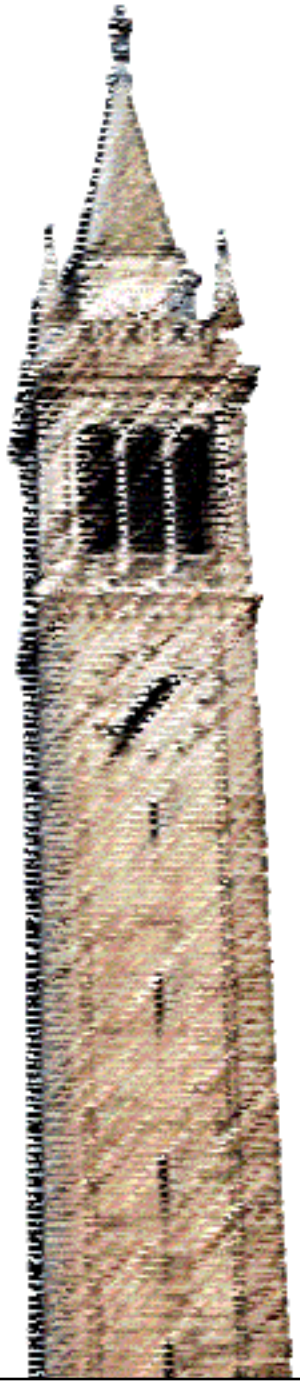
Fast Approximation Algorithms for Positive Linear Programs



Di Wang

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2017-126
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2017/EECS-2017-126.html

July 13, 2017

Acknowledgement

The results in this dissertation were based on collaboration with Satish Rao, Michael Mahoney, Peng Zhang and Nishanth Mohan. The results include previous publications~\cite{WRM16,WRMZ16,WMMR15}. I thank my collaborators for allowing the inclusion of coauthored work in this dissertation.

Fast Approximation Algorithms for Positive Linear Programs

by

Di Wang

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Satish Rao, Chair
Professor Dorit Hochbaum
Professor Prasad Raghavendra
Professor Michael Mahoney

Summer 2017

Fast Approximation Algorithms for Positive Linear Programs

Abstract

Fast Approximation Algorithms for Positive Linear Programs

by

Di Wang

Doctor of Philosophy in Computer Science

University of California, Berkeley

Professor Satish Rao, Chair

Positive linear programs (LPs), or equivalently, mixed packing and covering LPs, are LPs formulated with non-negative coefficients, constants, and variables. Notable special cases of positive LPs include packing LPs and covering LPs. Positive LPs model a wide range of fundamental problems both in theory of computation as well as in practice, thus have long drawn interest in theoretical computer science, operations research, and optimization communities.

In this work, we study iterative methods for approximately solving positive LPs efficiently. Given a positive LP of size N, we are interested in iterative methods that can converge to a $(1 \pm \epsilon)$-approximate optimal solution with complexity that is nearly linear in N, and polynomial in $\frac{1}{\epsilon}$.

For the special cases of packing LPs and covering LPs, we design improved sequential and parallel approximate solvers, both using first-order (i.e. gradient based) descent methods from (continuous) convex optimization. In particular, we build upon the linear coupling framework introduced by Allen-Zhu and Orecchia. The performance of first-order methods are sensitive to the geometry of the problem space, as well as properties of the objective function. At a high level, we design (discrete) techniques that exploit the combinatorial structures of the LPs, which lead to signicantly improved theoretical behavior of the optimization method. More specifically, our sequential (i.e., non-parallelizable) solver is based on a $\tilde{O}(N/\epsilon)$ algorithm for packing LPs in a previous breakthrough by Allen-Zhu and Orecchia, and we provide a unified method with running time $\tilde{O}(N/\epsilon)$ for both packing LPs and covering LPs. Our parallel algorithm has depth (i.e., parallel running time) $\tilde{O}(1/\epsilon^2)$, which is $\Omega(1/\epsilon)$ faster than the previous best algorithm.

For general positive LPs, we take a more combinatorial framework, called Lagrangian relaxation, but we crucially adapt the framework to leverage insights from continuous convex optimization. The incorporation of continuous optimization techniques provides a primal-dual perspective of the method, and leads to faster convergence. More specifically, we develop a $\tilde{O}(1/\epsilon^3)$ depth parallel algorithm, improving a long-standing bound of $\tilde{O}(1/\epsilon^4)$ for positive LPs.

At a high level, we benefit from a integrated view of continuous and combinatorial techniques to obtain the improved results in this work. This combination brings intriguing new perspectives to algorithm design, and is promising for further exciting progress.

To my parents,

who offered unconditional love and support,

and to Tiffany,

who has always been there for me.

Contents

Contents i

List of Tables ii

List of Algorithms iii

1 Introduction 1
- 1.1 Covering LPs and Packing LPs . . . 3
- 1.2 Positive LPs . . . 8
- 1.3 Organization . . . 9
- 1.4 Bibliographic Notes . . . 9

2 Sequential Algorithm for Packing LPs and Covering LPs 10
- 2.1 Preliminaries and Smoothing the Objective . . . 10
- 2.2 Technical Overview . . . 13
- 2.3 Diameter Reduction Method for Covering Problems . . . 15
- 2.4 Accelerated Solver for (Packing and) Covering LPs . . . 17
- 2.5 Missing Proofs . . . 26

3 Parallel Algorithms for Packing LPs and Covering LPs 31
- 3.1 Technical Overview . . . 31
- 3.2 $\tilde{O}(\frac{1}{\epsilon^2})$ Parallel Solver for Packing LPs . . . 32
- 3.3 $\tilde{O}(\frac{1}{\epsilon^2})$ Parallel Solver for Covering LPs . . . 40
- 3.4 $\tilde{O}(\frac{d}{\epsilon})$ Parallel Accelerated Solver for (Packing and) Covering LPs . . . 43

4 Parallel Algorithm for Positive LPs 47
- 4.1 Technical overview . . . 47
- 4.2 Parallel Algorithm for Mixed Packing and Covering LPs . . . 49
- 4.3 Missing Proof . . . 59

Bibliography 62

List of Tables

1.1 Selected work of sequential algorithms for covering LP and packing LP. 4
1.2 Selected work of parallel algorithms for covering LP and packing LP. 6

List of Algorithms

1 Accelerated sequential solver for packing LPs and covering LPs 18
2 $\tilde{O}(\frac{1}{\epsilon^2})$ time parallel packing LP solver 33
3 Fixing procedure for covering LPs 41
4 Accelerated $\tilde{O}(\frac{d}{\epsilon})$ time parallel solver for both packing and covering 45
5 Parallel algorithm for mixed packing and covering LPs 52

Acknowledgments

During my graduate studies at UC Berkeley, I have been privileged to have Satish Rao as my advisor. Satish has been a constant source of ideas and inspirations, and has greatly impacted my research. Beyond the stimulating discussions on research problems, Satish has provided support on many other aspects of my academic life. Over the years, Satish's patience and encouragement have instilled confidence and hope for me to get through the ups and (especially) downs, and I couldn't have made the journey without his guidance. I also feel fortunate to have the opportunity to work extensively with Michael Mahoney. Michael and Satish introduced me to the problems studied in this dissertation. Since then, Michael has been a great mentor, and I've greatly benefited from his advise and support. Most of this dissertation is based on joint work with Satish, Michael, Peng Zhang, and Richard Peng, and I thank my collaborators for all the effort and help.

I have also worked with some fabulous researchers not mentioned already, from whom I've learned a lot and greatly benefited, and I feel grateful: Monika Henzinger, Yu Cheng, Ho Yee Cheung, Shanghua Teng, Kimon Fountoulakis, Richard Zhang, Kamalika Chaudhuri, and Kevin Chen. I thank Kamal Jain, who have mentored me during my internship. I also thank the undergraduate students at Berkeley with whom I have worked with on research problems: Darren Kuo, Andrew Gambardella, Victor Huang, Chenyu Zhao and Nishanth Mohan.

I thank my fellow graduate students in the theory group for their friendship and the creation of an encouraging atmosphere: Nima Anari, Urmila Mahadev, Anand Bhaskar, James Cook, Anindya De, Piyush Srivastava, Chris Wilkens, Lorenzo Orecchia, Gregory Valiant, Isabelle Stanton, Siu Man Chan, Siu On Chan, Tom Watson, Thomas Vidick, Ilias Diakonikolas, Rafael Frongillo, Jonah Sherman, George Pierrakos, Seung Woo Shin, Sam Wong, Aviad Rubinstein, Antonio Blanca, Tselil Schramm, Ben Weitz, Jarett Schwartz, Alex Psomas, Jonah Brown-Cohen, Rishi Gupta, Paul Christiano, Fotis Iliopoulos, Manuel Sabin, Jingcheng Liu, Peihan Miao, Aaron Schild, Pasin Manurangsi, Marc Khoury, and Akshayaram Srinivasan. I also thank all the faculty members for making the theory group such a wonderful place.

My time at Berkeley was made pleasant also because of my friends, and I thank all of them for the adventures, the fun and the support along the journey. Finally I want to thank my family for the unconditional love and support I received in my life.

Chapter 1

Introduction

Positive linear programs (LPs) are LPs formulated with non-negative coefficients, constants, and variables (Example 1.0.1). Traditionally, the $\leq$ constraints are called *packing* constraints, while the $\geq$ constraints (apart from the non-negativity of variables) are called *covering* constraints, and positive LPs are equivalently called mixed packing and covering LPs. Notable special cases of positive LPs are (pure) packing LPs, which are positive LPs without any covering constraint, and (pure) covering LPs, which are positive LPs without any packing constraint. Positive LPs model a wide range of fundamental problems both in theory of computation as well as in practice, and thus have long drawn interest in theoretical computer science, operations research, and optimization communities.

Example 1.0.1. *(Two simple positive LPs) Non-negative coefficients, constants and variables.*

$$\begin{array}{lrl} \min & x_1 + x_2 + x_3 & \\ & x_1 + 3x_2 + x_3 & \geq \ 5 \\ & 2x_1 + x_2 + 2x_3 & \geq \ 7 \\ & x_1 + x_2 + 2x_3 & \leq \ 6 \\ & x_1, x_2, x_3 & \geq \ 0 \end{array} \qquad \begin{array}{lrl} \max & x_1 + 2x_2 + x_3 & \\ & 2x_1 + 3x_2 + x_3 & \geq \ 7 \\ & x_1 + 3x_2 + 2x_3 & \leq \ 6 \\ & x_1 + 3x_3 & \leq \ 5 \\ & x_1, x_2, x_3 & \geq \ 0 \end{array}$$

Given a positive LP of size N (i.e., the total number of non-zeros in the formulation), we want to find a $(1 \pm \epsilon)$-approximate optimal solution efficiently. Although one can use the interior point method or ellipsoid method, with computational complexity that has $\mathrm{poly}\log(\frac{1}{\epsilon})$ dependence on the approximation error ϵ, these methods typically have very high complexity dependence on N, as computations such as the Hessian matrix or linear system solving are involved in every iteration, and consequently are problematic for large-scale applications. In this work, we study fast low-precision iterative solvers for positive LPs. In particular, we are interested in methods of complexity nearly linear in N, although with of polynomial dependence on $\frac{1}{\epsilon}$.

This line of research was initiated by the seminal work of Luby and Nisan [LN93], which gave a iterative solver with $\tilde{O}(N/\epsilon^4)$[1] convergence rate. The focus of this line of work is mostly on

[1] We follow the standard practice of using $\tilde{O}$ to hide $\mathrm{poly}\log$ factors. We will spell out the exact bounds when we discuss the results in detail.

achieving better polynomial dependence on $\frac{1}{\epsilon}$, which is meaningful in two aspects. On one hand, it is a direct measure of the complexity of an iterative method, where a $\frac{1}{\epsilon^c}$ convergence rate means that to achieve one more (binary) bit of accuracy, the method takes 2^c times more work. On the other hand, as positive LP solvers are often used as subroutines in algorithm design, where ϵ is chosen to get the best trade-off among various subroutines of the algorithm, the dependence on $\frac{1}{\epsilon}$ determines the complexity of the entire algorithm (see Example 1.0.2 for a simple illustration).

Example 1.0.2. *(Bipartite Matching) One can first formulate the problem as a fractional packing LP, get an $(1-\epsilon)$-approximate optimal (fractional) solution, round it to an integral solution, and keep adding augmenting paths to improve the $(1-\epsilon)$-optimal solution to an optimal solution. Suppose the graph has m edges and n vertices, the size of the packing LP will be $O(m)$, and the optimal is $O(n)$. To improve the $(1-\epsilon)$-optimal solution to an optimal solution, one need to compute $O(\epsilon n)$ augmenting paths, which takes $O(m \cdot \epsilon n)$. Suppose the positive LP solver has convergence rate $\tilde{O}(m/\epsilon^c)$ for some constant c, then the optimal trade-off is to set $\epsilon = n^{1/(c+1)}$, which gives a $\tilde{O}(mn^{1-1/(c+1)})$ algorithm for bipartite matching. Clearly, smaller c is better.*

Moreover, based on whether the complexity depends on the *width* ρ, a parameter which typically amounts to the largest entry of A, positive LP solvers can be divided into width-dependent solvers and width-independent solvers. Width-dependent solvers are usually pseudo-polynomial, as the running time depends at least linearly on $\rho\,\mathrm{OPT}$, which itself can be large, while width-independent solvers are more efficient in the sense that they provide truly polynomial-time approximation solvers. The new results presented in this work are all width-independent methods.

Most previous work of iterative positive LP solvers follow one of two high-level approaches. The first approach is based on turning the original positive LP (with combinatorial packing and covering constraints) into continuous and convex objective functions with trivial or no constraints (See [Nes05]). The original positive LP is reduced to a (continuous) convex optimization problem, and general-purpose first-order iterative methods from the theory of optimization are usually applied directly. The other approach is based on the Lagrangian-relaxation framework, which is more of a combinatorial flavor. We note that the characterization of continuous versus combinatorial is only at a high level. Methods following the continuous optimization approach also involve components and techniques of combinatorial nature, and vice versa. Indeed, recent progress along this line of research, including the results in this work, can be attributed to the organic integration of techniques and insights from both continuous optimization and combinatorial methods.

In this work, we present algorithms for the special cases of packing LPs and covering LPs, as well as an algorithm for general positive LPs. The algorithms follow different approaches, where the method for packing LPs and covering LPs follows the convex optimization framework, while the method for general positive LPs uses the Lagrangian-relaxation framework, although in both cases, the improvement crucially relies on the integration of continuous and combinatorial perspectives of algorithm design. In the following sections, we discuss the two cases respectively.

1.1 Covering LPs and Packing LPs

Covering LPs and Packing LPs are important classes of positive LPs, and they have wide applications in theory, including the design of approximation algorithms, scheduling, and auction design, as well as practical applications in operation research. A covering LP, in its generic form, can be written as:

$$\min_{x \geq 0} \{c^T x : Ax \geq b\},$$

where $c \in \mathbb{R}^n_{\geq 0}, b \in \mathbb{R}^m_{\geq 0}$, and $A \in \mathbb{R}^{m \times n}_{\geq 0}$. Similarly, again with c, b, A non-negative, a packing LP can be written in its generic form as:

$$\max_{x \geq 0} \{c^T x : Ax \leq b\}.$$

This two classes of LP form a primal-dual pair, i.e., the dual LP of a covering LP is a packing LP. We denote by OPT the optimal value of a LP. In this case, we say that x is a $(1+\epsilon)$-*approximation* for the covering LP if $Ax \geq b$ and $c^T x \leq (1+\epsilon)\,\mathrm{OPT}$, and we say that x is a $(1-\epsilon)$-*approximation* for the packing LP if $Ax \leq b$ and $c^T x \geq (1-\epsilon)\,\mathrm{OPT}$.

The general approach of our methods for packing LPs and covering LPs is to pose the original problem as an optimization problem, where a smooth convex function is constructed to capture the original objective as well as the combinatorial packing or covering constraints. From here, we have a convex function, and we can apply efficient first-order methods from theory of optimization to minimize it. To do so, recall that first-order methods in optimization exploit the idea of using the negative gradient as the descent direction for each step of an iterative algorithm. In order to lower bound the improvement of successive steps of the algorithm, smoothness is at the core of the analysis, as it measures how much the gradient can change as we move in the problem space. Basically we want to move in the negative gradient as far as possible, without changing the gradient by too much; if the gradient changes too much, we no longer move in a descent direction. Consequently, the performance of an optimization schema, via smoothness, crucially depends on the geometry of the problem space and the properties of the function being optimized. Since our optimization problem is constructed in order to solve the original combinatorial problem (i.e., packing LPs and covering LPs), the problem space and objective function need to capture the structure of the LPs, which may not align well with the optimization tools we have. As a result, we need to design techniques to bridge the gap between our optimization problem and the optimization tools.

We study both sequential (i.e., non-parallelizable) algorithms and parallel algorithms for packing LP and covering LP. For sequential methods, the complexity is measured by the running time of the algorithm. For parallel algorithms, we measure complexity using (parallel) running time and total work. Time and work are standard notions from parallel algorithms that correspond to the longest chain of dependent operations and the total operations performed. In particular, time has a natural correspondence with iteration count, and these two measures have been used to measure the performance of previous work on positive LPs. The algorithms are stochastic, in the sense that the approximation bound of the computed solution hold with high probability, and the running time bounds are expected values.

Paper	Running Time
Width-dependent Solvers	
[PST91]	$\tilde{O}(\rho^2 \operatorname{OPT}^2 \frac{N}{\epsilon^2})$
[AHK12]	$\tilde{O}(\rho \operatorname{OPT} \frac{N}{\epsilon^2})$
[Nes05; Nem04]	$\tilde{O}(\rho \operatorname{OPT} \frac{N}{\epsilon})$
[BI04]	$\tilde{O}(\sqrt{dn}\frac{N}{\epsilon})$
Width-independent Solvers	
[LN93]	$\tilde{O}(\frac{N}{\epsilon^4})$
[You01; You14]	$\tilde{O}(\frac{N}{\epsilon^2})$
[KY14]	$\tilde{O}(N + \frac{n+m}{\epsilon^2})$
[AZO15a]: Packing LP	$\tilde{O}(\frac{N}{\epsilon})$
This work: Covering LP	$\tilde{O}(\frac{N}{\epsilon})$

Table 1.1: Selected work of sequential algorithms for covering LP and packing LP.

Accelerated Sequential Algorithm

For covering LPs, we present an iterative method that is width-independent and has a linear rate of convergence, that is, $O\left(N\frac{\log^2(N/\epsilon)\log(1/\epsilon)}{\epsilon}\right)$ running time. To simplify the following discussion, we will follow the standard practice of using $\tilde{O}$ to hide poly-log factors, in which case the running time of our algorithm for covering LP is $\tilde{O}\left(N/\epsilon\right)$.

In Table 1.1, we show the results of selected work on packing LP and covering LP. We group the results into width-dependent solvers and width-independent solvers. The result in [BI04] turns the dependence of the width $\rho \operatorname{OPT}$ to a more lenient term $\sqrt{dn}$, where d is the largest number of nonzeros in any row of A, which can be n in the worst case, so the running time is not nearly linear in the size of the LP.

As we discussed earlier, previous work at a high level mostly follows either the continuous optimization approach or the Lagrangian-relaxation approach. The continuous optimization approach gives algorithms that have better polynomial dependence on $\frac{1}{\epsilon}$. In particular, the best dependence on $\frac{1}{\epsilon}$ is linear, which is achieved with methods based on Nesterov's accelerated gradient descent [Nes05], and we refer to the linear dependence on $\frac{1}{\epsilon}$ as *Nesterov-like acceleration*. However, methods using the optimization approach are traditionally width-dependent, and it is widely believed to be very difficult to combine Nesterov-like acceleration and width-independence.

In a recent breakthrough [AZO15a], Allen-Zhu and Orecchia achieve the first such success to give a packing LP solver that is both width-independent and accelerated. They achieve the remarkable result by leveraging the linear coupling framework introduced recently by the same

authors in [ZO14]. This is a first-order method for solving convex optimization problems, and it provides a conceptually simple way to integrate gradient descent and mirror descent, which are two classical first-order methods from the theory of optimization. In the setting of standard smooth convex optimization, the method achieves the same convergence rate as that of the accelerated gradient descent method of Nesterov [Nes05], and indeed the former can be viewed as an insightful reinterpretation of the latter. The high-level view of the method as a coupling of gradient descent steps and mirror descent steps offers more flexibility to the framework, as the combination allows the two steps to complement each other in ways beyond simply Nesterov-like acceleration, and this is indeed crucial for the packing LP solver in [AZO15a] for getting width-independence.

One particular motivation for our work is a striking discrepancy between bounds provided for packing and covering LPs in [AZO15a]. In particular, they provide a $(1-\epsilon)$-approximation solver for the packing problem in $\tilde{O}(N/\epsilon)$, but they are only able to obtain $\tilde{O}(N/\epsilon^{1.5})$ for the covering problem. In the case of covering, they are unable to use the linear coupling method to achieve Nesterov-like acceleration, and even to get width-independence the authors need to integrate some ad-hoc and complicated techniques. This discrepancy between results for packing and covering LPs is rare, due to the duality between them. Filling this gap is of particular interest, as not being able to do so would suggest some fundamental structural differences between the two dual problems.

Among other things, our result is an improvement over the $\tilde{O}(N/\epsilon^{1.5})$ bound for the covering LP in [AZO15a], and gives a covering LP solver that is both Nesterov-like accelerated and width-independent, which matches the result for packing LP in [AZO15a].

At least as interesting as the $\tilde{O}(1/\epsilon^{0.5})$ improvement for covering LPs, however, is main technical contribution that we developed and exploited to achieve our improvement. At a high level, we also use the optimization approach, and exploit the linear coupling framework as in [AZO15a]. In the setting of covering LPs, we build a discrete lifting technique, which reorganizes the geometry of the problem space, so that the optimal solution won't be hidden in ill-shaped corners that are difficult to reach for the linear coupling method. Once the lifting is performed, covering LP shares all the essential properties necessary to achieve both width-independence and Nesterov-like acceleration as in the case of packing problems, and fits elegantly into the scheme and analysis from [AZO15a] that was developed for packing LPs. We obtain improved $\tilde{O}\left(N/\epsilon\right)$ results for covering LPs, and this provides a unified acceleration method (unified in the sense that it is with the same algorithm and almost identical analysis) for both packing and covering LPs.

Faster Parallel Algorithm

With the abundance of large-scale datasets, as well as the growing reliance on multiprocessors and cloud computing, iterative solvers that can be highly parallelized have drawn a lot of interest. For parallel algorithms, we focus on running time (or equivalently, *depth*) and total work, which are standard notions from parallel computing that correspond to the longest chain of dependent

Paper	Number of Iterations	Total Work
Width-dependent Solvers		
[PST91]	$\tilde{O}(\frac{\rho^2 \operatorname{OPT}^2}{\epsilon^2})$	$\tilde{O}(\frac{\rho^2 \operatorname{OPT}^2 N}{\epsilon^2})$
[AHK12]	$\tilde{O}(\frac{\rho \operatorname{OPT}}{\epsilon^2})$	$\tilde{O}(\frac{\rho \operatorname{OPT} N}{\epsilon^2})$
[Nes05; Nem04]	$\tilde{O}(\frac{\rho \operatorname{OPT}}{\epsilon})$	$\tilde{O}(\frac{\rho \operatorname{OPT} N}{\epsilon})$
[BI04]	$\tilde{O}(\frac{\sqrt{Kn}}{\epsilon})$	$\tilde{O}(\frac{\sqrt{Kn}N}{\epsilon})$
Width-independent Solvers		
[LN93; BBR97; BBR04; You01]	$\tilde{O}(\frac{1}{\epsilon^4})$	$\tilde{O}(\frac{N}{\epsilon^4})$
[You01; You14]	$\tilde{O}(\frac{1}{\epsilon^4})$	$\tilde{O}(\frac{N}{\epsilon^2})$
[AZO15b]	$\tilde{O}(\frac{1}{\epsilon^3})$	$\tilde{O}(\frac{N}{\epsilon^3})$
This work: Algorithm 2	$\tilde{O}(\frac{1}{\epsilon^2})$	$\tilde{O}(\frac{N}{\epsilon^2})$
This work: Algorithm 4	$\tilde{O}(\frac{d}{\epsilon})$	$\tilde{O}(\frac{dN}{\epsilon})$

Table 1.2: Selected work of parallel algorithms for covering LP and packing LP.

operations[2] and the total number of operations performed respectively. For iterative methods, time has a natural correspondence with iteration count, and these two measures have been used to study the performance of previous work on positive LPs.

In Table 1.1, we show selected results of parallel solvers for packing LPs and covering LPs. For all the iterative methods along this line of work, each iteration is computationally dominated by a matrix-vector multiplication, and takes only nearly-linear work in N and $O(\log N)$ depth. Since we are most interested here in the dependence on $\frac{1}{\epsilon}$, reducing the iteration count is the more interesting side of this line of work, and we will often just discuss the number of iterations instead of running time, as they are only off by a factor of $O(\log N)$.

In this work, we describe two parallel algorithms for packing LPs and covering LPs. Similar to our sequential solver, we follow the optimization approach, and develop the *dynamically-bucketed selective coordinate descent (DB-SCD)* method. This method is based on the linear coupling framework, and we integrate a discrete technique addressing the specific issue that slows down the convergence of previous results. With DB-SCD, we improve the dependence on $\frac{1}{\epsilon}$ for both iteration count and total work from $\tilde{O}(\frac{1}{\epsilon^3})$ to $\tilde{O}(\frac{1}{\epsilon^2})$ for packing LPs and covering LPs. When combined with ideas in the accelerated sequential solver discussed earlier, we also get a parallel solver with $\tilde{O}(\frac{d}{\epsilon})$ iterations (recall d is the max number of non-zeros in any row of A), which is always at least as good as the $\tilde{O}(\frac{\sqrt{dn}}{\epsilon})$ bound in [BI04]. We remark that, different from the case of sequential

[2]Following the standard model of parallel computation, when we consider the running time, we assume the algorithm has access to unlimited number of processors.

solvers, there is no parallel algorithm that obtains both width-independence and Nesterov-like acceleration. It is an interesting question whether this is fundamental for packing LPs and covering LPs, or whether there is a more efficient solution to approximate packing LPs and covering LPs in parallel.

