Mathematical Programming
https://doi.org/10.1007/s10107-024-02077-0

FULL LENGTH PAPER

Series B

An update-and-stabilize framework for the minimum-norm-point problem

Satoru Fujishige[1] · Tomonari Kitahara[2] · László A. Végh[3]

Received: 14 August 2023 / Accepted: 23 February 2024

Abstract

We consider the minimum-norm-point (MNP) problem over polyhedra, a well-studied problem that encompasses linear programming. We present a general algorithmic framework that combines two fundamental approaches for this problem: active set methods and first order methods. Our algorithm performs first order update steps, followed by iterations that aim to 'stabilize' the current iterate with additional projections, i.e., find a locally optimal solution whilst keeping the current tight inequalities. Such steps have been previously used in active set methods for the nonnegative least squares (NNLS) problem. We bound on the number of iterations polynomially in the dimension and in the associated circuit imbalance measure. In particular, the algorithm is strongly polynomial for network flow instances. Classical NNLS algorithms such as the Lawson–Hanson algorithm are special instantiations of our framework; as a consequence, we obtain convergence bounds for these algorithms. Our preliminary computational experiments show promising practical performance.

Mathematics Subject Classification 90C05 · 90C25 · 65K05

An extended abstract of this paper has appeared in Proceedings of the 24th Conference on Integer Programming and Combinatorial Optimization, IPCO 2023.

✉ László A. Végh
L.Vegh@lse.ac.uk

Satoru Fujishige
fujishig@kurims.kyoto-u.ac.jp

Tomonari Kitahara
tomonari.kitahara@econ.kyushu-u.ac.jp

[1] Research Institute for Mathematical Sciences, Kyoto University, Kyoto 606-8502, Japan

[2] Faculty of Economics, Kyushu University, Fukuoka 819-0395, Japan

[3] Department of Mathematics, London School of Economics and Political Science, London WC2A 2AE, UK

Published online: 18 April 2024

1 Introduction

We study the minimum-norm-point (MNP) problem

$$\text{Minimize } \tfrac{1}{2}||Ax-b||^2 \text{ subject to } \mathbf{0} \le x \le u\,,\ x \in \mathbb{R}^N\,, \qquad \text{(P)}$$

where m and n are positive integers, $M = \{1, \cdots, m\}$ and $N = \{1, \cdots, n\}$, $A \in \mathbb{R}^{M\times N}$ is a matrix with rank $\mathrm{rk}(A) = m$, $b \in \mathbb{R}^M$, and $u \in (\mathbb{R} \cup \{\infty\})^N$. We will use the notation $\mathbf{B}(u) := \{x \in \mathbb{R}^N \mid \mathbf{0} \le x \le u\}$ for the feasible set. The problem (P) generalizes the linear programming (LP) feasibility problem: the optimum value is 0 if and only if $Ax = b$, $x \in \mathbf{B}(u)$ is feasible. The case $u(i) = \infty$ for all $i \in N$ is also known as the *nonnegative least squares (NNLS) problem*, a fundamental problem in numerical analysis.

Two extensively studied approaches for MNP and NNLS are *active set methods* and *first order methods*. An influential active set method was proposed by Lawson and Hanson [19, Chapter 23] in 1974. Variants of this algorithm were also proposed by Stoer [28], Björck [2], Wilhelmsen [31], and Leichner, Dantzig, and Davis [20].[1] Closely related is Wolfe's classical minimum-norm-point algorithm [32]. These are iterative methods that maintain a set of active variables fixed at the lower or upper bounds, and passive (inactive) variables. In the main update steps, these algorithms fix the active variables at the lower or upper bounds, and perform unconstrained optimization on the passive variables. Such update steps require solving systems of linear equations. In all these methods, the set of columns corresponding to passive variables is linearly independent. The combinatorial nature of these algorithms enables to show termination with an exact optimal solution in a finite number of iterations. However, obtaining subexponential convergence bounds for such active set algorithms has remained elusive; see Sect. 1.1 for more work on NNLS and Wolfe's algorithm.

In the context of first order methods, the formulation (P) belongs to a family of problems for which Necoara, Nesterov, and Glineur [22] showed linear convergence bounds. That is, the number of iterations needed to find an ε-approximate solution depends linearly on $\log(1/\varepsilon)$. Such convergence has been known for strongly convex functions, but this property does not hold for (P). However, [22] shows that restricted variants of strong convexity also suffice for linear convergence. For problems of the form (P), the required property follows using Hoffman-proximity bounds [16]; see [26] and the references therein for recent results on Hoffman-proximity. In contrast to active set methods, first order methods are computationally cheaper as they do not require solving systems of linear equations. On the other hand, they do not find exact solutions.

