Rescaled Coordinate Descent Methods for Linear Programming

Daniel Dadush[1], László A. Végh[2(✉)], and Giacomo Zambelli[2]

[1] Centrum Wiskunde & Informatica, Amsterdam, The Netherlands
dadush@cwi.nl
[2] London School of Economics, London, UK
{l.vegh,g.zambelli}@lse.ac.uk

Abstract. We propose two simple polynomial-time algorithms to find a positive solution to $Ax = 0$. Both algorithms iterate between coordinate descent steps similar to von Neumann's algorithm, and rescaling steps. In both cases, either the updating step leads to a substantial decrease in the norm, or we can infer that the condition measure is small and rescale in order to improve the geometry. We also show how the algorithms can be extended to find a solution of maximum support for the system $Ax = 0$, $x \geq 0$. *This is an extended abstract. The missing proofs will be provided in the full version.*

1 Introduction

Let $A = [a_1, \ldots, a_n]$ be an integral $m \times n$ matrix with rank m, and let L denote the encoding size of A. We propose two simple polynomial algorithms for the linear feasibility problem, that is, to find a solution to systems of the form

$$\begin{aligned} Ax &= 0 \\ x &> 0. \end{aligned} \tag{1}$$

Our main contributions are: *(i)* new simple iterative methods for (1) with guaranteed finite convergence, *(ii)* a new geometric potential for these systems together with a rescaling method for improving it.

The algorithms we propose fit into a line of research developed over the past 10 years [2–6,8,15,16,20], where simple iterative updates, such as variants of perceptron [17] or of the relaxation method [1,11], are combined with some form of rescaling in order to get polynomial time algorithms for linear programming.

While these methods are slower than current interior point methods, they nevertheless yield important insights into the structure of linear programs. In particular, rescaling methods provide geometric potentials associated with a linear system which quantify how "well-conditioned" the system is, together with rescaling procedures for improving these potentials. Importantly, these potentials often provide more fine grained measures of the complexity of solving the linear system than the encoding length of the data, and help identify interesting subclasses of LPs that can be solved in strongly polynomial time (see for example [5]).

Q. Louveaux and M. Skutella (Eds.): IPCO 2016, LNCS 9682, pp. 26–37, 2016.
DOI: 10.1007/978-3-319-33461-5_3

Additionally, our algorithms can be adapted to solve the more general *maximum support* problem: find a solution to $Ax = 0, x \geq 0$ whose support $\{j : x_j > 0\}$ is inclusionwise maximum. Geometrically, this means finding the face of the convex hull of the columns of A that contains 0 in its relative interior. While general LP feasibility (and thus LP optimization) can be reduced to (1) via standard perturbation methods (see for example [18]), this is not desirable for numerical stability. On the other hand, any algorithm for the maximum support problem can be used directly to test feasibility of a system of the form $Ax = b$, $x \geq 0$ via simple homogenziation. We note that it is an open problem to devise any polynomial method for solving the maximum support problem that does not depend directly on the bit complexity L, but only on purely geometric parameters.

Preliminaries. Throughout the paper, we denote $\mathcal{L} := \{x \in \mathbb{R}^n : Ax = 0\}$, $\mathcal{L}_+ := \mathcal{L} \cap \mathbb{R}^n_+$, $\mathcal{L}_> := \mathcal{L} \cap \mathbb{R}^n_>$. We will also let $\mathcal{L}^\perp$ denote the orthogonal complement of $\mathcal{L}$; clearly, $\mathcal{L}^\perp = \{z \in \mathbb{R}^n : \exists y \in \mathbb{R}^m,\ z = A^\mathsf{T} y\}$. Let $\mathcal{L}^\perp_+ := \mathcal{L}^\perp \cap \mathbb{R}^n_+$ and $\mathcal{L}^\perp_> := \mathcal{L}^\perp \cap \mathbb{R}^n_>$. Therefore (1) is the problem of finding a point in $\mathcal{L}_>$. By strong duality, (1) is feasible if and only if $\mathcal{L}^\perp_+ = \{0\}$, that is,

$$A^\mathsf{T} y \geq 0, \tag{2}$$

has no solution other than $y = 0$.

Denote by $\operatorname{supp}(\mathcal{L}_+) \subseteq [n]$ the maximum support of a point in $\mathcal{L}_+$. Obviously $\operatorname{supp}(\mathcal{L}_+) \cap \operatorname{supp}(\mathcal{L}^\perp_+) = \emptyset$, whereas the strong duality theorem implies that $\operatorname{supp}(\mathcal{L}_+) \cup \operatorname{supp}(\mathcal{L}^\perp_+) = [n]$.

For any vector $v \in \mathbb{R}^m$ we denote by $\hat{v}$ the normal vector in the direction of v, that is $\hat{v} := v/\|v\|$. We let $\hat{A} := [\hat{a}_1, \ldots, \hat{a}_n]$. Note that, given $v, w \in \mathbb{R}^m$, $\hat{v}^\mathsf{T}\hat{w}$ is the cosine of the angle between them. Let $\mathbb{B}^m = \{x \in \mathbb{R}^m : \|x\| \leq 1\}$ denote the m-dimensional Euclidean ball. Let $\boldsymbol{e}_j$ denote the jth unit vector an $\boldsymbol{e}$ denote the all-ones vector of appropriate dimension (depending on the context).

Coordinate Descent Algorithms. Various coordinate descent methods are known for finding non-zero points in $\mathcal{L}_+$ or $\mathcal{L}^\perp_+$. Most algorithms address either the $\operatorname{supp}(\mathcal{L}_+) = [n]$ or the $\operatorname{supp}(\mathcal{L}^\perp_+) = [n]$ case; here we outline the common update steps.