Our basic improvement with the DB-SCD method comes from updating only a carefully-chosen subset of variables in each iteration. In particular, we bucket the coordinates of the gradient into buckets of roughly equal gradient magnitude, and then we update the coordinates in one of the buckets. While our particular approach is novel, the basic idea is hardly new to optimization. Indeed, for most non-trivial functions, variables "interfere" with each other, in the sense that variable i's gradient will be affected by the update of variable j (e.g., [Bra+11]). Thus, if we aim to move the variables while maintaining smoothness of the gradients, we have to take interference into consideration. In general, this limits the possible step size.

One way to alleviate the problem of interference is to update fewer variables in each iteration. This potentially permits better control over the changes of gradients for the updated variables, since for the variables not updated, the changes of their gradients don't affect the objective improvement for that iteration. One extreme of this idea is the coordinate descent method [Wri15], where in each iteration only one variable is updated. In this case, the step length of the update on the single variable is often larger than the step length when all variables are moved simultaneously. On the other hand, in most cases the computation of n successive partial derivatives can be more expensive than the computation of all the n partial derivatives for a fixed x, limiting the applicability of the coordinate descent method. When the tradeoff is good between the gain in the step length versus the loss in the computation, coordinate descent can be better than gradient descent in terms of total work ([LS13; AZO15a]). More generally, in the context of solving linear systems, we have the example of Jacobi iterations versus Gauss-Seidel iterations, and similar tradeoffs between interference and running time arise ([AM95; BT91], Chapter 4 of [Saa03]). Still more generally, various efforts to parallelize coordinate descent can be seen as explorations of tradeoffs among the smoothness parameter, the amount of computation, as well as the distributed iteration count ([FR13; RT12; RT14; Bra+11]).

To the best of our knowledge, all these works exhibit an inverse relationship between the number of variables updated each iteration, and the number of total iterations required, i.e., when fewer variables are updated, then more iterative steps are needed. This is what one would naturally expect. Moreover, these works mostly choose the subset of variables to update either by some fixed order, e.g., uniformly at random, or according to some static partition constructed from the sparsity structure of the given instance, e.g., the objective function is separable or the matrix in the problem is block diagonal ([TY09]). Rarely, if at all, is a subset of variables chosen dynamically using knowledge of the actual values of the gradients. Again, this is what one would naturally expect. For example, as Nesterov wrote in his seminal accelerated coordinate descent work [Nes12], if one already computed the whole gradient, then full-gradient methods seem to be better options.

With respect to both of these considerations, our method of selective coordinate descent is novel and quite different. First, at least for the case of packing and covering LPs, we can achieve better parallel running time and better total work by updating fewer (carefully-selected) variables each iteration. Second, our work shows that the extra computation of the whole gradient can help

us select a better subset dynamically (even if we don't update all coordinates). Our results show that both of these directions are worth additional exploration.

1.2 Positive LPs

More general than packing LPs and covering LPs, positive LPs further capture problems requiring both packing and covering constraints, including solving non-negative linear systems, computing tomography, and single/multi commodity flows on graphs.

Formally, positive LP, or equivalently, mixed packing and covering LP, can be written in the standard form:

$$\min\{\lambda : Px \leq \lambda p, Cx \geq c, x \geq 0\} \tag{1.1}$$

where P, C, p, c all have non-negative entries. A $(1+\epsilon)$-approximation is a feasible solution λ, x achieving $\lambda \leq (1+\epsilon)\lambda_{\text{OPT}}$.

In this work we focus on width-independent parallel algorithms that produce $1+\epsilon$ approximations in in $\text{poly}(\log n, \frac{1}{\epsilon})$ time and nearly-linear work. Despite the exciting recent progress following the optimization approach discussed in the previous section, the results are limited to the special cases of packing LPs and covering LPs, since the presence of both packing and covering constraints in one LP makes the smoothness properties of the optimization problem deteriorate significantly, which poses a fundamental obstacle to obtain width-independent solver following this approach. Moreover, almost all the width-independent solvers mentioned in the previous section (except [You01; You14]) don't extend to the more general case of positive LPs. Prior to our work, the best method for mixed packing and covering LPs takes time (and number of iterations) $\tilde{O}(\frac{1}{\epsilon^4})$ ([You01; You14]).

In this work, we present a parallel algorithm that in $\tilde{O}(\frac{1}{\epsilon^3})$ iterations computes a $(1+\epsilon)$-approximate solution, or correctly reports the original mixed packing and covering LP is infeasible. The algorithm is deterministic and width-independent. Again, the bottleneck of each iteration is a matrix-vector multiplication, and can be implemented in $O(\log N)$ depth, in which case the running time of our algorithm is $\tilde{O}(\frac{1}{\epsilon^3})$. The total work of the algorithm we present in this work is $\tilde{O}(\frac{N}{\epsilon^3})$.

We follow the Lagrangian-relaxation approach as in [You01; You14], which is a more combinatorial framework. Nonetheless, insights from continuous convex optimization still play a significant role, and the work demonstrates a different aspect of the integration of continuous and combinatorial approaches.

At a high level, we construct a continuous and smooth potential function to capture the combinatorial packing and covering constraints, which measures how far away the current solution is from satisfying all the captured constraints. We then start with a solution with all variables having very small values, and in each iteration, increase the values of a carefully chosen subset of variables. In particular, increasing the values of variables helps the covering constraints while hurting the packing constraints, and the subset of variables that we increase is chosen according to the potential function so that the trade-off between covering constraints and packing constraints is advantageous for us. Moreover, different from previous work using this approach, where all the

variables in the chosen subset are increased by the same fixed quantity, we further incorporate the gradient of the potential function into the size of the update of variables. This discriminative step size allows more aggressive updates on average, and is directly motivated by the line of work using gradient based optimization methods.

Furthermore, as the potential function captures the packing and covering constraints in a LP, there is an inherent connection between the gradients of the potential function and the variables in the dual LP of the original LP. By incorporating the gradients into the update of the variables, we exploit this connection, and essentially construct primal and dual solutions simultaneously. While previous methods analyze the convergence by tracking the progress made on the primal LP exclusively, our integration of the gradient into the update size gives us a primal-dual perspective of the convergence of our algorithm, and allows us to argue that either a feasible primal solution is obtained, or a dual solution can be constructed to certify the infeasibility of the primal LP.

1.3 Organization

In Chapter 2, we describe the preliminaries and the formulation of the smooth convex optimization problem for packing LPs and covering LPs, followed by the discrete lifting techniques for covering LPs, and the unified $\tilde{O}(fracN\epsilon)$ sequential algorithm for both packing LPs and covering LPs.

In Chapter 3, we describe the dynamically-bucketed selective coordinate descent, and the width-independent $\tilde{O}(\frac{1}{\epsilon^2})$ time parallel solver for packing LPs and covering LPs. We also present the $\tilde{O}(\frac{d}{\epsilon})$ time parallel algorithm based on DB-SCD as well as ideas from the accelerated sequential algorithm.

In Chapter 4, we describe the $\tilde{O}(\frac{1}{\epsilon^3})$ time parallel solver for general positive LPs, which is based on the Lagrangian-relaxation framework.

1.4 Bibliographic Notes

The results in this dissertation were based on collaboration with Satish Rao, Michael Mahoney, Peng Zhang and Nishanth Mohan. The results include previous publications [WRM16; Mah+16; Wan+15]. I thank my collaborators for allowing the inclusion of coauthored work in this dissertation.

Chapter 2

Sequential Algorithm for Packing LPs and Covering LPs

2.1 Preliminaries and Smoothing the Objective

Without loss of generality, given a packing LP in its generic form, one can scale the coefficients, in which case one can write the fractional packing LP in the standard form:

$$\max_{x \geq 0} \{\vec{1}^T x : Px \leq \vec{1}\}, \tag{2.1}$$

where $P \in \mathbb{R}_{\geq 0}^{m \times n}$. If the optimal value of (2.1) is OPT, we say that x is a $(1-\epsilon)$-*approximation* for the packing LP if $Px \leq \vec{1}$ and $\vec{1}^T x \geq (1-\epsilon)\,\mathrm{OPT}$.

Similarly, the fractional covering problem, can be written in the standard form as:

$$\min_{x \geq 0} \{\vec{1}^T x : Cx \geq \vec{1}\}. \tag{2.2}$$

In this case, we say that x is a $(1+\epsilon)$-*approximation* for the covering LP if $Cx \geq \vec{1}$ and $\vec{1}^T x \leq (1+\epsilon)\,\mathrm{OPT}$.

When our discussion applies to both packing LPs and covering LPs, we also use A to denote the constraint matrix of the LP, i.e., P and C for packing and covering respectively. We denote m as the number of rows in A, i.e., the number of constraints, n as the number of coordinates of x, i.e., the number of variables, N as the size of the formulation, i.e., the total number of nonzeros in A, ρ as the max entry in A, and d as the max number of nonzeros in any row of A.

Now we describe how to pose the LPs as smooth convex optimization problems. We discuss packing LP and covering LP separately, although due to the duality between them, the results (in this section) for them are symmetric in terms of both statements and proofs. Since we follow the same high level optimization approach and the same smoothing step as in [AZO15a], the lemmata and the proofs in this section are either the same or analogous to the results in [AZO15a]. We defer the proofs of this section to Section 2.5.

Packing LP

To start, let's assume that

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

This assumption is without loss of generality: since we are interested in multiplicative $(1-\epsilon)$-approximation, we can simply scale P for this to hold without sacrificing approximation quality. With this assumption, the following lemma holds.

Lemma 2.1.1. $\mathrm{OPT} \in [1, n]$

The first step (i.e., the smoothing step) is to turn the positive LPs into smoothed objective functions. For packing LP, we use

$$f_\mu(x) \stackrel{\text{def}}{=} -\vec{1}^T x + \max_{y\geq 0}\{y^T(Px-\vec{1}) + \mu H(y)\},$$

and it is a smoothed objective in the sense that it turns the packing constraints into soft penalties, with $H(y)$ being a regularization term. Here, same as in [AZO15b; AZO15a], we use the generalized entropy $H(y) = -\sum_j y_j \log y_j + y_j$, where μ is the smoothing parameter balancing the penalty and the regularization. It is straightforward to compute the optimal y, and write $f_\mu(x)$ explicitly, as stated in the following lemma.

Lemma 2.1.2. *$f_\mu(x) = -\vec{1}^T x + \mu\sum_{j=1}^m p_j(x)$, where $p_j(x) \stackrel{\text{def}}{=} \exp(\frac{1}{\mu}((Px)_j - 1))$ is what we call the exponential penalty of the j-th packing constraint.*

With OPT being at least 1, the error we introduce in the smoothing step will be small enough that $f_\mu(x)$ approximates the packing LP well enough with respect to ϵ around the optimum. If we let x^* be an optimal solution, and $u^* \stackrel{\text{def}}{=} (1-\epsilon/2)x^*$, then we have the properties in the following lemma.

Lemma 2.1.3. *Setting the smoothing parameter $\mu = \frac{\epsilon}{4\log(nm/\epsilon)}$, we have*

1. $f_\mu(u^*) \leq -(1-\epsilon)\,\mathrm{OPT}$.
2. $f_\mu(x) \geq -(1+\epsilon)\,\mathrm{OPT}$ *for every* $x \geq 0$.
3. *Letting $x_0 \geq 0$ be such that $x_0[i] = \frac{1-\epsilon/2}{n\|P_{:i}\|_\infty}$ for each $i \in [n]$, we have $f_\mu(x_0) \leq -\frac{1-\epsilon}{n}$.*
4. *For any $x \geq 0$ satisfying $f_\mu(x) \leq 0$, we must have $Px \leq (1+\epsilon)\vec{1}$, and thus $\vec{1}^T x \leq (1+\epsilon)\,\mathrm{OPT}$.*
5. *If $x \geq 0$ satisfies $f_\mu(x) \leq -(1-O(\epsilon))\,\mathrm{OPT}$, then $\frac{1}{1+\epsilon}x$ is a $(1-O(\epsilon))$-approximation to the packing LP.*

6. *The gradient of $f_\mu(x)$ is*

$$\nabla f_\mu(x) = -\vec{1} + P^T\overrightarrow{p(x)} \quad \textit{where} \quad p_j(x) \stackrel{\text{def}}{=} \exp(\frac{1}{\mu}((Px)_j - 1),$$

and $\nabla_i f_\mu(x) = -1 + \sum_j P_{ji} p_j(x) \in [-1, \infty]$.

Covering LP

The smoothing step for covering LP is basically the symmetric version of what we do for packing LP. This time, we can assume that

$$\min_{j\in[m]} \|C_{j:}\|_\infty = 1.$$

This assumption is without loss of generality: since we are interested in multiplicative $(1+\epsilon)$-approximation, we can simply scale C for this to hold without sacrificing approximation quality. With this assumption, the following lemma holds.

Lemma 2.1.4. $\mathrm{OPT} \in [1, m]$

Observation 2.1.5. *Since we are interested in a $(1+\epsilon)$-approximation, then with the above assumption, we can also eliminate the very small and very large entries from the matrix as follows. If some entry $C_{ji} \le \epsilon/(mn)$, then since $\mathrm{OPT} \le m$ we have that $C_{ji}x_i^* \le \epsilon/n$, and so we can just increase each variable by ϵ/n, in which case we can recover the loss from setting C_{ji} equal to 0 from the variable in the j-th constraint with coefficient at least 1. On the other hand, if some entry $C_{ji} \ge n/\epsilon$, then we can just set variable i to be at least ϵ/n and ignore constraint j. Thus, we can eliminate very small and very large entries from the matrix C, and we only incur an additional cost of ϵ, but since $\mathrm{OPT} \ge 1$, we still obtain a $(1+O(\epsilon))$-approximation.*

Analogously, for covering LP, the smooth function we optimize is

$$g_\mu(x) \stackrel{\text{def}}{=} \vec{1}^T x + \mu \sum_{j=1}^{m} q_j(x),$$

where $q_j(x) \stackrel{\text{def}}{=} \exp(\frac{1}{\mu}(1 - (Cx)_j))$ is the exponential penalty of the j-th covering constraint.

Analogous to f_μ, optimizing $g_\mu(x)$ gives a good approximation to OPT of the covering LP, in the following sense. If we let x^* be an optimal solution, and $u^* \stackrel{\text{def}}{=} (1+\epsilon/2)x^*$, then we have the properties in the following lemma.

Lemma 2.1.6. *Setting the smoothing parameter $\mu = \frac{\epsilon}{4\log(nm/\epsilon)}$, we have*

1. $g_\mu(u^*) \le (1+\epsilon)\,\mathrm{OPT}$.
2. $g_\mu(x) \ge (1-\epsilon)\,\mathrm{OPT}$ *for any* $x \ge 0$.
3. *For any $x \ge 0$ satisfying $g_\mu(x) \le 2\,\mathrm{OPT}$, we must have $Cx \ge (1-\epsilon)\vec{1}$.*

4. *If $x \geq 0$ satisfies $g_\mu(x) \leq (1 + O(\epsilon))\,\mathrm{OPT}$, then $\frac{1}{1-\epsilon}x$ is a $(1 + O(\epsilon))$-approximation to the covering LP.*

5. *The gradient of $g_\mu(x)$ is*

$$\nabla g_\mu(x) = \vec{1} - C^T\overrightarrow{q(x)} \quad \textit{where} \quad q_j(x) \stackrel{\text{def}}{=} \exp(\frac{1}{\mu}(1 - (Cx)_j),$$

and $\nabla_i g_\mu(x) = 1 - \sum_j C_{ji} q_j(x) \in [-\infty, 1]$.

Once the smoothing step is performed, the problem of approximately solving the original packing LP or covering LP is reduced to approximately optimize the function $f_\mu(x)$ or $g_\mu(x)$. We proceed to give the technique overview of how we solve the optimization problem.

2.2 Technical Overview

At a high level, we (as well as Allen-Zhu and Orecchia [AZO15b; AZO15a]) use the same two-step approach of Nesterov [Nes05]. The first step involves smoothing, which transforms the constrained problem into a smooth objective function with trivial or no constraints. In the setting of packing LP and covering LP, we have the smooth functions $f_\mu(x)$ and $g_\mu(x)$. By smooth, we mean that the gradient of the objective function has some property in the flavor of Lipschitz continuity. Once smoothing is accomplished, the second step uses one of several first order methods for convex optimization in order to obtain an approximate solution. Examples of standard application of this approach to covering LPs includes the width-dependent solvers of [Nes05; Nem04] as well as multiplicative weights update solvers [AHK12].

The first width-independent result following the optimization approach in [AZO15b] achieves width-independence by truncating the gradient, thus effectively reducing the width to 1. The algorithm uses, in a white-box way, the coupling of mirror descent and gradient descent from [ZO14], which can be viewed as a re-interpretation of Nesterov's accelerated gradient method [Nes05]. However, although [AZO15b] uses a coupling of mirror descent and gradient descent, the role of gradient descent is only for width-independence, i.e., to cover the loss incurred by the large component of the gradient (see Eqn. (2.7) below for the precise formulation of this loss), and it is independent of the mirror descent part acting on the truncated gradient. In addition, [AZO15b] deviates from the canonical smoothing with entropy, as it instead uses generalized entropy. Importantly, the objective function to be minimized is *not* smooth in the standard Lipschitz continuity sense, but it does satisfy a similar local Lipschitz property.

To improve the sequential packing solver in [AZO15b] with convergence $\tilde{O}(1/\epsilon^3)$ to $\tilde{O}(1/\epsilon)$, the same authors in [AZO15a] apply a stochastic coordinate descent method based on the linear coupling idea. Barring the difference between Lipschitz and local Lipschitz continuity, the results in [AZO15a] can be viewed as a variant of accelerated coordinate descent method [Nes12]. There are two places where the algorithm achieves an improvement over prior packing-covering results.

- One factor of improvement is due to the better coordinate-wise Lipschitz constant over the full dimensional Lipschitz constant. Intuitively, in the case of packing or covering, the gradient of variable x_i depends on the penalties of constraints involving x_i, which further depend on all the variables in those constraints. As a result, if we move all the variables simultaneously, we can only take a small step before changing the gradient of x_i drastically.

- The other factor of improvement comes from accelerating the gradient method. The role of gradient descent in the packing solver of [AZO15a] is twofold. First, it covers the loss incurred by the large component of the gradient as in [AZO15b] to give width-independence. Second, to accelerate the coupling as in [ZO14], the gradient descent also needs to cover the regret term incurred by the mirror descent step (see Eqn. (2.7) below for the precise formulation of this regret). The adoption of A-norm (defined in Eqn. (2.6) below) enables the acceleration. This A-norm works particularly well for packing problems, in the sense that it easily leads to good diameter bounds: since the packing constraints impose a naive upper bound of $x_i^* \leq 1/\|A_{:i}\|_\infty$ on each variable, thus the feasible region has a small diameter $\max_{x:f(x)\leq f(x_0)} \|x - x^*\|_A$.

  The importance of the small diameter is twofold. First, the diameter naturally arises in the convergence bound of gradient based methods, so we always need to use a norm or proximal setup giving small diameter to achieve good convergence. Second, and more importantly, when we confine the function $f_\mu(x)$ inside in this box type space with the particular upper-bound on each coordinate, the function behaves much smoother for the linear coupling method. More specifically, the small diameter $[0, 1/\|A_{:i}\|_\infty]$ on each coordinate relates the mirror descent step length and the gradient descent step length. As the regret term in mirror descent and the improvement of gradient descent step are both proportional to their respective step lengths, the small coordinate-wise diameter makes it possible to use gradient descent improvement to cover the mirror descent regret.

The combination of gradient truncation, stochastic coordinate descent, and acceleration due to small diameter in A-norm leads to the $\tilde{O}(N/\epsilon)$ solver for the packing LP [AZO15a].

Shifting to solvers for the covering LP, one obvious obstacle to reproducing the packing result is we no longer have the small diameter in A-norm. Indeed, a naive coordinate-wise upper bound from the covering constraints only gives $x_i^* \leq 1/\min_j\{A_{ji} : A_{ji} > 0\}$. Because of this, the covering solver in [AZO15a] instead use the proximal setup in their earlier work [AZO15b]. The particular proximal setup gives a good diameter for the feasible region they use, but it doesn't give a similarly good coordinate-wise diameter to enable the acceleration. To improve upon the $O(1/\epsilon^2)$ convergence of standard mirror descent, the authors use a negative-width technique as in [AHK12] (Theorem 3.3 with $l = \sqrt{\epsilon}$). This then leads to the (improved, but still worse than for packing) $\tilde{O}(1/\epsilon^{1.5})$ convergence rate. In addition, since they truncate the gradient at a smaller threshold to cover the loss incurred by the large component, they need a more complicated gradient step, leading to a more complicated algorithm than for the packing LP.

To get an $\tilde{O}(1/\epsilon)$ solver for the covering LP, it seems crucial to relate the gradient descent step and mirror descent step the same way as in the packing solver in [AZO15a]. Thus, we will stick

with the A-norm, and we will work directly to reduce the diameter. Our main result (presented next in Section 2.3) is a general discrete lifting technique that transform the feasible region of covering LP to achieve the same diameter property as in the case of packing LP, and this enables us to extend all the crucial ideas of the packing solver in [AZO15a], as outlined in this section, to get a covering solver with running time $\tilde{O}(N/\epsilon)$.

2.3 Diameter Reduction Method for Covering Problems

As we disccused in the overview, the optimization schema developed for packing LPs in [AZO15a] relies on the particular geometry of the problem space, where the optimal x^* lies in a box type space with good coordinate-wise diameters, i.e., $x_i^* \leq 1/\|P_{:i}\|_\infty$, thus we can optimize $f_\mu(x)$ over this particular region, where it has various properties that pair particularly well with the optimization schema. In this section, we describe our discrete lifting technique that reformulates a covering LP to give the similar geometry. Essentially, we "'lift"' the covering LP to a higher dimensional space, i.e., the reformulated LP will (at most a $\log \frac{mn}{\epsilon}$ factor) more variables, but in this space, we can guarantee the optimal of $g_\mu(x)$ is within a box with desired coordinate-wise diameters.

Given any covering LP of the form (2.2), characterized by a matrix C, we formulate an equivalent covering LP with good diameter properties. This will involve adding variables and redundant constraints. We use $i \in [n]$ to denote the indices of the variables (i.e., columns of C) and $j \in [m]$ to denote the indices of constraints (i.e., rows of C). For any $i \in [n]$, let

$$r_i \stackrel{\text{def}}{=} \frac{\max_j\{C_{ji} : C_{ji} > 0\}}{\min_j\{C_{ji} : C_{ji} > 0\}},$$

be the ratio between the largest non-zero coefficient and the smallest non-zero coefficient of variable $x[i]$ in all constraints, and let $n_i \stackrel{\text{def}}{=} \lceil \log r_i \rceil$. We first duplicate each original variable n_i times to obtain $\bar{x}[(i,l)], i \in [n], l \in [n_i]$ as the new variables. In terms of the coefficient matrix, we now have a new matrix, call it $\bar{C} \in \mathbb{R}_{\geq 0}^{m \times (\sum_i n_i)}$, which contains n_i copies of the i-th column $C_{:i}$. We denote a column of $\bar{C}$ by the tuple (i,l) with $l \in [n_i]$. Obviously, the covering LP given by $\bar{C}$ is equivalent to the original covering LP given by C. Adding additional copies of variables, however, will allow us to improve the diameter. To reduce the diameter of this new covering LP, we further decrease some of the coefficients in $\bar{C}$, and we put upper bounds on the variables. In particular, for j, i, l, we have

$$\bar{C}_{j,(i,l)} = \min\{C_{j,i}, 2^l \min_j\{C_{ji} : C_{ji} > 0\}\}, \tag{2.3}$$

and for variable $\bar{x}[(i,l)]$, we add the constraint

$$\bar{x}[(i,l)] \leq \frac{2}{2^l \min_j\{C_{ji} : C_{ji} > 0\}}. \tag{2.4}$$

The next lemma shows that the covering LP given by $\bar{C}$ and the covering LP given by C are equivalent.

Lemma 2.3.1. *Let* OPT *be the optimal value of the covering LP given by* C*, and let* $\overline{\mathrm{OPT}}$ *be the optimal of the covering LP given by* $\bar{C}$ *and* (2.4)*, as constructed above; then* $\mathrm{OPT} = \overline{\mathrm{OPT}}$.

Proof. Given any feasible solution $\bar{x}$, consider the solution x where $x[i] = \sum_{l=1}^{n_i} \bar{x}[(i,l)]$. It is obvious $\vec{1}^T x = \vec{1}^T \bar{x}$, and $Cx \geq \vec{1}$, as coefficients in $\bar{C}$ are no larger than coefficients in C. Thus $\mathrm{OPT} \leq \overline{\mathrm{OPT}}$.

