_\mu(u^*) \leq \frac{(1-\tau)}{\tau}(f_\mu(\mathsf{y}_{k-1}) - f_\mu(\mathsf{x}_k)) + \mathbf{E}_i\Big[\langle n\xi_k^{(i)}, \mathsf{z}_{k-1} - u^* \rangle\Big] + \mathbf{E}_i\Big[\langle n\widetilde{\eta}_k^{(i)}, -u^* \rangle\Big] + \varepsilon\mathsf{OPT} \enspace .$$

The proof of the above lemma is a simple repetition of that of (3.1), but replacing the classical distance upper bound with our adjusted one. See Appendix E for details.

D.3 Step 3: Mirror Descent Guarantee

Our update $\mathsf{z}_k^{(i)} \stackrel{\text{def}}{=} \arg\min_{z \in \Delta} \big\{ V_{\mathsf{z}_{k-1}}(z) + \langle (1+\gamma) n \alpha_k \xi_k^{(i)}, z \rangle \big\}$ is known as a mirror descent step from optimization theory.

We begin by explaining an attempt that is too weak for obtaining the $\varepsilon^{-1.5}$ convergence rate.

Using the classical theory of mirror descent (or multiplicative weight update), it is not hard to repeat the proof of Lemma 3.5 —although changing the distance function from $\|\cdot\|_A^2$ to $V_x(y)$— and obtain that, for every $u \in \Delta$,
$$\mathbf{E}_i\big[\alpha_k \langle n\xi_k^{(i)}, \mathsf{z}_{k-1} - u \rangle\big] \leq V_{\mathsf{z}_{k-1}}(u) - \mathbf{E}_i\Big[V_{\mathsf{z}_k^{(i)}}(u)\Big] + O(\alpha_k^2 n)\mathsf{OPT} \enspace .$$
The above inequality can be made true whenever ξ_i is between -1 and 1 for each coordinate i, but only yields the known ε^{-2} convergence rate. Here, ± 1 is also know as the *width* from multiplicative-weight-update languages [AHK12].

Fortunately, since we have required ξ_i to be only between $-\beta$ and 1, the $O(\alpha_k^2 n)$ factor can essentially be improved to $O(\alpha_k^2 \beta n)$. This is an improvement whenever $\beta \ll 1$, and we call it the *negative-width technique*.[13] Formally, we prove that

Lemma D.8. *Denoting by* $\gamma \stackrel{\text{def}}{=} 2\alpha_T n$*, we have*

$$\mathbf{E}_i\big[\alpha_k\big\langle n\xi_k^{(i)}, \mathsf{z}_{k-1} - u^*\big\rangle\big] \le V_{\mathsf{z}_{k-1}}\big(\frac{u^*}{1+\gamma}\big) - \mathbf{E}_i\Big[V_{\mathsf{z}_k^{(i)}}\big(\frac{u^*}{1+\gamma}\big)\Big] + 12\mathsf{OPT}\cdot\gamma\alpha_k\beta \ .$$

The proof can be found in Appendix E.

Although defined in a variational way, it is perhaps beneficial to explicitly describe how to implement this mirror step. The following proposition is straightforward but anyways proved in Appendix E:

Proposition D.9. *If* $\mathsf{z}_{k-1} \in \Delta$ *and* $\mathsf{z}_{k-1} > 0$*, the minimizer* $z = \arg\min_{z\in\Delta}\big\{V_{\mathsf{z}_{k-1}}(z) + \langle\delta\mathbf{e}_i, z\rangle\big\}$ *for any scalar* $\delta \in \mathbb{R}$ *and basis vector* $\mathbf{e}_i$ *can be computed as follows:*

1. $z \leftarrow \mathsf{z}_{k-1}$.
2. $z_i \leftarrow z_i \cdot e^{-\delta}$.
3. *If* $\mathbb{1}^T z > 2\mathsf{OPT}'$, $z \leftarrow \frac{2\mathsf{OPT}'}{\mathbb{1}^T z} z$.
4. *Return* z.

D.4 Step 4: Gradient Descent Guarantee

We claim that our gradient step $\mathsf{x}_k \to \mathsf{y}_k^{(i)}$ never increases the objective for all choices of i. In addition, it decreases the objective by an amount proportional to the adjusted large gradient $\widetilde{\eta}_k^{(i)}$.

Lemma D.10. *For every* $i \in [n]$*, we have*

(a) $f_\mu(\mathsf{x}_k) - f_\mu(\mathsf{y}_k^{(i)}) \ge 0$*, and*

(b) $f_\mu(\mathsf{x}_k) - f_\mu(\mathsf{y}_k^{(i)}) \ge \frac{\mu\beta}{12}\cdot\langle -\widetilde{\eta}_k^{(i)}, u^*\rangle$.

The proof of Lemma D.10 is quite technical and can be found in Appendix E.

At high level, one would generally hope to prove that the gradient step decreases the objective by an amount proportional to the large gradient $\eta_k^{(i)}$, rather than the adjusted large gradient $\widetilde{\eta}_k^{(i)}$. If that were true, the entire proof structure of our covering LP convergence would become much closer to that of packing LP, and there would be absolutely no need for the introduction of the distance adjustment in Section D.1, as well as the definitions of $\widetilde{A}$ and $\widetilde{\eta}$.

Unfortunately, if one replaces $\widetilde{\eta}$ with η in the above lemma, the inequality is *far* from being correct. The reason behind it is very similar to that we have summarized in Section D.1, and related to the unpleasant behavior of the exponential penalty function.

D.5 Step 5: Putting All Together

Combining Lemma D.7, Lemma D.8, and Lemma D.10, we obtain that

$$\begin{aligned}
&\alpha_k\big(f_\mu(\mathsf{x}_k) - f_\mu(u^*)\big) - \alpha_k\varepsilon\mathsf{OPT} \\
&\le \frac{(1-\tau)\alpha_k}{\tau}(f_\mu(\mathsf{y}_{k-1}) - f_\mu(\mathsf{x}_k)) + \mathbf{E}_i\Big[\alpha_k\langle n\xi_k^{(i)}, \mathsf{z}_{k-1} - u^*\rangle\Big] + \mathbf{E}_i\Big[\alpha_k\langle n\widetilde{\eta}_k^{(i)}, -u^*\rangle\Big] \\
&\le \frac{(1-\tau)\alpha_k}{\tau}(f_\mu(\mathsf{y}_{k-1}) - f_\mu(\mathsf{x}_k)) + V_{\mathsf{z}_{k-1}}\big(\frac{u^*}{1+\gamma}\big) - \mathbf{E}_i\Big[V_{\mathsf{z}_k^{(i)}}\big(\frac{u^*}{1+\gamma}\big)\Big]
\end{aligned}$$

[13]This negative width technique is strongly related to [AHK12, Definition 3.2], where the authors analyze the classical multiplicative weight update method in a special case when the oracle returns loss values only between $-\ell$ and ρ, for $\ell \ll \rho$. This technique is in fact related to a more general theory of mirror descent, known as the local-norm convergence, that we have summarized in a separate paper [ALO14].

$$+ 12\mathsf{OPT} \cdot \gamma\alpha_k\beta + \mathbf{E}_i\Big[\frac{12\alpha_k n}{\mu\beta}\big(f_\mu(\mathsf{x}_k) - f_\mu(\mathsf{y}_k^{(i)})\big)\Big]$$

Remark D.11. *Above, the quantity "$12\mathsf{OPT}\cdot\gamma\alpha_k\beta$" is the loss term introduced by the mirror descent. Unlike the packing LP case —see (3.2)— this loss term is not dominated by the gradient step. (If one could do so, this would turn our* `CovLPSolver` *into an ε^{-1} convergence rate.)*

The quantity "$\alpha_k\langle n\xi_k^{(i)}, \mathsf{z}_{k-1} - u^*\rangle$" is the loss introduced by the (adjusted) large gradient $\widetilde{\eta}$, and is dominated by our gradient step progress owing to Lemma D.10. This is similar to the packing LP case —see Lemma 3.9.