At every iteration, maintain a non-negative, non-zero vector $x \in \mathbb{R}^n$, and let $y = Ax$. If $y = 0$, then x is a non-zero point in $\mathcal{L}_+$. If $A^\mathsf{T} y > 0$, then $A^\mathsf{T} y \in \mathcal{L}^\perp_>$. Otherwise, choose an index $k \in [n]$ such that $a_k^\mathsf{T} y \leq 0$, and update x and y as follows:

$$y' := \alpha y + \beta \hat{a}_k; \quad x' := \alpha x + \frac{\beta}{\|a_k\|}\boldsymbol{e}_k, \tag{3}$$

where $\alpha, \beta > 0$ depend on the specific algorithm. Below we discuss various possible update choices. These can be seen as coordinate descent methods for minimizing $\|y\|^2$ subject to $y = Ax, x \geq 0$, and some further constraint is added, e.g. $\boldsymbol{e}^\mathsf{T} x = 1$ in the von Neumann algorithm.

An important quantity in the convergence analysis of the algorithms we will describe is (a slight variant) of the condition measure introduced by Goffin [10]:

$$\rho_A := \max_{\|y\|=1, y\in\mathbb{R}^m} \min_{j\in[n]} a_j^\mathsf{T} y \tag{4}$$

We will most often be concerned with the quantity $\rho_{\hat{A}}$, where the columns of A have been scaled to have norm 1, however both will be useful to us. While ρ_A and $\rho_{\hat{A}}$ can be very different, note that they always have the same sign.

Geometrically, $|\rho_A|$ is the distance of the origin from the boundary of $\operatorname{conv}(A)$, where $\rho_A > 0$ if and only if $\operatorname{supp}(\mathcal{L}_+^\perp) = [n]$ (in which case the origin is outside $\operatorname{conv}(A)$), $\rho_A < 0$ if and only if $\operatorname{supp}(\mathcal{L}_+) = [n]$ (in which case the origin is in the interior $\operatorname{conv}(\hat{A})$), and $\rho_A = 0$ otherwise. In particular, if $\rho_A < 0$, then $-\rho_A$ is the radius of the largest ball in $\mathbb{R}^n$ inscribed in $\operatorname{conv}(A)$ and centered at the origin. If $\rho_{\hat{A}} > 0$, then $\rho_{\hat{A}}$ is the width of the *dual cone* $\{y \in \mathbb{R}^m : A^\mathsf{T} y > 0\}$, that is, the radius of the largest ball in $\mathbb{R}^m$ inscribed in the dual cone and centered at a point at distance one from the origin.

von Neumann's algorithm maintains at every iteration the condition that y is a convex combination of $\hat{a}_1, \dots, \hat{a}_n$. The parameters $\alpha, \beta > 0$ are chosen so that $\alpha + \beta = 1$ and $\|y'\|$ is smallest possible. That is, y' is the point of minimum norm on the line segment joining y and $\hat{a}_k$. If we denote by y^t the vector at iteration t, a simple argument shows that $\|y^t\| \le 1/\sqrt{t}$ (see Dantzig [7]). If 0 is contained in the interior of the convex hull, that is $\rho_{\hat{A}} < 0$, Epelman and Freund [9] showed that $\|y^t\|$ decreases by a factor of $\sqrt{1-\rho_{\hat{A}}^2}$ in every iteration. Though the norm of y converges exponentially to 0, we note that this method may not actually terminate in finite time. If 0 is outside the convex hull however, that is, $\rho_{\hat{A}} > 0$, then the algorithm terminates after at most $1/\rho_{\hat{A}}^2$ iterations.

Betke [3] gave a polynomial time algorithm, based on a combinatorial variant of von Neumann's update, for the case $\operatorname{supp}(\mathcal{L}_+^\perp) = [n]$. Chubanov uses von Neumann's update on the columns of the projection matrix to $\mathcal{L}$, and is able to solve the maximum support problem in time $O(n^4 L)$.[1]

Perceptron chooses $\alpha = \beta = 1$ at every iteration. If $\rho_{\hat{A}} > 0$, then, similarly to the von Neumann algorithm, the perceptron algorithm terminates with a solution to the system $A^\mathsf{T} y > 0$ after at most $1/\rho_{\hat{A}}^2$ iterations (see Novikoff [13]). Peña and Soheili gave a smoothed variant of the perceptron update which guarantees termination in time $O(\sqrt{\log n}/\rho_{\hat{A}})$ [14], and showed how this gives rise to a polynomial-time algorithm [15] using the rescaling introduced by Betke in [3]. The same running time $O(\sqrt{\log n}/\rho_{\hat{A}})$ was achieved by Wei Yu et al. [21] by adapting the Mirror-Prox algorithm of Nemirovski [12].

Dunagan-Vempala [8] choose $\alpha = 1$ and $\beta = -(\hat{a}_k^\mathsf{T} y)$. The choice of β is the one that makes $\|y'\|$ the smallest possible when $\alpha = 1$. It can be readily computed that

[1] It had been suggested by Prof. C. Roos that Chubanov's algorithm could be further improved to $O(n^{3.5} L)$, but the paper was subsequently withdrawn due to a gap in the argument.

$$\|y'\| = \|y\|\sqrt{1-(\hat{a}_k^\mathsf{T}\hat{y})^2}. \tag{5}$$

In particular, the norm of y' decreases at every iteration, and the larger is the angle between a_k and y, the larger the decrease. If $\rho_{\hat{A}} < 0$, then $|\hat{a}_k^\mathsf{T}\hat{y}| \geq |\rho_{\hat{A}}|$, therefore this guarantees a decrease in the norm of at least $\sqrt{1-\rho_{\hat{A}}^2}$.