For the other direction, consider any feasible x. For each i, we can assume without loss of generality that

$$x[i] \leq \frac{1}{\min_j\{C_{ji} : C_{ji} > 0\}}.$$

Let l_i be the largest index such that

$$x[i] \leq \frac{2}{2^{l_i} \min_j\{C_{ji} : C_{ji} > 0\}},$$

and then let

$$\bar{x}[(i,l)] = \begin{cases} x[i] & \text{if } l = l_i \\ 0 & \text{if } l \neq l_i \end{cases}.$$

By construction, $\bar{x}$ satisfies all the upper bounds described in (2.4). Furthermore, for constraint j, we must have $\bar{C}_{j:}\bar{x} \geq 1$. Since for any i, $\bar{C}_{j,(i,l_i)}$ differs from C_{ji} only when $C_{ji} > 2^{l_i} \min_j\{C_{ji} : C_{ji} > 0\}$, and we must have $l_i < n_i$ in this case by definition of n_i, which gives $\bar{x}[(i,l_i)] = x[i] \geq \frac{1}{2^{l_i} \min_j\{C_{ji} : C_{ji} > 0\}}$ by our choice of l_i being the largest possible. Then we know $\bar{C}_{j,(i,l_i)} = 2^{l_i} \min_j\{C_{ji} : C_{ji} > 0\}$, so the j-th constraint is satisfied. Thus $\mathrm{OPT} \geq \overline{\mathrm{OPT}}$, and we can conclude $\mathrm{OPT} = \overline{\mathrm{OPT}}$. □

Given that we have shown that the covering LP defined by $\bar{C}$ and that defined by C are equivalent, we now point out that the seemingly-redundant constraints of (2.4) turn out to be crucial. The reason is that the feasible region now has a small diameter in the coordinate-wise weighted 2-norm $\|\cdot\|_C$. In particular, we can rewrite the constraints (2.4) to be

$$\bar{x}[(i,l)] \leq \frac{2}{\|\bar{C}_{:(i,l)}\|_\infty}.$$

For any i, consider the row $j^* = \mathrm{argmax}_j\{C_{ji}, C_{ji} > 0\}$, this is the same upper bound on $\bar{x}[(i,l)]$ for $l < n_i$, and it is a relaxation on $\bar{x}[(i,n_i)]$.

The price we pay for this diameter improvement is that the new LP defined by $\bar{C}$ is larger than that defined by C. Two comments on this are in order. First, by Observation 2.1.5, r_i is bounded by n^2m/ϵ^2, and so the diameter reduction step only increases the problem size by $O(\log(mn/\epsilon))$. Second, we have presented our diameter reduction as an explicit pre-processing step so we can use one unified optimization algorithm (Algorithm 1 below) for both packing and covering, but in practice the diameter reduction would not have to be carried out explicitly. It can equivalently be implemented implicitly within the algorithm (a trivially-modified version of Algorithm 1 below)

by randomly choosing a scale after picking the coordinate i and then computing $\bar{C}_{j,(i,l)}$ in (2.3) by shifting bits on the fly.

Given this reduction, in the rest of the paper, when we refer to the covering LP, we will implicitly be referring to the diameter reduced version, and we have the additional guarantee that there exists an optimal solution x^* to (2.2) such that

$$0 \leq x^*[i] \leq \frac{2}{\|C_{:i}\|_\infty} \quad \forall i \in [n]. \tag{2.5}$$

2.4 Accelerated Solver for (Packing and) Covering LPs

In this section, we will present our solver for covering LPs of the form (2.2). To motivate this, recall that for packing problems of the form (2.1), bounds of the form (2.5) automatically follow from the packing constraints $Px \leq \vec{1}$, so for both packing LP and covering LP, we can use a box type problem space

$$\Delta \stackrel{\text{def}}{=} \{x \in \mathbb{R}^n : 0 \leq x[i] \leq \frac{3}{\|A_{:i}\|_\infty}\},$$

and pose the problems as optimizing $f_\mu(x)$ and $g_\mu(x)$ over Δ respectively. In particular, the u^* (i.e., the approximate optimal for $f_\mu(x)$ or $g_\mu(x)$ in Lemma 2.1.3 and Lemma 2.1.6 will be inside Δ.

For readers familiar with the packing LP solver in [AZO15a], it should be plausible that—once we have this diameter property—the same stochastic coordinate descent optimization scheme will lead to a $\tilde{O}(N/\epsilon)$ covering LP solver. We now show that indeed the same optimization algorithm for packing LPs can be easily extended to solving covering LPs, thus establishing a unified acceleration method for packing and covering problems.

Accelerated Coordinate Descent Algorithm

Consider Algorithm 1, which is our main accelerated stochastic coordinate descent for both packing and covering. The correctness of this algorithm and its running time guarantees for the packing problem have already been nicely presented in [AZO15a], so here we will focus on the covering problem.[1] We use e_i to denote the standard basis vectors.

Recall that we will find a $(1+\epsilon)$-approximation of the covering LP by approximately minimizing

$$g_\mu(x) \stackrel{\text{def}}{=} \vec{1}^T x + \mu \sum_{j=1}^{m} q_j(x),$$

[1]For packing LPs, the input will be the matrix P, function f_μ, and $\Delta \stackrel{\text{def}}{=} \{x \in \mathbb{R}^n : 0 \leq x[i] \leq \frac{1}{\|P_{:i}\|_\infty}\}$, with $x^{\text{start}} = \frac{1-\epsilon/2}{n\|P_{:i}\|_\infty}$

Algorithm 1 Accelerated sequential solver for packing LPs and covering LPs

Input: $C \in \mathbb{R}^{m\times n}_{\geq 0}, x^{\text{start}} \in \Delta, g_\mu, \epsilon$ **Output**: $y_T \in \Delta$

$\mu \leftarrow \frac{\epsilon}{4\log(nm/\epsilon)}, L \leftarrow \frac{4}{\mu}, \tau \leftarrow \frac{1}{8nL}$

$T \leftarrow \lceil 8nL\log(1/\epsilon)\rceil = \tilde{O}(\frac{n}{\epsilon})$

$x_0, y_0, z_0 \leftarrow x^{\text{start}}, \alpha_0 \leftarrow \frac{1}{nL}$

for $k = 1$ to T **do**

  $\alpha_k \leftarrow \frac{1}{1-\tau}\alpha_{k-1}$

  $x_k \leftarrow \tau z_{k-1} + (1-\tau)y_{k-1}$

  Select $i \in [n]$ uniformly at random.

  ▷ Gradient truncation:

  Let $(\xi_k^{(i)}) \leftarrow \begin{cases} -\mathrm{e}_i & \nabla_i g_\mu(x_k) < -1 \\ \nabla_i g_\mu(x_k)\cdot \mathrm{e}_i & \nabla_i g_\mu(x_k) \in [-1,1] \\ \mathrm{e}_i & \nabla_i g_\mu(x_k) > 1 \end{cases}$

  ▷ Mirror descent step:

  $z_k \leftarrow z_k^{(i)} \stackrel{\text{def}}{=} \operatorname{argmin}_{z\in\Delta}\{V_{z_{k-1}}(z) + \langle z, n\alpha_k\xi_k^{(i)}\rangle\}$.

  ▷ Gradient descent step:

  $y_k \leftarrow y_k^{(i)} \stackrel{\text{def}}{=} x_k + \frac{1}{n\alpha_k L}(z_k^{(i)} - z_{k-1})$

end for

return y_T.

where $q_j(x) \stackrel{\text{def}}{=} \exp(\frac{1}{\mu}(1-(Cx)_j))$, over the region

$$\Delta \stackrel{\text{def}}{=} \{x \in \mathbb{R}^n : 0 \leq x[i] \leq \frac{3}{\|C_{:i}\|_\infty}\}.$$

Our main result is summarized in the following theorems.

Theorem 2.4.1. *With x^{start} computable in time $\tilde{O}(N)$ to be specified later, Algorithm 1 outputs y_T satisfying $\mathbb{E}[g_\mu(y_T)] \leq (1+6\epsilon)\,\mathrm{OPT}$, and the running time is $\tilde{O}(N/\epsilon)$.*

Given Theorem 2.4.1, a standard application of Markov bound, together with Lemma 2.1.6(5), gives the following theorem as a corollary.

Theorem 2.4.2. *There is a algorithm that, with probability at least $9/10$, computes a $(1+O(\epsilon))$-approximation to the fractional covering problem and has $\tilde{O}(N/\epsilon)$ expected running time.*

Not surprisingly, due to the structural similarities of packing and covering problems after diameter reduction, the correctness of Algorithm 1 for covering can be established using the same approach as [AZO15a] did for packing.

Before proceeding with our proof of these theorems, we discuss briefly the optimization scheme from [AZO15a] we will use. First, observe that the C-norm, where

$$\|x\|_C = \sqrt{\sum_i \|C_{:i}\|_\infty x[i]^2}, \tag{2.6}$$

is used as the proximal setup for mirror descent. The corresponding distance generating function is $w(x) = \frac{1}{2}\|x\|_C^2$, and the Bregman divergence is $V_x(y) = \frac{1}{2}\|x-y\|_C^2$.[2]

Next, observe that Algorithm 1 works as follows. Each iteration integrates a mirror descent step and a gradient descent step. Different from the coupling of gradient descent and mirror descent in the parallel packing algorithm in Section 3.2, in Algorithm 1, the gradient step and mirror descent step are two seperate steps, and the update takes a convex combination of the two steps.

The role of gradient descent is twofold. First, it covers the loss incurred by the large component of the gradient as in the parallel packing case to give width-independence. Second, to accelerate the coupling as in [ZO14], the gradient descent also needs to cover the regret term incurred by the mirror descent step (see Eqn. (2.7) for the precise formulation of this regret). With the scale based lifting, we get a small diameter $[0, 3/\|C_{:i}\|_\infty]$ on each coordinate. We will see this relates the mirror descent step length and the gradient descent step length, and makes it possible to use gradient descent improvement to cover the mirror descent regret. This enables us to telescope both the loss and the regret through all iterations and to bound the total by the gap between $g_\mu(x^{\text{start}})$ and the optimal. The remaining terms in the mirror descent also telescope through the algorithm, and they are bounded in total by the distance (in C-norm) from x^{start} to $u^* \in \Delta$. Then, given these, all we need is an initial condition x^{start} that is not too far away from the optimal in terms of the function value and not too far away from u^* in C-norm. For packing, starting with all 0's will work. For covering, we will show later a good enough x^{start} can be obtained in $\tilde{O}(N)$.

Analysis of Algorithm 1 for Covering LPs

Once we perform the discrete lifting, we have that inside the region Δ, our function $g_\mu(x)$ is *locally Lipschitz continuous*, in a sense quantified by the following lemma, and so we have a good improvement with a gradient step within certain range.

Lemma 2.4.3. *Let $L \stackrel{\text{def}}{=} \frac{4}{\mu}$, for any $x \in \Delta$, and $i \in [n]$*

1. *If $\nabla_i g_\mu(x) \in (-1, 1)$, then for all $|\gamma| \le \frac{1}{L\|C_{:i}\|_\infty}$, we have*

$$|\nabla_i g_\mu(x) - \nabla_i g_\mu(x + \gamma\, \mathrm{e}_i)| \le L\|C_{:i}\|_\infty |\gamma|.$$

2. *If $\nabla_i g_\mu(x) \le -1$, then for all $\gamma \le \frac{1}{L\|C_{:i}\|_\infty}$, we have*

$$\nabla_i g_\mu(x + \gamma\, \mathrm{e}_i) \le (1 - \frac{L\|C_{:i}\|_\infty}{2}|\gamma|)\nabla_i g_\mu(x).$$

[2] In particular, w is a 1-strongly convex function with respect to $\|\cdot\|_C$, and $V_x(y) \stackrel{\text{def}}{=} w(y) - \langle \nabla w(x), y - x\rangle - w(x)$. See [ZO14] for a detailed discussion of mirror descent as well as and several interpretations.

Proof. First, observe the following:

$$\left|\log\frac{1-\nabla_i g_\mu(x+\gamma\,\mathrm{e}_i)}{1-\nabla_i g_\mu(x)}\right| = \left|\int_0^\gamma -\frac{\nabla_{ii} g_\mu(x+\nu\,\mathrm{e}_i)}{1-\nabla_i g_\mu(x+\nu\,\mathrm{e}_i)}d\nu\right| = \left|\frac{1}{\mu}\int_0^\gamma \frac{\sum_j C_{ji}^2 q_j(x+\nu\,\mathrm{e}_i)}{\sum_j C_{ji} q_j(x+\nu\,\mathrm{e}_i)}d\nu\right|$$
$$\leq \left|\frac{1}{\mu}\int_0^\gamma \|C_{:i}\|_\infty d\nu\right| = \frac{1}{\mu}|\gamma|\|C_{:i}\|_\infty = \frac{L\|C_{:i}\|_\infty}{4}|\gamma|.$$

Then, we have

$$\exp(-\frac{L\|C_{:i}\|_\infty}{4}|\gamma|) \leq \frac{1-\nabla_i g_\mu(x+\gamma\,\mathrm{e}_i)}{1-\nabla_i g_\mu(x)} \leq \exp(\frac{L\|C_{:i}\|_\infty}{4}|\gamma|).$$

Since $\frac{L\|C_{:i}\|_\infty}{4}|\gamma| \leq \frac{1}{4}$ by our assumption, we have $x \leq e^x - 1 \leq 1.2x$ for $x \in [-\frac{1}{4}, \frac{1}{4}]$. Thus, it follows that

$$-\frac{L\|C_{:i}\|_\infty}{4}|\gamma| \leq \frac{\nabla_i g_\mu(x) - \nabla_i g_\mu(x+\gamma\,\mathrm{e}_i)}{1-\nabla_i g_\mu(x)} \leq 1.2\frac{L\|C_{:i}\|_\infty}{4}|\gamma|.$$

Finally, to prove the lemma we consider the following two cases:

1. If $\nabla_i g_\mu(x) \in (-1, 1)$, then we have

$$|\nabla_i g_\mu(x) - \nabla_i g_\mu(x+\gamma\,\mathrm{e}_i)| \leq 1.2(1-\nabla_i g_\mu(x))\frac{L\|C_{:i}\|_\infty}{4}|\gamma| \leq L\|C_{:i}\|_\infty|\gamma|.$$

2. If $\nabla_i g_\mu(x) \leq -1$, then $1 - \nabla_i g_\mu(x) \leq -2\nabla_i g_\mu(x)$, and

$$\nabla_i g_\mu(x+\gamma\,\mathrm{e}_i) \leq \nabla_i g_\mu(x) + (1-\nabla_i g_\mu(x))\frac{L\|C_{:i}\|_\infty}{4}|\gamma| \leq (1 - \frac{L\|C_{:i}\|_\infty}{2}|\gamma|)\nabla_i g_\mu(x).$$

□

We call $L\|C_{:i}\|_\infty$ the *coordinate-wise local Lipschitz constant*. For readers familiar with accelerated coordinate descent method (ACDM) [Nes12], the C-norm is essentially the $\|\cdot\|_{1-\alpha}$ in ACDM [Nes12] with $\alpha = 0$, except we use the coordinate-wise local Lipschitz constant instead of the Lipschitz constant to weight each coordinate. The significance of Lemma 2.4.3 is that for covering LPs the coordinate-wise diameter is inversely proportional to the coordinate-wise local Lipschitz constant. (This fact has been established previously for the case of packing LPs [AZO15a].)

The following two lemmas are invariant to the differences between packing and covering problems, and so they follow directly from the same results in [AZO15a] (but, for completeness, we include the proofs in Section 2.5). The values of parameters μ, L, τ, α_k can be found in the description of Algorithm 1. The first lemma says that the gradient step we take is always valid (i.e., in Δ), which is crucial in the sense that the gradient descent improvement is proportional to the step length, and we need the step length to be at least $\frac{1}{n\alpha_k L}$ of the mirror descent step length for the coupling to work.

Lemma 2.4.4. *We have $x_k, y_k, z_k \in \Delta$ for all $k = 0, 1, \ldots, T$.*

The second lemma is clearly crucial to achieve the nearly linear time $\tilde{O}(N/\epsilon)$ algorithm.

Lemma 2.4.5. *Each iteration can be implemented in expected $O(N/n)$ time.*

Mirror Descent Step

We now analyze the mirror descent step of Algorithm 1:

$$z_k \leftarrow z_k^{(i)} \stackrel{\text{def}}{=} \operatorname*{argmin}_{z\in\Delta}\{V_{z_{k-1}}(z) + \langle z, n\alpha_k\xi_k^{(i)}\rangle\}.$$

A lemma of the following form, which here applies to both covering and packing LPs, is needed, and it's proof follows from the textbook mirror descent analysis (or, e.g., Lemma 3.5 in [AZO15a]).

Lemma 2.4.6. $\langle n\alpha_k\xi_k^{(i)}, z_{k-1} - u^*\rangle \le n^2\alpha_k^2 L\langle \xi_k^{(i)}, x_k - y_k^{(i)}\rangle + V_{z_{k-1}}(u^*) - V_{z_k}(u^*)$

Proof. The lemma follows from the following chain of equalities and inequalities.

$$\begin{aligned}
\langle n\alpha_k\xi_k^{(i)}, z_{k-1} - u^*\rangle &= \langle n\alpha_k\xi_k^{(i)}, z_{k-1} - z_k\rangle + \langle n\alpha_k\xi_k^{(i)}, z_k - u^*\rangle \\
&= n^2\alpha_k^2 L\langle \xi_k^{(i)}, x_k - y_k^{(i)}\rangle + \langle n\alpha_k\xi_k^{(i)}, z_k - u^*\rangle \\
&\le n^2\alpha_k^2 L\langle \xi_k^{(i)}, x_k - y_k^{(i)}\rangle + \langle -\nabla V_{z_{k-1}}(z_k^{(i)}), z_k - u^*\rangle \\
&\le n^2\alpha_k^2 L\langle \xi_k^{(i)}, x_k - y_k^{(i)}\rangle + V_{z_{k-1}}(u^*) - V_{z_k^{(i)}}(u^*) - V_{z_{k-1}}(z_k^{(i)}) \\
&\le n^2\alpha_k^2 L\langle \xi_k^{(i)}, x_k - y_k^{(i)}\rangle + V_{z_{k-1}}(u^*) - V_{z_k}(u^*).
\end{aligned}$$

The first equality follows by adding and subtracting z_k, and the second equality comes from the gradient step $y_k^{(i)} = x_k + \frac{1}{n\alpha_k L}(z_k^{(i)} - z_{k-1})$. The first inequality is due to the the minimality of $z_k^{(i)}$, which gives

$$\langle \nabla V_{z_{k-1}}(z_k^{(i)}) + n\alpha_k\xi_k^{(i)}, u - z_k\rangle \ge 0 \quad \forall u \in \Delta,$$

the second inequality is due to the standard three point property of Bregman divergence, that is $\forall x, y \ge 0$

$$\langle -\nabla V_x(y), y - u\rangle = V_x(u) - V_y(u) - V_x(y),$$

and the last inequality just drops the term $-V_{z_k}(u^*)$, which is always negative. □

Also, we note that the mirror descent step, defined above in a variational way, can be explicitly written as

1. $z_k^{(i)} \leftarrow z_{k-1}$
2. $z_k^{(i)} \leftarrow z_k^{(i)} - n\alpha_k\xi_k^{(i)}/\|C_{:i}\|_\infty$
3. If $z_k^{(i)}[i] < 0, z_k^{(i)}[i] \leftarrow 0$; if $z_k^{(i)}[i] > 3/\|C_{:i}\|_\infty, z_k^{(i)}[i] \leftarrow 3/\|C_{:i}\|_\infty$.

This is invariant to the difference of packing and covering, and so it follows directly from Proposition 3.6 in [AZO15a]. It is fairly easy to derive, and so we omit the proof.

Gradient Descent Step

We now analyze the gradient descent step of Algorithm 1. In particular, from the explicit formulation of the mirror descent step, we have that $|z_k^{(i)}[i] - z_{k-1}[i]| \leq \frac{n\alpha_k|\xi_k^{(i)}|}{\|C_{:i}\|_\infty}$, which gives

$$|y_k^{(i)}[i] - x_k[i]| = \frac{1}{n\alpha_k L}|z_k^{(i)}[i] - z_{k-1}[i]| \leq \frac{|\xi_k^{(i)}|}{L\|C_{:i}\|_\infty}.$$

The gradient step we take is within the local region, and so Lemma 2.4.3 applies. We bound the improvement from the gradient descent step in the following lemma, which is symmetric[3] to Lemma 3.8 in [AZO15a].

Lemma 2.4.7. $g_\mu(x_k) - g_\mu(y_k^{(i)}) \geq \frac{1}{2}\langle \nabla g_\mu(x_k), x_k - y_k^{(i)}\rangle$

Proof. Since x_k and $y_k^{(i)}$ differ only at coordinate i, denote $\gamma = y_k^{(i)}[i] - x_k[i]$, we have

$$g_\mu(x_k) - g_\mu(y_k^{(i)}) = g_\mu(x_k) - g_\mu(x_k + \gamma\,\mathrm{e}_i) = \int_0^\gamma -\nabla_i g_\mu(x_k + \nu\,\mathrm{e}_i)d\nu.$$

Since γ satisfies $|\gamma| \leq \frac{|\xi_k^{(i)}|}{L\|C_{:i}\|_\infty} \leq \frac{1}{L\|C_{:i}\|_\infty}$, we can apply Lemma 2.4.3. There are two cases to consider.

If $\nabla_i g_\mu(x_k) \in (-1, 1)$, then we have $|\gamma| \leq \frac{|\xi_k^{(i)}|}{L\|C_{:i}\|_\infty} = \frac{|\nabla_i g_\mu(x_k)|}{L\|C_{:i}\|_\infty}$, and by Lemma 2.4.3 we have $-\nabla_i g_\mu(x_k + \nu\,\mathrm{e}_i) \geq -\nabla_i g_\mu(x_k) - L\|C_{:i}\|_\infty|\nu|$ in the above integration. Thus,

$$\begin{aligned}
g_\mu(x_k) - g_\mu(y_k^{(i)}) &\geq \int_0^\gamma -\nabla_i g_\mu(x_k + \nu\,\mathrm{e}_i)d\nu \\
&\geq \int_0^\gamma -\nabla_i g_\mu(x_k) - L\|C_{:i}\|_\infty|\nu|d\nu \\
&= -\nabla_i g_\mu(x_k)\gamma - \frac{L\|C_{:i}\|_\infty}{2}\gamma^2 \\
&\geq -\nabla_i g_\mu(x_k)\gamma - \frac{L\|C_{:i}\|_\infty}{2}|\gamma|\frac{|\nabla_i g_\mu(x_k)|}{L\|C_{:i}\|_\infty} \\
&= -\frac{1}{2}\langle \nabla_i g_\mu(x_k), \gamma\rangle = \frac{1}{2}\langle \nabla g_\mu(x_k), x_k - y_k^{(i)}\rangle.
\end{aligned}$$

If $\nabla_i g_\mu(x_k) \leq -1$, then again by Lemma 2.4.3 we have

$$-\nabla_i g_\mu(x_k + \nu\,\mathrm{e}_i) \geq -(1 - \frac{L\|C_{:i}\|_\infty}{2}|\nu|)\nabla_i g_\mu(x_k) \geq -\frac{1}{2}\nabla_i g_\mu(x_k).$$

[3]The symmetry is between Lemma 2.6 in [AZO15a] and Lemma 2.4.3, as the gradient descent improvement follows directly from the corresponding Lipschitz properties. The actual improvement guarantee is the same as Lemma 3.8 in [AZO15a].

Thus,

$$\begin{aligned} g_\mu(x_k) - g_\mu(y_k^{(i)}) &\geq \int_0^\gamma -\nabla_i g_\mu(x_k + \nu\, \mathrm{e}_i) d\nu \\ &\geq \int_0^\gamma -\frac{1}{2}\nabla_i g_\mu(x_k) d\nu = \frac{1}{2}\langle \nabla g_\mu(x_k), x_k - y_k^{(i)} \rangle. \end{aligned}$$

□

Coupling of Gradient and Mirror Descent

Here, we will analyze the coupling between the gradient descent and mirror descent steps. This and the next section will give a proof of Theorem 2.4.1.

As we take steps on random coordinates, we will write the full gradient as

$$\nabla g_\mu(x_k) = \mathbb{E}_i[n\nabla_i g_\mu(x_k)] = \mathbb{E}_i[n\eta_k^{(i)} + n\xi_k^{(i)}].$$

As discussed earlier, we have the small component $\xi_k^{(i)} \in (-1, 1)\, \mathrm{e}_i$ and the large component $\eta_k^{(i)} = \nabla_i g_\mu(x_k) - \xi_k^{(i)} \in (-\infty, 0]\, \mathrm{e}_i$. We put the gradient and mirror descent steps together, and we bound the gap to optimality at iteration k:

$$\begin{aligned} &\alpha_k(g_\mu(x_k) - g_\mu(u^*)) \\ \leq &\langle \alpha_k \nabla g_\mu(x_k), x_k - u^* \rangle \\ = &\langle \alpha_k \nabla g_\mu(x_k), x_k - z_{k-1} \rangle + \langle \alpha_k \nabla g_\mu(x_k), z_{k-1} - u^* \rangle \\ = &\langle \alpha_k \nabla g_\mu(x_k), x_k - z_{k-1} \rangle + \mathbb{E}_i[\langle n\alpha_k \eta_k^{(i)}, z_{k-1} - u^* \rangle + \langle n\alpha_k \xi_k^{(i)}, z_{k-1} - u^* \rangle] \\ = &\frac{1-\tau}{\tau}\alpha_k \langle \nabla g_\mu(x_k), y_{k-1} - x_k \rangle + \mathbb{E}_i[\langle n\alpha_k \eta_k^{(i)}, z_{k-1} - u^* \rangle] \\ &+ \mathbb{E}_i[\langle n\alpha_k \xi_k^{(i)}, z_{k-1} - u^* \rangle] \\ \leq &\frac{1-\tau}{\tau}\alpha_k (g_\mu(y_{k-1}) - g_\mu(x_k)) + \mathbb{E}_i[\langle n\alpha_k \eta_k^{(i)}, z_{k-1} - u^* \rangle] \\ &+ \mathbb{E}_i[n^2\alpha_k^2 L \langle \xi_k^{(i)}, x_k - y_k^{(i)} \rangle + V_{z_{k-1}}(u^*) - V_{z_k^{(i)}}(u^*)]. \end{aligned}$$

The first line is due to convexity. The next two lines just break and regroup the terms. The fourth line is due to $x_k = \tau z_{k-1} + (1-\tau)y_{k-1}$, so $\tau(x_k - z_{k-1}) = (1-\tau)(y_{k-1} - x_k)$. The last line is by Lemma 2.4.6.