From here, let us use the special choice of $\tau = \frac{\mu\beta}{12n}$. We obtain that

$$\begin{aligned}&-\alpha_k\big(f_\mu(u^*) + \varepsilon\mathsf{OPT}\big)\\ &\leq 12\gamma\alpha_k\beta\mathsf{OPT} + \frac{(1-\tau)\alpha_k}{\tau}f_\mu(\mathsf{y}_{k-1}) + V_{\mathsf{z}_{k-1}}\big(\frac{u^*}{1+\gamma}\big) - \mathbf{E}_i\Big[\frac{\alpha_k}{\tau}f_\mu(\mathsf{y}_k^{(i)}) + V_{\mathsf{z}_k^{(i)}}\big(\frac{u^*}{1+\gamma}\big)\Big] \ .\end{aligned}$$

Use the choice $\alpha_k = \frac{\alpha_{k-1}}{1-\tau}$ and telescoping the above inequality for $k = 1, \dots, T$, we have

$$-\Big(\sum_{k=1}^T \alpha_k\Big)\big(f_\mu(u^*) + \varepsilon\mathsf{OPT}\big) \leq \Big(\sum_{k=1}^T \alpha_k\Big)\cdot 12\gamma\beta\mathsf{OPT} + \frac{\alpha_0}{\tau}f_\mu(\mathsf{y}_0) + V_{\mathsf{z}_0}\big(\frac{u^*}{1+\gamma}\big) - \frac{\alpha_T}{\tau}\mathbf{E}\big[f_\mu(\mathsf{y}_T)\big] \ .$$

We compute that $\sum_{k=1}^T \alpha_k = \alpha_T \cdot \sum_{k=0}^{T-1}(1-\tau)^k = \alpha_T \cdot \frac{1-(1-\tau)^T}{\tau} < \frac{\alpha_T}{\tau}$, and recall that $\gamma = 2\alpha_T n$. Therefore, we rearrange and get

$$\begin{aligned}&\frac{\alpha_T}{\tau}\mathbf{E}\big[f_\mu(\mathsf{y}_T)\big] \leq \frac{\alpha_T}{\tau}\big(f_\mu(u^*) + \varepsilon\mathsf{OPT}\big) + \frac{\alpha_T}{\tau}\cdot 12\gamma\beta\mathsf{OPT} + \frac{\alpha_0}{\tau}f_\mu(\mathsf{y}_0) + V_{\mathsf{z}_0}\big(\frac{u^*}{1+\gamma}\big) \ ,\\ &\implies \mathbf{E}\big[f_\mu(\mathsf{y}_T)\big] \leq f_\mu(u^*) + \varepsilon\mathsf{OPT} + 24\alpha_T n\beta\mathsf{OPT} + (1-\tau)^T f_\mu(\mathsf{y}_0) + \frac{\tau}{\alpha_T}V_{\mathsf{z}_0}\big(\frac{u^*}{1+\gamma}\big) \ . \qquad \text{(D.1)}\end{aligned}$$

From this point, we need to use our special choice of the initial point $\mathsf{x}_0 = \mathsf{y}_0 = \mathsf{z}_0 = x^{\mathsf{start}}$ (see Proposition C.2.d), which implies that $f_\mu(\mathsf{y}_0) \leq 4\mathsf{OPT}$ and $\mathbb{1}^T x^{\mathsf{start}} \leq 4\mathsf{OPT}$. We also have

$$\begin{aligned}V_{\mathsf{z}_0}\big(\frac{u^*}{1+\gamma}\big) = V_{x^{\mathsf{start}}}\big(\frac{u^*}{1+\gamma}\big) &= \sum_{i=1}^n \frac{u_i^*}{1+\gamma}\log\frac{u_i^*}{(1+\gamma)x_i^{\mathsf{start}}} + x_i^{\mathsf{start}} - \frac{u_i^*}{1+\gamma}\\ &\overset{①}{\leq} \sum_{i=1}^n u_i^*\log(u_i^*\cdot n) + 4\mathsf{OPT} \overset{②}{\leq} (2\log(nm)+4)\cdot\mathsf{OPT} \ .\end{aligned}$$

Above, inequality ① follows because $x_i^{\mathsf{start}} \geq 1/n$ for all $i \in [n]$ according to the definition in Proposition C.2.d; inequality ② follows because $u_i^* \leq (1+\varepsilon/2)x_i^* \leq (1+\varepsilon/2)\mathsf{OPT} \leq (1+\varepsilon/2)m$ and $\mathbb{1}^T u_i^* = (1+\varepsilon/2)\mathsf{OPT}$, as well as the fact that ε is sufficiently small.

Finally, we choose $\beta = \sqrt{\varepsilon}$, $\alpha_T = \frac{\varepsilon}{12n\beta}$, and $T = \lceil\frac{1}{\tau}\log(1/\varepsilon)\rceil$. Substituting into (D.1) all of these parameters, along with the aforementioned inequalities $f_\mu(\mathsf{y}_0) \leq 4\mathsf{OPT}$ and $V_{\mathsf{z}_0}\big(\frac{u^*}{1+\gamma}\big) \leq (2\log(nm)+4)\cdot\mathsf{OPT}$, as well as $f_\mu(u^*) \leq (1+\varepsilon)\mathsf{OPT}$ from Proposition C.2.b, we obtain that

$$\mathbf{E}\big[f_\mu(\mathsf{y}_T)\big] \leq (1+\varepsilon)\mathsf{OPT} + \varepsilon\mathsf{OPT} + 2\varepsilon\mathsf{OPT} + \varepsilon f_\mu(\mathsf{y}_0) + \frac{\mu\beta/12n}{\varepsilon/12n\beta}(2\log(nm)+4)\mathsf{OPT} = (1+9\varepsilon)\mathsf{OPT} \ .$$

This finishes the proof of Theorem D.3. □

It is now straightforward to use Markov inequality to turn the expected guarantee in Theorem D.3 into a probabilistic one:

Corollary D.12. *With probability at least* $9/10$, `CovLPSolver`$(A, x^{\mathsf{start}}, \varepsilon)$ *outputs a $(1+O(\varepsilon))$ approximate solution to the covering LP program. The expected running time is $O(\frac{\log(nm/\varepsilon)\log(1/\varepsilon)}{\varepsilon^{1.5}}N)$.*

Proof. Since for every $x \in \Delta$ it satisfies $f_\mu(x) \geq (1-\varepsilon)\mathsf{OPT}$ according to Proposition C.2.c, we obtain that $f_\mu(y_T) - (1-\varepsilon)\mathsf{OPT}$ is a random variable that is non-negative, whose expectation $\mathbf{E}[f_\mu(y_T) - (1-\varepsilon)\mathsf{OPT}] \leq 10\varepsilon$. By Markov bound, with at least probability 9/10, we obtain some y_T satisfying $f_\mu(y_T) \leq (1+O(\varepsilon))\mathsf{OPT}$, which yields some $(1+O(\varepsilon))$ approximate solution according to Proposition C.2.f.

The running time follows from our efficient implementation in Section G. □

E Missing Proofs for Section D

Lemma D.5.

$$\begin{aligned} f_\mu(\mathsf{x}_k) - f_\mu(u^*) &\leq \langle \mathbb{1} - A^T p(\mathsf{x}_k), \mathsf{x}_k - u^* \rangle + \langle \widetilde{A}^T p(\mathsf{x}_k) - A^T p(\mathsf{x}_k), u^* \rangle + \varepsilon\mathsf{OPT} \\ &= \langle \nabla f_\mu(\mathsf{x}_k), \mathsf{x}_k - u^* \rangle + \langle \widetilde{A}^T p(\mathsf{x}_k) - A^T p(\mathsf{x}_k), u^* \rangle + \varepsilon\mathsf{OPT} \end{aligned}$$

Proof.

$$\begin{aligned} f_\mu(\mathsf{x}_k) - f_\mu(u^*) &= \mu \sum_{j=1}^m \Big(\exp^{\frac{1}{\mu}(1-(A\mathsf{x}_k)_j)} - \exp^{\frac{1}{\mu}(1-(Au^*)_j)} \Big) + \langle \mathbb{1}, \mathsf{x}_k - u^* \rangle \\ &\overset{①}{\leq} \mu \sum_{j=1}^m \Big(\exp^{\frac{1}{\mu}(1-(A\mathsf{x}_k)_j)} - \exp^{\frac{1}{\mu}(1-(\widetilde{A}u^*)_j)} \Big) + \langle \mathbb{1}, \mathsf{x}_k - u^* \rangle + \mu \cdot m \cdot \exp^{-1/\mu} \\ &\overset{②}{\leq} \sum_{j=1}^m \exp^{\frac{1}{\mu}(1-(A\mathsf{x}_k)_j)} \cdot \big((\widetilde{A}u^*)_j - (A\mathsf{x}_k)_j \big) + \langle \mathbb{1}, \mathsf{x}_k - u^* \rangle + \varepsilon\mathsf{OPT} \\ &= \sum_{j=1}^m p_j(\mathsf{x}_k) \cdot \big((\widetilde{A}u^*)_j - (A\mathsf{x}_k)_j \big) + \langle \mathbb{1}, \mathsf{x}_k - u^* \rangle + \varepsilon\mathsf{OPT} \\ &= \sum_{j=1}^m p_j(\mathsf{x}_k) \cdot \big((Au^*)_j - (A\mathsf{x}_k)_j \big) + \langle \mathbb{1}, \mathsf{x}_k - u^* \rangle + \sum_{j=1}^m p_j(\mathsf{x}_k) \cdot \big((\widetilde{A}u^*)_j - (Au^*)_j \big) + \varepsilon\mathsf{OPT} \\ &= \langle -Ap(\mathsf{x}_k), \mathsf{x}_k - u^* \rangle + \langle \mathbb{1}, \mathsf{x}_k - u^* \rangle + \langle \widetilde{A}^T p(\mathsf{x}_k) - A^T p(\mathsf{x}_k), u^* \rangle + \varepsilon\mathsf{OPT} \ . \end{aligned}$$