Our Algorithms. Both our algorithms use Dunagan-Vempala updates: Algorithm 1 on the columns of A, and Algorithm 2 on the orthogonal projection matrix $\varPi$ to the space $\mathcal{L}^\perp$. These iterations are performed as long as we obtain a substantial decrease in $\|y\|$. Otherwise, a rescaling is performed in order to improve a volumetric potential which serves as a proxy to the condition measure $|\rho_{\hat{A}}|$. The rescaling in Algorithm 1 is the same as in Dunagan-Vempala [8], even though they solve the dual problem of finding a point in $\mathcal{L}^\perp_>$. We will describe the differences after the description of the algorithm.

Our Algorithm 2 is inspired by the work of Chubanov [6], and it uses the same rescaling. Our algorithms are in some sense dual to each other however: Chubanov uses von Neumann updates on the projection matrix to $\mathcal{L}$ whereas we use Dunagan-Vempala on the projection $\varPi$ to $\mathcal{L}^\perp$. For the same algorithm, we provide two entirely different analyses, one similar to Chubanov's, and another volumetric one, as for Algorithm 1. Thus, while the rescaling is seemingly very different from the one used in Algorithm 1, there is indeed a similar underlying geometry. We compare our algorithm to Chubanov's at the end of Sect. 3.

The running time of our Algorithm 1 is $O((m^3n + n^2m)L)$, where as Algorithm 2 runs in $O(mn^4L)$ time. Although the second running time bound is worse, this algorithm can be extended to solve the full support problem within the same running time estimation. We note that Algorithm 1 could also be extended to solve the maximum support problem, but in a more indirect way, and at the expense of substantially increasing the running time.

2 Algorithm 1

Algorithm 1, described below, solves (1) (that is, finding a point in $\mathcal{L}_>$), using the Dunagan-Vempala (DV) update. It uses the parameters

$$\varepsilon := \frac{1}{11m}, \quad N := 6mL, \quad \delta := \min_{j\in[n]} \frac{1}{\|(AA^\mathsf{T})^{-1}a_j\|}. \tag{6}$$

It follows from (5) that, if in a given iteration there exists $k \in [n]$ such that $\hat{a}_k^\mathsf{T}\hat{y} \leq -\varepsilon$, then we obtain a substantial decrease in the norm, namely

$$\|y'\| \leq \|y\|\sqrt{1-\varepsilon^2}. \tag{7}$$

On the other hand, if $\hat{a}_j^\mathsf{T}\hat{y} \geq -\varepsilon$ for all $j \in [n]$, then it follows that $|\rho_{\hat{A}}| < \varepsilon$, that is, the condition measure is small. Our aim is to perform a geometric rescaling that improves the condition measure. As a proxy for $|\rho_{\hat{A}}|$, we use the volume of the polytope P_A defined by

$$P_A := \mathrm{conv}(\hat{A}) \cap (-\mathrm{conv}(\hat{A})). \tag{8}$$

Recall that $|\rho_{\hat{A}}|$ is the radius of the largest ball around the origin inscribed in P_A.

Algorithm 1

```
Input: A matrix $A \in \mathbb{Z}^{m\times n}$ with rank $m$.
Output: Either a solution to the system (1) or the statement that
    (1) is infeasible.
 1: Set $x_j := 1$ for all $j \in [n]$ and $y := Ax$. Set $t := 0$.
 2: while $\|y\| \geq \delta$ and $t \leq N$ do
 3:     if $A^\top y \geq 0$ then Terminate, return (1) is infeasible.
 4:     else
 5:         Let $k := \arg\min_{j\in[n]} \hat{a}_j^\top \hat{y}$;
 6:         if $\hat{a}_k^\top \hat{y} < -\varepsilon$ then
 7:             update $x := x - \frac{a_k^\top y}{\|a_k\|^2}\boldsymbol{e}_k$;   $y := y - (\hat{a}_k^\top y)\hat{a}_k$.
 8:         else
 9:             rescale $A := \left(I + \hat{y}\hat{y}^\top\right) A$;   $y := 2y$;   $t := t+1$;
10: if $\|y\| < \delta$ then return feasible solution $x - A^\top(AA^\top)^{-1}Ax$;
11: else return (1) is infeasible.
```

If $\hat{a}_j^\top \hat{y} \geq -\varepsilon$ for all $j \in [n]$, then P_A is contained in a "narrow strip" of width 2ε, namely $P_A \subseteq \{z \in \mathbb{R}^m : -\varepsilon \leq \hat{y}^\top \hat{z} \leq \varepsilon\}$. If we replace A with the matrix $A' := (I + \hat{y}\hat{y}^\top)A$, Lemma 2.2 shows that the volume of $P_{A'}$ is at least 3/2 times the volume of P_A. Geometrically, A' is obtained by applying to the columns of A the linear transformation that "stretches" them by a factor of two in the direction of $\hat{y}$ (see Fig. 1).