We try to use the improvement from the gradient step given in Lemma 2.4.7 to cover the loss from $\eta_k^{(i)}$, and the regret from the mirror descent step:

$$\underbrace{\mathbb{E}_i[\langle n\alpha_k \eta_k^{(i)}, z_{k-1} - u^* \rangle]}_{\text{loss from } \eta_k^{(i)}} + \underbrace{\mathbb{E}_i[n^2\alpha_k^2 L \langle \xi_k^{(i)}, x_k - y_k^{(i)} \rangle]}_{\text{regret from mirror descent}}, \tag{2.7}$$

and we will use the fact $z_{k-1}, z_k^{(i)}, u^* \in \Delta$. Consider the following cases.

1. $\eta_k^{(i)}[i] = 0$: In this case, the loss term is 0. We only need to worry about the regret term, and by Lemma 2.4.7

$$n^2\alpha_k^2 L\langle \xi_k^{(i)}, x_k - y_k^{(i)}\rangle \le 2n^2\alpha_k^2 L(g_\mu(x_k) - g_\mu(y_k^{(i)})).$$

2. $\eta_k^{(i)}[i] < 0, z_k^{(i)}[i] < \frac{3}{\|C_{:i}\|_\infty}$: In this case, we increased the i-th variable in both the gradient and mirror descent step, and because $z_k^{(i)}[i]$ is inside Δ without any projection, we know the step length of gradient descent is exactly $y_k^{(i)}[i] - x_k[i] = \frac{1}{n\alpha_k L}\frac{n\alpha_k}{\|C_{:i}\|_\infty} = \frac{1}{L\|C_{:i}\|_\infty}$, together with $z_{k-1} \ge 0$, and $u_i^* \le \frac{3}{\|C_{:i}\|_\infty}$, we have

$$\langle n\alpha_k\eta_k^{(i)}, z_{k-1} - u^*\rangle \le \langle n\alpha_k\eta_k^{(i)}, -u^*\rangle \le -n\alpha_k\nabla_i g_\mu(x_k)\frac{3}{\|C_{:i}\|_\infty} = 3n\alpha_k L\langle \nabla g_\mu(x_k), x_k - y_k^{(i)}\rangle,$$

and

$$\begin{aligned}\langle n\alpha_k\eta_k^{(i)}, z_{k-1} - u^*\rangle + n^2\alpha_k^2 L\langle \xi_k^{(i)}, x_k - y_k^{(i)}\rangle \le& (3n\alpha_k L + n^2\alpha_k^2 L)\langle \nabla g_\mu(x_k), x_k - y_k^{(i)}\rangle \\ \le& (6n\alpha_k L + 2n^2\alpha_k^2 L)(g_\mu(x_k) - g_\mu(y_k^{(i)})).\end{aligned}$$

The last step is by Lemma 2.4.7.

3. $\eta_k^{(i)}[i] < 0, z_k^{(i)}[i] = \frac{3}{\|C_{:i}\|_\infty}$: In this case, as we know $u_i^* \le \frac{3}{\|C_{:i}\|_\infty}$, we have

$$\langle n\alpha_k\eta_k^{(i)}, z_{k-1} - u^*\rangle \le \langle n\alpha_k\eta_k^{(i)}, z_{k-1} - z_k^{(i)}\rangle = n^2\alpha_k^2 L\langle \eta_k^{(i)}, x_k - y_k^{(i)}\rangle,$$

and

$$\begin{aligned}\langle n\alpha_k\eta_k^{(i)}, z_{k-1} - u^*\rangle + n^2\alpha_k^2 L\langle \xi_k^{(i)}, x_k - y_k^{(i)}\rangle \le& 2n^2\alpha_k^2 L\langle \nabla g_\mu(x_k), x_k - y_k^{(i)}\rangle \\ \le& 4n^2\alpha_k^2 L(g_\mu(x_k) - g_\mu(y_k^{(i)})).\end{aligned}$$

Again, the last step is due to Lemma 2.4.7.

Since $n\alpha_k < 1$ for all k, we have in all above cases,

$$\mathbb{E}_i[\langle n\alpha_k\eta_k^{(i)}, z_{k-1} - u^*\rangle] + \mathbb{E}_i[n^2\alpha_k^2 L\langle \xi_k^{(i)}, x_k - y_k^{(i)}\rangle] \le \mathbb{E}_i[8n\alpha_k L(g_\mu(x_k) - g_\mu(y_k^{(i)}))].$$

Back to our earlier derivation, we have

$$\begin{aligned}\alpha_k(g_\mu(x_k) - g_\mu(u^*)) \le& \frac{1-\tau}{\tau}\alpha_k(g_\mu(y_{k-1}) - g_\mu(x_k)) + \mathbb{E}_i[\langle n\alpha_k\eta_k^{(i)}, z_{k-1} - u^*\rangle] \\ &+ \mathbb{E}_i[n^2\alpha_k^2 L\langle \xi_k^{(i)}, x_k - y_k^{(i)}\rangle + V_{z_{k-1}}(u^*) - V_{z_k^{(i)}}(u^*)] \\ \le& \frac{1-\tau}{\tau}\alpha_k(g_\mu(y_{k-1}) - g_\mu(x_k)) + \mathbb{E}_i[8n\alpha_k L(g_\mu(x_k) - g_\mu(y_k^{(i)})] \\ &+ \mathbb{E}_i[V_{z_{k-1}}(u^*) - V_{z_k^{(i)}}(u^*)].\end{aligned}$$

With our choice of $\tau = \frac{1}{8nL}, \alpha_k = \frac{1}{1-\tau}\alpha_{k-1}$, we have

$$-\alpha_k g_\mu(u^*) \leq 8nL\alpha_{k-1}g_\mu(y_{k-1}) - \mathbb{E}_i[8nL\alpha_k g_\mu(y_k^{(i)})] + \mathbb{E}_i[V_{z_{k-1}}(u^*) - V_{z_k^{(i)}}(u^*)].$$

Telescoping the above inequality [4] along $k = 1, \ldots, T$, we get

$$\mathbb{E}[8nL\alpha_T g_\mu(y_T)] \leq \sum_{k=1}^{T} \alpha_k g_\mu(u^*) + 8nL\alpha_0 g_\mu(y_0) + V_{z_0}(u^*),$$

and thus

$$\mathbb{E}[g_\mu(y_T)] \leq \frac{\sum_{k=1}^{T}\alpha_k}{8nL\alpha_T} g_\mu(u^*) + \frac{\alpha_0}{\alpha_T} g_\mu(y_0) + \frac{1}{8nL\alpha_T} V_{z_0}(u^*).$$

We have $\sum_{k=1}^{T}\alpha_k = \alpha_T \sum_{k=0}^{T-1}(1-\frac{1}{8nL})^k = 8nL\alpha_T(1-(1-\frac{1}{8nL})^T) \leq 8nL\alpha_T$, and by our choice of $T = \lceil 8nL\log(1/\epsilon)\rceil$, we also have

$$\frac{\alpha_0}{\alpha_T} = (1-\frac{1}{8nL})^T \leq \epsilon, \frac{1}{8nL\alpha_T} \leq \frac{\epsilon}{8nL\alpha_0} = \frac{\epsilon}{8},$$

and thus

$$\mathbb{E}_i[g_\mu(y_T)] \leq g_\mu(u^*) + \epsilon g_\mu(y_0) + \frac{\epsilon}{8} V_{z_0}(u^*).$$

Finding a Good Starting Point

Here, we will describe how to find a good starting point for the algorithm. This will permit us to establish the quality-of-approximation and running time guarantees of Theorem 2.4.1.

A good starting point $y_0 = x^{\text{start}}$ for Algorithm 1 is an initial condition x^{start} that is not too far away from the optimal in terms of the function value (i.e small $g_\mu(y_0)$), and not too far away from u^* in C-norm (i.e. small $V_{z_0}(u^*)$). For packing problems, starting with all the all-0's vector will work, but this will not work for covering problems. Instead, for covering problems, we will show now a good enough x^{start} can be obtained in $\tilde{O}(N)$.

To do so, recall that we can get a 2-approximation $x^{\#}$ to the original covering LP in time $\tilde{O}(N)$ using various nearly linear time covering solvers, e.g., those of [KY14; You14]. Without loss of generality, we can assume $x[i]^{\#} \in [0, \frac{2}{\|C_{:i}\|_\infty}]$, since we can use the diameter reduction process as specified in Lemma 2.3.1 to get a equivalent solution satisfying the conditions. Then, we have the following lemma.

Lemma 2.4.8. *Let $x^{\text{start}} = (1+\epsilon/2)x^{\#}$, we have $x^{\text{start}} \in \Delta$, $g_\mu(x^{\text{start}}) \leq 4\,\text{OPT}$, and $V_{x^{\text{start}}}(u^*) \leq 6\,\text{OPT}$*

[4] More accurately, the telescoping works on

$$-\alpha_k g_\mu(u^*) \leq 8nL\alpha_{k-1}\mathbb{E}_{I_{k-1}}[g_\mu(y_{k-1})] - \mathbb{E}_{I_k}[8nL\alpha_k g_\mu(y_k^{(i)})] + \mathbb{E}_{I_{k-1}}[V_{z_{k-1}}(u^*)] - \mathbb{E}_{I_k}[V_{z_k^{(i)}}(u^*)].$$

where I_k is all the random coordinate choices made through the first iteration till k-th iteration. The final expectation on $g_\mu(y_T)$ is over all the Trandom choices.

Proof. It is obvious that $x^{\text{start}} \in \Delta$. Thus,

$$\vec{1}^T x^{\text{start}} = (1+\epsilon/2)\vec{1}^T x^{\#} \leq (1+\epsilon/2)2\,\text{OPT} \leq 3\,\text{OPT}\,.$$

Furthermore, we have $Cx^{\text{start}} - \vec{1} \geq (1+\epsilon/2)Cx^{\#} - \vec{1} \geq \frac{\epsilon}{2}\vec{1}$, and so

$$g_\mu(x^{\text{start}}) = \mu \sum_j q_j(x^{\text{start}}) + \vec{1}^T x^{\text{start}} \leq \mu \sum_j \exp(-\frac{\epsilon/2}{\mu}) + 3\,\text{OPT} \leq \frac{\mu m}{(nm)^2} + 3\,\text{OPT} < 4\,\text{OPT}\,.$$

For the divergence, we have that

$$\begin{aligned}
V_{x^{\text{start}}}(u^*) =& \frac{1}{2}\sum_i \|C_{:i}\|_\infty (x_i^{\text{start}} - u_i^*)^2 \\
=& \frac{1}{2}\sum_i \|C_{:i}\|_\infty ((x_i^{\text{start}})^2 + (u^*)_i^2 - 2x_i^{\text{start}} u_i^*) \\
\leq& \frac{3}{2}\sum_i x_i^{\text{start}} + u_i^* \\
\leq& \frac{3}{2}(3\,\text{OPT} + \text{OPT}) \leq 6\,\text{OPT},
\end{aligned}$$

which proves the lemma. □

It is now clear that we have

$$\mathbb{E}_i[g_\mu(y_T)] \leq g_\mu(u^*) + \epsilon g_\mu(y_0) + \frac{\epsilon}{8} V_{z_0}(u^*) \leq (1+\epsilon)\,\text{OPT} + 4\epsilon\,\text{OPT} + \epsilon\,\text{OPT} = (1+6\epsilon)\,\text{OPT}\,.$$

Thus, we have the approximation guarantee in Theorem 2.4.1. The running time follows directly from Lemma 2.4.5 and $T = \tilde{O}(n/\epsilon)$.

2.5 Missing Proofs

Lemma 2.1.1. $\text{OPT} \in [1, n]$

Proof. By the assumption $\min_{i\in[n]} \|P_{:i}\|_\infty = 1$, we know at least one variable has all coefficients at most 1, so we can just set that variable to 1, which gives $\text{OPT} \geq 1$. On the other hand, since each variable has a coefficient of 1 in some constraint, no variable can be larger than 1, thus $\text{OPT} \leq n$. □

Lemma 2.1.3. *Setting the smoothing parameter* $\mu = \frac{\epsilon}{4\log(nm/\epsilon)}$*, we have*

1. $f_\mu(u^*) \leq -(1-\epsilon)\,\text{OPT}$.
2. $f_\mu(x) \geq -(1+\epsilon)\,\text{OPT}$ *for every* $x \geq 0$.

3. *Letting $x_0 \geq 0$ be such that $x_0[i] = \frac{1-\epsilon/2}{n\|P_{:i}\|_\infty}$ for each $i \in [n]$, we have $f_\mu(x_0) \leq -\frac{1-\epsilon}{n}$.*

4. *For any $x \geq 0$ satisfying $f_\mu(x) \leq 0$, we must have $Px \leq (1+\epsilon)\vec{1}$, and thus $\vec{1}^T x \leq (1+\epsilon)\,\mathrm{OPT}$.*

5. *If $x \geq 0$ satisfies $f_\mu(x) \leq -(1-O(\epsilon))\,\mathrm{OPT}$, then $\frac{1}{1+\epsilon}x$ is a $(1-O(\epsilon))$-approximation to the packing LP.*

6. *The gradient of $f_\mu(x)$ is*

$$\nabla f_\mu(x) = -\vec{1} + P^T\overrightarrow{p(x)} \quad \textit{where} \quad p_j(x) \stackrel{\text{def}}{=} \exp(\frac{1}{\mu}((Px)_j - 1),$$

and $\nabla_i f_\mu(x) = -1 + \sum_j P_{ji}p_j(x) \in [-1, \infty]$.

Proof. 1. Since $Px^* \leq \vec{1}$, and $u^* = (1-\epsilon/2)x^*$, we have $(Pu^*)_j - 1 \leq -\epsilon/2$ for all j. Then $p_j(u^*) \leq \exp(-\frac{1}{\mu}\frac{\epsilon}{2}) = (\frac{\epsilon}{mn})^2$, and $f_\mu(u^*) = -\vec{1}^T u^* + \mu\sum_{j=1}^m p_j(u^*) \leq -(1-\epsilon/2)\,\mathrm{OPT} + \mu m(\frac{\epsilon}{mn})^2 \leq -(1-\epsilon)\,\mathrm{OPT}$.

2. By contradiction, suppose $f_\mu(x) < -(1+\epsilon)\,\mathrm{OPT}$, since $f_\mu(x) > -\vec{1}^T x$, we must have $\vec{1}^T x > (1+\epsilon)\,\mathrm{OPT}$. Suppose $\vec{1}^T x = (1+v)\,\mathrm{OPT}$ for some $v > \epsilon$. There must exits a j, such that $(Px)_j > 1+v$. Then we have $p_j(x) > \exp(v/\mu) = ((\frac{mn}{\epsilon})^4)^{v/\epsilon}$, which implies

$$f_\mu(x) \geq -(1+v)\,\mathrm{OPT} + \mu p_j(x) \geq \frac{\epsilon}{4\log(mn/\epsilon)}((\frac{mn}{\epsilon})^4)^{v/\epsilon} - (1+v)\,\mathrm{OPT} > 0$$

since $\mathrm{OPT} \leq n$, and $v > \epsilon$. This gives a contradiction.

3. The x_0 we use satisfies $Px_0 - \vec{1} \leq -\epsilon/2 - \vec{1}$, thus

$$f_\mu(x_0) = \mu\sum_j p_j(x_0) - \vec{1}^T x_0 \leq \frac{\mu m}{(nm)^2} - \frac{1-\epsilon/2}{n} \leq -\frac{1-\epsilon}{n}$$

4. By contradiction, suppose there is some j such that $(Px)_j - 1 \geq \epsilon$. Let $v > \epsilon$ be the smallest v such that $Px \leq (1+v)\,\mathrm{OPT}$, and denote j the constraint that has $(Px)_j - 1 = v$. We must have $\vec{1}^T x \leq (1+v)\,\mathrm{OPT}$ by definition of OPT. Then

$$f_\mu(x) \geq \mu p_j(x) - (1+v)\,\mathrm{OPT} \geq \frac{\epsilon}{4\log(mn/\epsilon)}((\frac{mn}{\epsilon})^4)^{v/\epsilon} - (1+v)\,\mathrm{OPT} > 0,$$

which gives a contradiction.

5. By the above part, $f_\mu(x) \leq -(1-O(\epsilon))\,\mathrm{OPT} \leq 0$ suggests $\frac{x}{1+\epsilon}$ is feasible. Furthermore, $-\vec{1}^T x < f_\mu(x) \leq -(1-O(\epsilon))\,\mathrm{OPT}$ gives $\vec{1}^T x \geq (1-O(\epsilon))\,\mathrm{OPT}$, thus $\vec{1}^T\frac{x}{1+\epsilon} \geq (1-O(\epsilon))\,\mathrm{OPT}$ is approximately optimal.

6. This is by straightforward computation.

□

Lemma 2.1.4. $\mathrm{OPT} \in [1, m]$

Proof. By the assumption $\min_{j\in[m]} \|C_{j:}\|_\infty = 1$, we know at least one constraint has all coefficients at most 1, so to satisfy that constraint, we must have the sum of the variables to be at least 1. On the other hand, since each constraint has a variable with coefficient at least 1 in it, $x = \vec{1}$ clearly satisfies all constraints, so $\mathrm{OPT} \le m$. □

Lemma 2.1.6. *Setting the smoothing parameter $\mu = \frac{\epsilon}{4\log(nm/\epsilon)}$, we have*

1. $g_\mu(u^*) \le (1+\epsilon)\,\mathrm{OPT}$.
2. $g_\mu(x) \ge (1-\epsilon)\,\mathrm{OPT}$ *for any* $x \ge 0$.
3. *For any* $x \ge 0$ *satisfying* $g_\mu(x) \le 2\,\mathrm{OPT}$, *we must have* $Cx \ge (1-\epsilon)\vec{1}$.
4. *If* $x \ge 0$ *satisfies* $g_\mu(x) \le (1+O(\epsilon))\,\mathrm{OPT}$, *then* $\frac{1}{1-\epsilon}x$ *is a* $(1+O(\epsilon))$*-approximation to the covering LP.*
5. *The gradient of* $g_\mu(x)$ *is*

$$\nabla g_\mu(x) = \vec{1} - C^T\overrightarrow{q(x)} \quad \textit{where} \quad q_j(x) \stackrel{\text{def}}{=} \exp(\frac{1}{\mu}(1-(Cx)_j),$$

and $\nabla_i g_\mu(x) = 1 - \sum_j C_{ji}q_j(x) \in [-\infty, 1]$.

Proof. 1. Since $Cx^* \ge \vec{1}$, and $u^* = (1+\epsilon/2)x^*$, we have $(Cu^*)_j - 1 \ge \epsilon/2$ for all j. Then $q_j(u^*) \le \exp(-\frac{1}{\mu}\frac{\epsilon}{2}) = (\frac{\epsilon}{mn})^2$, and

$$g_\mu(u^*) = \vec{1}^T u^* + \mu\sum_{j=1}^m q_j(u^*) \le (1+\epsilon/2)\,\mathrm{OPT} + \mu m(\frac{\epsilon}{mn})^2 \le (1+\epsilon)\,\mathrm{OPT}\,.$$

2. By contradiction, suppose $g_\mu(x) < (1-\epsilon)\,\mathrm{OPT}$, since $g_\mu(x) < \mathrm{OPT} \le m$, we must have $q_j(x) < m/\mu$ for any j, which implies $(Cx)_j \ge 1-\epsilon$. By definition of OPT, we have $\vec{1}^T x \ge (1-\epsilon)\,\mathrm{OPT}$, since $Cx \ge (1-\epsilon)\vec{1}$. This gives a contradiction as $g_\mu(x) > \vec{1}^T x \ge (1-\epsilon)\,\mathrm{OPT}$.

3. By contradiction, suppose there is some j such that $(Cx)_j - 1 \le -\epsilon$, then as in the last part, we have $\mu q_j(x) \ge \mu(\frac{mn}{\epsilon})^4 > 2\,\mathrm{OPT}$, contradicting $g_\mu(x) \le 2\,\mathrm{OPT}$.

4. For any x satisfying $g_\mu(x) \le (1+O(\epsilon))\,\mathrm{OPT} \le 2\,\mathrm{OPT}$, by last part we know $Cx \ge (1-\epsilon)\vec{1}$, so $C(\frac{1}{1-\epsilon}x) \ge \vec{1}$. We also have $\vec{1}^T(\frac{1}{1-\epsilon}x) = \frac{1}{1-\epsilon}\vec{1}^T x < \frac{1}{1-\epsilon}g_\mu(x) \le (1+O(\epsilon))\,\mathrm{OPT}$.

5. This is by straightforward computation.

□

Lemma 2.4.4. *We have $x_k, y_k, z_k \in \Delta$ for all $k = 0, 1, \ldots, T$.*

Proof. At the start $x_0 = y_0 = z_0 = x^{\text{start}} \in \Delta$ by assumption. z_k is always in Δ as we take the projection in the mirror descent step. If we can further show $y_k \in \Delta$ for all k, we are done, since x_k is a convex combination of y_{k-1}, z_{k-1}. To show $y_k \in \Delta$, we write y_k as a convex combination of $z_0, \ldots, z_k$, $y_k = \sum_{l=0}^{k} c_k^l z_l$. At $k = 0$, we have $y_0 = z_0$, and at $k = 1$, $y_1 = x_1 + \frac{1}{n\alpha_1 L}(z_1 - z_0) = \frac{1}{n\alpha_1 L} z_1 + (1 - \frac{1}{n\alpha_1 L}) z_0$, as $x_1 = y_0 = z_0$. For $k \geq 2$, we can verify

$$c_k^l = \begin{cases} (1-\tau) c_{k-1}^l & l = 0, \ldots, k-2 \\ (\frac{1}{n\alpha_{k-1}L} - \frac{1}{n\alpha_k L}) + \tau(1 - \frac{1}{n\alpha_{k-1}L}) & l = k-1 \\ \frac{1}{n\alpha_k L} & l = k \end{cases}$$

since

$$\begin{aligned} y_k &= x_k + \frac{1}{n\alpha_k L}(z_k - z_{k-1}) \\ &= \tau z_{k-1} + (1-\tau) y_{k-1} + \frac{1}{n\alpha_k L}(z_k - z_{k-1}) \\ &= \tau z_{k-1} + (1-\tau)(\sum_{l=0}^{k-2} c_{k-1}^l z_l + \frac{1}{n\alpha_{k-1}L} z_{k-1}) + \frac{1}{n\alpha_k L}(z_k - z_{k-1}) \\ &= (\sum_{l=0}^{k-2} (1-\tau) c_{k-1}^l z_l) + ((\frac{1}{n\alpha_{k-1}L} - \frac{1}{n\alpha_k L}) + \tau(1 - \frac{1}{n\alpha_{k-1}L})) z_{k-1} + \frac{1}{n\alpha_k L} z_k \end{aligned}$$

As $\alpha_k \geq \alpha_{k-1}$, and $\alpha_0 = \frac{1}{nL}$, we have $c_k^l \geq 0$ for all l, k, and it is easy to check the coefficients sum to 1 for each k. □

Lemma 2.4.5. *Each iteration can be implemented in expected $O(N/n)$ time.*

Proof. We show how to implement a iteration conditioned on i in time $O(\|C_{:i}\|_0)$, where $\|C_{:i}\|_0$ is the number of non-zeros in column i, thus give a expected running time of $O(N/n)$ for each iteration. We maintain the following quantities

$$z_k \in \mathbb{R}^n_{\geq 0}, cz_k \in \mathbb{R}^m_{\geq 0}, y'_k \in \mathbb{R}^n, cy_k \in \mathbb{R}^m, B_{k,1}, B_{k,2} \in \mathbb{R}_+$$

with the following invariants always satisfied throughout the algorithm

$$Cz_k = cz_k \tag{2.8}$$

$$y_k = B_{k,1} z_k + B_{k,2} y'_k, \quad Cy_k = B_{k,1} cz_k + B_{k,2} cy_k \tag{2.9}$$

When $k = 0$, we let $cz_k = Cz_0, y'_k = y_0, cy_k = Cy_0, B_{k,1} = 0, B_{k,2} = 1$, and it is clear all the invariants are satisfied. For $k = 1, 2, \ldots, T$:

- The step $x_k = \tau z_{k-1} + (1-\tau)y_{k-1}$ does not need to be implemented.
- Computation of $\nabla_i f(x_k)$ requires the value of $q_j(x_k) = \exp(\frac{1}{\mu}(1-(Cx_k)_j))$ for each j such that $C_{ji} \neq 0$, and we can get the value

  $$(Cx_k)_j = \tau(Cz_{k-1})_j + (1-\tau)(Cy_{k-1})_j = (\tau + (1-\tau)B_{k-1,1})(cz_{k-1})_j + (1-\tau)B_{k-1,2}cy_{k-1,j}$$

  for each such j. This can be computed in $O(1)$ time for each j, and $O(\|C_{:i}\|_0)$ time in total.
- The mirror descent step $z_k^{(i)} \stackrel{\text{def}}{=} \operatorname{argmin}_{z\in\Delta}\{V_{z_{k-1}}(z) + \langle z, n\alpha_k \xi_k^{(i)}\rangle\}$ is simply $z_k = z_k + \delta\,\mathrm{e}_i$ where $\delta \in \mathbb{R}$ can be computed in $O(1)$ time. $z_k = z_{k-1} + \delta\,\mathrm{e}_i$ yields $y_k = \tau z_{k-1} + (1-\tau)y_{k-1} + \frac{\delta}{n\alpha_k L}\,\mathrm{e}_i$ by the gradient descent step. Therefore, we can update the values accordingly

  $$z_k \leftarrow z_{k-1} + \delta\,\mathrm{e}_i, \quad cz_k \leftarrow cz_{k-1} + \delta C_{:i}$$

  and

  $$\begin{aligned} &B_{k,1} \leftarrow \tau + (1-\tau)B_{k-1,1} && B_{k,2} \leftarrow (1-\tau)B_{k-1,2} \\ &y'_k \leftarrow y'_{k-1} + \delta(-\tfrac{B_{k,1}}{B_{k,2}} + \tfrac{1}{n\alpha_k L}\tfrac{1}{B_{k,2}})\,\mathrm{e}_i && cy_k \leftarrow cy_{k-1} + \delta(-\tfrac{B_{k,1}}{B_{k,2}} + \tfrac{1}{n\alpha_k L}\tfrac{1}{B_{k,2}})C_{:i} \end{aligned}$$

  We can verify that after the updates, the invariants still hold

  $$\begin{aligned} y_k =& B_{k,1}z_k + B_{k,2}y'_k = B_{k,1}(z_{k-1} + \delta\,\mathrm{e}_i) + B_{k,2}(y'_{k-1} + \delta(-\frac{B_{k,1}}{B_{k,2}} + \frac{1}{n\alpha_k L}\frac{1}{B_{k,2}})\,\mathrm{e}_i) \\ =& B_{k,1}z_{k-1} + B_{k,2}(y'_{k-1} + \delta(\frac{1}{n\alpha_k L}\frac{1}{B_{k,2}})\,\mathrm{e}_i) \\ =& B_{k,1}z_{k-1} + B_{k,2}y'_{k-1} + \frac{\delta}{n\alpha_k L}\,\mathrm{e}_i \\ =& (\tau + (1-\tau)B_{k-1,1})z_{k-1} + ((1-\tau)B_{k-1,2})y'_{k-1} + + \frac{\delta}{n\alpha_k L}\,\mathrm{e}_i \\ =& \tau z_{k-1} + (1-\tau)y_{k-1} + + \frac{\delta}{n\alpha_k L}\,\mathrm{e}_i \end{aligned}$$

  It is also straightforward to verify $Cy_k = B_{k,1}cz_k + B_{k,2}cy_k$ equals $Cy_k = \tau Cz_{k-1} + (1-\tau)Cy_{k-1} + + \frac{\delta}{n\alpha_k L}C\,\mathrm{e}_i$. The updates are dominated by the updates on cz_k and cy_k, which take $O(\|C_{:i}\|_0)$ time.