We propose a new algorithmic framework for the minimum-norm-point problem (P) that can be seen as a blend of active set and first order methods. Our algorithm performs *stabilizing steps* between first order updates, and terminates with an exact optimal solution in a finite number of iterations. Moreover, we show $\mathrm{poly}(n, \kappa)$ running time bounds for multiple instantiations of the framework, where κ is the *circuit*

[1] While there are minor differences in the details, these are essentially the same algorithm. Henceforth, we refer only to the Lawson–Hanson algorithm for simplicity.

imbalance measure associated with the matrix $(A \mid I_M)$ (see Sect. 2.1). This gives strongly polynomial bounds whenever κ is constant; in particular, $\kappa = 1$ for network flow feasibility. We note that if $A \in \mathbb{Z}^{M\times N}$, then $\kappa \le \Delta(A)$ for the maximum subdeterminant $\Delta(A)$. Still, κ can be exponential in the encoding length of the matrix.

The stabilizing step is similar to the one used by Björck [2] who considered the same formulation (P). The Lawson–Hanson algorithm for the NNLS problem can be seen as special instantiations of our framework, and we obtain an $O(n^2m^2\cdot\kappa^2\cdot\|A\|^2\cdot\log(n+\kappa))$ iteration bound. These algorithms only use coordinate updates as first order steps, and maintain linear independence of the columns corresponding to passive variables. Our framework is signficantly more general: we waive the linear independence requirement and allow for arbitarty active and passive sets. This provides much additional flexibility, as our framework can be implemented with a variety of first order methods. This feature also yields a significant advantage in our computational experiments.

Overview of the algorithm A key concept in our algorithm is the *centroid mapping*, defined as follows. For disjoint subsets $I_0, I_1 \subseteq N$, we let $\mathbf{L}(I_0, I_1)$ denote the affine subspace of $\mathbb{R}^N$ where $x(i) = 0$ for $i \in I_0$ and $x(i) = u(i)$ for $i \in I_1$. For $x \in \mathbf{B}(u)$, let $I_0(x)$ and $I_1(x)$ denote the subsets of coordinates i with $x(i) = 0$ and $x(i) = u(i)$, respectively. The *centroid mapping* $\Psi : \mathbf{B}(u) \to \mathbb{R}^N$ is a mapping with the property that $\Psi(x) \in \arg\min_y\{\frac{1}{2}\|Ay - b\|^2 \mid y \in \mathbf{L}(I_0(x), I_1(x))\}$. This mapping may not be unique, since the columns of A corresponding to $J(x) := \{i \in N \mid 0 < x(i) < u(i)\}$ may not be independent: the optimal *centroid set* is itself an affine subspace. The point $x \in \mathbf{B}(u)$ is *stable* if $\Psi(x) = x$. This generalizes an update used by Björck [2]. However, in his setting $J(x)$ is always linearly independent and thus the centroid set is always a single point. Stable sets can also be seen as the analogues of *corral* solutions in Wolfe's minimum-norm point algorithm.

Every major cycle starts with an update step and ends with a stable point. The update step could be any first-order step satisfying some natural requirements, such as variants of Frank–Wolfe, projected gradient, or coordinate updates. As long as the current iterate is not optimal, this update strictly improves the objective. Finite convergence follows by the fact that there can be at most 3^n stable points.

After the update step, we start a sequence of minor cycles. From the current iterate $x \in \mathbf{B}(u)$, we move to $\Psi(x)$ in case $\Psi(x) \in \mathbf{B}(u)$, or to the intersection of the boundary of $\mathbf{B}(u)$ and the line segment $[x, \Psi(x)]$ otherwise. The minor cycles finish once $x = \Psi(x)$ is a stable point. The objective $\frac{1}{2}\|Ax - b\|^2$ is decreasing in every minor cycle, and at least one new coordinate $i \in N$ is set to 0 or to $u(i)$. Thus, the number of minor cycles in any major cycle is at most n. One can use various centroid mappings satisfying a mild requirement on Ψ, described in Sect. 2.3.