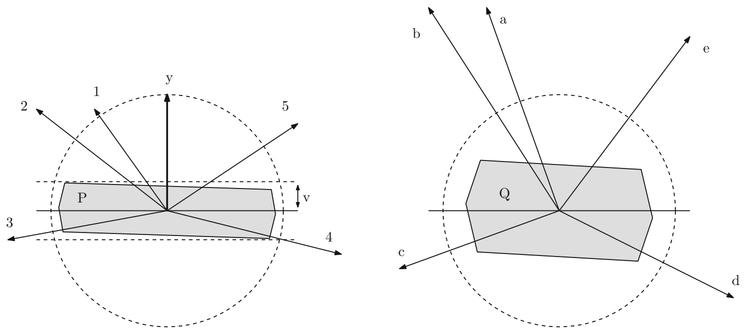

Fig. 1. Effect of rescaling. The dashed circle represent the set of points of norm 1. The shaded areas are P_A and $P_{A'}$.

Thus, at every iteration we either have a substantial decrease in the length of the current y, or we have a constant factor increase in the volume of P_A. Since the volume of P_A is bounded by the volume of the unit ball in $\mathbb{R}^m$, it follows that the algorithm cannot perform too many rescalings, unless (1) is infeasible.

After a polynomial number of iterations we either conclude that (1) is infeasible or we achieve a vector $y = Ax$ of tiny norm. In the latter case, it can be shown that projecting x to the kernel of A yields a positive kernel solution. We now state our main result.

Theorem 2.1. *For any input matrix $A \in \mathbb{Z}^{m\times n}$, Algorithm 1 returns a feasible solution x for (1) if and only if (1) is feasible. The total number of iterations of the while cycle is $O(m^3 L)$, and the total number of arithmetic operations performed is $O\left((m^3 n + mn^2)L\right)$.*

Relation to Previous Work. Even though our update step and rescaling are the same as the one used by Dunagan and Vempala [8], the algorithm and analysis are substantially different. In fact [8] assumes that $\mathrm{supp}(\mathcal{L}_+^\perp) = [n]$, and shows that the dual cone width $\rho_{\hat{A}}$ increases with a high probability. Their algorithm makes use of both perceptron as well as the DV update steps. The latter is always restarted from a random point y in the unit sphere (so in their algorithm y is not a conic combination of the a_i's). In contrast, our algorithm is fully deterministic, and uses the coordinate descent method in a more natural and direct way for the primal full dimensional case $\mathrm{supp}(\mathcal{L}_+) = [n]$.

An earlier volumetric rescaling for the $\mathrm{supp}(\mathcal{L}_+^\perp) = [n]$ case was introduced by Betke [3]. Given any convex combination x, $y = Ax$, $\|y\| \le 1/(\sqrt{m}n)$, Betke's rescaling shrinks each column of A in the direction of the a_i that has the largest coefficient x_i, i.e. $a_j \leftarrow a_j - 1/2(\hat{a}_i^\top a_j)\hat{a}_i$. This has the effect of increasing the volume of the intersection of the cone $A^\top z > 0$ with the unit Euclidean ball, which can be interpreted as a smooth proxy for $\rho_{\hat{A}}$. Here, one can view our potential as the natural primal counterpart to Betke's.

Convergent Coordinate Descent. Let us consider a modification of Algorithm 1 without any rescaling: first normalize the columns of A to have unit norm (i.e. set $A = \hat{A}$), initialize δ as above (with the normalized A) and x to $\boldsymbol{e}$, perform Dunagan-Vempala updates until $\|y\| \le \delta$, and terminate with the orthogonal projection of x onto $\mathcal{L}$. As we explain below, this yields a new "pure" coordinate descent method for (Algorithm 1) (perhaps with the exception of the rounding step) with finite convergence.

If (1) is feasible (that is, $\mathcal{L}_> \neq \emptyset$ and $\rho_{\hat{A}} < 0$), the above will terminate with a feasible solution in at most $O(\log(n/|\rho_{\hat{A}}|)/\rho_{\hat{A}}^2)$ iterations. To understand this, note that after normalizing A, the initial $y = A\boldsymbol{e}$ has norm at most n by the triangle inequality. Since the rate of norm decrease for the DV updates is still $\sqrt{1-\rho_{\hat{A}}^2}$, after $O(\log(n/|\rho_{\hat{A}}|)/\rho_{\hat{A}}^2)$ iterations the norm of y is less that $|\rho_{\hat{A}}|$. Termination then follows by $|\rho_{\hat{A}}| = |\rho_A| \le \delta$.

The modified algorithm just described can in fact be seen as the first coordinate descent method with termination bounded in terms of $|\rho_{\hat{A}}|$ and n for the

$\rho_{\hat{A}} < 0$ case. In comparison, perceptron and von Neumann may not even terminate in a finite amount of iterations in this setting. In contrast, for the case $\rho_{\hat{A}} > 0$ these algorithms converge to a feasible solution $\mathcal{L}_>^\perp$ in $O(1/\rho_{\hat{A}}^2)$ iterations, whereas our algorithm need not finitely converge. We note that Wolfe's algorithm [22], also implicitly used by Betke [3], is a simple finite method for arbitrary values of $\rho_{\hat{A}}$, including $\rho_{\hat{A}} = 0$. However, finiteness is due to the Simplex-like nature and there is no bound known on the number of iterations in terms of $|\rho_{\hat{A}}|$ if $\rho_{\hat{A}} \leq 0$.

Analysis. The crucial part of the analysis is to bound the volume increase of P_A at every rescaling iteration.