□

Chapter 3

Parallel Algorithms for Packing LPs and Covering LPs

In this chapter we present the parallel algorithms for packing LPs and covering LPs. At a high level, we still use the optimization approach as our sequential solver from previous chapter, where we optimize the smooth function $f_\mu(u)$ or $g_\mu(u)$. However, in our parallel algorithms, as we are updating multiple variables simultaneously, we face a different issue regarding the smoothness of the function.

3.1 Technical Overview

At the core of most gradient based optimization framework is the smoothness issue, where the update step is bounded by the local region where the gradients don't change too wildly. In parallel algorithms where we move all variables simultaneously, we have to take interference into consideration, where by interference we mean that variable i's gradient will be affected by the update of other variables $j \neq i$. In particular, the bottleneck of the convergence in [AZO15b], as we see it, is that the step size of the update is too small due to interference. In a typical iteration, they aim to move all variables simultaneously proportional to the respective gradients, without changing the gradient of any variable by too much. A natural obstacle arises when the gradients of the variables are not on the same scale. In such case, even when variable i has a large gradient, we are constrained from moving i by a large step to harness the improvement, due to the interference of i with some other variable j, which has a tiny gradient.

We tackle this bottleneck by reducing interference with dynamically-bucketed selective coordinate descent (DB-SCD). In particular, as specified in Section 3.2, in each iteration, we group all the variables according to the magnitudes of their gradients such that variables in the same group all have approximately the same magnitudes up to some constant factor. If we only update variables of one randomly chosen group, we don't suffer from the ratio of the largest gradients and the smallest gradients of the variables we move. We further show that we only need $\log \frac{1}{\epsilon}$ groups, so each iteration we update a large fraction of the variables on expectation.

For the $\tilde{O}(\frac{1}{\epsilon^2})$ time algorithm, we use a setup that multiplicative update the variables (comparing to the addictive updates in the sequential solver), and get a explicit algorithm for packing LPs. The algorithm for covering LPs is a bit different, as it first construct its dual packing LP, run the algorithm on the packing LP, and construct a solution for the original covering LP using the gradients from all the iterations.

For the accelerated $\tilde{O}(\frac{d}{\epsilon})$ time algorithm, we use the same additive update setup as in our sequential solver (including discrete lifting for covering LPs), and the techniques in DB-SCD. We lose a factor of d (i.e., the max number of non-zeros per row in A) in the running time bound, since to stay in local smooth region, we need to bound our update's impact on the gradient, which depends on the exponential penalties of the constraints. As we are updating variables additively, we need to scale back the step size by $\frac{1}{d}$ since a constraint may have d variables involved in the update.

3.2 $\tilde{O}(\frac{1}{\epsilon^2})$ Parallel Solver for Packing LPs

In this section we present the randomized $\tilde{O}(1/\epsilon^2)$ time parallel algorithm for packing and covering. We will start with our packing algorithm and main results, followed by the analysis and proofs. Then we show the algorithm and analysis for the dual covering problem, using largely the primal packing solver.

Algorithm and Main Theorems of Packing Solver

Our algorithm to approximately solve packing LPs is specified in Algorithm 2. Recall to find a $(1-\epsilon)$-approximation of a packing LP, we will approximately minimize the following convex function over the region $x \geq \vec{0}$

$$f_\mu(x) = -\vec{1}^T x + \mu \sum_{j=1}^{m} \exp(\frac{1}{\mu}((Px)_j - 1)),$$

In each iteration k, for each variable $x_k[i]$, we break the gradients $\nabla_i f_\mu(x_k)$ into small, medium, and large components, let

$$\zeta_k[i] = \begin{cases} \nabla_i f_\mu(x_k) & \nabla_i f_\mu(x_k) \in [-\epsilon, \epsilon] \\ 0 & \text{otherwise} \end{cases}$$

$$\xi_k[i] = \begin{cases} 0 & \nabla_i f_\mu(x_k) \in [-\epsilon, \epsilon] \\ \nabla_i f_\mu(x_k) & \nabla_i f_\mu(x_k) \in [-1, 1] \backslash [-\epsilon, \epsilon] \\ 1 & \nabla_i f_\mu(x_k) > 1 \end{cases}$$

$$\eta_k[i] = \begin{cases} \nabla_i f_\mu(x_k) - 1 & \nabla_i f_\mu(x_k) > 1 \\ 0 & \text{otherwise} \end{cases}$$

Algorithm 2 $\tilde{O}(\frac{1}{\epsilon^2})$ time parallel packing LP solver

Input: $P \in \mathbb{R}^{m\times n}_{\geq 0}$, $f_\mu, \epsilon \in (0, 1/2]$ **Output**: $x \in \mathbb{R}^n_{\geq 0}$

$\mu \leftarrow \frac{\epsilon}{4\log(nm/\epsilon)}, \alpha \leftarrow \frac{\mu}{20}$
$T \leftarrow \frac{\lceil 10\log(1/\epsilon)\log(2n)\rceil}{\alpha\epsilon} = \tilde{O}(\frac{1}{\epsilon^2}), w \leftarrow \lceil\log(\frac{1}{\epsilon})\rceil$
$x_0 \leftarrow \frac{1-\epsilon/2}{n\|P_{:i}\|_\infty}$
for $k = 0$ to $T-1$ **do**
    Select $t \in \{0, \ldots, w-1\}$ uniformly at random
    for $i = 1$ to n **do**
    ▷ Gradient truncation:
        Compute $\nabla_i f_\mu(x_k)$, and get $\xi_k^{(t)}[i]$ as defined in (3.2)
    ▷ Update step:
        $x_{k+1}[i] \leftarrow x_{k+1}^{(t)}[i] \stackrel{\text{def}}{=} x_k[i]\exp(-\alpha\xi_k^{(t)}[i])$
    end for
end for
return x_T.

We have

$$\nabla f_\mu(x_k) = \zeta_k + \xi_k + \eta_k \tag{3.1}$$

As we will see in Lemma 3.2.3, if the gradients are all within a factor of 2 from each other, we can take a multiplicative step with step size $\alpha = \Theta(\mu) = \tilde{O}(\epsilon)$. To exploit that, we will further partition the variables into groups, such that for variables in the same group, the absolute values of their truncated gradients will be within a factor 2 of each other. For $t \in \{0, \ldots, \lceil\log(\frac{1}{\epsilon}) - 1\rceil\}$, let

$$\xi_k^{(t)}[i] = \begin{cases} \xi_k[i] & \xi_k[i] \in (\epsilon 2^t, \epsilon 2^{t+1}] \\ & \cup[-\epsilon 2^{t+1}, -\epsilon 2^t) \\ 0 & \text{otherwise} \end{cases} \qquad \eta_k^{(t)}[i] = \begin{cases} \eta_k[i] & t = \lceil\log(\frac{1}{\epsilon})\rceil - 1 \\ 0 & \text{otherwise} \end{cases} \tag{3.2}$$

In each iteration, we will pick a group t uniformly at random, and update all variables using $\xi_k^{(t)}$.

Our main result is summarized in the following theorems.

Theorem 3.2.1. *Algorithm 2 outputs x_T satisfying $\mathbb{E}[f_\mu(x_T)] \leq -(1-5\epsilon)\,\mathrm{OPT}$, and the algorithm can be implemented with $\tilde{O}(\frac{1}{\epsilon^2})$ iterations with total work $\tilde{O}(N/\epsilon^2)$, where N is the total number of non-zeros in P.*

Given Theorem 3.2.1, a standard application of Markov bound, together with Lemma 2.1.3(5), gives the following corollary.

Corollary 3.2.2. *There is an algorithm that, with probability at least $9/10$, computes a $(1 - O(\epsilon))$-approximation to the fractional packing problem and has $\tilde{O}(N/\epsilon^2)$ expected total work, and implementable in $\tilde{O}(1/\epsilon^2)$ distributed iterations.*

Analysis of Algorithm 2 for Packing LPs

For preliminaries, see Section 2.1, especially Lemma 2.1.3.

Same as in [AZO15b], we interpret our update step $x_{k+1}[i] \leftarrow x_{k+1}^{(t)}[i] \stackrel{\text{def}}{=} x_k[i]\exp(-\alpha\xi_k^{(t)}[i])$ in Algorithm 2 as both a mirror descent step, as well as a gradient descent step. We proceed with respective analysis for the two interpretations.

Gradient Descent Step

We will first analyze the update step in Algorithm 2 as a gradient descent step. As in most gradient descent analysis, we need to bound our step's impact on the gradients. We show $f_\mu(x)$ is *locally multiplicative Lipschitz continuous*, in a sense quantified by the following lemma. Note the result is a stronger version of Proposition 3.6 in [AZO15b], in the sense that the step size α is $1/\epsilon$ larger. This improvement is achieved due to the reduced interference from our selective updating.

Lemma 3.2.3. *Let $x_{k+1}^{(t)}[i] = x_k[i]\exp(-\alpha\xi_k^{(t)}[i])$ for any $t = 0, \ldots, w-1$ as in Algorithm 2. Let $B_t = \{i|\xi_k^{(t)}[i] > 0\}$ be the set of variables we update. If $Px_k \leq (1+\epsilon)\vec{1}$, then for any $x = \tau x_k + (1-\tau)x_{k+1}^{(t)}$ where $\tau \in [0,1]$, we have $\forall i \in B_t, \nabla_i f_\mu(x)$ is between $\frac{1}{2}\nabla_i f_\mu(x_k)$ and $\frac{3}{2}\nabla_i f_\mu(x_k)$.*

Proof. Because for all $i \in B_t$, $\xi_k^{(t)}[i] \in (\epsilon 2^t, \epsilon 2^{t+1}] \cup [-\epsilon 2^{t+1}, -\epsilon 2^t)$, each variable changes multiplicatively by at most $\exp(\pm\alpha\epsilon 2^{t+1})$, and since $\alpha\epsilon 2^{t+1} \leq 1/4$, we must have for all i,

$$x[i] \in x_k[i] \cdot [1 - \frac{8}{3}\alpha\epsilon 2^t, 1 + \frac{8}{3}\alpha\epsilon 2^t] \tag{3.3}$$

Now we look at the impact of the step on the exponential penalties

$$p_j(x) = \exp(\frac{1}{\mu}((Px)_j - 1))$$

Due to (3.3), and $(Px_k)_j \leq (1+\epsilon)$ for any j, we have

$$|(Px)_j - (Px_k)_j| \leq \frac{8}{3}\alpha\epsilon 2^t (Px_k)_j \leq \frac{10}{3}\alpha\epsilon 2^t$$

Then by our choice of α, we have

$$p_j(x) \geq p_j(x_k)\exp(-\frac{10\alpha\epsilon 2^t}{3\mu}) = p_j(x_k)\exp(-\frac{\epsilon 2^t}{6})$$

Since $\epsilon 2^t \leq 1$, we have $\exp(-\frac{\epsilon 2^t}{6}) \geq (1 - \frac{\epsilon 2^t}{4})$. By a similar calculation for the upper bound, we have

$$p_j(x) \in p_j(x_k) \cdot [1 - \frac{\epsilon 2^t}{4}, 1 + \frac{\epsilon 2^t}{4}] \tag{3.4}$$

For any $i \in B_t$, if $\xi_k^{(t)}[i] \in (\epsilon 2^t, \epsilon 2^{t+1}]$, we have

$$\begin{aligned}\nabla_i f_\mu(x) &= (P^T p(x))_i - 1 \\ &> (P^T p(x_k))(1 - \frac{\epsilon 2^t}{4}) - 1 \\ &= (\nabla_i f_\mu(x_k) + 1)(1 - \frac{\epsilon 2^t}{4}) - 1 \\ &\geq \frac{\nabla_i f_\mu(x_k)}{2}\end{aligned}$$

where the last step is due to $\epsilon 2^t \leq 1$ and $\nabla_i f_\mu(x_k) \geq \xi_k^{(t)}[i] > \epsilon 2^t$. By similar calculation, we get $\nabla_i f_\mu(x) \leq \frac{3}{2}\nabla_i f_\mu(x_k)$. The same holds for the case $\xi_k^{(t)}[i] \in [-\epsilon 2^{t+1}, -\epsilon 2^t)$. □

We will see in Claim 3.2.5, the condition of $Px_k \leq (1+\epsilon)\vec{1}$ holds for all $k = 0, \ldots, T$. Once we established smoothness of the gradients within the range of our update step, we can lower bound the improvement we make. In particular, the term $\langle \alpha \eta_k^{(t)}, x_k - u \rangle$ is the loss incurred from the truncation, as our update step doesn't act on the truncated part, but it shows up when we use convexity to bound the gap to optimality.

Lemma 3.2.4. *For all $t = 0, \ldots, w-1$, any $u \geq 0$*

$$\langle \alpha \eta_k^{(t)}, x_k - u \rangle \leq 4(f_\mu(x_k) - f_\mu(x_{k+1}^{(t)}))$$

Proof.

$$\begin{aligned}f_\mu(x_k) - f_\mu(x_{k+1}^{(t)}) &= \int_0^1 \langle \nabla f_\mu(x_{k+1}^{(t)} + \tau(x_k - x_{k+1}^{(t)})), x_k - x_{k+1}^{(t)} \rangle d\tau \\ &= \sum_{i \in B_t} \int_0^1 \nabla_i f_\mu(x_{k+1}^{(t)} + \tau(x_k - x_{k+1}^{(t)})) d\tau \times (x_k[i] - x_{k+1}^{(t)}[i])\end{aligned}$$

where the last equality is because $x_k[i] - x_{k+1}^{(t)}[i] = 0$ for $i \notin B_t$. By Lemma 3.2.3, we have $\nabla_i f_\mu(x_{k+1}^{(t)} + \tau(x_k - x_{k+1}^{(t)}))$ has the same sign as $\nabla_i f_\mu(x_k)$ for all $\tau \in [0,1]$. Furthermore, by our update rule, $x_k[i] - x_{k+1}^{(t)}[i]$ also has the same sign as $\nabla_i f_\mu(x_k)$, so we have $f_\mu(x_k) - f_\mu(x_{k+1}^{(t)}) \geq 0$ for all t. If $t < w - 1$, we know $\eta_k^{(t)} = \vec{0}$, thus

$$\langle \alpha \eta_k^{(t)}, x_k - u \rangle = 0 \leq 4(f_\mu(x_k) - f_\mu(x_{k+1}^{(t)}))$$

When $t = w - 1$, let $B = \{i | \nabla_i f_\mu(x_k) > 1\} \supseteq B_t$ be the set of variables with nonzero $\eta_k^{(t)}[i]$, we know for $i \in B$, $\xi_k^{(t)}[i] = 1$, so $x_{k+1}^{(t)}[i] = x_k[i] \exp(-\alpha)$, and

$$\begin{aligned} f_\mu(x_k) - f_\mu(x_{k+1}^{(t)}) &= \sum_{i \in B_t} \int_0^1 \nabla_i f_\mu(x_{k+1}^{(t)} + \tau(x_k - x_{k+1}^{(t)}))d\tau \times (x_k[i] - x_{k+1}^{(t)}[i]) \\ &\geq \sum_{i \in B} \int_0^1 \nabla_i f_\mu(x_{k+1}^{(t)} + \tau(x_k - x_{k+1}^{(t)}))d\tau \times (x_k[i] - x_{k+1}^{(t)}[i]) \\ &\geq \sum_{i \in B} \frac{1}{2} \nabla_i f_\mu(x_k) \times x_k[i](1 - \exp(-\alpha)) \\ &\geq \sum_{i \in B} \frac{\alpha}{4} \nabla_i f_\mu(x_k) x_k[i] \end{aligned}$$

The first inequality is due to $B_t \subseteq B$, and every i has non-negative contribution to the sum. The second inequality is from Lemma 3.2.3, and the last inequality is because $(1 - \exp(-\alpha)) > \alpha/2$ when $\alpha < 1/10$. Then we have

$$\langle \alpha \eta_k^{(t)}, x_k - u \rangle \leq \sum_{i \in B} \alpha \nabla_i f_\mu(x_k) x_k[i] \leq 4(f_\mu(x_k) - f_\mu(x_{k+1}^{(t)}))$$

where the first inequality is because $\nabla_i f_\mu(x_k) > \eta_k^{(t)} \geq 0$ in this case, and $u \geq 0$. □

We see $f_\mu(x_k) - f_\mu(x_{k+1}^{(t)}) \geq 0$ for any $t = 0, \ldots, w - 1$, we have

Claim 3.2.5. *$f_\mu(x_k)$ is non-increasing with k. By part $(3), (4)$ of Lemma 2.1.3, $Px_k \leq (1 + \epsilon)\vec{1}$, and $\vec{1}^T x_k \leq (1 + \epsilon)\,\mathrm{OPT}$ for all k.*

Mirror Descent Step

We now interpret the update step as a mirror desent step. We use the same proximal setup as in [AZO15b]. The distance generating function will be the generalized entropy function, where

$$w(x) \stackrel{\text{def}}{=} \sum_{i \in [n]} x[i] \log(x[i]) - x[i]$$

and the corresponding Bregman divergence function will be

$$V_x(y) = \sum_{i \in [n]} (y[i] \log \frac{y[i]}{x[i]} + x[i] - y[i])$$

This is the standard proximal setup when one works with L_1-norm with the simplex as the feasible region. In our case, since the feasible region is $x \geq 0$, we don't have the standard strongly convexity of the Bregman divergence, but one can verify

$$V_x(y) = \sum_{i \in [n]} (y[i] \log \frac{y[i]}{x[i]} + x[i] - y[i]) \geq \sum_{i \in [n]} \frac{(x[i] - y[i])^2}{2 \max\{x[i], y[i]\}} \tag{3.5}$$

To interpret the update step as a mirror descent step. The following claim is the same as Claim 3.7 in [AZO15b] applied to different vectors.

Claim 3.2.6. *For all $t = 0, \ldots, w-1$, we have*

$$x_{k+1}^{(t)} = \operatorname*{argmin}_{z \geq 0}\{V_{x_k}(z) + \langle z - x_k, \alpha\xi_k^{(t)}\rangle\}$$

Proof. Since the function $V_{x^{(k)}}(z)$, the dot product and the constraint $z \geq 0$ are all coordinate-wise separable, we look at each coordinate independently. Thus we only need to check

$$x_{k+1}^{(t)}[i] = \operatorname*{argmin}_{z[i] \geq 0}\{(z[i]\log\frac{z[i]}{x_k[i]} + x_k[i] - z[i]) + \alpha\xi_k^{(t)}[i](z[i] - x_k[i])\}$$

This univariate function being optimized is convex and has a unique minimizer. We find it by taking the derivative to get

$$\log\frac{z[i]}{x_k[i]} + \alpha\xi_k^{(t)}[i] = 0$$

, which gives $x_{k+1}^{(t)}[i] \stackrel{\text{def}}{=} z[i] = x_k[i]\exp(-\alpha\xi_k^{(t)}[i])$. □

Once we see the update step is indeed a mirror descent step, we can derive the following result from the textbook mirror descent analysis (or, e.g., Lemma 3.3 in [AZO15b]).

Lemma 3.2.7. *For all $t = 0, \ldots, w-1$, we have for any $u \geq 0$*

$$\langle\alpha\xi_k^{(t)}, x_k - u\rangle \leq \alpha^2\,\mathrm{OPT} + V_{x_k}(u) - V_{x_{k+1}^{(t)}}(u)$$

Proof. The lemma follows from the following chain of equalities and inequalities.

$$\begin{aligned}
\langle\alpha\xi_k^{(t)}, x_k - u\rangle &= \langle\alpha\xi_k^{(t)}, x_k - x_{k+1}^{(t)}\rangle + \langle\alpha\xi_k^{(t)}, x_{k+1}^{(t)} - u\rangle \\
&\leq \langle\alpha\xi_k^{(t)}, x_k - x_{k+1}^{(t)}\rangle + \langle-\nabla V_{x_k}(x_{k+1}^{(t)}), x_{k+1}^{(t)} - u\rangle \\
&\leq \langle\alpha\xi_k^{(t)}, x_k - x_{k+1}^{(t)}\rangle + V_{x_k}(u) - V_{x_{k+1}^{(t)}}(u) - V_{x_k}(x_{k+1}^{(t)}) \\
&\leq \sum_{i\in[n]}\left(\alpha\xi_k^{(t)}[i](x_k[i] - x_{k+1}^{(t)}[i]) - \frac{(x_k[i] - x_{k+1}^{(t)}[i])^2}{2\max\{x_k[i], x_{k+1}^{(t)}[i]\}}\right) + V_{x_k}(u) - V_{x_{k+1}^{(t)}}(u) \\
&\leq \sum_{i\in[n]}\frac{(\alpha\xi_k^{(t)}[i])^2\max\{x_k[i], x_{k+1}^{(t)}[i]\}}{2} + V_{x_k}(u) - V_{x_{k+1}^{(t)}}(u) \\
&\leq \frac{2}{3}\alpha^2\vec{1}^T x_k + V_{x_k}(u) - V_{x_{k+1}^{(t)}}(u) \\
&\leq \alpha^2\,\mathrm{OPT} + V_{x_k}(u) - V_{x_{k+1}^{(t)}}(u)
\end{aligned}$$

The first equality follows by adding and subtracting $x_{k+1}^{(t)}$. The first inequality is due to the the minimality of $x_{k+1}^{(t)}$, which gives

$$\langle \nabla V_{x_k}(x_{k+1}^{(t)}) + \alpha \xi_k^{(t)}, u - x_{k+1}^{(t)} \rangle \geq 0 \quad \forall u \geq 0,$$

the second inequality is due to the standard three point property of Bregman divergence, that is $\forall x, y \geq 0$

$$\langle -\nabla V_x(y), y - u \rangle = V_x(u) - V_y(u) - V_x(y),$$

the third inequality is from (3.5), the fourth inequality follows from $2ab - a^2 \leq b^2$, the next inequality is due to $x_{k+1}^{(t)}[i] \leq x_k[i](1+\epsilon)$, and $\xi_k^{(t)}[i] \leq 1$. The last inequality is by Claim 3.2.5, $\vec{1}^T x_k \leq (1+\epsilon)\,\mathrm{OPT}$. □

Coupling of Gradient and Mirror Descent

In this section we show convergence using the results we derived by analyzing the update step as both a gradient descent step and a mirror descent step. This section will give a proof of Theorem 3.2.1.

Recall we break the gradients into small, medium and large components. The proof follows a similar approach as Lemma 3.4 of [AZO15b], where we bound the three components respectively, and telescope along all iterations. Furthermore, we divide the medium and large components into $w = \log(\lceil \frac{1}{\epsilon} \rceil)$ groups.

$$\nabla f_\mu(x_k) = \zeta_k + \xi_k + \eta_k$$

and

$$\xi_k = w\mathbb{E}_t[\xi_k^{(t)}], \eta_k = w\mathbb{E}_t[\eta_k^{(t)}]$$

We bound the gap to optimality at iteration k:

$$\begin{aligned}
\alpha(f_\mu(x_k) - f_\mu(u^*)) \leq & \langle \alpha \nabla f_\mu(x_k), x_k - u^* \rangle \\
= & \alpha\langle \zeta_k, x_k - u^* \rangle + \alpha\langle \xi_k, x_k - u^* \rangle + \alpha\langle \eta_k, x_k - u^* \rangle \\
= & \alpha\langle \zeta_k, x_k - u^* \rangle + w\mathbb{E}_t[\langle \alpha\xi_k^{(t)}, x_k - u^* \rangle] + w\mathbb{E}_t[\langle \alpha\eta_k^{(t)}, x_k - u^* \rangle]
\end{aligned}$$

The first line is due to convexity. The next two lines just break and regroup the terms. Now we upperbound each of the three terms

Lemma 3.2.8. *1.* $\langle \zeta_k, x_k - u^* \rangle \leq 3\epsilon\,\mathrm{OPT}$.