We present a poly(n,κ) convergence analysis for the NNLS problem with coordinate updates, which corresponds to the Lawson–Hanson algorithm and its variants. We expect that similar arguments extend to the capacitated case. The proof has two key ingredients. First, we show linear convergence of the first-order update steps (Theorem 4.5). Such a bound follows already from [22]; we present a simple self-contained proof exploiting properties of stable points and the uncapacitated setting. The second step of the analysis shows that in every poly(n,κ) iterations, we can identify a new variable that will never become zero in subsequent iterations (Theorem 4.1). The proof relies on proximity arguments: we show that for any iterate x and any subsequent iterate x', the distance $\|x - x'\|$ can be upper bounded in terms of n, κ, and the optimality gap at x.

In Sect. 5, we present preliminary computational experiments using randomly generated problem instances of various sizes. We compare the performance of different variants of our algorithm to standard gradient methods. For the choice of update steps, projected gradient performs much better than coordinate updates used in the NNLS algorithms. We compare an 'oblivious' centroid mapping and one that chooses $\Psi(x)$ as the nearest point to x in the centroid set in the *'local norm'* (see Sect. 2.2). The latter one appears to be significantly better. For choices of parameters $n \geq 2m$, the running time of our method with projected gradient updates and local norm mapping is typically within a factor two of `TNT-NN`, the state-of-the-art practical active set heuristic for NNLS [21], despite the fact that we only use simple linear algebra tools and have not made any attempts for practical speed ups. The performance is often better than projected accelerated gradient descent, the best first order approach.

Proximity arguments and strongly polynomial algorithms Arguments that show strongly polynomial convergence by gradually revealing the support of an optimal solution are prevalent in combinatorial optimization. These date back to Tardos's [29] groundbreaking work giving the first strongly polynomial algorithm for minimum-cost flows. Our proof is closer to the dual 'abundant arc' arguments by Fujishige [12] and Orlin [24]. Tardos generalized the above result for general LP's, giving a running time dependence $\mathrm{poly}(n, \log \Delta(A))$, where $\Delta(A)$ is the largest subdeterminant of the constraint matrix. In particular, her algorithm is strongly polynomial as long as the entries of the matrix are polynomially bounded integers. This framework was recently strengthened in [7] to $\mathrm{poly}(n, \log \kappa(A))$ running time for the circuit imbalance measure $\kappa(A)$. They also highlight the role of Hoffman-proximity and give such a bound in terms of $\kappa(A)$. We note that the above algorithms—along with many other strongly polynomial algorithms in combinatorial optimization—modify the problem directly once new information is learned about the optimal support. In contrast, our algorithm does not require any such modifications, nor a knowledge or estimate on the condition number κ. Arguments about the optimal support only appear in the analysis.

Strongly polynomial algorithms with $\mathrm{poly}(n, \log \kappa(A))$ running time bounds can also be obtained using layered least squares interior point methods. This line of work was initiated by Vavasis and Ye [30] using a related condition measure $\bar{\chi}(A)$. An improved version that also established the relation between $\bar{\chi}(A)$ and $\kappa(A)$ was recently given by Dadush et al. [6]. We refer the reader to the survey [9] for properties and further applications of circuit imbalances.

1.1 Further related work

The Lawson–Hanson algorithm remains popular for the NNLS problem, and several variants are known. Bro and De Jong [3], and by Myre et al. [21] proposed empirically faster variants. In particular, [21] allows bigger changes in the active and passive sets, thus waiving the linear independence on passive variables, and reports a significant speedup. However, there are no theoretical results underpinning the performance of these heuristics.

Wolfe's minimum-norm-point algorithm [32] considers the variant of (P) where the box constraint $x \in \mathbf{B}(u)$ is replaced by $\sum_{i\in N} x_i = 1, x \geq \mathbf{0}$. It has been successfully employed as a subroutine in various optimization problems, e.g., submodular function minimization [14], see also [1, 11, 13]. Beyond the trivial 2^n bound, the convergence analysis remained elusive; the first bound with $1/\varepsilon$-dependence was given by Chakrabarty et al. [4] in 2014. Lacoste-Julien and Jaggi [17] gave a $\log(1/\varepsilon)$ bound, parametrized by the *pyramidal width* of the polyhedron. Recently, De Loera et al. [8] showed an example of exponential time behaviour of Wolfe's algorithm for the *min-norm insertion rule* (the analogue of a pivot rule); no exponential example for other insertion rules such as the *linopt* rule used in the application for submodular minimization.[2]

Our Update-and-Stabilize algorithm is also closely related to the Gradient Projection Method, see [5] and [23, Section 16.7]. This method also maintains a non-independent set of passive variables. For each gradient update, a more careful search is used in the gradient direction, 'bending' the movement direction whenever a constraint is hit. The analogues of stabilizer steps are conjugate gradient iterations. Thus, this method avoids the computationally expensive step of exact projections; on the other hand, finite termination is not guaranteed. We further discuss the relationship between the two algorithms in Sect. 6.