Lemma 2.2. *Assume* (1) *is feasible. For some* $0 < \varepsilon < 1/(11m)$*, let* $v \in \mathbb{R}^m$*,* $\|v\| = 1$*, such that* $\hat{a}_j^\top v \geq -\varepsilon\ \forall j \in [n]$*. Let* $A' = (I + vv^\top)A$*. Then* $\mathrm{vol}(P_{A'}) \geq \frac{3}{2}\mathrm{vol}(P_A)$*.*

We now sketch the proof of Theorem 2.1. It can be shown that if $\mathrm{conv}(A)$ contains the origin in its interior, then at the beginning P_A would contain a ball of radius at least 2^{-3L}. During the entire algorithm, P_A remains inside the unit ball. These facts, together with Lemma 2.2, imply that if the algorithm does not terminate within N rescalings, then $\mathrm{conv}(A)$ cannot contain the origin in its interior. If the algorithm terminates for the condition $\|y\| \leq \delta$, then one can show using the definition of δ that the returned solution is feasible.

To bound the number of iterations, note that by (7), $\|y\|^2$ decreases by a factor of $(1 - \varepsilon^2)$ every time we perform an update. Every time we perform a rescaling, $\|y\|^2$ increases by a factor of 4; however, this may only happen $N = O(mL)$ times. We terminate once $\|y\|^2 \leq \delta^2$. Combining these bounds gives a bound $O(m^3 L)$ on the total number of iterations. Every update can be computed in linear time. The number of rescalings is $O(mL)$, and each of them can be performed in $O(n^2)$ arithmetic operations, provided that we had previously computed $A^\top A$. Therefore the total number of arithmetic operations is $O((m^3 n + mn^2)L)$.

3 Algorithm 2: A Dual Chubanov Algorithm

Let $\Pi = A^\top(AA^\top)^{-1}A$ denote the orthogonal projection matrix to $\mathcal{L}^\perp$ (i.e., the space spanned by the rows of A), and let $\pi_1, \ldots, \pi_n$ denote the columns of Π and π_{ij} $(i, j \in [n])$ denote the (i, j) entry of Π. We recall the following well known properties of the projection matrix Π.

Proposition 3.1. *Let* $A \in \mathbb{R}^{m\times n}$ *and let* $\Pi = A^\top(AA^\top)^{-1}A$*. The following hold (i) For all* $x, z \in \mathbb{R}^n$*,* $\Pi x = 0$ *if and only if* $x \in \mathcal{L}$*, and* $\Pi z = z$ *if and only if* $z \in \mathcal{L}^\perp$*; (ii)* $\Pi^2 = \Pi$*; (iii) For every* $w \in \mathbb{R}^n$*,* $\|\Pi w\| \leq \|w\|$*; (iv) For all* $j \in [n]$*,* $\pi_j = \Pi \boldsymbol{e}_j$*, thus* $\|\pi_j\| \leq 1$*; (v)* $\pi_{jj} = \|\pi_j\|^2$ *for all* $j \in [n]$*; (vi)* $\mathrm{trace}(\Pi) = \sum_{j=1}^n \|\pi_j\|^2 = m$*.*

In Algorithm 2 below, we set $\varepsilon := \frac{1}{16n\sqrt{3m}}$. Throughout this section, for every $I \subseteq [n]$ we denote by D_I the diagonal matrix with $d_{jj} = 1/2$ if $j \in I$, $d_{jj} = 1$ if $j \notin I$. Thus $D_I = I - (1/2)\sum_{j\in I} \boldsymbol{e}_j \boldsymbol{e}_j^\mathsf{T}$.

Note that, since $z_j = \pi_j^\mathsf{T} z$ for all $j \in [n]$, the update step is just the Dunagan-Vempala update applied to the matrix Π instead of on A. Thus, at each update the norm of the current z decreases by at least a multiplicative factor $\sqrt{1-\varepsilon^2}$.

Observe also that at every iteration $w_j \geq 1$ for all $j \in [n]$, so in particular $\|z\| < 1$ immediately implies $w - z > 0$, thus the algorithm terminates with the solution $x := w - z$ if $\|z\| \leq 1$.

We give a proof of correctness of the algorithm. Afterwards, we provide a different analysis, reminiscent of Lemma 2.2, which relates the rescaling step to the change of a certain geometric quantity related to the condition measure of Π.

Algorithm 2

```
Input: A matrix $A \in \mathbb{Z}^{m\times n}$ with rank $m$.
Output: Either a solution $x \in \mathcal{L}_>$, or a set $R \subseteq [n]$ disjoint from the
    support of $\mathcal{L}_+$.
 1: Compute $\Pi = A^\mathsf{T}(AA^\mathsf{T})^{-1}A$. Set $D = I_n$.
 2: Set $w_j := 1$ for all $j \in [n]$, $z := \Pi w$, count$_j := 0$ for all $j \in [n]$.
 3: while count$_j < L$ for all $j \in [n]$ do
 4:     if $w - z > 0$ then Terminate, return $x := D^{-1}(w - z)$.
 5:     if $z \geq 0$ then Terminate, return $R := \{j \in [n] : z_j \neq 0\}$.
 6:     else
 7:         Let $i := \arg\min_{j\in[n]} \frac{z_j}{\|z\|\|\pi_j\|}$;
 8:         if $\frac{z_i}{\|z\|\|\pi_i\|} < -\varepsilon$ then
 9:             update $w := w - \frac{z_i \boldsymbol{e}_i}{\|\pi_i\|^2}$;    $z := z - \frac{z_i \pi_i}{\|\pi_i\|^2}$;
10:         else rescale
11:             Let $I := \{j \in [n] : \frac{z_j}{\|z\|} > \frac{1}{\sqrt{3n}}\}$;   $D := DD_I$;
12:             Recompute $\Pi = DA^\mathsf{T}(AD^2A^\mathsf{T})^{-1}AD$;
13:             Set $w_j := 1$ for all $j \in [n]$, $z := \Pi w$;
14:     count$_j :=$ count$_j + 1$ for all $j \in I$;
    return $R := \{j : \text{count}_j = L\}$.
```

Correctness of the Algorithm. For any $a \in \mathbb{R}$, we let $a^+ := \max\{0, a\}$ and $a^- = (-a)^+$. The correctness of the algorithm is based on the following simple bound due to Roos [16].