Above, ① is because if $(Au^*)_j \neq (\widetilde{A}u^*)_j$ for some j, then it must satisfy that $(\widetilde{A}u^*)_j = 2$, and therefore $-\exp^{\frac{1}{\mu}(1-(Au^*)_j)} \leq -\exp^{\frac{1}{\mu}(1-(\widetilde{A}u^*)_j)} + \exp^{-1/\mu}$. ② uses the convexity inequality of $e^b - e^a \leq \langle e^b, b-a \rangle$, and the fact that $\mu m \exp^{-1/\mu} \ll \varepsilon\mathsf{OPT}$. □

Lemma D.7.

$$f_\mu(\mathsf{x}_k) - f_\mu(u^*) \leq \frac{(1-\tau)}{\tau}(f_\mu(\mathsf{y}_{k-1}) - f_\mu(\mathsf{x}_k)) + \mathbf{E}_i\Big[\langle n\xi_k^{(i)}, \mathsf{z}_{k-1} - u^* \rangle\Big] + \mathbf{E}_i\Big[\langle n\widetilde{\eta}_k^{(i)}, -u^* \rangle\Big] + \varepsilon\mathsf{OPT} \ .$$

Proof.

$$\begin{aligned} &\big(f_\mu(\mathsf{x}_k) - f_\mu(u^*)\big) - \varepsilon\mathsf{OPT} \\ &\overset{①}{\leq} \langle \nabla f_\mu(\mathsf{x}_k), \mathsf{x}_k - u^* \rangle + \langle \widetilde{A}^T p(\mathsf{x}_k) - A^T p(\mathsf{x}_k), u^* \rangle \\ &= \langle \nabla f_\mu(\mathsf{x}_k), \mathsf{x}_k - \mathsf{z}_{k-1} \rangle + \langle \nabla f_\mu(\mathsf{x}_k), \mathsf{z}_{k-1} - u^* \rangle + \langle \widetilde{A}^T p(\mathsf{x}_k) - A^T p(\mathsf{x}_k), u^* \rangle \\ &\overset{②}{=} \frac{(1-\tau)}{\tau} \langle \nabla f_\mu(\mathsf{x}_k), \mathsf{y}_{k-1} - \mathsf{x}_k \rangle + \langle \nabla f_\mu(\mathsf{x}_k), \mathsf{z}_{k-1} - u^* \rangle + \langle \widetilde{A}^T p(\mathsf{x}_k) - A^T p(\mathsf{x}_k), u^* \rangle \\ &\overset{③}{\leq} \frac{(1-\tau)}{\tau}(f_\mu(\mathsf{y}_{k-1}) - f_\mu(\mathsf{x}_k)) + \langle \nabla f_\mu(\mathsf{x}_k), \mathsf{z}_{k-1} - u^* \rangle + \langle \widetilde{A}^T p(\mathsf{x}_k) - A^T p(\mathsf{x}_k), u^* \rangle \\ &= \frac{(1-\tau)}{\tau}(f_\mu(\mathsf{y}_{k-1}) - f_\mu(\mathsf{x}_k)) + \langle \xi_k + \eta_k, \mathsf{z}_{k-1} - u^* \rangle + \langle \widetilde{A}^T p(\mathsf{x}_k) - A^T p(\mathsf{x}_k), u^* \rangle \end{aligned}$$

$$\overset{④}{\le} \frac{(1-\tau)}{\tau}(f_\mu(\mathsf{y}_{k-1}) - f_\mu(\mathsf{x}_k)) + \langle \xi_k, \mathsf{z}_{k-1} - u^* \rangle + \langle \widetilde{A}^T p(\mathsf{x}_k) - A^T p(\mathsf{x}_k) - \eta_k, u^* \rangle$$
$$\overset{⑤}{\le} \frac{(1-\tau)}{\tau}(f_\mu(\mathsf{y}_{k-1}) - f_\mu(\mathsf{x}_k)) + \langle \xi_k, \mathsf{z}_{k-1} - u^* \rangle + \langle -\widetilde{\eta}_k, u^* \rangle$$
$$= \frac{(1-\tau)}{\tau}(f_\mu(\mathsf{y}_{k-1}) - f_\mu(\mathsf{x}_k)) + \mathbf{E}_i\Big[\langle n\xi_k^{(i)}, \mathsf{z}_{k-1} - u^* \rangle + \langle -n\widetilde{\eta}_k^{(i)}, u^* \rangle\Big] \ .$$

Above, ① is due to Lemma D.5. ② is because $\mathsf{x}_k = \tau \mathsf{z}_{k-1} + (1-\tau)\mathsf{y}_{k-1}$, which implies that $\tau(\mathsf{x}_k - \mathsf{z}_{k-1}) = (1-\tau)(\mathsf{y}_{k-1} - \mathsf{x}_k)$. ③ is by the convexity of $f_\mu(\cdot)$. ④ is because $\langle \eta_k, \mathsf{z}_{k-1} \rangle \le 0$, since $\eta_k \le 0$ while $\mathsf{z}_{k-1} \ge 0$.

⑤ needs some careful justification: for every $i \notin B_k$, we have $(\widetilde{A}^T p(\mathsf{x}_k) - A^T p(\mathsf{x}_k))_i - \eta_{k,i} \le 0 - 0 = -\widetilde{\eta}_{k,i}$; for every $i \in B_k$, we have

$$\begin{aligned}(\widetilde{A}^T p(\mathsf{x}_k) - A^T p(\mathsf{x}_k))_i - \eta_{k,i} &= (\widetilde{A}^T p(\mathsf{x}_k) - A^T p(\mathsf{x}_k))_i - \big((1+\beta) - (A^T p(\mathsf{x}_k))_i\big) \\ &= -\big((1+\beta) - (\widetilde{A}^T p(\mathsf{x}_k))_i\big) = -\widetilde{\eta}_{k,i} \ ,\end{aligned}$$

where the two equalities follow from the definitions of $\eta_{k,i}$ and $\widetilde{\eta}_{k,i}$ (see Definition D.6). □

Lemma D.8. *Denoting by $\gamma \overset{\text{def}}{=} 2\alpha_T n$, we have*

$$\mathbf{E}_i\big[\alpha_k \langle n\xi_k^{(i)}, \mathsf{z}_{k-1} - u^* \rangle\big] \le V_{\mathsf{z}_{k-1}}\big(\frac{u^*}{1+\gamma}\big) - \mathbf{E}_i\Big[V_{\mathsf{z}_k^{(i)}}\big(\frac{u^*}{1+\gamma}\big)\Big] + 12\mathsf{OPT} \cdot \gamma\alpha_k\beta \ .$$

Proof. Define $w(x) \overset{\text{def}}{=} \sum_i x_i \log(x_i) - x_i$ and accordingly, $V_x(y) = w(y) - \langle w'(x), y - x \rangle - w(x) = \sum_i y_i \log\frac{y_i}{x_i} + x_i - y_i$. We first compute using the classical analysis of mirror descent step as follows:

$$\begin{aligned}
&\gamma\alpha_k\big\langle n\xi_k^{(i)}, \mathsf{z}_{k-1}\big\rangle + \alpha_k\big\langle n\xi_k^{(i)}, \mathsf{z}_{k-1} - u^*\big\rangle \\
&= (1+\gamma)\alpha_k\Big\langle n\xi_k^{(i)}, \mathsf{z}_k^{(i)} - \frac{u^*}{1+\gamma}\Big\rangle + (1+\gamma)\alpha_k\big\langle n\xi_k^{(i)}, \mathsf{z}_{k-1} - \mathsf{z}_k^{(i)}\big\rangle \\
&\overset{①}{\le} \Big\langle w'(\mathsf{z}_{k-1}) - w'(\mathsf{z}_k^{(i)}), \mathsf{z}_k^{(i)} - \frac{u^*}{1+\gamma}\Big\rangle + (1+\gamma)\alpha_k\big\langle n\xi_k^{(i)}, \mathsf{z}_{k-1} - \mathsf{z}_k^{(i)}\big\rangle \\
&= \Big(w\big(\frac{u^*}{1+\gamma}\big) - w(\mathsf{z}_{k-1}) - \Big\langle w'(\mathsf{z}_{k-1}), \frac{u^*}{1+\gamma} - \mathsf{z}_{k-1}\Big\rangle\Big) - \Big(w\big(\frac{u^*}{1+\gamma}\big) - w(\mathsf{z}_k^{(i)}) - \Big\langle w'(\mathsf{z}_k^{(i)}), \frac{u^*}{1+\gamma} - \mathsf{z}_k^{(i)}\Big\rangle\Big) \\
&\quad + \Big(w(\mathsf{z}_{k-1}) - w(\mathsf{z}_k^{(i)}) - \big\langle w'(\mathsf{z}_{k-1}), \mathsf{z}_{k-1} - \mathsf{z}_k^{(i)}\big\rangle\Big) + (1+\gamma)\alpha_k\big\langle n\xi_k^{(i)}, \mathsf{z}_{k-1} - \mathsf{z}_k^{(i)}\big\rangle \\
&= V_{\mathsf{z}_{k-1}}\big(\frac{u^*}{1+\gamma}\big) - V_{\mathsf{z}_k^{(i)}}\big(\frac{u^*}{1+\gamma}\big) + \boxed{(1+\gamma)\alpha_k\big\langle n\xi_k^{(i)}, \mathsf{z}_{k-1} - \mathsf{z}_k^{(i)}\big\rangle - V_{\mathsf{z}_{k-1}}(\mathsf{z}_k^{(i)})} \ .
\end{aligned} \tag{E.1}$$