There are similarities between our algorithm and the Iteratively Reweighted Least Squares (IRLS) method that has been intensively studied since the 1960s [18, 25]. For some $p \in [0, \infty]$, $A \in \mathbb{R}^{M\times N}$ and $b \in \mathbb{R}^M$, the goal is to approximately solve $\min\{\|x\|_p \mid Ax = b\}$. At each iteration, a weighted minimum-norm point problem $\min\{\sum_{i=1}^n w_i^{(t)} x_i^2 \mid Ax = b\}$ is solved, where the weights $w^{(t)}$ are iteratively updated. The LP-feasibility problem $Ax = b$, $\mathbf{0} \leq x \leq \mathbf{1}$ for finite upper bounds $u = \mathbf{1}$ can be phrased as an ℓ_∞-minimization problem $\min\{\|x\|_\infty \mid Ax = b - A\mathbf{1}/2\}$. Ene and Vladu [10] gave an efficient variant of IRLS for ℓ_1 and ℓ_∞-minimization; see their paper for further references. Some variants of our algorithm solve a weighted least squares problem with changing weights in the stabilizing steps. There are, however significant differences between IRLS and our method. The underlying optimization problems are different, and IRLS does not find an exact optimal solution in finite time. Applied to LP in the ℓ_∞ formulation, IRLS satisfies $Ax = b$ throughout while violating the box constraints $\mathbf{0} \leq x \leq u$. In contrast, iterates of our algorithm violate $Ax = b$ but maintain $\mathbf{0} \leq x \leq u$. The role of the least squares subroutines is also rather different in the two settings.

2 Preliminaries

Notation We use $N \oplus M$ for disjoint union (or direct sum) of the copies of the two sets. For a matrix $A \in \mathbb{R}^{M\times N}$, $i \in M$ and $j \in N$, we denote the ith row of A by A_i and jth column by A^j. Also for any matrix X denote by $X^\top$ the matrix transpose

[2] The *linopt* rule corresponds to the coordinate updates in the terminology of this paper.

of X. We let $\|\cdot\|_p$ denote the ℓ_p vector norm; we use $\|\cdot\|$ to denote the Euclidean norm $\|\cdot\|_2$. For a matrix $A \in \mathbb{R}^{M\times N}$, we let $\|A\|$ denote the spectral norm, that is, the $\ell_2 \to \ell_2$ operator norm.

For any $x, y \in \mathbb{R}^M$ we define $\langle x, y\rangle = \sum_{i\in M} x(i)y(i)$. We will use this notation also in other dimensions. We let $[x, y] := \{\lambda x + (1-\lambda)y \mid \lambda \in [0, 1]\}$ denote the line segment between the vectors x and y.

2.1 Elementary vectors and circuits

For a linear space $W \subsetneq \mathbb{R}^N$, $g \in W$ is an *elementary vector* if g is a support minimal nonzero vector in W, that is, no $h \in W\setminus\{\mathbf{0}\}$ exists such that $\mathrm{supp}(h) \subsetneq \mathrm{supp}(g)$, where supp denotes the support of a vector. We let $\mathcal{F}(W) \subseteq W$ denote the set of elementary vectors. A *circuit* in W is the support of some elementary vector; these are precisely the circuits in the associated linear matroid $\mathcal{M}(W)$.

The subspaces $W = \{\mathbf{0}\}$ and $W = \mathbb{R}^N$ are called trivial subspaces; all other subspaces are nontrivial. We define the *circuit imbalance measure*

$$\kappa(W) := \max\left\{\left|\frac{g(j)}{g(i)}\right| \mid g \in \mathcal{F}(W), i, j \in \mathrm{supp}(g)\right\}$$

for nontrivial subspaces and $\kappa(W) = 1$ for trivial subspaces. For a matrix $A \in \mathbb{R}^{M\times N}$, we use the notation $\kappa(A)$ to denote $\kappa(\ker(A))$.