Lemma 3.2 (Roos). *Let $z \in \mathcal{L}^\perp$ and let $k \in [n]$ such that $z_k > 0$. Then, for every $x \in \mathcal{L} \cap [0,1]^n$.*

$$x_k \leq \frac{\sum_{j=1}^n z_j^-}{z_k}. \tag{9}$$

This bound justifies the rescaling in our algorithm, as stated in the following lemma.

Lemma 3.3. *Let A be the current matrix at a given iteration of Algorithm 2. Suppose that the current $z = \Pi w$ satisfies $z_j \geq -\varepsilon \|z\| \|\pi_j\|$. Then the set*

$$I = \left\{ j \in [n] \,:\, \frac{z_j}{\|z\|} > \frac{1}{\sqrt{3n}} \right\}$$

is nonempty. Furthermore, every $x \in \mathcal{L} \cap [0,1]^n$ satisfies $x_k \leq \frac{1}{2}$ for all $k \in I$.

Observe that rescaling has the effect of replacing the null space $\mathcal{L}$ of A with $D_I^{-1}\mathcal{L}$, that is, multiplying by 2 the components indexed by I of all vectors in $\mathcal{L}$. Let $\mathcal{L}^0$ be the null space of the input matrix A (i.e. before any rescaling). Lemmas 3.2 and 3.3 show that, at any iteration of the algorithm, $\mathcal{L}^0 \cap [0,1] \subseteq \{x \in \mathbb{R}^n \,:\, x_j < 2^{-\mathrm{count}_j}\}$. It is well-known (see for example Schrijver [18]) that, if $j \in [n]$ is in the support $Ax = 0$, $x \geq 0$, then there exists a solution with $x_j \geq 2^{-L}$. This shows that, whenever $\mathrm{count}_j = L$ for some $j \in [n]$, j cannot be in the support.

Running Time. At the beginning of the algorithm and after each rescaling, $z = \Pi \boldsymbol{e}$, therefore $\|z\| \leq \|\boldsymbol{e}\| = \sqrt{n}$. Every Dunagan-Vempala update decreases $\|z\|^2$ by a factor $1 - \varepsilon^2$, and the algorithm terminates with $x := w - z > 0$ when $\|z\| < 1$. This gives the strongly polynomial bound $O(n^2 m \log(n))$ on the number of iterations between any two rescaling. Since the algorithm performs at most L rescaling for every variable, and each update requires $O(n)$ operations, therefore the running-time of the algorithm is $O(n^4 m \log(n) L)$. (It should be noted here that the recomputation of the matrix Π at every rescaling can be performed in $O(|I|n^2)$ arithmetic operations using the Sherman-Morrison formula [19], therefore the total number of arithmetic operations performed during the rescalings is $O(n^3 L)$). Finally, one can improve the running time bound to $O(n^4 m L)$ by slightly modifying the algorithm, choosing the next w after each rescaling more carefully, using the following lemma.

Lemma 3.4. *Let $A \in \mathbb{R}^{m \times n}$, $\Pi = A^\top (AA^\top)^{-1} A$. Let $D > 0$ be an $n \times n$ diagonal matrix, and let $\Pi' = DA^\top (AD^2A^\top)^{-1} AD$. Given $z = \pi w$ for some $w \in \mathbb{R}^n$, if we let $w' = D^{-1} w$ and $z' = \Pi' w'$, then $\|z'\| \leq \left(\max_{i \in [n]} D_{ii}^{-1}\right) \|z\|$.*

The Maximum Support Problem. Algorithm 2 can be used to identify the support of $Ax = 0$, $x \geq 0$: whenever the algorithm returns a set R of indices not in the support, we set $x_j := 0$ for all $j \in R$, remove the columns of A indexed by R, and repeat. If the algorithm terminates with a feasible solution $x > 0$ for the current system, this defines a maximal support solution for the original problem. In the worst case, we need to run Algorithm 2 n times, giving a naïve running time estimate of $O(n^5 m L)$. However, observe that whenever Algorithm 2 terminates with a set R of indices, at the subsequent call to the algorithm we can initialize count_j, $j \notin R$, to the values computed at the end of the last call. Therefore, the total number of arithmetic operations needed to compute a maximum support solution is $O(n^4 m L)$, the same as the worst-case running time of Algorithm 2.

Analysis Based on a Geometric Potential. An alternative volumetric analysis can be given, similar to the one in Sect. 2. Let $Q_\Pi := \operatorname{conv}(\Pi) \cap \operatorname{conv}(-\Pi)$. Let us denote by $\widehat{\operatorname{vol}}(\cdot)$ the volume with respect to the measure induced on $\mathcal{L}^\perp$. We will consider as a potential $\widehat{\operatorname{vol}}(Q_\Pi)$.