Above, ① is because $\mathsf{z}_k^{(i)} = \arg\min_{z\in\Delta}\big\{V_{\mathsf{z}_{k-1}}(z) + \langle (1+\gamma)\alpha_k n\xi_k^{(i)}, z \rangle\big\}$, which is equivalent to saying

$$\begin{aligned} &\forall u \in \Delta, \quad \langle V'_{\mathsf{z}_{k-1}}(\mathsf{z}_k^{(i)}) + (1+\gamma)\alpha_k n\xi_k^{(i)}, u - \mathsf{z}_k^{(i)} \rangle \ge 0 \\ \iff &\forall u \in \Delta, \quad \langle w'(\mathsf{z}_k^{(i)}) - w'(\mathsf{z}_{k-1}) + (1+\gamma)\alpha_k n\xi_k^{(i)}, u - \mathsf{z}_k^{(i)} \rangle \ge 0 \ .\end{aligned}$$

In particular, we have $\mathbb{1}^T \frac{u^*}{1+\gamma} = \mathbb{1}^T \frac{(1+\varepsilon/2)x^*}{1+\gamma} < 2\mathsf{OPT} \le 2\mathsf{OPT}'$ and therefore substituting $u = \frac{u^*}{1+\gamma} \in \Delta$ into the above inequality we get ①.

Next, we upper bound the term in the box:

$$\begin{aligned}&(1+\gamma)\alpha_k\langle n\xi_k^{(i)}, \mathsf{z}_{k-1} - \mathsf{z}_k^{(i)} \rangle - V_{\mathsf{z}_{k-1}}(\mathsf{z}_k^{(i)}) \\ &\overset{①}{\le} (1+\gamma)\alpha_k n\xi_{k,i} \cdot (\mathsf{z}_{k-1,i} - \mathsf{z}_{k,i}^{(i)}) - \Big(\mathsf{z}_{k,i}^{(i)} \log\frac{\mathsf{z}_{k,i}^{(i)}}{\mathsf{z}_{k-1,i}} + \mathsf{z}_{k-1,i} - \mathsf{z}_{k,i}^{(i)}\Big)\end{aligned}$$

$$
\begin{aligned}
&\overset{②}{\le} (1+\gamma)\alpha_k n\xi_{k,i}\cdot(\mathsf{z}_{k-1,i}-\mathsf{z}^{(i)}_{k,i}) - \frac{|\mathsf{z}^{(i)}_{k,i}-\mathsf{z}_{k-1,i}|^2}{2\max\{\mathsf{z}^{(i)}_{k,i},\mathsf{z}_{k-1,i}\}}\\
&\overset{③}{\le} (1+\gamma)\alpha_k n\xi_{k,i}\cdot(\mathsf{z}_{k-1,i}-\mathsf{z}^{(i)}_{k,i}) - \frac{|\mathsf{z}^{(i)}_{k,i}-\mathsf{z}_{k-1,i}|^2}{4\mathsf{z}_{k-1,i}}\\
&\overset{④}{\le} (1+\gamma)^2\mathsf{z}_{k-1,i}\cdot(\alpha_k n\xi_{k,i})^2 \overset{⑤}{\le} 2\mathsf{z}_{k-1,i}\cdot(\alpha_k n\xi_{k,i})^2 \overset{⑥}{\le} \mathsf{z}_{k-1,i}\cdot\gamma\alpha_k n|\xi_{k,i}|\\
&\overset{⑦}{\le} \mathsf{z}_{k-1,i}\cdot\gamma\alpha_k n\xi_{k,i} + 2\mathsf{z}_{k-1,i}\cdot\gamma\alpha_k n\beta = \gamma\alpha_k\langle n\xi^{(i)}_k,\mathsf{z}_{k-1}\rangle + 2\mathsf{z}_{k-1,i}\cdot\gamma\alpha_k n\beta\ . \qquad \text{(E.2)}
\end{aligned}
$$

Above, ① uses the fact that for every $i'\neq i$, $\mathsf{z}^{(i)}_{k,i'}\log\frac{\mathsf{z}^{(i)}_{k,i'}}{\mathsf{z}_{k-1,i'}}+\mathsf{z}_{k-1,i'}-\mathsf{z}^{(i)}_{k,i}\ge 0$. ② uses the inequality that for every $a,b>0$, we have $a\log\frac{a}{b}+b-a\ge\frac{(a-b)^2}{2\max\{a,b\}}$.[14] ③ uses the fact that $\mathsf{z}^{(i)}_{k,i}\le 2\mathsf{z}_{k-1,i}$.[15] ④ uses Cauchy-Shwarz: $ab-b^2/4\le a^2$. ⑤ uses $(1+\gamma)^2<2$. ⑥ uses $|\xi_{k,i}|\le 1$ and $\gamma=2\alpha_T n\ge 2\alpha_k n$. ⑦ uses $\xi_{k,i}\ge-\beta$.

Next, we combine (E.1) and (E.2) to conclude that

$$
\alpha_k\big\langle n\xi^{(i)}_k,\mathsf{z}_{k-1}-u^*\big\rangle \le V_{\mathsf{z}_{k-1}}\big(\frac{u^*}{1+\gamma}\big)-V_{\mathsf{z}^{(i)}_k}\big(\frac{u^*}{1+\gamma}\big)+2\mathsf{z}_{k-1,i}\cdot\gamma\alpha_k n\beta\ .
$$

Taking expectation on both sides with respect to i, and using the property that $\mathbb{1}^T\mathsf{z}_{k-1}\le 3\mathsf{OPT}'\le 6\mathsf{OPT}$, we obtain that

$$
\mathbf{E}_i\big[\alpha_k\big\langle n\xi^{(i)}_k,\mathsf{z}_{k-1}-u^*\big\rangle\big] \le V_{\mathsf{z}_{k-1}}\big(\frac{u^*}{1+\gamma}\big)-\mathbf{E}_i\Big[V_{\mathsf{z}^{(i)}_k}\big(\frac{u^*}{1+\gamma}\big)\Big]+12\mathsf{OPT}\cdot\gamma\alpha_k\beta\ . \qquad \square
$$

Proposition D.9. *If $\mathsf{z}_{k-1}\in\Delta$ and $\mathsf{z}_{k-1}>0$, the minimizer $z=\arg\min_{z\in\Delta}\big\{V_{\mathsf{z}_{k-1}}(z)+\langle\delta\mathbf{e}_i,z\rangle\big\}$ for any scalar $\delta\in\mathbb{R}$ and basis vector $\mathbf{e}_i$ can be computed as follows:*

1. *$z\leftarrow\mathsf{z}_{k-1}$.*
2. *$z_i\leftarrow z_i\cdot e^{-\delta}$.*
3. *If $\mathbb{1}^Tz>2\mathsf{OPT}'$, $z\leftarrow\frac{2\mathsf{OPT}'}{\mathbb{1}^Tz}z$.*
4. *Return z.*

Proof. Let us denote by z the returned value of the described procedure, and $g(u)\overset{\text{def}}{=}V_{\mathsf{z}_{k-1}}(u)+\langle\delta\mathbf{e}_i,u\rangle$. Since Δ is a convex body and $g(\cdot)$ is convex, to show $z=\arg\min_{z\in\Delta}\{g(u)\}$, it suffices for us to prove that for every $u\in\Delta$, $\langle\nabla g(z),u-z\rangle\ge 0$. Since the gradient $\nabla g(z)$ can be written explicitly, this is equivalent to

$$
\delta(u_i-z_i)+\sum_{\ell=1}^n\log\frac{z_\ell}{\mathsf{z}_{k-1,\ell}}\cdot(u_\ell-z_\ell)\ge 0\ .
$$

If the re-scaling in step 3 is not executed, then we have $z_\ell=\mathsf{z}_{k-1,\ell}$ for every $\ell\neq i$, and $z_i=\mathsf{z}_{k-1,i}\cdot e^{-\delta}$; thus, the left-hand side is zero so the above inequality is true for every $u\in\Delta$.

Otherwise, we have $\mathbb{1}^Tz=2\mathsf{OPT}'$ and there exists some constant factor $Z>1$ such that, $z_\ell=\mathsf{z}_{k-1,\ell}/Z$ for every $\ell\neq i$, and $z_i=\mathsf{z}_{k-1,i}\cdot e^{-\delta}/Z$. In such a case, the left-hand side equals to

$$
(u_i-z_i)\cdot(\delta-\delta)+\sum_{\ell=1}^n-\log Z\cdot(u_\ell-z_\ell)\ .
$$

[14]This inequality in fact corresponds to a local strong convexity property of $w(\cdot)$. We have used this technique in our paper [AO15].

[15]This is because, our parameter choices ensure that $(1+\gamma)\alpha_k n<1/2\beta$, which further means $-(1+\gamma)\alpha_k n\xi^{(i)}_k\le 1/2$. As a result, we must have $\mathsf{z}^{(i)}_{k,i}\le\mathsf{z}_{k-1,i}\cdot e^{0.5}<2\mathsf{z}_{k-1,i}$ (see the explicit definition of the mirror step at Proposition D.9).

