Does this simple coordinate descent solve linear programming in $O(M^3L)$ operations ?

Adrien CHAN-HON-TONG

December 19, 2021

Abstract

This document presents an algorithm for linear programming whose complexity seems to be $O(M^3L)$ with a main loop which is just a simple coordinate descent. This results, which should be validated, is interesting for linear program.

1 Introduction

1.1 Background

Linear programming which consists to solve $\min_{x \ / \ Ax \geq b} c^T x$ for $A \in \mathbb{Z}^{M \times N}$ a matrix, $b, c \in \mathbb{Z}^M \times \mathbb{Z}^N$ two vectors is a central optimization problem.

Today, state of the art algorithms is based on interior point and has almost not changed since path-following algorithm [6] which solves linear programs with $O(M)$ variables and constraints, and L total binary size in less than $\widetilde{O}(\sqrt{M}L)$ Newton steps. Each Newton step is mainly the resolution of a $M \times M$ linear system which can be done $\widetilde{O}(M^\omega \sqrt{M}L)$ where ω is the coefficient of matrix multiplication (3 with simple algorithm but 2.38 with [1]). There exists faster randomized algorithm like [2] which are not in the scope of this paper.

This paper presents an algorithm which is given[1] to have a complexity of $\widetilde{O}(M^3L + M^{\omega+1})$. Currently, $3 \geq \omega + 0.5$ with efficient matrix multiplication algorithms, so the offered complexity is not better than the state of the art. But, if one has to rely on matrix multiplication with $\omega > 2.5$ (for example for having simpler formal verification on less complex algorithm), then, the offered algorithm is better than the state of the art.

1.2 Contribution

As linear programming is equivalent to linear feasibility: given $A \in \mathbb{Z}^{M \times N}$ such that $X_A = \{\chi, A\chi > \mathbf{0}\} \neq \emptyset$, linear feasibility consists to find $x \in X_A$ (see hal-02399129v14 and/or hal-02491694v11). The offered algorithm will deal with this

[1] Due to the novelty of this result, a deeper validation should be performed !

shape. This way, this algorithm is similar to [3, 5]: slightly better than [5] and deterministic contrary to [3].

Precisely:

- The algorithm solve a linear feasibility instance (finding $x \in X_A$) by solving (at most) M linear weak feasibility instance. Weak feasibility consists to solve x such that $Ax \geq \mathbf{0}$ (with $x \neq \mathbf{0}$) with $Ker(A) = \{\mathbf{0}\}$ and the same hypothesis than in linear feasibility.
- This last problem is solved by finding x, $Ax \geq -\epsilon\mathbf{1}$. Indeed, if $A \in \mathbb{Z}^{M\times M}$, $b \in \mathbb{Z}^M$, one could consider $\min_{x,t,Ax+t\geq b,t\geq 0} t$. Then, Cramer rules implies that $t^* = \frac{Det(A_R)}{Det(A_S)}$. So either $t^* = 0$ or $t^* \geq \frac{1}{Det(A_S)}$. Thus, if one find χ, τ such that $A\chi + \tau \geq b, 0 \leq \tau \leq 2^{-O(L)}$, then, one could be sure that $t^* \leq \tau$ i.e. $t^* = 0$.

2 Algorithm

2.1 Framework

Let $\mathcal{A} \in \mathbb{Z}^{M\times N}$ such that $X_{\mathcal{A}} = \{\chi, A\chi > \mathbf{0}\} \neq \emptyset$ and $Ker(\mathcal{A}) = \{\mathbf{0}\}$ and total binary size L. Thus, there exists $\chi, A\chi \geq \mathbf{1}$ and $\log(\chi^T\chi) = \widetilde{O}(L)$ (due to Cramer rule on χ + Hadamard bound on the determinant from the Cramer rule).

Let A the matrix with $A_m = \sqrt{\frac{1}{\mathcal{A}_m\mathcal{A}_m^T}}\mathcal{A}_m$. Let, $\Gamma = \max_m \mathcal{A}_m\mathcal{A}_m^T$ then, it holds that $A \times (\sqrt{\Gamma}\chi) \geq \mathbf{1}$. So there also exists x such that $Ax \geq \mathbf{1}$ and $\log(x^Tx) = \widetilde{O}(L + \log(\frac{1}{\varepsilon}))$. Then, let $F_A(v) = \frac{1}{2}\Upsilon v^T AA^T v - \sum_m \log(v_m)$

Seeing that $\forall v \geq \mathbf{0}$, $1^T v \leq (Ax)^T v = x^T A^T v \leq \sqrt{x^Tx \times v^T AA^T v}$, it holds that F is bounded with $F^* \geq M\log(x^Tx) = \widetilde{O}(M(L + \log(\frac{1}{\epsilon})))$. see hal-02399129v14 and/or hal-02491694v11 for a more complete proof.

Independently, one could consider an initial point $v_{start} = \frac{1}{\sqrt{\Upsilon}}\mathbf{1}$ which leads to $F(v_{start}) \leq M\log(\Upsilon) + M$

Thus, if one is able to decrease F by a constant value ($O(1)$) using a block of operations under some conditions, then, those conditions can not be meet $\widetilde{O}(M(L + \log(\Upsilon)))$ times successively.

hal-02399129v14 and/or hal-02491694v11 offer to rely on Newton descent to perform this minimization, leading to a $\widetilde{O}(M^\omega ML)$ algorithm. However, this paper offers a clever way.