Lemma 3.5. *Let $\varepsilon' = 1/(16\sqrt{3n}m)$. Let $z \in \mathcal{L}^\perp$ such that $z_j \geq -\varepsilon'\|z\|\|\pi_j\|$ for all $j \in [n]$. Let $I = \{j \in [n] : \frac{z_j}{\|z\|} > \frac{1}{\sqrt{3n}}\}$, and $\Pi' = D_I A^\mathsf{T}(AD_I^2A^\mathsf{T})^{-1}AD_I$. Then*

$$\widehat{\operatorname{vol}}(Q_{\Pi'}) \geq e^{1/8}\,\widehat{\operatorname{vol}}(Q_\Pi).$$

Since $\varepsilon \leq \varepsilon'$, it follows that when Algorithm 2 performs a rescaling, the current point $z = \Pi w$ satisfies the hypothesis of Lemma 3.5, thus after rescaling, $\widehat{\operatorname{vol}}(Q_\Pi)$ increases by a constant factor. As in Sect. 2, the total number of rescalings can be bounded by $O(mL)$. In particular, in $O(mL)$ rescalings one can either find a solution to $Ax = 0$, $x > 0$, or argue that none exists. Since $m \leq n$, this means that typically we may be able to prove that $Ax = 0$, $x > 0$ has no solution before we are actually able to identify any index j not in the support.

3.1 Refinements

Note that the two analyses we provided are somewhat “loose”, in the sense that the parameters in Algorithm 2 have been chosen to ensure that both analyses hold. Here we propose a few refinements and variants.

(a) To optimize the algorithm based on the potential $\widehat{\operatorname{vol}}(Q_\Pi)$, we can use $\varepsilon' = 1/(8\sqrt{n}m)$ instead of $\varepsilon = 1/(8\sqrt{m}n)$. This improves the total running time to $O(n^2m^3L)$.

(b) The analysis of the algorithm based on the argument in Sect. 3 can be simplified if we set $\bar{\varepsilon} = 1/(2\sqrt{mn})$, and do an update when the condition $z_i \leq -\bar{\varepsilon}\|\pi_i\|$ is satisfied by some $i \in [n]$ (rather then when $z_i \leq -\varepsilon\|z\|\|\pi_i\|$). This implies that the norm of $z' := z - (z_i/\|\pi_i\|^2)\pi_i$ satisfies $\|z'\|^2 \leq \|z\|^2(1-(\bar{\varepsilon}/\|z\|)^2) = \|z\|^2 - 1/(4mn)$. Since after each rescaling $\|z\| \leq \sqrt{n}$, this ensures that between every two rescalings there are at most $4mn^2$ updates (without the need of resorting to Lemma 3.4). When $z_j \geq -\bar{\varepsilon}\|\pi_j\|$ for every $j \in [n]$, it follows that there must be at least one $k \in [n]$ such that the bound in (9) is at most 1/2. Indeed, for any k such that $z_k \geq 1$ (one such k must exist because $w - z \not> 0$ and $w_j \geq 1$ for all $j \in [n]$) we have $(\sum_{j=1}^n z_j^-)/z_k \leq \varepsilon \sum_{j=1}^n \|\pi_j\| \leq \varepsilon\sqrt{nm} = 1/2$.

(c) A variant of the algorithm that gives the same running time but could potentially be more efficient in practice is the following. Define $\tilde{\varepsilon} = 1/(2\sqrt{n})$. At each iteration, let $N(z) := \{j : z_j < 0\}$, and compute $q := \sum_{j \in N(z)} \pi_j$. Note that $\|q\| \leq \sqrt{|N(z)|}$, since q is the projection onto $\mathcal{L}^\perp$ of the incidence vector of $N(z)$.

Instead of checking if there exists $i \in [n]$ such that $z_i \leq -\varepsilon\|z\|\|\pi_i\|$, check if $q^\mathsf{T}z \leq -\tilde{\varepsilon}\|q\|$. If such an index exists, then update as follows

$$z' := z - q\frac{q^\top z}{\|q\|^2}; \quad w' := w - \frac{q^\top z}{\|q\|^2}\sum_{j\in N(z)} \boldsymbol{e}_j.$$

It follows that $\|z'\|^2 \le \|z\|^2 - 1/(4n)$, hence the maximum number of updates between rescalings is $4n^2$. If instead $q^\top z > -\tilde{\varepsilon}\|q\|$, then for every $k \in [n]$ such that $z_k \ge 1$, we have $(\sum_{j=1}^n z_j^-)/z_k = (-q^\top z)/z_k \le \tilde{\varepsilon}\|q\| \le \tilde{\varepsilon}\sqrt{n} = \frac{1}{2}$.

Note that the total number of updates performed by the algorithm is $O(n^3 L)$, which is better than $O(mn^3L)$ updates performed by Algorithm 2. However, the number of arithmetic operations needed to compute q is, in the worst case, $O(n^2)$, therefore the total number of arithmetic operations is still $O(n^5L)$. Nevertheless, this variant may be better in practice because it provides faster convergence.