It is clear at this moment that since $\log Z > 0$ and $\mathbb{1}^T u \le 2\mathsf{OPT}' = \mathbb{1}^T z$, the above quantity is always non-negative, finishing the proof. $\square$

Lemma D.10. *For every $i \in [n]$, we have*

(a) $f_\mu(\mathsf{x}_k) - f_\mu(\mathsf{y}_k^{(i)}) \ge 0$, *and*

(b) $f_\mu(\mathsf{x}_k) - f_\mu(\mathsf{y}_k^{(i)}) \ge \frac{\mu\beta}{12} \cdot \langle -\widetilde{\eta}_k^{(i)}, u^* \rangle$.

Proof of Lemma D.10 part (a). Since if $i \notin B_k$ is not a large index we have $\mathsf{y}_k^{(i)} = \mathsf{x}_k$ and the claim is trivial, we focus on $i \in B_k$ in the remaining proof. Recall that $\mathsf{y}_k^{(i)} = \mathsf{x}_k + \delta \mathbf{e}_i$ for some $\delta > 0$ defined in Algorithm 2, so we have

$$f_\mu(\mathsf{x}_k) - f_\mu(\mathsf{y}_k^{(i)}) = \int_{\tau=0}^{\delta} \langle -\nabla f_\mu(\mathsf{x}_k + \tau \mathbf{e}_i), \mathbf{e}_i \rangle d\tau = \int_{\tau=0}^{\delta} \big(\langle A_{:i}, p(\mathsf{x}_k + \tau \mathbf{e}_i)\rangle - 1\big) d\tau \ .$$

It is clear that $\langle A_{:i}, p(\mathsf{x}_k + \tau \mathbf{e}_i)\rangle$ decreases as τ increases, and therefore it suffices to prove that $\langle A_{:i}, p(\mathsf{x}_k + \delta \mathbf{e}_i)\rangle \ge 1$.

Suppose that the rows of $A_{:i}$ are sorted (for the simplicity of notation) by the increasing order of $A_{j,i}$. Now, by the definition of the algorithm, there exists some $j^* \in [m]$ satisfying that

$$\sum_{j<j^*} A_{j,i} \cdot p_j(\mathsf{x}_k) < 1 + \beta \quad \text{and} \quad \sum_{j \le j^*} A_{j,i} \cdot p_j(\mathsf{x}_k) \ge 1 + \beta \ .$$

Next, by our choice of δ which satisfies $\delta = \frac{\mu\beta}{2A_{j^*,i}} \le \frac{\mu\beta}{2A_{j,i}}$ for every $j \le j^*$, we have

$$p_j(\mathsf{x}_k + \delta \mathbf{e}_i) = p_j(\mathsf{x}_k) \cdot \exp^{-\frac{A_{j,i}\delta}{\mu}} \ge p_j(\mathsf{x}_k) \cdot \exp^{-\beta/2} \ge p_j(\mathsf{x}_k) \cdot (1 - \beta/2) \ ,$$

and as a result,

$$\langle A_{:i}, p(\mathsf{x}_k + \delta \mathbf{e}_i) \ge \sum_{j \le j^*} A_{j,i} \cdot p_j(\mathsf{x}_k + \delta \mathbf{e}_i) \ge (1-\beta/2) \sum_{j \le j^*} A_{j,i} \cdot p_j(\mathsf{x}_k) \ge (1-\beta/2)(1+\beta) \ge 1 \ . \ \square$$

Proof of Lemma D.10 part (b). Owing to part (a), for every coordinate i such that $\widetilde{\eta}_{k,i} \ge 0$, we automatically have $f_\mu(\mathsf{x}_k) - f_\mu(\mathsf{y}_k^{(i)}) \ge 0$ so the lemma is obvious. Therefore, let us focus only on coordinates i such that $\widetilde{\eta}_{k,i} < 0$; these are necessarily large indices $i \in B$. Recall from Definition D.6 that $\widetilde{\eta}_{k,i} = (1+\beta) - (\widetilde{A}^T p(\mathsf{x}_k))_i$, so we have

$$\sum_{j=1}^{m} \widetilde{A}_{j,i} \cdot p_j(\mathsf{x}_k) - (1+\beta) > 0 \ .$$

For the simplicity of description, suppose again that the rows of the i-th column is sorted in the non-decreasing order of $A_{j,i}$. That is, $A_{1,i} \le \cdots A_{m,i}$. The definition of j^* can be simplified as

$$\sum_{j<j^*} A_{j,i} \cdot p_j(\mathsf{x}_k) < 1 + \beta \quad \text{and} \quad \sum_{j \le j^*} A_{j,i} \cdot p_j(\mathsf{x}_k) \ge 1 + \beta \ .$$

Let $j^\flat \in [m]$ be the row such that

$$\sum_{j<j^\flat} \widetilde{A}_{j,i} \cdot p_j(\mathsf{x}_k) < 1 + \beta \quad \text{and} \quad \sum_{j \le j^\flat} \widetilde{A}_{j,i} \cdot p_j(\mathsf{x}_k) \ge 1 + \beta \ .$$

Note that such a $j^\flat$ must exist because $\sum_{j=1}^m \widetilde{A}_{j,i} \cdot p_j > 1 + \beta$. It is clear that $j^\flat \ge j^*$, owing to the definition that $\widetilde{A}_{ji} \le A_{ji}$ for all $i \in [n], j \in [m]$. Defining $\delta^\flat = \frac{\mu\beta}{2A_{j^\flat,i}} \le \delta$, the objective decrease is lower bounded as

$$f_\mu(\mathsf{x}_k) - f_\mu(\mathsf{y}_k^{(i)}) = \int_{\tau=0}^{\delta} \langle -\nabla f_\mu(\mathsf{x}_k + \tau \mathbf{e}_i), \mathbf{e}_i \rangle d\tau = \int_{\tau=0}^{\delta} \big(\langle A_{:i}, p(\mathsf{x}_k + \tau \mathbf{e}_i)\rangle - 1\big) d\tau$$

$$
\begin{aligned}
&\geq \int_{\tau=0}^{\delta^\flat} \big(\langle A_{:i}, p(\mathsf{x}_k+\tau\mathbf{e}_i)\rangle - 1\big) d\tau \\
&= \underbrace{\int_{\tau=0}^{\delta^\flat} \Big(\sum_{j\leq j^\flat} A_{j,i}\cdot p_j(\mathsf{x}_k+\tau\mathbf{e}_i) - 1\Big) d\tau}_{I} + \underbrace{\sum_{j>j^\flat}\int_{\tau=0}^{\delta^\flat} A_{j,i}\cdot p_j(\mathsf{x}_k+\tau\mathbf{e}_i) d\tau}_{I'}
\end{aligned}
$$

where the inequality is because $\delta^\flat \leq \delta$ and $\langle A_{:i}, p(\mathsf{x}_k+\tau\mathbf{e}_i)\rangle \geq 1$ for all $\tau\leq\delta$ (see the proof of part (a)).

Part *I*. To lower bound I, we use the monotonicity of $p_j(\cdot)$ and obtain that

$$
I = \int_{\tau=0}^{\delta^\flat} \Big(\sum_{j\leq j^\flat} A_{j,i}\cdot p_j(\mathsf{x}_k+\tau\mathbf{e}_i) - 1\Big) d\tau \geq \delta^\flat \cdot \Big(\sum_{j\leq j^\flat} A_{j,i}\cdot p_j(\mathsf{x}_k+\delta^\flat\mathbf{e}_i) - 1\Big) \ .
$$

However, our choice of $\delta^\flat = \frac{\mu\beta}{2A_{j^\flat,i}} \leq \frac{\mu\beta}{2A_{j,i}}$ for all $j\leq j^\flat$ ensures that

$$
\sum_{j\leq j^\flat} A_{j,i}\cdot p_j(\mathsf{x}_k+\delta^\flat\mathbf{e}_i) \geq \sum_{j\leq j^\flat} A_{j,i}\cdot p_j(\mathsf{x}_k)\cdot \exp^{\frac{-A_{j,i}\cdot\delta^\flat}{\mu}} \geq \sum_{j\leq j^\flat} A_{j,i}\cdot p_j(\mathsf{x}_k)\cdot(1-\beta/2) \ .
$$

Therefore, we obtain that

$$
I \geq \delta^\flat\Big((1-\beta/2)\sum_{j\leq j^\flat} A_{j,i}\cdot p_j(\mathsf{x}_k) - 1\Big) \geq \frac{\delta^\flat}{3}\Big(\sum_{j\leq j^\flat} A_{j,i}\cdot p_j(\mathsf{x}_k) - 1\Big) \ ,
$$

where the inequality is because $\left(\frac{2}{3}-\frac{\beta}{2}\right)\sum_{j\leq j^\flat} A_{j,i}\cdot p_j(\mathsf{x}_k) \geq \frac{4-3\beta}{6}\cdot(1+\beta) \geq \frac{2}{3}$ whenever $\beta\leq\frac{1}{3}$ (or equivalently, whenever $\varepsilon\leq 1/9$).