The following theorem shows the relation to totally unimodular (TU) matrices. Recall that a matrix is *totally unimodular (TU)* if the determinant of every square submatrix is 0, +1, or −1.

Theorem 2.1 [Cederbaum, 1957, see c 3,4]ENV22] *Let $W \subset \mathbb{R}^N$ be a linear subspace. Then $\kappa(W) = 1$ if and only if there exists a TU matrix $A \in \mathbb{R}^{M\times N}$ such that* $W = \ker(A)$.

We also note that if $A \in \mathbb{Z}^{M\times N}$ is an integer matrix, then $\kappa(A) \le \Delta(A)$ for the maximum subdeterminant $\Delta(A)$.

Conformal circuit decompositions We say that the vector $y \in \mathbb{R}^N$ *conforms to* $x \in \mathbb{R}^N$ if $x(i)y(i) > 0$ whenever $y(i) \neq 0$. Given a subspace $W \subseteq \mathbb{R}^N$, a *conformal circuit decomposition* of a vector $v \in W$ is a decomposition

$$v = \sum_{k=1}^{\ell} h^k,$$

where $\ell \le n$ and $h^1, h^2, \ldots, h^\ell \in \mathcal{F}(W)$ are elementary vectors that conform to v. A fundamental result on elementary vectors asserts the existence of a conformal circuit decomposition; see e.g., [15, 27]. Note that there may be multiple conformal circuit decompositions of a vector.

Lemma 2.2 *For every subspace $W \subseteq \mathbb{R}^N$, every $v \in W$ admits a conformal circuit decomposition.*

Given $A \in \mathbb{R}^{M\times N}$, we define the extended subspace $\mathcal{X}_A \subset \mathbb{R}^{N\oplus M}$ as $\mathcal{X}_A := \ker([A \mid -I_M])$. Hence, for every $v \in \mathbb{R}^N$, $(v, Av) \in \mathcal{X}_A$. For $v \in \mathbb{R}^N$, the *generalized path-circuit decomposition of v with respect to A* is a decomposition $v = \sum_{k=1}^{\ell} h^k$, where $\ell \le n$, and for each $1 \le k \le \ell$, $(h^k, Ah^k) \in \mathbb{R}^{N\oplus M}$ is an elementary vector in $\mathcal{X}_A$ that conforms to (v, Av). Moreover, h^k is an *inner vector* in the decomposition if $Ah^k = \mathbf{0}$ and an *outer vector* otherwise.

We say that $v \in \mathbb{R}^N$ is *cycle-free with respect to A*, if all generalized path-circuit decompositions of v contain outer vectors only. The following lemma will play a key role in analyzing our algorithms.

Lemma 2.3 *For any $A \in \mathbb{R}^{M\times N}$, let $v \in \mathbb{R}^N$ be cycle-free with respect to A. Then,*

$$\|v\|_\infty \le \kappa(\mathcal{X}_A)\cdot\|Av\|_1 \quad \textit{and} \quad \|v\|_2 \le m\cdot\kappa(\mathcal{X}_A)\cdot\|Av\|_2\,.$$

Proof Consider a generalized path-circuit decomposition $v = \sum_{k=1}^{\ell} h^k$. By assumption, $Ah^k \neq \mathbf{0}$ for each k. Thus, for every $j \in \mathrm{supp}(h^k)$ there exists an $i \in M$, such that $|h^k(j)| \le \kappa(\mathcal{X}_A)|A_i h^k|$. For every $j \in N$, the conformity of the decomposition implies $|v(j)| = \sum_{k=1}^{\ell} |h^k(j)|$. Similarly, for every $i \in M$, $|A_i v| = \sum_{k=1}^{\ell} |A_i h^k|$. These imply the inequality $\|v\|_\infty \le \kappa(\mathcal{X}_A)\|Av\|_1$.

For the second inequality, note that for any outer vector $(h^k, Ah^k) \in \mathcal{X}_A$, the columns in $\mathrm{supp}(h^k)$ must be linearly independent. Consequently, $\|h^k\|_2 \le \sqrt{m}\cdot\kappa(\mathcal{X}_A)\cdot|(Ah^k)_i|$ for each k and $i \in \mathrm{supp}(Ah^k)$. This implies

$$\|v\|_2 \le \sum_{k=1}^{\ell}\|h^k\|_2 \le \sqrt{m}\cdot\kappa(\mathcal{X}_A)\cdot\|Av\|_1 \le m\cdot\kappa(\mathcal{X}_A)\cdot\|Av\|_2\,,$$

completing the proof. □

Remark 2.4 We note that a similar argument shows that $\|A\| \le \sqrt{m\tau(A)}\cdot\kappa(\mathcal{X}_A)$, where $\tau(A) \le m$ is the maximum size of $\mathrm{supp}(Ah)$ for an elementary vector $(h, Ah) \in \mathcal{X}_A$.