Comparison with Chubanov's Algorithm. Chubanov's algorithm works on the projection matrix $\bar{\Pi} = [\bar{\pi}_1, \ldots, \bar{\pi}_n]$ to the null space $\mathcal{L}$ of A, that is, $\bar{\Pi} = I - \Pi$. At every iteration, Chubanov maintains a vector $v \in \mathbb{R}^n_+$ such that $\boldsymbol{e}^\top v = 1$, starting from $y = \bar{\pi}_j$ for some $j \in [n]$, and computes $y = \bar{\Pi}v$. If $y > 0$, then Chubanov's algorithm terminates with $y \in \mathcal{L}_>$, else it selects an index $i \in [n]$ with $y_i \le 0$ and performs a von Neumann step $y' = \lambda y + (1-\lambda)\bar{\pi}_i$. By Dantzig's analysis of von Neumann's algorithm [7], $\|y'\|^{-2} \ge \|y\|^{-2} + 1$, hence after at most $4n^3$ operations $\|y\| \le 1/(2n\sqrt{n})$. Now, if $k = \arg\max_{j\in[n]} v_j$, then $v_k \ge 1/n$, therefore we have that for every $x \in \mathcal{L}_+ \cap [0,1]^n$, $x_k \le (v^\top x)/v_k = (y^\top x)/v_k \le (\|x\|\|y\|)/v_k \le \sqrt{n}\|y\|/v_k \le 1/2$. Thus, after at most $O(n^3)$ steps, Chubanov's algorithm performs the same rescaling as Algorithm 2 using $I := \{j \in [n] : \|y\|/v_k \le 1/(2\sqrt{n})\}$.

Note that, while the rescaling used by Algorithm 2 and Chubanov's algorithm are the same, and both algorithm ultimately produce a point in $\mathcal{L}_>$ if one exists, the updating steps work in the opposite direction. Indeed, both algorithms maintain a nonnegative vector in $\mathbb{R}^n$, but every von Neumann step in Chubanov's algorithm decreases the norm of the orthogonal projection of the nonnegative vector onto $\mathcal{L}$, whereas every Dunagan-Vempala update of Algorithm 2 decreases the norm of the orthogonal projection z onto $\mathcal{L}^\perp$. Also, Chubanov's iterations guarantee a fixed increase in $\|y\|^{-2}$, and rescaling occurs when $\|y\|$ is small enough, whereas Algorithm 2 terminates when $\|z\|$ is small enough (that is, when $\|z\| \le 1$), and rescaling occurs when the updating step would not produce a sufficient decrease in $\|z\|$.

We note that Chubanov's algorithm solves the maximum support problem in $O(n^4L)$, and hence is faster than ours. The full version of the paper will include an enhanced version of Algorithm 2 with running time bound $O(n^3mL)$.

References

1. Agmon, S.: The relaxation method for linear inequalities. Can. J. Math. **6**, 382–392 (1954)
2. Basu, A., De Loera, J., Junod, M.: On Chubanov's method for linear programming. INFORMS J. Comput. **26**(2), 336–350 (2014)

3. Betke, U.: Relaxation, new combinatorial and polynomial algorithms for the linear feasibility problem. Discrete Comput. Geom. **32**, 317–338 (2004)
4. Chubanov, S.: A strongly polynomial algorithm for linear systems having a binary solution. Math. Prog. **134**, 533–570 (2012)
5. Chubanov, S.: A polynomial algorithm for linear optimization which is strongly polynomial under certain conditions on optimal solutions (2015). http://www.optimization-online.org/DB_FILE/2014/12/4710.pdf
6. Chubanov, S.: A polynomial projection algorithm for linear programming. Math. Prog. **153**, 687–713 (2015)
7. Dantzig, G.B.: An ε-precise feasible solution to a linear program with a convexity constraint in $1/\varepsilon^2$ iterations independent of problem size, Report SOL 92–5, Stanford University (1992)
8. Dunagan, J., Vempala, S.: A simple polynomial-time rescaling algorithm for solving linear programs. Math. Prog. **114**, 101–114 (2006)
9. Epelman, M., Freund, R.M.: Condition number complexity of an elementary algorithm for computing a reliable solution of a conic linear system. Math. Prog. **88**(3), 451–485 (2000)
10. Goffin, J.: The relaxation method for solving systems of linear inequalities. Math. Oper. Res. **5**, 388–414 (1980)
11. Motzkin, T., Schoenberg, I.J.: The relaxation method for linear inequalities. Can. J. Math. **6**, 393–404 (1954)
12. Nemirovski, A.: Prox-method with rate of convergence $o(1/t)$ for variational inequalities with Lipschitz continuous monotone operators and smooth convex-concave saddle point problems. SIAM J. Optim. **15**, 229–251 (2004)
13. Novikoff, A.B.J.: On convergence proofs for perceptrons. In: Proceedings of the Symposium on the Mathematical Theory of Automata XII, pp. 615–622 (1962)
14. Soheili, N., Peña, J.: A smooth perceptron algorithm. SIAM J. Optim. **22**, 728–737 (2012)
15. Peña, J., Soheili, N.: A deterministic rescaled perceptron algorithm. Math. Prog. **155**(1), 497–510 (2016)
16. Roos, K.: On Chubanov's method for solving a homogeneous inequality system. Numer. Anal. Optim. **134**, 319–338 (2015)
17. Rosenblatt, F.: The perceptron: a probabilistic model for information storage and organization in the brain. Psychol. Rev. **65**, 386–408 (1958). Cornell Aeronautical Laboratory
18. Schrijver, A.: Theory of Linear and Integer Programming. Wiley, New York (1986)
19. Sherman, J., Morrison, W.J.: Adjustment of an inverse matrix corresponding to a change in one element of a given matrix. Ann. Math. Stat. **21**, 124–127 (1949)
20. Végh, L.A., Zambelli, G.: A polynomial projection-type algorithm for linear programming. Oper. Res. Lett. **42**, 91–96 (2014)
21. Yu, A.W., Kılınç-Karzan, F., Carbonell, J.: Saddle points and accelerated perceptron algorithms. In: Proceedings of the 31st International Conference on Machine Learning. Journal of Machine Learning Research **32**, 1827–1835 (2014)
22. Wolfe, P.: Finding the nearest point in a polytope. Math. Prog. **11**(1), 128–149 (1976)