2. $\forall t$, $\langle \alpha\xi_k^{(t)}, x_k - u^* \rangle \leq \alpha^2\,\mathrm{OPT} + V_{x_k}(u^*) - V_{x_{k+1}^{(t)}}(u^*)$.

3. $\forall t$, $\langle \alpha\eta_k^{(t)}, x_k - u^* \rangle \leq 4(f_\mu(x_k) - f_\mu(x_{k+1}^{(t)}))$.

Proof. 1. We know $|\zeta_k[i]| \leq \epsilon$ for all i, $\vec{1}^T x_k \leq (1+\epsilon)\,\mathrm{OPT}$ from Claim 3.2.5, and $\vec{1}^T u^* \leq \mathrm{OPT}$, thus

$$\langle \zeta_k, x_k - u^* \rangle \leq \epsilon(\vec{1}^T x_k + \vec{1}^T u^*) \leq 3\epsilon\,\mathrm{OPT}$$

2. This is just Lemma 3.2.7 applied to $u = u^*$.

3. This is just Lemma 3.2.4 applied to $u = u^*$.

□

Then we have

$$\begin{aligned}\alpha(f_\mu(x_k) - f_\mu(u^*)) \leq & \alpha\langle\zeta_k, x_k - u^*\rangle + w\mathbb{E}_t[\langle\alpha\xi_k^{(t)}, x_k - u^*\rangle] + w\mathbb{E}_t[\langle\alpha\eta_k^{(t)}, x_k - u^*\rangle] \\ \leq & 3\alpha\epsilon\,\mathrm{OPT} + w\mathbb{E}_t[\alpha^2\,\mathrm{OPT} + V_{x_k}(u^*) - V_{x_{k+1}^{(t)}}(u^*)] \\ & + 4wf_\mu(x_k) - 4w\mathbb{E}_t[f_\mu(x_{k+1}^{(t)})]\end{aligned}$$

The above inequality holds for x_k following any sequence of random choices (i.e. t's in the first $k-1$ iterations). Let I_k denote the random choices over the first k iterations, we take expectation of the above inequality to get

$$\begin{aligned}\alpha(\mathbb{E}_{I_k}[f_\mu(x_k)] - f_\mu(u^*)) \leq & 3\alpha\epsilon\,\mathrm{OPT} + \alpha^2 w\,\mathrm{OPT} + w\mathbb{E}_{I_k}[V_{x_k}(u^*)] - w\mathbb{E}_{I_{k+1}}[V_{x_{k+1}^{(t)}}(u^*)] \\ & + 4w\mathbb{E}_{I_k}[f_\mu(x_k)] - 4w\mathbb{E}_{I_{k+1}}[f_\mu(x_{k+1}^{(t)})] \qquad (3.6)\end{aligned}$$

Telescoping (3.6) for $k = 0, \ldots, T-1$, we get

$$\begin{aligned}& \alpha\sum_{k=0}^{T-1}(\mathbb{E}_{I_k}[f_\mu(x_k)] - f_\mu(u^*)) \\ \leq & 3T\alpha\epsilon\,\mathrm{OPT} + wT\alpha^2\,\mathrm{OPT} + wV_{x_0}(u^*) + 4wf_\mu(x_0) - 4w\mathbb{E}_{I_T}[f_\mu(x_T^{(t)})] \\ \leq & 3T\alpha\epsilon\,\mathrm{OPT} + wT\alpha^2\,\mathrm{OPT} + 2w\log 2n\,\mathrm{OPT} - 4w\mathbb{E}_{I_T}[f_\mu(x_T^{(t)})] \qquad (3.7)\end{aligned}$$

where the last inequality is due to $f_\mu(x_0) < 0$, and

Claim 3.2.9. $V_{x_0}(u^*) \leq 2\log 2n\,\mathrm{OPT}$

Proof.

$$\begin{aligned}V_{x_0}(u^*) = \sum_i u^*[i]\log\frac{u^*[i]}{x_0[i]} + x_0[i] - u^*[i] &\leq \sum_i u^*[i]\log\frac{u^*[i]}{x_0[i]} + x_0[i] \\ &\leq \sum_i u^*[i]\log\frac{1/\|P_{:i}\|_\infty}{(1-\epsilon/2)/(n\|P_{:i}\|_\infty)} + \frac{1-\epsilon/2}{n\|P_{:i}\|_\infty} \\ &\leq \vec{1}^T u^*\log(2n) + 1 \leq 2\log(2n)\,\mathrm{OPT}\end{aligned}$$

where we have used $u_i^* \leq \frac{1}{\|P_{:i}\|_\infty}$ in the second line, since $Pu^* \leq \vec{1}$. The third line is due to $\vec{1}^T u^* \leq \mathrm{OPT}$, and $\mathrm{OPT} \geq 1$. □

We prove that $E_{I_T}[f_\mu(x_T^{(t)})] \leq -(1-5\epsilon)\,\mathrm{OPT}$ (i.e. Theorem 3.2.1) by contradiction. If $E_{I_T}[f_\mu(x_T^{(t)})] > -(1-5\epsilon)\,\mathrm{OPT}$, we have $-4wE_{I_T}[f_\mu(x_T^{(t)})] \leq 4w\,\mathrm{OPT}$. Divide both sides of (3.7) by αT, we have

$$\begin{aligned}
&\frac{1}{T\alpha}\sum_{k=0}^{T-1}\alpha(\mathbb{E}_{I_k}[f_\mu(x_k)] - f_\mu(u^*)) \\
\leq&\frac{1}{T\alpha}(3T\alpha\epsilon\,\mathrm{OPT} + wT\alpha^2\,\mathrm{OPT} + 2w\log 2n\,\mathrm{OPT} - 4w\mathbb{E}_{I_T}[f_\mu(x_T^{(t)})]) \\
\leq&3\epsilon\,\mathrm{OPT} + w\alpha\,\mathrm{OPT} + \frac{2w\log 2n}{T\alpha}\,\mathrm{OPT} + \frac{4w}{T\alpha}\,\mathrm{OPT}
\end{aligned}$$

Recall $\alpha = \mu/20 = \frac{\epsilon}{20\log\frac{mn}{\epsilon}}$, we have $w\alpha \leq \epsilon/20$. By our choice of $T = \frac{10w\log 2n}{\alpha\epsilon}$, we have $\frac{2w\log 2n}{T\alpha} < \epsilon$, and $\frac{4w}{T\alpha} < \epsilon$. Thus

$$\frac{1}{T}\sum_{k=0}^{T-1}(\mathbb{E}_{I_k}[f_\mu(x_k)] - f_\mu(u^*)) \leq 4\epsilon\,\mathrm{OPT}$$

From part (1) of Lemma 2.1.3, we know $f_\mu(u^*) \leq -(1-\epsilon)\,\mathrm{OPT}$, which suggests there exists a x_k, such that $f_\mu(x_k) \leq -1(1-5\epsilon)\,\mathrm{OPT}$. This gives a contradiction of $E_{I_T}[f_\mu(x_T^{(t)})] > -(1-5\epsilon)\,\mathrm{OPT}$ by Claim 3.2.5, as $f_\mu(x_k)$ is non-increasing, so is $\mathbb{E}_{I_k}[f_\mu(x_k)]$.

3.3 $\tilde{O}(\frac{1}{\epsilon^2})$ Parallel Solver for Covering LPs

A benefit of actually computing all the gradients in each iteration is that we can exploit the same primal-dual structure as in [AZO15b] to get a covering LP solver. Given a covering LP instance in the form of (2.2), we can construct its dual, which is a packing LP with $P = C^T$. If we run Algorithm 2 on the packing instance for T iterations, the average of the exponential penalties used in the computation of gradients

$$\bar{y} = \frac{1}{T}\sum_{k=0}^{T-1}\overrightarrow{p(x_k)} \geq 0 \tag{3.8}$$

will be with constant probability a $(1+\epsilon)$-approximation of the covering problem after some simple fixing step. Here $\overrightarrow{p(x_k)}$ is the vector with the exponential penalties of the packing constraints at iteration k.

The primal-dual property we will exploit is that the slackness of the i-th covering constraint with $\bar{y}$ is the average gradient of the i-th variable in the primal packing LP.

$$(C\bar{y})_i - 1 = \frac{1}{T}\sum_{k=0}^{T-1}(P^T\overrightarrow{p(x_k)})_i - 1 = \frac{1}{T}\sum_{k=0}^{T-1}\nabla_i f_\mu(x_k)$$

We use a similar approch as in [AZO15b; ZLO15]. We specify the fixing procedure in Algorithm 3 from [AZO15b], which explicitly makes all the dual constraints satisfied. We will run Algorithm 2

Algorithm 3 Fixing procedure for covering LPs

Input: $P^T = C \in \mathbb{R}^{m\times n}_{\geq 0}, \epsilon \in (0, 1/10], \bar{y} \in R^m_{\geq 0}$ **Output**: $y \in \mathbb{R}^m_{\geq 0}$ such that $Cy \geq \vec{1}$.

1: $\bar{y}' \leftarrow \bar{y}$
2: **for all** i such that $\lambda_i \stackrel{\text{def}}{=} (P^T\bar{y})_i - 1 + \epsilon \leq -2\epsilon$ **do**
3: Let $j = \text{argmax}'_j P_{i,j'}$, i.e. $P_{i,j} = \|P_{:i}\|_\infty$.
4: $\bar{y}'_j \leftarrow \bar{y}'_j + \frac{-\lambda_i}{P_{i,j}}$.
5: **end for**
6: **return** $\frac{\bar{y}'}{1-3\epsilon}$.

for T iterations, with $T \geq \max\{\frac{6w}{\alpha\epsilon}, \frac{2w^2 \log \frac{n}{\epsilon}}{\epsilon^2}\}$, and fix the $\bar{y}$ as in (3.8) with Algorithm 3. Let y be the output of Algorithm 3, y is feasible by construction, i.e. $y \geq 0, Cy = P^T y \geq \vec{1}$. Furthermore, we can show

Theorem 3.3.1. $\mathbb{E}[\vec{1}^T y] \leq (1 + 10\epsilon)\,\text{OPT}$, *and* $y \geq 0, Cy = P^T y \geq \vec{1}$.

Apply Markov's inequality, we get

Corollary 3.3.2. *There is a algorithm that, with probability at least* $9/10$*, computes a* $(1 + O(\epsilon))$*-approximation to the fractional covering problem and has* $\tilde{O}(N/\epsilon^2)$ *expected total work, and implementable in* $\tilde{O}(1/\epsilon^2)$ *distributed iterations.*

Analysis of Algorithm 2 for Covering LPs

We first show $\vec{1}^T\bar{y}$ is close to OPT on expectation.

Lemma 3.3.3. *For any* $T \geq \frac{6w}{\alpha\epsilon}$*, we have* $\mathbb{E}[\vec{1}^T\bar{y}] \leq (1 + 5\epsilon)\,\text{OPT}$

The proof follows directly from Lemma $D.1$ of [AZO15b], only with the additional $w = \log(\lceil \frac{1}{\epsilon} \rceil)$ due to our dynamic grouping, and the expectation. The expectation holds since all the inequalities used in the proof hold universally (i.e. in any outcome of the random choices). We omit the proof here, and encourage interested readers to look at [AZO15b].

Now we look at the i-th constraint of the covering LP, which corresponds to the variable $x[i]$ in the dual packing instance. Let $Z^{(i)}_k$ be the indicator random variable of whether $x[i]$ is in the group being updated in iteration k of Algorithm 2, and

$$S_i = w \sum_{k=0}^{T-1} Z^{(i)}_k (\min\{\nabla_i f_\mu(x_k), 1\} + \epsilon)$$

Using the notations of Algorithm 2, we have for all i, $\frac{S_i}{w} \geq \sum_{k=0}^{T-1} \xi_k^{(t)}[i]$. We know the cumulative update on variable $x[i]$ must be bounded, due to Claim 3.2.5, $x_k[i] \leq \frac{1+\epsilon}{\|P_{:i}\|_\infty}$ for all k, and in particular

$$\frac{1+\epsilon}{\|P_{:i}\|_\infty} \geq x_T[i] = x_0[i] \exp(-\alpha \sum_{k=0}^{T-1} \xi_k^{(t)}[i]) \geq x_0[i] \exp(-\alpha \frac{S_i}{w}) = \frac{1-\epsilon/2}{n\|P_{:i}\|_\infty} \exp(-\alpha \frac{S_i}{w})$$

which gives the lower bound on the random variable S_i

$$S_i \geq -\frac{w \log 2n}{\alpha} \tag{3.9}$$

Notice that the slackness of the i-th covering constraint with the solution $\bar{y}$ is

$$\begin{aligned}
(P^T \bar{y})_i - 1 + \epsilon =& \frac{1}{T} \sum_{k=0}^{T-1} (P^T \overrightarrow{p(x_k)})_i - 1 + \epsilon \\
=& \frac{1}{T} \sum_{k=0}^{T-1} \nabla_i f_\mu(x_k) + \epsilon \\
\geq& \frac{1}{T} \sum_{k=0}^{T-1} \min\{\nabla_i f_\mu(x_k), 1\} + \epsilon \\
\stackrel{\text{def}}{=}& \frac{1}{T} U_i
\end{aligned}$$

with the definition of the random variable $U_i = \sum_{k=0}^{T-1} \min\{\nabla_i f_\mu(x_k), 1\} + \epsilon$. If the i-th variable is updated in all iterations, i.e. $Z_k^{(i)} = 1$ for all k, we have $U_i = S_i$ in that case, and when $T \geq \frac{6w \log 2n}{\alpha\epsilon}$

$$(P^T \bar{y})_i - 1 + \epsilon \geq \frac{1}{T} S_i \geq -\frac{w \log 2n}{\alpha T} \geq -\epsilon$$

so we know $(P^T \bar{y})_i \geq 1 - 2\epsilon$ for all i, which means all covering constraints are approximately feasible. However, we don't always update variable i in all iterations of Algorithm 2, so we need to bound the difference $S_i - U_i$.

Lemma 3.3.4. *For any $T \geq \frac{2w^2 \log \frac{n}{\epsilon}}{\epsilon^2}$, we have* $\Pr[S_i - U_i \geq \epsilon T] \leq \frac{\epsilon}{n}$

Proof. The randomness of U_i and S_i comes from the random choice of which group to update in each iteration of Algorithm 2. Let

$$D_k^{(i)} = w \sum_{k' \leq k} Z_{k'}^{(i)} (\min\{\nabla_i f_\mu(x_k), 1\} + \epsilon) - \sum_{k' \leq k} \min\{\nabla_i f_\mu(x_k), 1\} + \epsilon$$

Let G_k be the random choice (i.e. the group to update) made at k-th iteration of Algorithm 2. Since $Z_k^{(i)}$ is an indicator random variables with probability of $\frac{1}{w}$ being 1, and it is independent from $G_0, \ldots, G_{k-1}$, $D_k^{(i)}$ is a martingale with respect to G_k, as

$$E[D_k^{(i)}|G_0, \ldots, G_{k-1}] = D_{k-1}^{(i)} + \frac{1}{w} w(\min\{\nabla_i f_\mu(x_k), 1\} + \epsilon) - (\min\{\nabla_i f_\mu(x_k), 1\} + \epsilon) = D_{k-1}^{(i)}$$

and we have

$$D_0^{(i)} = E[S_i - U_i] = 0, \quad D_T^{(i)} = S_i - U_i$$

Furthermore, $|D_k^{(i)} - D_{k+1}^{(i)}| \leq w$ for all k, so we can apply Azuma's inequality, and get

$$\Pr[S_i - U_i \geq \epsilon T] \leq \exp(\frac{\epsilon^2 T^2}{2Tw^2}) \leq \frac{\epsilon}{n}$$

□

The above lemma shows that with high probability, $\bar{y}$ satisfies the i-th covering constraint up to -3ϵ. In the rare case it doesn't, we use Algorithm 3 to fix it, and get $\bar{y}'$. We show on expectation this step doesn't add too much to the total cost.

Lemma 3.3.5. $\mathbb{E}[\vec{1}^T \bar{y}'] \leq (1 + 6\epsilon)\,\mathrm{OPT}$.

Proof. When $S_i - U_i \leq \epsilon T$, we have

$$(P^T \bar{y})_i - 1 + \epsilon \geq \frac{1}{T}(S_i - \epsilon T) \geq -\frac{w \log 2n}{\alpha T} - \epsilon \geq -2\epsilon$$

so we don't need to fix the i-th constraint. When that is not the case, since $(P^T \bar{y})_i \geq 0$, and $\|P_{:i}\|_\infty \geq 1$, we need to add at most 1 to some variable $\bar{y}'_j$ to fix the i-th covering constraint. For all the n covering constraints, we add on expectation at most $n\frac{\epsilon}{n} \leq \epsilon \leq \epsilon\,\mathrm{OPT}$ to $\bar{y}$ to get $\bar{y}'$. Together with Lemma 3.3.3, we have $\mathbb{E}[\vec{1}^T \bar{y}'] \leq (1 + 6\epsilon)\,\mathrm{OPT}$. □

We complete the proof of Theorem 3.3.1 by noticing $(P^T \bar{y}')_i \geq 1 - 3\epsilon$ for all i, so the output of Algorithm 3, $\frac{\bar{y}'}{1-3\epsilon}$ satisfies the properties in Theorem 3.3.1.

3.4 $\tilde{O}(\frac{d}{\epsilon})$ Parallel Accelerated Solver for (Packing and) Covering LPs

We show how to combine the idea of updating selective variables with similar gradients to parallelize Algorithm 4 to run in $\tilde{O}(\frac{d}{\epsilon})$ time. The presentation is based on covering LPs, but it is straightforward to see it also works for packing LP. We use the same notations as in Chapter 2 and Section 2.4.

We break the gradient into small, medium, and large components, let

$$\zeta_k[i] = \begin{cases} \nabla_i g_\mu(x_k) & \nabla_i g_\mu(x_k) \in [-\epsilon/n, \epsilon/n] \\ 0 & \text{otherwise} \end{cases}$$

$$\xi_k[i] = \begin{cases} 0 & \nabla_i g_\mu(x_k) \in [-\epsilon/n, \epsilon/n] \\ \nabla_i g_\mu(x_k) & \nabla_i g_\mu(x_k) \in [-1, 1] \backslash [-\epsilon/n, \epsilon/n] \\ -1 & \nabla_i g_\mu(x_k) < -1 \end{cases}$$

$$\eta_k[i] = \begin{cases} \nabla_i g_\mu(x_k) + 1 & \nabla_i g_\mu(x_k) < -1 \\ 0 & \text{otherwise} \end{cases}$$

We have

$$\nabla g_\mu(x_k) = \zeta_k + \xi_k + \eta_k$$

We will further partition the variables into groups, such that for variables in the same group, the absolute values of their truncated gradients will be within a factor 2 of each other. For $t \in \{0, \ldots, \lceil \log(\frac{n}{\epsilon}) - 1 \rceil\}$, let

$$\xi_k^{(t)}[i] = \begin{cases} \xi_k[i] & \xi_k[i] \in (2^t\epsilon/n, 2^{t+1}\epsilon/n] \\ & \cup [-\epsilon 2^{t+1}/n, -\epsilon 2^t/n) \\ 0 & \text{otherwise} \end{cases} \qquad \eta_k^{(t)}[i] = \begin{cases} \eta_k[i] & t = \lceil \log(\frac{n}{\epsilon}) \rceil - 1 \\ 0 & \text{otherwise} \end{cases} \tag{3.10}$$

We have $\xi_k = w\mathbb{E}_t[\xi_k^{(t)}], \eta_k = w\mathbb{E}_t[\eta_k^{(t)}]$, with $w = \lceil \log(\frac{n}{\epsilon}) \rceil$ being the total number of groups.

Theorem 3.4.1. *With x^{start} computable in total work $\tilde{O}(N)$ and $\tilde{O}(1)$ distributed iterations to be specified later, Algorithm 4 outputs y_T satisfying $\mathbb{E}[g_\mu(y_T)] \leq (1+6\epsilon)\,\text{OPT}$, and the total expected work is $\tilde{O}(dN/\epsilon)$ with $\tilde{O}(d/\epsilon)$ expected iterations, where d is the maximum number of variables in any constraint.*

A standard application of Markov bound, together with part 5 of Lemma 2.1.6, gives

Corollary 3.4.2. *There is a algorithm such that with probability at least $9/10$, computes a $(1 + O(\epsilon))$-approximation to the fractional covering problem. The expected total work is $\tilde{O}(dN/\epsilon)$, and the expected number of iterations is $\tilde{O}(d/\epsilon)$.*

We will prove the smoothness property and gradient descent guarantee. The rest of the analysis follows trivially from the analysis of the accelerated stochastic coordinate descent algorithm in Section 2.4, with n replaced by dw, so we omit the proofs.

Lemma 3.4.3. *Let $x_{k+1}^{(t)} = x_k + \frac{z_k^{(t)} - z_{k-1}}{4dw\alpha_k L}$ for any $t = 0, \ldots, w-1$ as in Algorithm 4. Let $B_t = \{i | \xi_k^{(t)}[i] > 0\}$ be the set of variables we update, and $\vec{\gamma} = x_{k+1}^{(t)} - x_k$. For any $x = x_k + \nu\vec{\gamma}$ where $\nu \in [0, 1]$, we have $\forall i \in B_t, \nabla_i g_\mu(x)$ is between $\frac{1}{2}\nabla_i g_\mu(x_k)$ and $\frac{3}{2}\nabla_i g_\mu(x_k)$.*

Algorithm 4 Accelerated $\tilde{O}(\frac{d}{\epsilon})$ time parallel solver for both packing and covering

Input: $C \in \mathbb{R}^{m\times n}_{\geq 0}, x^{\text{start}} \in \Delta, g_\mu, \epsilon$ **Output**: $y_T \in \Delta$

$w \leftarrow \lceil \log(\frac{n}{\epsilon}) \rceil, d \leftarrow \max_j \|C_{j,:}\|_0$
$\mu \leftarrow \frac{\epsilon}{4\log(nm/\epsilon)}, L \leftarrow \frac{4}{\mu}, \tau \leftarrow \frac{1}{12dwL}$
$T \leftarrow \lceil 12dwL\log(1/\epsilon) \rceil = \tilde{O}(\frac{d}{\epsilon})$
$x_0, y_0, z_0 \leftarrow x^{\text{start}}, \alpha_0 \leftarrow \frac{1}{dwL}$
for $k = 1$ to T **do**
    $\alpha_k \leftarrow \frac{1}{1-\tau}\alpha_{k-1}$
    $x_k \leftarrow \tau z_{k-1} + (1-\tau)y_{k-1}$
    Select $t \in [w]$ uniformly at random.
    ▷ Gradient truncation:
    Let $\xi_k^{(t)}$ be as specified in (3.10)
    ▷ Mirror descent step:
    $z_k \leftarrow z_k^{(t)} \stackrel{\text{def}}{=} \text{argmin}_{z\in\Delta}\{V_{z_{k-1}}(z) + \langle z, w\alpha_k\xi_k^{(t)}\rangle\}$.
    ▷ Gradient descent step:
    $y_k \leftarrow y_k^{(t)} \stackrel{\text{def}}{=} x_k + \frac{1}{4dw\alpha_k L}(z_k^{(t)} - z_{k-1})$
end for
return y_T.