Now, suppose that $\sum_{j\leq j^\flat}\widetilde{A}_{j,i}\cdot p_j(\mathsf{x}_k) - (1+\beta) = b\cdot\widetilde{A}_{j^\flat,i}\cdot p_{j^\flat}(\mathsf{x}_k)$ for some $b\in[0,1]$. Note that we can do so by the very definition of $j^\flat$. Then, we must have

$$
\begin{aligned}
\sum_{j\leq j^\flat} A_{j,i}\cdot p_j(\mathsf{x}_k) - 1 &\geq \sum_{j<j^\flat}\widetilde{A}_{j,i}\cdot p_j(\mathsf{x}_k) + A_{j^\flat,i}\cdot p_{j^\flat}(\mathsf{x}_k) - 1 \\
&= (1+\beta) - (1-b)\widetilde{A}_{j^\flat,i}\cdot p_{j^\flat}(\mathsf{x}_k) + A_{j^\flat,i}\cdot p_{j^\flat} - 1 \\
&\geq \beta + b\cdot A_{j^\flat,i}\cdot p_{j^\flat}(\mathsf{x}_k) \ .
\end{aligned}
$$

Therefore, we conclude that

$$
\begin{aligned}
I &\geq \frac{\delta^\flat}{3}\Big(\sum_{j\leq j^\flat} A_{j,i}\cdot p_j(\mathsf{x}_k) - 1\Big) > \frac{\delta^\flat}{3}\cdot b\cdot A_{j^\flat,i}\cdot p_{j^\flat}(\mathsf{x}_k) = \frac{\mu\beta}{6\widetilde{A}_{j^\flat,i}}\cdot b\cdot\widetilde{A}_{j^\flat,i}\cdot p_{j^\flat}(\mathsf{x}_k) \\
&= \frac{\mu\beta}{6\widetilde{A}_{j^\flat,i}}\cdot\Big(\sum_{j\leq j^\flat}\widetilde{A}_{j,i}\cdot p_j(\mathsf{x}_k) - (1+\beta)\Big) \geq \frac{\mu\beta}{12}\cdot u_i^*\cdot\Big(\sum_{j\leq j^\flat}\widetilde{A}_{j,i}\cdot p_j(\mathsf{x}_k) - (1+\beta)\Big) \ .
\end{aligned}
$$

Above, the last inequality is because $u_i^*\cdot\widetilde{A}_{j^\flat,i} \leq \langle\widetilde{A}_{j^\flat:}, u^*\rangle \leq 2$ by our definition of the adjusted $\widetilde{A}$.

Part *I'*. To lower bound I', consider every $j>j^\flat$ and the integral

$$
\int_{\tau=0}^{\delta^\flat} A_{j,i}\cdot p_j(\mathsf{x}_k+\tau\mathbf{e}_i) d\tau \ .
$$

Note that whenever $\tau \leq \frac{\mu\beta}{2A_{j,i}} \leq \frac{\mu\beta}{2A_{j^\flat,i}} = \delta^\flat$, we have that $p_j(\mathsf{x}_k+\tau\mathbf{e}_i) \geq p_j(\mathsf{x}_k)\cdot e^{-\beta/2} \geq \frac{1}{2}p_j(\mathsf{x}_k)$. Therefore, the above integral is at least $\frac{\mu\beta}{2A_{j,i}}\cdot A_{j,i}\cdot\frac{1}{2}p_j(\mathsf{x}_k)$. This implies a lower bound on I':

$$
I' \geq \sum_{j>j^\flat}\frac{\mu\beta}{4A_{j,i}}\cdot A_{j,i}\cdot p_j(\mathsf{x}_k) \geq \frac{\mu\beta}{8}\cdot\sum_{j>j^\flat} u_i^*\cdot\widetilde{A}_{j,i}\cdot p_j(\mathsf{x}_k) \ ,
$$

where again in the last inequality we have used $u_i^* \cdot \widetilde{A}_{j^\flat,i} \leq \langle \widetilde{A}_{j^\flat:}, u^* \rangle \leq 2$ by our definition of $\widetilde{A}$.

Together. Combining the lower bounds on I and I', we obtain

$$f_\mu(\mathsf{x}_k) - f_\mu(\mathsf{y}_k^{(i)}) \geq I + I' \geq \frac{\mu\beta}{12} \cdot u_i^* \cdot \Big(\sum_{j=1}^m \widetilde{A}_{j,i} \cdot p_j(\mathsf{x}_k) - (1+\beta) \Big) = \frac{\mu\beta}{12} \cdot \langle -\widetilde{\eta}_k^{(i)}, u^* \rangle \ . \qquad \square$$

F Efficient Implementation of `PacLPSolver`

In this section, we illustrate how to implement each iteration of `PacLPSolver` to run in an expected $O(N/n)$ time. We maintain the following quantities

$$\mathsf{z}_k \in \mathbb{R}^n_{\geq 0}, \quad \mathsf{az}_k \in \mathbb{R}^m_{\geq 0}, \quad \mathsf{y}'_k \in \mathbb{R}^n, \quad \mathsf{ay}_k \in \mathbb{R}^m, \quad B_{k,1}, B_{k,2} \in \mathbb{R}_+$$

throughout the algorithm, so as to ensure the following invariants are always satisfied

$$A\mathsf{z}_k = \mathsf{az}_k \ , \tag{F.1}$$

$$\mathsf{y}_k = B_{k,1} \cdot \mathsf{z}_k + B_{k,2} \cdot \mathsf{y}'_k \ , \qquad A\mathsf{y}_k = B_{k,1} \cdot A\mathsf{z}_k + B_{k,2} \cdot \mathsf{ay}_k \ . \tag{F.2}$$

It is clear that when $k = 0$, letting $\mathsf{az}_k = A\mathsf{z}_0$, $\mathsf{y}'_k = \mathsf{y}_0$, $\mathsf{ay}_k = A\mathsf{y}_0$, $B_{k,1} = 0$, and $B_{k,2} = 1$, we can ensure that all the invariants are satisfied initially. We denote $\|A_{:i}\|_0$ the number of nonzeros elements in vector $A_{:i}$. In each iteration $k = 1, 2, \dots, T$:

- The step $\mathsf{x}_k = \tau \mathsf{z}_{k-1} + (1-\tau)\mathsf{y}_{k-1}$ does not need to be implemented.

- The value $\nabla_i f(\mathsf{x}_k)$ requires the knowledge of $p_j(\mathsf{x}_k) = \exp^{\frac{1}{\mu}((A\mathsf{x}_k)_j - 1)}$ for each j such that $A_{ij} \neq 0$. Accordingly, we need to know the value

  $$(A\mathsf{x}_k)_j = \tau(A\mathsf{z}_{k-1})_j + (1-\tau)(A\mathsf{y}_{k-1})_j = \big(\tau + (1-\tau)B_{k-1,1}\big)(A\mathsf{z}_{k-1})_j + (1-\tau)B_{k-1,2}\mathsf{ay}_{k-1,j}$$

  for each such j. This can be computed in $O(1)$ time for each j, and $O(\|A_{:i}\|_0)$ time in total.

- Recall that the step $\mathsf{z}_k \leftarrow \arg\min_{z \in \Delta} \big\{ \frac{1}{2}\|z - \mathsf{z}_{k-1}\|_A^2 + \langle n\alpha_k \xi_k^{(i)}, z \rangle \big\}$ can be written as $\mathsf{z}_k = \mathsf{z}_{k-1} + \delta \mathbf{e}_i$ for some $\delta \in \mathbb{R}$ that can be computed in $O(1)$ time (see Proposition 3.6). Observe also that $\mathsf{z}_k = \mathsf{z}_{k-1} + \delta \mathbf{e}_i$ yields $\mathsf{y}_k = \tau \mathsf{z}_{k-1} + (1-\tau)\mathsf{y}_{k-1} + \frac{\delta \mathbf{e}_i}{n\alpha_k L}$ due to Line 6 and Line 10 of Algorithm 1. Therefore, we can perform two explicit updates on z_k and az_k as

  $$\mathsf{z}_k \leftarrow \mathsf{z}_{k-1} + \delta \mathbf{e}_i \ , \quad \mathsf{az}_k \leftarrow A\mathsf{z}_{k-1} + \delta A_{:i}$$

  and two implicit updates on y_k as

  $$\begin{aligned} &B_{k,1} = \tau + (1-\tau)B_{k-1,1} &&, \quad B_{k,2} = (1-\tau)B_{k-1,2} \ , \\ &\mathsf{y}'_k \leftarrow \mathsf{y}'_{k-1} + \delta \mathbf{e}_i \cdot \Big(-\tfrac{B_{k,1}}{B_{k,2}} + \tfrac{1}{n\alpha_k L}\tfrac{1}{B_{k,2}} \Big) &&, \quad \mathsf{ay}_k \leftarrow \mathsf{ay}_{k-1} + \delta A_{:i} \cdot \Big(-\tfrac{B_{k,1}}{B_{k,2}} + \tfrac{1}{n\alpha_k L}\tfrac{1}{B_{k,2}} \Big) \end{aligned}$$