2.2 Coordinate descent

Let $f_k(v_k) = v_k \to F(v)$ a 1 variable self concordant function from $]0, \infty[$ to $\mathbb{R}$.

Thus, results from self concordant theory holds (see for example [4]). In particular, if $\frac{f_k'^2(v_k)}{f_k''(v_k)} \geq \frac{1}{4}$, then, $f_k\left(v_k - \frac{1}{1+\frac{f_k'^2(v_k)}{f_k''(v_k)}}\frac{f_k'(v_k)}{f_k''(v_k)}\right) \leq f_k(v_k) - \frac{1}{50}$.

Yet, $f_k'(v_k) = \Upsilon A_k A^T v - \frac{1}{v_k}$ and $f_k''(v_k) = \Upsilon A_k A_k^T + \frac{1}{v_k^2} = \Upsilon + \frac{1}{v_k^2}$.

Now, if $A_k A^T v \leq -\frac{1}{\sqrt{\Upsilon}}$, then the 2 terms of $f'_k(v_k)$ are negative, and, thus $f'_k(v_k) = -(\Upsilon |A_k A^T v| + \frac{1}{v_k}) \leq -(\sqrt{\Upsilon} + \frac{1}{v_k})$.

In particular, $\frac{f'^2_k(v_k)}{f''_k(v_k)} \geq \frac{(\sqrt{\Upsilon}+\frac{1}{v_k})^2}{\Upsilon+\frac{1}{v_k^2}} \geq 1$.

So, as long as, there is $k \in \{1, ..., M\}$ such that $A_m A^T v \leq --\frac{1}{\sqrt{\Upsilon}}$, then, it is possible with a simple 1D step on k to decrease F by a constant value. But $F(v_{start}) - F^* \leq \widetilde{O}(ML + M\log(\Upsilon))$. So, this process can not last more than $ML + M\log(\Upsilon)$ steps !

Theorem 1:
If $A \in \mathbb{Z}^{M \times N}$ is a matrix with total binary size L and such that $X_A \neq \emptyset$, then, it is possible to find x such that $Ax \geq -\frac{1}{\sqrt{\Upsilon}}$ in $\widetilde{O}(M(L + \log(\Upsilon)))$ steps.

2.3 Purification

Now, using $\Upsilon = 2^{\widetilde{O}(L)}$, one can extract from x a vector such that $Ax \geq \mathbf{0}$ with M matrix inversion (i.e. $\widetilde{O}(M^{\omega+1})$. Importantly, this term does not depend on L ! Thus, using simple matrix multiplication i.e. $\omega = 3$ is not an issue from complexity point of view.

Now, by using simple pre processing to ensure that $Ker(A) = \{\mathbf{0}\}$, one can thus produce χ such that $A\chi \geq \mathbf{0},\ A\chi \neq \mathbf{0}$.

Finally, it is then possible to restart the process with $A_{\{m, A_m\chi=0\}}$ and this process converges as $\{m, A_m\chi = 0\} \neq \{1, ..., M\}$.

Thus, one is able to solve linear program by tackling linear feasibility and more precisely, with $\widetilde{O}(M)$ weak linear feasibility problems - each of them can be solved in $\widetilde{O}(ML)$ 1D coordinate descent as presented in Theorem 1.

2.4 Implementation detail

Importantly, if done naively, each step of the coordinate descent cost M^2 (the product of a matrix vector). This may result in a $\widetilde{O}(M^4 L)$ algorithm not really competitive with state of the art.

Now, using the correct data structure, one is able to perform the 1D coordinate descent with only M operations, resulting in a $\widetilde{O}(M^3 L)$ total complexity. The underlying idea to speed up this coordinate descent is to precompute $A_i A_j^T$ and $A^T \mathbf{1}_k$ for all k.

This way, both, $A^T v$ and $AA^T v$ can be represented by M value, and, when performing $v_k \leftarrow v_k + \delta$, one could update those $2M$ values with

- $A_i A^T(v + \delta \mathbf{1}_k) = A_i A^T v + \delta A_i A_k^T$ i.e. $A_i A^T(v_k + \delta) += \delta A_i A_k^T$ with $A_i A_k^T$ being precomputed
- $A^T(v + \delta \mathbf{1}_k) = A^T v + \delta A^T \mathbf{1}_k$ with $A^T \mathbf{1}_k$ being precomputed

This way, updating both v, $A^T v$, and, $AA^T v$ can be done in $\widetilde{O}(M)$.

References

[1] Andris Ambainis, Yuval Filmus, and François Le Gall. Fast matrix multiplication: limitations of the coppersmith-winograd method. In *Proceedings of the forty-seventh annual ACM symposium on Theory of Computing*, pages 585–593, 2015.

[2] Michael B Cohen, Yin Tat Lee, and Zhao Song. Solving linear programs in the current matrix multiplication time. In *Proceedings of the 51st annual ACM SIGACT symposium on theory of computing*, 2019.

[3] John Dunagan and Santosh Vempala. A simple polynomial-time rescaling algorithm for solving linear programs. *Mathematical Programming*, 114(1):101–114, 2008.

[4] Arkadi Nemirovski. Interior point polynomial time methods in convex programming. *Lecture notes*, 42(16):3215–3224, 2004.

[5] Javier Peña and Negar Soheili. A deterministic rescaled perceptron algorithm. *Mathematical Programming*, 155(1-2):497–510, 2016.

[6] James Renegar. A polynomial-time algorithm, based on newton's method, for linear programming. *Mathematical programming*, 40(1):59–93, 1988.