Proof. Because for all $i \in B_t$, $\xi_k^{(t)}[i] \in (\epsilon 2^t/n, \epsilon 2^{t+1}/n] \cup [-\epsilon 2^{t+1}/n, -\epsilon 2^t/n)$, we know the gradient step on each variable is bounded by $|\vec{\gamma}_i| \leq \frac{\epsilon 2^{t+1}/n}{4dL\|C_{:i}\|_\infty}$, we must have for all i,

$$x[i] \in [x_k[i] - \frac{\epsilon 2^{t+1}/n}{4dL\|C_{:i}\|_\infty}, x[i] + \frac{\epsilon 2^{t+1}/n}{4dL\|C_{:i}\|_\infty}] \tag{3.11}$$

Now we look at the impact of the step on the exponential penalties

$$p_j(x) = \exp(\frac{1}{\mu}((Cx)_j - 1))$$

Due to (3.11), and d is the max degree of any constraint, we have

$$|(Cx)_j - (Cx_k)_j| \leq \frac{\epsilon 2^t/n}{2L}$$

Then by our choice of $L = \frac{4}{\mu}$, we have

$$p_j(x) \geq p_j(x_k)\exp(-\frac{\epsilon 2^t/n}{2L\mu}) = p_j(x_k)\exp(-\frac{\epsilon 2^t/n}{8})$$

Since $\epsilon 2^t/n \leq 1$, we have $\exp(-\frac{\epsilon 2^t/n}{8}) \geq (1 - \frac{\epsilon 2^t/n}{4})$. By a similar calculation for the upper bound, we have

$$p_j(x) \in p_j(x_k) \cdot [1 - \frac{\epsilon 2^t/n}{4}, 1 + \frac{\epsilon 2^t/n}{4}] \tag{3.12}$$

For any $i \in B_t$, if $\xi_k^{(t)} \in (\epsilon 2^t/n, \epsilon 2^{t+1}/n]$, we have

$$\begin{aligned}\nabla_i g_\mu(x) &= 1 - (C^T p(x))_i \\ &> 1 - (C^T p(x_k))(1 + \frac{\epsilon 2^t/n}{4}) \\ &= 1 - (1 - \nabla_i g_\mu(x_k))(1 + \frac{\epsilon 2^t/n}{4}) \\ &\geq \frac{\nabla_i g_\mu(x_k)}{2}\end{aligned}$$

where the last step is due to $\epsilon 2^t/n \leq 1$ and $\nabla_i g_\mu(x_k) \geq \xi_k^{(t)}[i] > \epsilon 2^t/n$. By similar calculation, we get $\nabla_i g_\mu(x) \leq \frac{3}{2}\nabla_i g_\mu(x_k)$. The same holds for the case $\xi_k^{(t)}[i] \in [-\epsilon 2^{t+1}/n, -\epsilon 2^t/n)$. □

We bound the improvement from the gradient descent step in the following lemma

Lemma 3.4.4. $g_\mu(x_k) - g_\mu(y_k^{(t)}) \geq \frac{1}{2}\langle \nabla g_\mu(x_k), x_k - y_k^{(t)} \rangle$

Proof. Let $\vec{\gamma}$ be the step, i.e. $\vec{\gamma}[i] = y_k^{(t)}[i] - x_k[i]$, we have

$$g_\mu(x_k) - g_\mu(y_k^{(t)}) = g_\mu(x_k) - g_\mu(x_k + \vec{\gamma}) = \int_0^1 -\langle \nabla g_\mu(x_k + \nu\vec{\gamma}), \vec{\gamma} \rangle d\nu$$

Since $\vec{\gamma}_i$'s are non-zero only at coordinates i's, such that $\epsilon 2^t/n \leq |\xi_k[i]| \leq \epsilon 2^{t+1}/n$, denote the set of those variables B_t, we have

$$g_\mu(x_k) - g_\mu(y_k^{(t)}) = \sum_{i \in G_t} -\vec{\gamma}[i] \int_0^1 \nabla_i g_\mu(x_k + \nu\vec{\gamma}) d\nu$$

By Lemma 3.4.3, $\nabla_i g_\mu(x_k + \nu\vec{\gamma}) \geq \frac{1}{2}\nabla_i g_\mu(x_k)$ when $-\vec{\gamma}[i] > 0$, and $\nabla_i g_\mu(x_k + \nu\vec{\gamma}) \leq \frac{1}{2}\nabla_i g_\mu(x_k)$ when $-\vec{\gamma}[i] < 0$, we have

$$g_\mu(x_k) - g_\mu(y_k^{(t)}) \geq \sum_{i \in G_t} -\vec{\gamma}[i]\frac{1}{2}\nabla_i g_\mu(x_k) = \frac{1}{2}\langle \nabla g_\mu(x_k), x_k - y_k^{(t)} \rangle$$

□

Chapter 4

Parallel Algorithm for Positive LPs

In this chapter we describe our parallel algorithm for general positive LPs, i.e., LPs with both packing and covering constraints. Unlike the algorithms from previous chapters, we use the Lagrangian-relaxation framework for mixed packing and covering LPs.

Via standard reductions, we can use binary search on the optimal value of the positive LP, and rewrite the objective function as a packing constraint (for minimization problems) or a covering constraint (for maximization problems), thus turning the optimization problem into a feasibility problem. We present a parallel algorithm that, given a mixed packing and covering (feasibility) LP of size N, in $\tilde{O}(\frac{1}{\epsilon^3})$ iterations computes a $(1+\epsilon)$-approximate (feasible) solution, or correctly reports the original mixed packing and covering LP is infeasible. The algorithm is deterministic and width-independent.

The bottleneck of each iteration is a matrix-vector multiplication, and can be implemented in $O(\log N)$ depth, in which case the running time of our algorithm is $\tilde{O}(\frac{1}{\epsilon^3})$. The total work of the algorithm we present in the paper is $\tilde{O}(\frac{N}{\epsilon^3})$. In particular, our result improves upon the current fastest parallel algorithm of mixed packing and covering LPs in [You01; You14], where the running time is $\tilde{O}(\frac{1}{\epsilon^4})$. The work of the parallel algorithm in [You01; You14] is $\tilde{O}(\frac{1}{\epsilon^2})$. We note that using a simple lazy update modification on the algorithm, which is the same technique used in [You01; You14], we can reduce the work of our algorithm to $\tilde{O}(\frac{1}{\epsilon^2})$. Same as in [You01; You14], this comes at the cost of requiring a centralized step in the parallel algorithm. Since the iteration count is the more interesting side of this line of work, we will not incorporate the lazy update in our algorithm for simplicity.

Furthermore, in the case of pure packing problem or pure covering problem, our algorithm allows a similar but simplified analysis, and will converge in $\tilde{O}(\frac{1}{\epsilon^2})$ iterations.

4.1 Technical overview

To compute $(1+\epsilon)$-approximation of a mixed packing and covering LP in the form (1.1), via standard reduction and scaling (e.x. See [You01]), it suffices to solve a $(1+O(\epsilon))$-feasibility problem as specified in (4.1) and (4.2).

At a high level, our work follows the Lagrangian-relaxation approach as in [You01; You14]. In particular, we replace the hard packing and covering constraints with a scalar-valued potential function, which is continuous and smooth. The potential function measures how far away the current solution is from satisfying all the captured constraints. We then start with a x of very small values, and iteratively increase x by a carefully chosen increment vector Δ, which is guided by the gradients of the potential function. The increment vector Δ will keep driving down the potential function, until the function gets small enough, at which point we terminate, and conclude we either have an approximate solution or the original LP is infeasible.

Same as in [You01; You14], each iteration a subset of the variables are picked based on the gradients of the potential function, and are increased within a local smooth region so we can bound the change of the potential function. However, unlike the parallel algorithm in [You01; You14] that multiplicatively updates all variables in the subset with a uniform step size, we further incorporate the gradients into step sizes of individual variables' updates. This discriminative multiplicative step size allows more aggressive updates on average, and is directly motivated by the line of works using gradient based optimization methods [AZO15b; AZO15a; Wan+15]. However, we move away from the optimization oriented view of these update steps in favor of more localized and adhoc analyses, which was developed to analyze direct adaptations of Young's algorithm for purely-packing SDPs [PT12], leading to bounds similar to the optimization based approaches [ZLO15].

In particular, for each variable there will be a packing gradient capturing the packing part of the potential function, and there will be a covering gradient for the covering part of the potential function. The packing gradient captures the overall impact of a variable on the packing constraints, so a large packing gradient suggests the variable should not be increased. The covering gradient, on the other hand, gauges the impact of the variable on the covering constraints, and when large, it will be advantageous to increase the variable. To decide whether or not to increase a variable, we look at both gradients, and update only when the covering gradient is larger. Furthermore, to determine a variable's update step size, we use the ratio of its two gradients.

Similar to Young's algorithm [You01], the overall progress of the algorithm is captured by how large the constraints become, which in turn depend on how large the variables are. A sufficient condition for the algorithm to terminate is when a variable is increased by more than a certain amount. To bound the number of iterations before any variable gets too large, we combine the notion of phases from [You01] with the more refined analysis of gradient updates from more recent works [Wan+15]. This is owing to the clearer combinatorial structure of our interpretation of discriminative multiplicative steps

On a high level, the algorithm demonstrates the behavior that it will update a subset of variables for a while, until certain global progress is made, before shifting to other variables. The phases are created to capture these local windows, such that within a phase the algorithm only makes limited global progress. Since the update only increases variables, thus only increasing the values of the constraints, this translate to certain monotonicity-like property on the gradients within the interactions captured by a single phase. In [You01], the phases are defined so that any variable being increased at the last iteration of a phase must have been increased in every iteration of the phase, which, coupled with a lower bound on each increase, gives a bound of the number of iterations in a phase. In our analysis, we significantly expand the phases to capture larger global

progress, leading to a smaller number of phases.

The larger phases lead to a weaker monotonicity property on the gradients within a phase. To bound the number of iterations in our phase, we divide the iterations into two groups: some initial warm-up *bad* iterations followed by subsequent *good* iterations containing more interesting segments of the path to convergence (See Definition 4.2.1 for formal definitions). The bad iterations are ones where packing gradients are much smaller than covering gradients. Such steps create difficulties in the analysis, but an analysis identical to Young's algorithm [You01] shows that they must occur near the very start of a phase. In the subsequent good iterations, the packing gradients for all variables are relatively large comparing to their covering counterparts, so we only get weak signals as to which variables to move, and we can only move them by small steps. To bound the number of good iterations, we construct a dual solution from a suitable average of the packing and covering gradients over all the good iterations in a phase. Intuitively, if the primal LP is feasible, the dual solution certifies that there must be some key variable(s) we increase during the phase to achieve the fixed global progress. Particularly, we know that there is at least one variable that on average has smaller packing gradients than covering gradients. Moreover, since in the good iterations, the packing gradients are all at least on the same scale as the covering gradients, an argument in the spirit of Markov's inequality then implies that the corresponding variable was increased by a large amount, which in turn leads to a bound on the number of iterations of a phase.

Remarks

We note that the phases in our result are virtual: we only need them in the analysis, but not in the actual execution of the algorithm. In particular, this modification to Young's algorithm [You01] removes the dependency of updates on the phase of the overall algorithm as well as the current gradient. We believe this direct removal of phases also apply to other variants of Young's algorithm [You14].

Furthermore, we believe that there is a more natural variant of the analysis that does not rely on phases, and treats all the iterations in a completely symmetric manner. Such an analysis is likely crucial for extending our results to the SDP setting, where the gradients exhibit much weaker monotonicity behaviors [ZLO15]. We are optimistic that it will lead to an $\tilde{O}(\frac{1}{\epsilon^2})$ bound for the mixed packing-covering case, which we believe is the more likely asymptotic behaviors of phaseless, gradient update variants of Young's algorithm [You01; You14].

4.2 Parallel Algorithm for Mixed Packing and Covering LPs

Preliminaries

To compute $(1+\epsilon)$-approximation of a mixed packing and covering LP in the form (1.1), via standard reduction and scaling (e.x. See [You01]), it suffices to solve the following $(1+O(\epsilon))$-feasibility problem, that is, either find $x \geq 0$ such that

$$0 < \max Px \leq (1+\epsilon)\min Cx \tag{4.1}$$

or conclude the following LP is infeasible

$$\begin{aligned} Cx &\geq \vec{1} \\ Px &\leq (1-10\epsilon)\vec{1} \\ x &\geq 0 \end{aligned} \tag{4.2}$$

We present our parallel $\tilde{O}(1/\epsilon^3)$ routine in Algorithm 5 for solving the $(1+O(\epsilon))$-feasibility problem above, that is, either find $x \geq 0$ satisfying (4.1), or certify the infeasibility of (4.2). The input contains a packing constraint matrix $P \in \mathbb{R}_{\geq 0}^{n_P \times m}$, a covering constraint matrix $C \in \mathbb{R}_{\geq 0}^{n_C \times m}$, and an error parameter $0 < \epsilon$. That is, there are m variables, n_P packing constraints, and n_C covering constraints. We also use $n = n_P + n_C$ to denote the total number of constraints.

To certify that (4.2) is infeasible, we rely on the dual LP of (4.2).

Lemma 4.2.1. *By duality,* (4.2) *is infeasible if there exists* $y, z \geq 0$ *s.t.*

$$(1-10\epsilon)\frac{C^T z}{\vec{1}^T z} < \frac{P^T y}{\vec{1}^T y}. \tag{4.3}$$

Proof. Eqn. (4.3) is a direct reformulation of the dual LP of (4.2). Since we only need the sufficient condition, the result is by weak duality. If there exists any $x \geq 0$ satisfying (4.2), we have

$$\begin{aligned} (1-10\epsilon)\frac{x^T C^T z}{\vec{1}^T z} &\geq (1-10\epsilon)\frac{\vec{1}^T z}{\vec{1}^T z} = 1-10\epsilon, \\ \frac{x^T P^T y}{\vec{1}^T y} &\leq (1-10\epsilon)\frac{\vec{1}^T y}{\vec{1}^T y} = 1-10\epsilon. \end{aligned}$$

Together they give $1 < 1$, contradiction. □

As in [You01], we use the soft-max $\operatorname{lmax}(Px)$ and soft-min $\operatorname{lmin}(Cx)$ in our potential function

$$\operatorname{lmax}(Px) = \ln(\sum_j \exp(Px)_j) = \ln(\sum_j \exp(P_j^T x))$$

and

$$\operatorname{lmin}(Cx) = -\ln(\sum_j \exp(-Cx)_j) = -\ln(\sum_j \exp(-C_j^T x)),$$

where P_j^T, C_j^T are the j-th row of P, C respectively. In particular, these functions give smooth approximation to $\max Px$ and $\min Cx$:

$$\begin{aligned} \max Px \leq \operatorname{lmax}(Px) &\leq \max Px + \ln n \\ \min Cx \geq \operatorname{lmin}(Cx) &\geq \min Cx - \ln n. \end{aligned} \tag{4.4}$$

Algorithm

We start with small $x_i^{(0)} = \frac{1}{m\|P_{:i}\|_\infty}, \forall\ i \in [m]$, and keep increasing x properly, until it reaches the terminate condition in line 5, that is, $\max\{\max Px, \min Cx\} \geq K = \frac{10 \ln n}{\epsilon}$. The reason of the chosen K value is stated in Lemma 4.2.5. Roughly, when the difference of $\min Cx$ and $\max Px$ becomes ϵ factor smaller than $\max\{\max Px, \min Cx\}$, we know that $\max Px \leq (1 + O(\epsilon)) \min Cx$.

In each iteration of the while-loop, we first delete all covering constraints which has already reached K. Since x never decreases, we know that once a row is deleted, we no longer need to look at it. Note the covering matrix cannot be empty, since we enter the iteration with $\min Cx < K$. We compute the vectors y, z, which are exponentials of the values of the packing and covering constraints respectively. We then compute a and b, which can be considered as gradients of $\mathrm{lmax}(Px)$ and $\mathrm{lmin}(Cx)$ respectively, and use them to guide our update on x. In particular, we update x_i if $a_i \leq (1 - \epsilon/50)b_i$ (i.e. $i \in B$). Furthermore, we update x_i multiplicatively by a factor depends on the ratio of $\frac{a_i}{b_i}$, as specified in Eqn. (4.5) and line 13. Note that the smallest update in our algorithm is by a factor of $(1 + \Omega(\frac{\epsilon^2}{\ln n}))$, which is the same as the fixed update step size in [You14], and in general our updates take larger steps.

Note that in our analysis, we equivalently view z as the full n_C-dimensional vector, where the coordinates corresponding to deleted constraints are filled by 0's. In particular, the matrix-vector product of the original C with the n_C-dimensional z will be the same as the product of the reduced covering matrix $C^{(t)}$ and reduced z.

Proof of Correctness

In this section we will show Algorithm 5 will terminate, and output the correct answer.

Lemma 4.2.2 shows that empty B certifies the infeasibility of the input instance (4.2), which proves the correctness if we end up in the case of line 11.

Lemma 4.2.2. *If the problem instance* (4.2) *is feasible, then*

$$\forall\ x \geq 0, B = \{i : a_i \leq (1 - \frac{\epsilon}{50})b_i\} \neq \emptyset.$$

Proof. Let x^* be a feasible solution of (4.2). Assume by contradiction, $\exists\ x \geq 0, \forall\ i \in [m], a_i^{(t)} > (1 - \frac{\epsilon}{50})b_i^{(t)}$. By definition of a, b, it is equivalent to $\exists\ y, z \geq 0$ such that

$$(1 - \frac{\epsilon}{50})\frac{C^T z}{\vec{1}^T z} < \frac{P^T y}{\vec{1}^T y}.$$

Then the result follows directly from Lemma 4.2.1. □

If Algorithm 5 doesn't terminate with line 11, it must increase at least one variable by at least a factor of $(1 + \frac{\epsilon^2}{10 \ln n})$ each iteration, so the algorithm must reach the termination condition of

Algorithm 5 Parallel algorithm for mixed packing and covering LPs

Input: P, C, ϵ

Output: "infeasible" or $x \geq 0$ s.t. $Px \leq (1+\epsilon)Cx$

Let $K = \frac{10 \ln n}{\epsilon}$, $\alpha = \frac{1}{K}$, where n is the number of constraints.
Initialize $x_i^{(0)} = \frac{1}{m\|P_{:i}\|_\infty}, \forall\, i \in [m]$, where m is the number of variables.

Let $t = 0$.
while $\max Px < K$ and $\min Cx < K$ **do**
    Let $C^{(t)}$ be C with rows j such that $\left(Cx^{(t)}\right)_j \geq K$ deleted.
    Let $y^{(t)} = \exp\left(Px^{(t)}\right), z^{(t)} = \exp\left(-C^{(t)}x^{(t)}\right)$.

        $a^{(t)} = \frac{P^T y^{(t)}}{\vec{1}^T y^{(t)}}, b^{(t)} = \frac{(C^{(t)})^T z^{(t)}}{\vec{1}^T z^{(t)}}$.
    Define $B^{(t)} = \{i : a_i^{(t)} \leq (1 - \frac{\epsilon}{50}) b_i^{(t)}\}$.
    If $B^{(t)} = \emptyset$, then return "infeasible".
    Let

$$\Delta_i^{(t)} = \begin{cases} \frac{1}{2}(1 - \frac{a_i^{(t)}}{b_i^{(t)}}) \in [\epsilon/100, \frac{1}{2}] & \text{if } i \in B^{(t)} \\ 0 & \text{if } i \notin B^{(t)} \end{cases} \tag{4.5}$$

    $x_i^{(t+1)} \leftarrow x_i^{(t)}(1 + \alpha\Delta_i^{(t)})$.
    $t \leftarrow t + 1$.
end while
return $x = \frac{x^{(t)}}{K}$.

the while loop at some point, and we need to show the output x satisfies (4.1). We consider the following potential function,

$$f(x^{(t)}) = \mathrm{lmax}(Px^{(t)}) - \mathrm{lmin}(C^{(t)}x^{(t)}) = \ln(\vec{1}^T y^{(t)}) + \ln(\vec{1}^T z^{(t)}).$$

We first quantify the changes of lmax and lmin when we update the variables. This type of smoothness analysis is standard in analyzing algorithms that make updates using gradient information. Similar results are derived in other works on packing and covering (See [AZO15b; You01]). The particular analysis we develop can deal with larger gradient steps. In particular, the approach of our analysis allows updates that may move the gradients of some variables out of their respective coordinate-wise smooth regions, as long as we can still bound the combined impact on the potential function from updates of all variables. This approach can extend straightforwardly to show larger updates also work in [AZO15b], and improve their pure packing algorithm to run in $\tilde{O}(\frac{N}{\epsilon^2})$ iterations. Since the proof is technically tedious, we defer the proof to Section 4.3.

Lemma 4.2.3. *At each iteration t,*

$$\operatorname{lmax}(Px^{(t+1)}) \le \operatorname{lmax}(Px^{(t)}) + \alpha\langle a^{(t)}, (1+\Delta^{(t)})\cdot\Delta^{(t)}\cdot x^{(t)}\rangle$$

and

$$\operatorname{lmin}(C^{(t+1)}x^{(t+1)}) \ge \operatorname{lmin}(C^{(t)}x^{(t)}) + \alpha\langle b^{(t)}, (1-\Delta^{(t)})\cdot\Delta^{(t)}\cdot x^{(t)}\rangle,$$

where $\Delta\cdot x$ is the entry-wise product vector, i.e., $(\Delta\cdot x)_i = \Delta_i x_i$.

With the above bounds on the changes of the two components $\operatorname{lmax}(Px)$ and $\operatorname{lmin}(Cx)$, we can show how our updates move the potential function $f(x)$.

Lemma 4.2.4. *Given $\max Px^{(t)} < \frac{10\ln n}{\epsilon}$ and $\min Cx^{(t)} < \frac{10\ln n}{\epsilon}$, we always have $f(x^{(t)}) \le 2\ln n$ during the execution of Algorithm 5.*

Proof. Initially, $x_i^{(0)} = \frac{1}{m\|P_{:i}\|_\infty}$, we have $Px^{(0)} \le \vec{1}$ and $Cx \ge 0$, thus $f(x^{(0)}) \le 2\ln n$. To show $f(x) \le 2\ln n$ for all iterations t before terminate, it suffices to show that $f(x)$ is non-increasing during the process. From Lemma 4.2.3,

$$\begin{aligned} f(x^{(t+1)}) - f(x^{(t)}) &\le \alpha\langle a, (1+\Delta)\cdot\Delta\cdot x^{(t)}\rangle - \alpha\langle b, (1-\Delta)\cdot\Delta\cdot x^{(t)}\rangle \\ &= \sum_i \alpha\Delta_i x_i(a_i(1+\Delta_i) - b_i(1-\Delta_i)). \end{aligned}$$

For each $i\in[m]$, by our update rule (4.5), either Δ_i, or $\Delta_i = \frac{1}{2}(1-\frac{a_i}{b_i})$, in which case

$$a_i(1+\Delta_i) - b_i(1-\Delta_i) = \frac{3a_ib_i - a_i^2}{2b_i} - \frac{a_i+b_i}{2} = \frac{2a_ib_i - a_i^2 - b_i^2}{2b_i} \le 0,$$

so all the summands are non-positive, thus $f(x)$ is non-increasing. □

The above lemma guarantees that the difference between $\operatorname{lmax}(Px)$ and $\operatorname{lmin}(Cx)$ is bounded by $2\ln n$, which by Eqn. (4.4) suggests $\max Px \le \min Cx + O(\ln n)$.Then when the two terms are large at termination, we are approximately feasible as the difference is a factor of ϵ smaller.

Lemma 4.2.5. *If Algorithm 5 terminates with line 16, then it returns an $x \ge 0$ with $0 < \max Px \le (1+\epsilon)\min Cx$.*

Proof. Suppose the algorithm terminates at iteration T, that is, $\max Px^{(T)} \ge \frac{10\ln n}{\epsilon}$ or $\min Cx^{(T)} \ge \frac{10\ln n}{\epsilon}$. Consider iteration $T-1$, the covering matrix is not empty (otherwise, the algorithm terminates before iteration T). Since $x^{(T)} = x^{(T-1)}\cdot(1+\alpha\Delta^{(T-1)}) \le (1+\frac{5\epsilon}{\ln n})x^{(T-1)}$, we have $\max Px^{(T-1)} \ge \frac{5\ln n}{\epsilon}$ or $\min Cx^{(T-1)} \ge \frac{5\ln n}{\epsilon}$.

By Lemma 4.2.4,

$$\max Px^{(T-1)} \le \operatorname{lmax}(Px^{(T-1)}) \le \operatorname{lmin}(C^{(T-1)}x^{(T-1)}) + 2\ln n \le \min Cx^{(T-1)} + 2\ln n.$$

Since $2\ln n \leq \epsilon \cdot \frac{5\ln n}{\epsilon}$, we have

$$\max Px^{(T-1)} \leq (1+\epsilon)\min Cx^{(T-1)}.$$

This also gives $\max Px^{(T)} \leq (1+\epsilon)\min Cx^{(T)}$, since x^T is within in multiplicative factor $1+\frac{\epsilon}{10\ln n}$ of x^{T-1}. Since we start with $x > 0$, and only increase x, we also have $\max Px > 0$. So the x we return at the end satisfies (4.1). □

Analysis of Convergence

So far we have proved that Algorithm 5 will terminate, and will either output x satisfying (4.1) at the end, or terminate earlier and correctly certify (4.2) is infeasible. In this section we show that if (4.2) is feasible, Algorithm 5 must finish with the first case in $\tilde{O}(\frac{1}{\epsilon^3})$ iterations, so if the algorithm takes more than $\frac{1000\ln n\ln(\frac{m}{\epsilon})}{\epsilon^3}$ iterations to complete, we can terminate it, and correctly output that (4.2) is infeasible.

We adapt the concept *phase* from Young's algorithm. Note phase is only used in our analysis, and our algorithm does not contain phase. Formally, phase s contains the iterations t such that

$$\frac{n_P}{n_C}\cdot 2^s \leq \frac{\vec{1}^T y^{(t)}}{\vec{1}^T z^{(t)}} < \frac{n_P}{n_C}\cdot 2^{s+1}$$

where n_P is the number of packing constraints and n_C is the number of covering constraints.

Since we only increase x, $\frac{\vec{1}^T y}{\vec{1}^T z}$ is monotonically increasing, so each phase covers a consecutive sequence of iterations. Furthermore, as $\ln(\frac{\vec{1}^T y}{\vec{1}^T z}) = \mathrm{lmax}(Px) + \mathrm{lmin}(Cx)$ measures global progress towards termination, each phase captures a fixed amount of progress. From our definition of phases, and the termination condition, we have

Lemma 4.2.6. *The total number of phases in Algorithm 5 is* $O(\frac{\log n}{\epsilon})$.

Proof. Since x is monotonically increasing, $y = \exp(Px)$ and $z = \exp(-Cx)$ are monotonically increasing and decreasing respectively, which implies that the quantity $\frac{\vec{1}^T y}{\vec{1}^T z}$ is monotonically increasing. Initially $Px^{(0)} \geq 0, Cx^{(0)} \geq 0$, we know $\frac{\vec{1}^T y^{(0)}}{\vec{1}^T z^{(0)}} \geq \frac{n_P}{n_C}$. By the termination condition in Algorithm 5, the ratio never goes beyond $n_P \exp(\frac{10\log n}{\epsilon})$. Therefore, the total number of phases is $O(\frac{\log n}{\epsilon})$. □

We now bound the number of iterations in a single phase. The iterations of a phase are divided into two groups, the bad iterations and the good iterations, formally defined as follows.

Definition 4.2.1. *If in an iteration* t*, we have for all* i

$$\frac{a_i^{(t)}}{b_i^{(t)}} > \frac{1}{3}, \tag{4.6}$$

then we call it a good *iteration. Otherwise we call it a* bad *iteration.*

Note a phase may contain only bad iterations or only good iterations. We bound the total number of iterations in the two groups separately.

As discussed earlier, the bad iterations capture the initial warm-up iterations of a phase, where in any bad iteration, we can identify some variable x_i with a strong signal (i.e. $\frac{a_i}{b_i} \leq \frac{1}{3}$), so we can increase the variable by a lot. This restricts the warm-up sequence from getting too long, and we formalize the intuition in the following lemma.