  It is not hard to verify that after these updates, we have

  $$\begin{aligned} \mathsf{y}_k &= B_{k,1} \cdot \mathsf{z}_k + B_{k,2} \cdot \mathsf{y}'_k = B_{k,1} \cdot (\mathsf{z}_{k-1} + \delta \mathbf{e}_i) + B_{k,2} \cdot \Big(\mathsf{y}'_{k-1} + \delta \mathbf{e}_i \cdot \Big(-\frac{B_{k,1}}{B_{k,2}} + \frac{1}{n\alpha_k L}\frac{1}{B_{k,2}} \Big) \Big) \\ &= B_{k,1} \cdot \mathsf{z}_{k-1} + B_{k,2} \cdot \Big(\mathsf{y}'_{k-1} + \delta \mathbf{e}_i \cdot \Big(\frac{1}{n\alpha_k L}\frac{1}{B_{k,2}} \Big) \Big) \\ &= B_{k,1} \cdot \mathsf{z}_{k-1} + B_{k,2} \cdot \mathsf{y}'_{k-1} + \frac{\delta \mathbf{e}_i}{n\alpha_k L} \\ &= \big(\tau + (1-\tau)B_{k-1,1}\big) \cdot \mathsf{z}_{k-1} + \big((1-\tau)B_{k-1,2}\big) \cdot \mathsf{y}'_{k-1} + \frac{\delta \mathbf{e}_i}{n\alpha_k L} \\ &= \tau \mathsf{z}_{k-1} + (1-\tau)\mathsf{y}_{k-1} + \frac{\delta \mathbf{e}_i}{n\alpha_k L} \ . \end{aligned}$$

One can similarly verify that $A\mathsf{y}_k = B_{k,1} \cdot A\mathsf{z}_k + B_{k,2} \cdot \mathsf{ay}_k$ equals $A\mathsf{y}_k = \tau A\mathsf{z}_{k-1} + (1-\tau)A\mathsf{y}_{k-1} + \frac{\delta A e_i}{n\alpha_k L}$. In sum, these updates are dominated by the updates on $A\mathsf{z}_k$ and ay_k, each costing an $O(\|A_{:i}\|_0)$ running time, and ensure that the invariants in (F.1) and (F.2) are satisfied at iteration k.

In sum, we only need $O(\|A_{:i}\|_0)$ time to perform the updates in `PacLPSolver` for an iteration k if the coordinate i is selected. Therefore, each iteration of `PacLPSolver` can be implemented to run in an expected $O(\mathbf{E}_i[\|A_{:i}\|_0]) = O(N/n)$ time.

G Efficient Implementation of `CovLPSolver`

In this section we illustrate how to implement each iteration of `CovLPSolver` to run in an expected $O(N/n)$ time. We maintain the following quantities

$$\mathsf{z}'_k \in \mathbb{R}^n_+, \quad \mathsf{sz}_k \in \mathbb{R}_+, \quad \mathsf{sumz}_k \in \mathbb{R}_+, \quad \mathsf{az}_k \in \mathbb{R}^m_{\geq 0}, \quad \mathsf{y}'_k \in \mathbb{R}^n, \quad \mathsf{ay}_k \in \mathbb{R}^m, \quad B_{k,1}, B_{k,2} \in \mathbb{R}_+$$

throughout the algorithm, so as to maintain the following invariants are always satisfies

$$\mathsf{z}_k = \mathsf{z}'_k / \mathsf{sz}_k, \qquad \mathsf{sumz}_k = \mathbb{1}^T \mathsf{z}'_k, \qquad A\mathsf{z}_k = \mathsf{az}_k / \mathsf{sz}_k, \tag{G.1}$$

$$\mathsf{y}_k = B_{k,1} \cdot \mathsf{z}'_k + B_{k,2} \cdot \mathsf{y}'_k, \qquad A\mathsf{y}_k = B_{k,1} \cdot \mathsf{az}_k + B_{k,2} \cdot \mathsf{ay}_k \ . \tag{G.2}$$

It is clear that when $k = 0$, letting $\mathsf{z}'_k = \mathsf{z}_0$, $\mathsf{sz}_k = 1$, $\mathsf{sumz}_k = \mathbb{1}^T \mathsf{z}_0$, $\mathsf{az}_k = A\mathsf{z}_0$, $\mathsf{y}'_k = \mathsf{y}_0$, $\mathsf{ay}_k = A\mathsf{y}_0$, $B_{k,1} = 0$, and $B_{k,2} = 1$, we can ensure that all the invariants are satisfied initially.

We denote by $\|A_{:i}\|_0$ the number of nonzero elements in vector $A_{:i}$. In each iteration $k = 1, 2, \dots, T$:

- The step $\mathsf{x}_k = \tau \mathsf{z}_{k-1} + (1-\tau)\mathsf{y}_{k-1}$ does not need to be implemented.
- The value $p_j(\mathsf{x}_k) = \exp^{\frac{1}{\mu}(1-(A\mathsf{x}_k)_j)}$ for each j only requires the knowledge of

  $$(A\mathsf{x}_k)_j = \tau(A\mathsf{z}_{k-1})_j + (1-\tau)(A\mathsf{y}_{k-1})_j = \big(\tau + (1-\tau)B_{k-1,1}\big)\frac{\mathsf{az}_{k-1,j}}{\mathsf{sz}_{k-1}} + (1-\tau)B_{k-1,2}\mathsf{ay}_{k-1,j} \ .$$

  This can be computed in $O(1)$ time.
- The value $\nabla_i f(\mathsf{x}_k)$ requires the knowledge of $p_j(\mathsf{x}_k)$ for each $j \in [m]$ such that $A_{ij} \neq 0$. Since we have $\|A_{:i}\|_0$ such j's, we can compute $\nabla_i f(\mathsf{x}_k)$ in $O(\|A_{:i}\|_0)$ time.
- Letting $\delta = (1+\gamma)n\alpha_k \xi^{(i)}_{k,i}$, recall that the mirror step $\mathsf{z}_k \leftarrow \arg\min_{z\in\Delta}\big\{V_{\mathsf{z}_{k-1}}(z) + \langle \delta \mathbf{e}_i, z\rangle\big\}$ has a very simple form (see Proposition D.9): first multiply the i-th coordinate of z_{k-1} by $e^{-\delta}$ and then, if the sum of all coordinates have exceeded $2\mathsf{OPT}'$, scale everything down so as to sum up to $2\mathsf{OPT}'$. This can be implemented as follows: setting $\delta_1 = \mathsf{z}'_{k-1,i}(e^{-\delta} - 1)$,

  $$\begin{aligned} &\mathsf{z}'_k \leftarrow \mathsf{z}'_{k-1} + \delta_1 \mathbf{e}_i \quad , \quad \mathsf{az}_k \leftarrow \mathsf{az}_{k-1} + \delta_1 A_{:i} \ , \\ &\mathsf{sumz}_k \leftarrow \mathsf{sumz}_{k-1} + \delta_1 \quad , \quad \mathsf{sz}_k \leftarrow \mathsf{sz}_k \cdot \max\Big\{1, \tfrac{\mathsf{sumz}_k}{\mathsf{sz}_{k-1}\cdot 2\mathsf{OPT}'}\Big\} \ . \end{aligned}$$

  These updates can be implemented to run in $O(\|A_{:i}\|_0)$ time, and they together ensure that the invariants in (G.1) are satisfied at iteration k.
- Recall that the gradient step is of the form $\mathsf{y}_k \leftarrow \mathsf{x}_k + \delta_2 \cdot \mathbf{e}_i$ for some value $\delta_2 \geq 0$. This value δ_2 can be computed in $O(\|A_{:i}\|_0)$ time, since each $p_j(\mathsf{x}_k)$ can be computed in $O(1)$ time, and we can sort the rows of each column of A by preprocessing.

  Since $\mathsf{y}_k = \mathsf{x}_k + \delta_2 \cdot \mathbf{e}_i = \tau \mathsf{z}_{k-1} + (1-\tau)\mathsf{y}_{k-1} + \delta_2 \mathbf{e}_i$, we can implement this update by letting

  $$\begin{aligned} &B_{k,1} = \tfrac{\tau}{\mathsf{sz}_{k-1}} + (1-\tau)B_{k-1,1} \quad , \quad B_{k,2} = (1-\tau)B_{k-1,2} \\ &\mathsf{y}'_k \leftarrow \mathsf{y}'_{k-1} + \mathbf{e}_i \cdot \Big(-\tfrac{B_{k,1}\delta_1}{B_{k,2}} + \tfrac{\delta_2}{B_{k,2}}\Big) \quad , \quad \mathsf{ay}_k \leftarrow \mathsf{ay}_{k-1} + A_{:i} \cdot \Big(-\tfrac{B_{k,1}\delta_1}{B_{k,2}} + \tfrac{\delta_2}{B_{k,2}}\Big) \end{aligned}$$

It is not hard to verify that after these updates, we have

$$\begin{aligned}\mathsf{y}_k &= B_{k,1}\cdot \mathsf{z}'_k + B_{k,2}\cdot \mathsf{y}'_k = B_{k,1}\cdot \big(\mathsf{z}'_{k-1}+\delta_1\mathbf{e}_i\big) + B_{k,2}\cdot\Big(\mathsf{y}'_{k-1}+\mathbf{e}_i\cdot\Big(-\frac{B_{k,1}\delta_1}{B_{k,2}}+\frac{\delta_2}{B_{k,2}}\Big)\Big)\\ &= B_{k,1}\cdot \mathsf{z}'_{k-1} + B_{k,2}\cdot\big(\mathsf{y}'_{k-1}+\delta_2\mathbf{e}_i/B_{k,2}\big)\\ &= B_{k,1}\cdot \mathsf{z}'_{k-1} + B_{k,2}\cdot \mathsf{y}'_{k-1}+\delta_2\mathbf{e}_i\\ &= \big(\frac{\tau}{\mathsf{sz}_{k-1}}+(1-\tau)B_{k-1,1}\big)\cdot \mathsf{z}'_{k-1}+\big((1-\tau)B_{k-1,2}\big)\cdot\mathsf{y}'_{k-1}+\delta_2\mathbf{e}_i\\ &= \tau\mathsf{z}_{k-1}+(1-\tau)\mathsf{y}_{k-1}+\delta_2\mathbf{e}_i\ \ .\end{aligned}$$

One can similarly verify that $A\mathsf{y}_k = B_{k,1}\cdot\mathsf{az}_k + B_{k,2}\cdot\mathsf{ay}_k$ equals $A\mathsf{y}_k = \tau A\mathsf{z}_{k-1}+(1-\tau)A\mathsf{y}_{k-1}+\delta_2 A_{:i}$. These updates can be implemented to run in $O(\|A_{:i}\|_0)$ time, and they together ensure that the invariants in (G.2) are satisfied at iteration k.

In sum, we only need $O(\|A_{:i}\|_0)$ time to perform the updates in `CovLPSolver` for an iteration k if the coordinate i is selected. Therefore, each iteration of `CovLPSolver` can be implemented to run in an expected $O(\mathbf{E}_i[\|A_{:i}\|_0]) = O(N/n)$ time.

References

[AHK12] Sanjeev Arora, Elad Hazan, and Satyen Kale. The Multiplicative Weights Update Method: a Meta-Algorithm and Applications. *Theory of Computing*, 8:121–164, 2012.

[AK08] Baruch Awerbuch and Rohit Khandekar. Stateless distributed gradient descent for positive linear programs. *Proceedings of the fourtieth annual ACM symposium on Theory of computing - STOC 08*, page 691, 2008.

[AKR12] Baruch Awerbuch, Rohit Khandekar, and Satish Rao. Distributed algorithms for multicommodity flow problems via approximate steepest descent framework. *ACM Transactions on Algorithms*, 9(1):1–14, December 2012.

[ALO14] Zeyuan Allen-Zhu, Zhenyu Liao, and Lorenzo Orecchia. Linear-sized spectral sparsification in almost quadratic time and regret minimization beyond matrix multiplicative weight updates. Technical report, November 2014. Manuscript.

[AO14] Zeyuan Allen-Zhu and Lorenzo Orecchia. Linear coupling of gradient and mirror descent: A novel simple interpretation of Nesterov's accelerated method. *ArXiv e-prints*, abs/1407.1537, July 2014.

[AO15] Zeyuan Allen-Zhu and Lorenzo Orecchia. Using optimization to break the epsilon barrier: A faster and simpler width-independent algorithm for solving positive linear programs in parallel. In *Proceedings of the 26th ACM-SIAM Symposium on Discrete Algorithms*, SODA '15, 2015.

[BBR97] Yair Bartal, John W. Byers, and Danny Raz. Global optimization using local information with applications to flow control. In *Proceedings 38th Annual Symposium on Foundations of Computer Science*, pages 303–312. IEEE Comput. Soc, 1997.

[BBR04] Yair Bartal, John W. Byers, and Danny Raz. Fast, Distributed Approximation Algorithms for Positive Linear Programming with Applications to Flow Control. *SIAM Journal on Computing*, 33(6):1261–1279, January 2004.

[BI04] D. Bienstock and G. Iyengar. Faster approximation algorithms for packing and covering problems. Technical report, 2004. Preliminary version published in STOC '04.

[BN00] John Byers and Gabriel Nasser. Utility-based decision-making in wireless sensor networks. In *Mobile and Ad Hoc Networking and Computing, 2000. MobiHOC. 2000 First Annual Workshop on*, pages 143–144. IEEE, 2000.

[BN13] Aharon Ben-Tal and Arkadi Nemirovski. *Lectures on Modern Convex Optimization*. Society for Industrial and Applied Mathematics, January 2013.

[CE05] Fabián A. Chudak and Vânia Eleutério. Improved Approximation Schemes for Linear Programming Relaxations of Combinatorial Optimization Problems. In *Proceedings of the 11th International IPCO Conference on Integer Programming and Combinatorial Optimization*, pages 81–96, 2005.

[CL06] Nicolo Cesa-Bianchi and Gabor Lugosi. *Prediction, Learning, and Games*. Cambridge University Press, Cambridge, 2006.

[DP14] Ran Duan and Seth Pettie. Linear-Time Approximation for Maximum Weight Matching. *Journal of the ACM*, 61(1):1–23, January 2014.

[Fle00] Lisa K. Fleischer. Approximating Fractional Multicommodity Flow Independent of the Number of Commodities. *SIAM Journal on Discrete Mathematics*, 13(4):505–520, January 2000.

[FR13] Olivier Fercoq and Peter Richtárik. Accelerated, Parallel and Proximal Coordinate Descent. *ArXiv e-prints*, abs/1312.5799:25, December 2013.

[GK07] Naveen Garg and Jochen Könemann. Faster and Simpler Algorithms for Multicommodity Flow and Other Fractional Packing Problems. *SIAM Journal on Computing*, 37(2):630–652, January 2007.

[KY99] Philip Klein and Neal Young. On the number of iterations for dantzig-wolfe optimization and packing-covering approximation algorithms. In Gérard Cornuéjols, Rainer E. Burkard, and Gerhard J. Woeginger, editors, *Integer Programming and Combinatorial Optimization*, volume 1610 of *Lecture Notes in Computer Science*, pages 320–327. Springer Berlin Heidelberg, 1999.

[KY13] Christos Koufogiannakis and Neal E. Young. A Nearly Linear-Time PTAS for Explicit Fractional Packing and Covering Linear Programs. *Algorithmica*, pages 494–506, March 2013. Previously appeared in FOCS '07.

[LN93] Michael Luby and Noam Nisan. A parallel approximation algorithm for positive linear programming. In *Proceedings of the twenty-fifth annual ACM symposium on Theory of computing - STOC '93*, pages 448–457, New York, New York, USA, 1993. ACM Press.

[Mad10] Aleksander Madry. Faster approximation schemes for fractional multicommodity flow problems via dynamic graph algorithms. In *Proceedings of the 42nd ACM symposium on Theory of computing - STOC '10*, page 121, New York, New York, USA, 2010. ACM Press.

[Nem04] Arkadi Nemirovski. Prox-Method with Rate of Convergence $O(1/t)$ for Variational Inequalities with Lipschitz Continuous Monotone Operators and Smooth Convex-Concave Saddle Point Problems. *SIAM Journal on Optimization*, 15(1):229–251, January 2004.

[Nes83] Yurii Nesterov. A method of solving a convex programming problem with convergence rate $O(1/k^2)$. In *Doklady AN SSSR (translated as Soviet Mathematics Doklady)*, volume 269, pages 543–547, 1983.

[Nes04] Yurii Nesterov. *Introductory Lectures on Convex Programming Volume: A Basic course*, volume I. Kluwer Academic Publishers, 2004.

[Nes05] Yurii Nesterov. Smooth minimization of non-smooth functions. *Mathematical Programming*, 103(1):127–152, December 2005.

[Nes08] Yu Nesterov. Rounding of convex sets and efficient gradient methods for linear programming problems. *Optimisation Methods and Software*, 23(1):109–128, 2008.

[PST95] Serge A. Plotkin, David B. Shmoys, and Éva Tardos. Fast Approximation Algorithms for Fractional Packing and Covering Problems. *Mathematics of Operations Research*, 20(2):257–301, May 1995.

[ST11] Shai Shalev-Shwartz and Ambuj Tewari. Stochastic methods for l1-regularized loss minimization. *Journal of Machine Learning Research*, 12:1865–1892, 2011.

[Tre98] Luca Trevisan. Parallel Approximation Algorithms by Positive Linear Programming. *Algorithmica*, 21(1):72–88, May 1998.

[You01] Neal E. Young. Sequential and parallel algorithms for mixed packing and covering. In *42nd Annual IEEE Symposium on Foundations of Computer Science (FOCS'01)*, pages 538–546. IEEE Comput. Soc, 2001.

[You14] Neal E. Young. Nearly linear-time approximation schemes for mixed packing/covering and facility-location linear programs. *ArXiv e-prints*, abs/1407.3015, July 2014.

[ZN01] Edo Zurel and Noam Nisan. An efficient approximate allocation algorithm for combinatorial auctions. In *Proceedings of the 3rd ACM conference on Electronic Commerce*, pages 125–136. ACM, 2001.