Lemma 4.2.7. *In a single phase, the number of bad iterations is at most* $O(\ln n \ln(\frac{m}{\epsilon})/\epsilon)$.

Proof. We will prove the result by showing that there cannot be any bad iteration after the initial $100 \ln n \ln(\frac{m}{\epsilon})/\epsilon$ iterations of a phase. By contradiction, if for any variable i, after $\Omega(\ln n \ln(\frac{m}{\epsilon})/\epsilon)$ iterations of a phase, we have at iteration t such that for some i,

$$\frac{a_i^{(t)}}{b_i^{(t)}} = \frac{(P^T y^{(t)})_i}{\vec{1}^T y^{(t)}} \frac{\vec{1}^T z^{(t)}}{(C^T z^{(t)})_i} \leq \frac{1}{3},$$

then this ratio is at most $\frac{2}{3}$ in all previous iterations of this phase, since $\frac{\left(P^T y\right)_i}{(C^T z)_i}$ is monotonically increasing, and $2^s \leq \frac{\vec{1}^T y}{\vec{1}^T z} < 2^{s+1}$ in this phase. Equivalently, this is saying $a_i \leq \frac{2}{3} b_i$, so $i \in B$ in all previous $\Omega(\ln n \ln(\frac{m}{\epsilon})/\epsilon)$ iterations of the phase, and $\Delta_i \geq \frac{1}{6}$ in all those iterations.

Each iteration the multiplicative update on x_i is $(1 + \alpha\Delta_i)$, which is $(1 + \Theta(\frac{\epsilon}{10 \ln n}))$ since $\Delta_i \geq \frac{1}{6}$. As x_i starts with $\frac{1}{m\|P_{:i}\|_\infty}$, after $100 \ln n \ln(\frac{m}{\epsilon})/\epsilon$ updates, we have $x_i \gg \frac{10 \ln n}{\epsilon \|P_{:i}\|_\infty}$, which gives $\max Px \gg \frac{10 \ln n}{\epsilon}$, so the algorithm must have terminated. □

The above lemma guarantees that all iterations after the first $100 \ln n \ln(\frac{m}{\epsilon})/\epsilon$ must be good iterations, so we proceed to bound the number of these good iterations in a single phase. Without loss of generality, we index these good iterations in a phase as $1, \ldots, T$ by shifting t.

We first identify one variable that must be updated extensively in these iterations.

Claim 4.2.8. *Suppose the instance* (4.2) *is feasible, then There exists* $i \in [m]$ *such that*

$$\sum_{t=1}^{T} b_i^{(t)} - a_i^{(t)} \geq 10\epsilon \sum_{t=1}^{T} b_i^{(t)}. \tag{4.7}$$

Proof. Define $\overline{y}$ and $\overline{z}$ to be the sum of the normalized gradients of iterations $1, \ldots, T$, that is,

$$\overline{y} = \sum_{t=1}^{T} \frac{y^{(t)}}{\vec{1}^T y^{(t)}}, \quad \overline{z} = \sum_{t=1}^{T} \frac{z^{(t)}}{\vec{1}^T z^{(t)}}.$$

Note $\vec{1}^T \overline{y} = \vec{1}^T \overline{z} = T$. Recall $a_i^{(t)}$ and $b_i^{(t)}$ are respectively $\frac{(P^T y^{(t)})_i}{\vec{1}^T y^{(t)}}$ and $\frac{(C^T z^{(t)})_i}{\vec{1}^T z^{(t)}}$, then

$$\sum_{t=1}^{T} a_i^{(t)} = \frac{T(P^T \overline{y})_i}{\vec{1}^T \overline{y}}, \quad \sum_{t=1}^{T} b_i^{(t)} = \frac{T(C^T \overline{z})_i}{\vec{1}^T \overline{z}}.$$

Assume by contradiction, $\forall\, i \in [m]$, $\sum_{t=1}^{T} a_i^{(t)} > (1-10\epsilon)\sum_{t=1}^{T} b_i^{(t)}$, that is,

$$\frac{P^T\overline{y}}{\vec{1}^T\overline{y}} > (1-10\epsilon)\frac{C^T\overline{z}}{\vec{1}^T\overline{z}}.$$

By Lemma 4.2.1, $\overline{y}, \overline{z}$ certify infeasibility of the instance (4.2), which contradicts the assumption. □

The above claim gives us a variable that on average has smaller packing gradients than covering gradients in this iteration. Together with the property we have on the good iterations (4.6), we can bound the number of good iterations.

Lemma 4.2.9. *In a single phase, the number of good iterations is at most* $O(\ln n \ln(\frac{m}{\epsilon})/\epsilon^2)$.

Proof. Let x_i be a variable satisfying Eqn. (4.7). We want to turn Eqn. (4.7) into some lower bound on the total multiplicative update on x_i through these iterations. Intuitively, a bad case is that in some iteration t, $a_i^{(t)}, b_i^{(t)}$ are much larger than the values in other iterations, since they can dominate the terms from other iterations in Eqn. (4.7), but not much to the total update of x_i, since their ratio is what matters to the update. However, since we are inside one single phase, and only looking at good iterations, we can show the bad scenario will not show up.

Formally, let $l = a_i^{(1)}$ and $u = b_i^{(1)}$. Since $(P^T y)_i$ monotonically increases, and $\vec{1}^T y$ will increase but not by more than a factor of 2 in a phase, we have

$$a_i^{(t)} = \frac{(P^T y^{(t)})_i}{\vec{1}^T y^{(t)}} \geq l/2 \qquad \forall t = 1, \ldots, T \tag{4.8}$$

Similarly, we have

$$b_i^{(t)} = \frac{(C^T z^{(t)})_i}{\vec{1}^T z^{(t)}} \leq 2u \qquad \forall t = 1, \ldots, T \tag{4.9}$$

Furthermore, since we are looking at the good iterations, we have

$$l \geq \frac{1}{3}u. \tag{4.10}$$

The inequalities above allow us to turn the difference-based guarantee from Eqn. (4.7) into lower bounds on ratios we need.

By the update (4.5), we have

$$\Delta_i^{(t)} \geq \frac{(1-\frac{\epsilon}{50})b_i^{(t)} - a_i^{(t)}}{2b_i^{(t)}}.$$

So we can lower bound the total update on x_i as follows

$$\begin{aligned} x_i^{(T)} \geq & x_i^{(1)} \exp\left(\frac{\alpha \sum_t \Delta_i^{(t)}}{2}\right) \\ = & x_i^{(1)} \exp\left(\frac{\alpha}{4} \sum_t \frac{(1-\frac{\epsilon}{50}) b_i^{(t)} - a_i^{(t)}}{b_i^{(t)}}\right) \\ \geq & x_i^{(1)} \exp\left(\frac{\alpha}{4} \sum_t \left(\frac{b_i^{(t)} - a_i^{(t)}}{2u} - \frac{\epsilon}{50}\right)\right) \end{aligned}$$

where we used (4.9) in the last line.

From Eqn. (4.7), we have

$$\begin{aligned} \sum_t b_i^{(t)} - a_i^{(t)} \geq & 10\epsilon \sum_i b_i^{(t)} \\ \geq & \frac{10\epsilon}{1-10\epsilon} \sum_t a_i^{(t)} \\ \geq & \frac{\epsilon u T}{1-10\epsilon} \geq \epsilon u T. \end{aligned}$$

The first two lines both follow from Eqn. (4.7), the next line follows from $a_i^{(t)} \geq l/2 \geq u/6$.

Thus

$$x_i^{(T)} \geq x_i^{(1)} \exp\left(\frac{\epsilon\alpha T}{8} - \frac{\epsilon\alpha T}{200}\right) \geq \frac{1}{m\|P_{:i}\|_\infty} \exp\left(\frac{\epsilon\alpha T}{10}\right).$$

If $T \geq \frac{100 \ln n \ln \frac{m}{\epsilon}}{\epsilon^2} \geq \frac{100 \ln \frac{m}{\epsilon}}{\epsilon\alpha}$, we have $x_i^{(T)} \gg \frac{10 \ln n}{\epsilon \|P_{:i}\|_\infty}$. So the algorithm must have terminated since $\max Px \gg \frac{10 \ln n}{\epsilon}$. □

Lemma 4.2.7 and Lemma 4.2.9 bound the total number of iterations in a phase by $\tilde{O}(\frac{1}{\epsilon^2})$, together with the bound on the number of phases, which is $\tilde{O}(\frac{1}{\epsilon})$, we guarantee the total number of iterations in Algorithm 5 is $\tilde{O}(\frac{1}{\epsilon^3})$ if the LP in (4.2) is feasible.

Theorem 4.2.10. *Algorithm 5 solves the $(1+\epsilon)$-feasibility problem correctly. It runs in parallel time $\tilde{O}(1/\epsilon^3)$ with the total work $\tilde{O}(N/\epsilon^3)$, where N is the number of non-zero entries in the constraint matrix.*

Proof. The correctness and convergence follows from the lemmas in the prior sections. We only need to look at the running time and total work.

At each iteration, we compute all updated values in $O(\log N)$ parallel time. Since the total number of iterations is $\tilde{O}(\frac{1}{\epsilon^3})$, the algorithm terminates in parallel time $\tilde{O}(\frac{1}{\epsilon^3})$.

To see the total work, consider the following implementation. For each $i \in [m]$, we maintain $P_{ji}x_i$ if $P_{ij} \neq 0$; similarly we maintain $C_{ji}x_i$ if $C_{ij} \neq 0$. Besides, we maintain the values of

$y, z, P^T y, C^T z, \vec{1}^T y$ and $\vec{1}^T z$. When we update x_i, we update these values accordingly, with work proportional to the number of non-zero entries in the ith column of the constraint matrix. For each fixed variable x_i, the total time of updates is at most $\tilde{O}(\frac{1}{\epsilon^2})$. Thus, the work on this part is $\tilde{O}(\frac{N}{\epsilon^2})$. Additionally, we need to compute the a_i, b_i for all variables at the beginning of each iteration to determine which variables to update, this takes $\tilde{O}(N)$ work each iteration, so the total work is $\tilde{O}(\frac{N}{\epsilon^3})$.

We see the majority of the work is actually on computing the gradients for the variables we may not update. We point out that we can implement the same lazy update as in [You14], which on a high level is just that if a variable has a large $\frac{a_i}{b_i}$ in an iteration, and is not updated, we don't recompute its gradients, until $\frac{\vec{1}^T y}{\vec{1}^T z}$ grows by more than a factor of $1 + \epsilon$. This can reduce the work to $\tilde{O}(\frac{N}{\epsilon^2})$, but requires a centralized step to control the phases. We omit the details as it is a straightforward adaptation. □

Pure Packing and Pure Covering LPs

We point out that in the case of pure packing or pure covering LPs, Algorithm 5 converges in $\tilde{O}(\frac{1}{\epsilon^2})$ iterations. This has the advantages over Algorithm 2, as Algorithm 5 is deterministic, and doesn't need centralized steps.

We will look at pure packing LPs, and the case for pure covering LPs will be symmetric.

Given packing LP in the optimization form

$$\max_{x \geq 0} \{\vec{1}^T x : Px \leq \vec{1}\}$$

via standard reduction and scaling, we need to solve a $(1 + \epsilon)$-feasibility problem the same as in the mixed packing and covering case specified in (4.1) and (4.2). In the case of pure packing, we will have $C = c\vec{1}^T$ for some constant c.

The correctness proof follows from the mixed case, and we discuss how we get faster convergence for pure packing.

The special structure of C greatly simplifies the convergence analysis, as now z is a scalar, and $b_i^{(t)} = c$ for all variables x_i across all iterations t. This allows us to aggregate the good iterations from all phases, and bound the total number of good phases by $\tilde{O}(\frac{1}{\epsilon^2})$, which will lead to $\tilde{O}(\frac{1}{\epsilon^2})$ total iterations.

In particular, now we look at all the good iterations across all phases together, and WLOG number them $1, \ldots, T$. Claim 4.2.8 still holds, as it doesn't rely on phases. Then we can prove a stronger version of Lemma 4.2.9

Lemma 4.2.11. *The total number of good iterations T is at most $O(\ln n \ln(\frac{m}{\epsilon})/\epsilon^2)$.*

The proof is a straightforward adaptation of the proof of Lemma 4.2.9. The proof is simpler, since now b_i is a constant across all iterations, so the property (4.6) we have on the good iterations directly put all the values on the same scale, so we can lower bound the total update on x_i across all good iterations.

4.3 Missing Proof

Lemma 4.2.3. *At each iteration t,*

$$\operatorname{lmax}(Px^{(t+1)}) \le \operatorname{lmax}(Px^{(t)}) + \alpha\langle a^{(t)}, (1+\Delta^{(t)})\cdot\Delta^{(t)}\cdot x^{(t)}\rangle$$

and

$$\operatorname{lmin}(C^{(t+1)}x^{(t+1)}) \ge \operatorname{lmin}(C^{(t)}x^{(t)}) + \alpha\langle b^{(t)}, (1-\Delta^{(t)})\cdot\Delta^{(t)}\cdot x^{(t)}\rangle,$$

where $\Delta\cdot x$ is the entry-wise product vector, i.e., $(\Delta\cdot x)_i = \Delta_i x_i$.

Proof. To simplify, we omit superscript (t) in the proof.

$$\begin{aligned}
\operatorname{lmax}(Px^{(t+1)}) &= \ln\sum_j \exp(P_j^T(x+\alpha\Delta\cdot x)) \\
&= \ln\sum_j \exp(P_j^T x)\cdot\exp(\alpha P_j^T(\Delta\cdot x)) \\
&\le \ln\sum_j \exp(P_j^T x)(1+\alpha P_j^T(\Delta\cdot x)+\alpha^2(P_j^T(\Delta\cdot x))^2).
\end{aligned}$$

Recall P_j^T is the j-th row of P (i.e. the j-th packing constraint). The last inequality is by Taylor expansion, with $\epsilon \le \Delta_i \le \frac{1}{2}$, $\alpha = \epsilon/10\log n$, and $P_j^T x \le 10\log n/\epsilon$ for all j, so $\alpha P_j^T(\Delta\cdot x) \le \frac{1}{2}$. We can control the second order term as follows

$$\begin{aligned}
\alpha^2(P_j^T(\Delta\cdot x))^2 &= \alpha^2\left(\sum_i P_{ji}(\Delta_i x_i)\right)^2 \\
&\le \alpha^2\left(\sum_i \Delta_i P_{ji}(\Delta_i x_i)\right)\left(\sum_i P_{ji}x_i\right) \\
&\le \alpha^2\left(\sum_i \Delta_i P_{ji}(\Delta_i x_i)\right)\frac{10\ln n}{\epsilon} \\
&= \alpha\left(\sum_i \Delta_i P_{ji}(\Delta_i x_i)\right) = \alpha P_j^T(\Delta\cdot\Delta\cdot x).
\end{aligned}$$

The first inequality is by the Cauchy-Schwarz inequality

$$\langle u, v\rangle^2 \le \|u\|^2\|v\|^2$$

, with $u_i = \sqrt{\Delta_i P_{ji}(\Delta_i x_i)}$ and $v_i = \sqrt{P_{ji}x_i}$. The second inequality is due to $P_j^T x \le \frac{10\ln n}{\epsilon}$ for all j. The last line is by our choice of $\alpha = \frac{\epsilon}{10\ln n}$.

So far we have bounded the impact of the updates on each individual constraint, and we have

$$\operatorname{lmax}(Px^{(t+1)}) \le \ln\sum_j \exp(P_j^T x)(1+\alpha P_j^T(\Delta\cdot x)+\alpha P_j^T(\Delta\cdot\Delta\cdot x)).$$

We then translate the changes on each constraint to the combined change on $\operatorname{lmax}(Px)$. Intuitively, the combined change is a convex combination on the changes of each constraint, weighted by their exponential values $y_j = \exp(P_j^T x)$.

$$\begin{aligned}
&\ln \sum_j \exp(P_j^T x)(1 + \alpha P_j^T(\Delta \cdot x) + \alpha P_j^T(\Delta \cdot \Delta \cdot x)) \\
=&\ln \sum_j y_j(1 + \alpha P_j^T(\Delta \cdot x) + \alpha P_j^T(\Delta \cdot \Delta \cdot x)) \\
=&\ln \left((\vec{1}^T y) \left(1 + \sum_j \frac{y_j}{\vec{1}^T y} \left(\alpha P_j^T(\Delta \cdot x) + \alpha P_j^T(\Delta \cdot \Delta \cdot x) \right) \right) \right) \\
=&\ln \left((\vec{1}^T y) \left(1 + \alpha \langle \frac{y}{\vec{1}^T y}, P((\vec{1} + \Delta) \cdot \Delta \cdot x) \rangle \right) \right).
\end{aligned}$$

We can then write out the change of $\operatorname{lmax}(Px)$ explicitly as

$$\begin{aligned}
\operatorname{lmax}(Px^{(t+1)}) \leq& \ln \left((\vec{1}^T y) \left(1 + \alpha \langle \frac{y}{\vec{1}^T y}, P((\vec{1} + \Delta) \cdot \Delta \cdot x) \rangle \right) \right) \\
=& \ln \left((\vec{1}^T y) \left(1 + \alpha \langle \frac{P^T y}{\vec{1}^T y}, (\vec{1} + \Delta) \cdot \Delta \cdot x \rangle \right) \right) \\
=& \ln \left((\vec{1}^T y) \left(1 + \alpha \langle a, (\vec{1} + \Delta) \cdot \Delta \cdot x \rangle \right) \right) \\
=& \operatorname{lmax}(Px) + \ln \left(1 + \alpha \langle a, (\vec{1} + \Delta) \cdot \Delta \cdot x \rangle \right) \\
\leq& \operatorname{lmax}(Px) + \alpha \langle a, (1 + \Delta) \cdot \Delta \cdot x \rangle.
\end{aligned}$$

Recall $a = \frac{P^T y}{\vec{1}^T y}$ as defined in (9), and the last line is by $\ln(1 + x) \leq x$ for $x \geq 0$.

For the $\operatorname{lmin}(C^{(t)}x)$ part, we follow the same approach.

$$-\operatorname{lmin}(C^{(t+1)}x^{(t+1)}) = \ln \sum_k \exp(-C^{(t+1)}x^{(t+1)})_k,$$

since $C^{(t+1)}$ can only have same or more rows dropped from $C^{(t)}$ due to the overly satisfied constraints, we know $\ln \sum_k \exp(-C^{(t+1)}x^{(t+1)})_k \leq \ln \sum_k \exp(-C^{(t)}x^{(t+1)})_k$. Again omitting the superscript (t), we have

$$\begin{aligned}
-\operatorname{lmin}(C^{(t+1)}x^{(t+1)}) \leq& \ln \sum_k \exp(-C(x + \alpha\Delta \cdot x))_k \\
=& \ln \sum_k \exp(-Cx)_k \exp(-\alpha C(\Delta \cdot x))_k \\
\leq& \ln \sum_k \exp(-Cx)_k (1 - \alpha C(\Delta \cdot x) + \alpha^2 (C(\Delta \cdot x))^2)_k.
\end{aligned}$$

The last inequality is due to $(\alpha C(\Delta \cdot x))_k \leq \frac{1}{2}$ for all k, since in $C^{(t)}$ we only keep those constraints that are not above $\frac{10 \ln n}{\epsilon}$ yet, i.e. $(Cx)_k \leq \frac{1}{\alpha}$ for all k. Again using Cauchy-Schwarz inequality as before and derivations similar to the $\operatorname{lmax}(Px)$ case, we get

$$-\operatorname{lmin}(C^{(t+1)}x^{(t+1)}) \leq -\operatorname{lmin}(C^{(t)}x^{(t)}) - \alpha\langle b, (1-\Delta) \cdot \Delta \cdot x\rangle.$$

□

Bibliography

[AHK12] Sanjeev Arora, Elad Hazan, and Satyen Kale. "The Multiplicative Weights Update Method: a Meta-Algorithm and Applications". In: *Theory of Computing* 8.6 (2012), pp. 121–164.

[AM95] Pierluigi Amodio and Francesca Mazzia. "A Parallel Gauss-Seidel Method for Block Tridiagonal Linear Systems". In: *SIAM J. Sci. Comput.* 16.6 (Nov. 1995), pp. 1451–1461.

[AZO15a] Zeyuan Allen-Zhu and Lorenzo Orecchia. "Nearly-Linear Time Positive LP Solver with Faster Convergence Rate". In: *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing*. STOC '15. Newer version available at `http://arxiv.org/abs/1411.1124`. 2015, pp. 229–236.

[AZO15b] 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 Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*. SODA '15. Full version with title "Using Optimization to Solve Positive LPs Faster in Parallel" available at `http://arxiv.org/abs/1407.1925`. 2015, pp. 1439–1456.

[BBR04] Yair Bartal, John W. Byers, and Danny Raz. "Fast, Distributed Approximation Algorithms for Positive Linear Programming with Applications to Flow Control". In: *SIAM J. Comput.* 33.6 (2004), pp. 1261–1279.

[BBR97] Yair Bartal, John W. Byers, and Danny Raz. "Global Optimization Using Local Information with Applications to Flow Control". In: *38th Annual Symposium on Foundations of Computer Science, FOCS '97, Miami Beach, Florida, USA, October 19-22, 1997*. 1997, pp. 303–312.

[BI04] D. Bienstock and G. Iyengar. "Faster Approximation Algorithms for Packing and Covering Problems". In: (2004). Preliminary version appeared in the proceeding of STOC 04.

[Bra+11] Joseph K. Bradley et al. "Parallel Coordinate Descent for L1-Regularized Loss Minimization". In: *Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011*. 2011, pp. 321–328.

[BT91] Dimitri P. Bertsekas and John N. Tsitsiklis. “Some aspects of parallel and distributed iterative algorithms - A survey› ”. In: *Automatica* 27.1 (1991), pp. 3–21.

[FR13] Olivier Fercoq and Peter Richtárik. “Accelerated, Parallel and Proximal Coordinate Descent”. In: *CoRR* abs/1312.5799 (2013).

[KY14] Christos Koufogiannakis and Neal E. Young. “A Nearly Linear-Time PTAS for Explicit Fractional Packing and Covering Linear Programs”. In: *Algorithmica* 70.4 (2014), pp. 648–674.

[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, May 16-18, 1993, San Diego, CA, USA*. 1993, pp. 448–457.

[LS13] Yin Tat Lee and Aaron Sidford. “Efficient Accelerated Coordinate Descent Methods and Faster Algorithms for Solving Linear Systems”. In: *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*. 2013, pp. 147–156.

[Mah+16] Michael W. Mahoney et al. “Approximating the Solution to Mixed Packing and Covering LPs in Parallel $\tilde{O}(\epsilon^{-3})$ Time”. In: *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*. 2016, 52:1–52:14.

[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”. In: *SIAM Journal on Optimization* 15.1 (2004), pp. 229–251.

[Nes05] Yurii Nesterov. “Smooth minimization of non-smooth functions”. In: *Math. Program.* 103.1 (2005), pp. 127–152.

[Nes12] Yurii Nesterov. “Efficiency of Coordinate Descent Methods on Huge-Scale Optimization Problems”. In: *SIAM Journal on Optimization* 22.2 (2012), pp. 341–362.

[PST91] Serge A. Plotkin, David B. Shmoys, and Éva Tardos. “Fast Approximation Algorithms for Fractional Packing and Covering Problems”. In: *32nd Annual Symposium on Foundations of Computer Science, San Juan, Puerto Rico, 1-4 October 1991*. 1991, pp. 495–504.

[PT12] Richard Peng and Kanat Tangwongsan. “Faster and simpler width-independent parallel algorithms for positive semidefinite programming”. In: *Proceedinbgs of the 24th ACM symposium on Parallelism in algorithms and architectures*. SPAA ’12. Available at http://arxiv.org/abs/1201.5135. Pittsburgh, Pennsylvania, USA, 2012, pp. 101–108.

[RT12] Peter Richtárik and Martin Takác. “Parallel Coordinate Descent Methods for Big Data Optimization”. In: *CoRR* abs/1212.0873 (2012).

[RT14] Peter Richtárik and Martin Takác. "Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function". In: *Math. Program.* 144.1-2 (2014), pp. 1–38.

[Saa03] Yousef Saad. *Iterative methods for sparse linear systems*. Siam, 2003.

[TY09] Paul Tseng and Sangwoon Yun. "A coordinate gradient descent method for nonsmooth separable minimization". In: *Math. Program.* 117.1-2 (2009), pp. 387–423.

[Wan+15] Di Wang et al. "Faster Parallel Solver for Positive Linear Programs via Dynamically-Bucketed Selective Coordinate Descent". In: *CoRR* abs/1511.06468 (2015).

[Wri15] Stephen J. Wright. "Coordinate descent algorithms". In: *Math. Program.* 151.1 (2015), pp. 3–34.

[WRM16] Di Wang, Satish Rao, and Michael W. Mahoney. "Unified Acceleration Method for Packing and Covering Problems via Diameter Reduction". In: *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*. 2016, 50:1–50:13.

[You01] Neal E. Young. "Sequential and Parallel Algorithms for Mixed Packing and Covering". In: *42nd Annual Symposium on Foundations of Computer Science, FOCS 2001, 14-17 October 2001, Las Vegas, Nevada, USA*. 2001, pp. 538–546.

[You14] Neal E. Young. "Nearly Linear-Time Approximation Schemes for Mixed Packing/-Covering and Facility-Location Linear Programs". In: *CoRR* abs/1407.3015 (2014).

[ZLO15] Zeyuan Allen Zhu, Yin Tat Lee, and Lorenzo Orecchia. "Using Optimization to Obtain a Width-Independent, Parallel, Simpler, and Faster Positive SDP Solver". In: *CoRR* abs/1507.02259 (2015).

[ZO14] Zeyuan Allen Zhu and Lorenzo Orecchia. "Linear Coupling: An Ultimate Unification of Gradient and Mirror Descent". In: *CoRR* abs/1407.1537 (2014).