Example 2.5 If $A \in \mathbb{R}^{M\times N}$ is the node-arc incidence matrix of a directed graph $D = (M, N)$. The system $Ax = b, x \in \mathbf{B}(u)$ corresponds to a network flow feasibility problem. Here, $b(i)$ is the demand of node $i \in M$, i.e., the inflow minus the outflow at i is required to be $b(i)$. Recall that A is a TU matrix; consequently, $(A| - I_M)$ is also TU, and $\kappa(\mathcal{X}_A) = 1$. Our algorithm is strongly polynomial in this setting. Note that inner vectors correspond to cycles and outer vectors to paths; this motivates the term 'generalized path-circuit decomposition.' We also note $\tau(A) = 2$, and thus $\|A\| \le \sqrt{2|M|}$ in this case.

2.2 Optimal solutions and proximity

Let

$$Z(A, u) := \{Ax \mid x \in \mathbf{B}(u)\}. \tag{1}$$

Thus, Problem (P) is to find the point in $Z(A, u)$ that is nearest to b with respect to the Euclidean norm. We note that if the upper bounds u are finite, $Z(A, u)$ is called a *zonotope*.

Throughout, we let p^* denote the optimum value of (P). Note that whereas the optimal solution x^* may not be unique, the vector $b^* := Ax^*$ is unique by strong convexity; we have $p^* = \frac{1}{2}\|b - b^*\|^2$. We use

$$\eta(x) := \tfrac{1}{2}\|Ax - b\|^2 - p^*$$

to denote the optimality gap for $x \in \mathbf{B}(u)$. The point $x \in \mathbf{B}(u)$ is an *ε-approximate solution* if $\eta(x) \le \varepsilon$.

For a point $x \in \mathbf{B}(u)$, let

$$\begin{aligned}&I_0(x) := \{i \in N \mid x(i) = 0\}\,, \quad I_1(x) := \{i \in N \mid x(i) = u(i)\}\,, \quad \text{and}\\ &J(x) := N \setminus (I_0(x) \cup I_1(x))\,.\end{aligned}$$

The gradient of the objective $\frac{1}{2}\|Ax - b\|^2$ in (P) can be written as

$$g^x := A^\top(Ax - b)\,. \tag{2}$$

We recall the first order optimality conditions.

Lemma 2.6 *The point $x \in \mathbf{B}(u)$ is an optimal solution to* (P) *if and only if $g^x(i) = 0$ for all $i \in J(x)$, $g^x(i) \ge 0$ for all $i \in I_0(x)$, and $g^x(i) \le 0$ for all $i \in I_1(x)$.*

Using Lemma 2.3, we can bound the distance of any x from the nearest optimal solution.

Lemma 2.7 *For any $x \in \mathbf{B}(u)$, there exists an optimal solution x^* to* (P) *such that*

$$\begin{aligned}\|x - x^*\|_\infty &\le \kappa(\mathcal{X}_A) \cdot \|Ax - b^*\|_1\,, \text{ and}\\ \|x - x^*\|_2 &\le m \cdot \kappa(\mathcal{X}_A) \cdot \|Ax - b^*\|_2\,.\end{aligned}$$

Proof Let us select an optimal solution x^* to (P) such that $\|x - x^*\|_2$ is minimal. We show that $x - x^*$ is cycle-free w.r.t. A; the statements then follow from Lemma 2.3.

For a contradiction, assume a generalized path-circuit decomposition of $x - x^*$ contains an inner vector g, i.e., $Ag = \mathbf{0}$. By conformity of the decomposition, for $\bar{x} = x^* + g$ we have $\bar{x} \in \mathbf{B}(u)$ and $A\bar{x} = Ax^*$. Thus, $\bar{x}$ is another optimal solution, but $\|x - \bar{x}\|_2 < \|x - x^*\|_2$, a contradiction. □

2.3 The centroid mapping

Let us denote by 3^N the set of all ordered pairs (I_0, I_1) of disjoint subsets $I_0, I_1 \subseteq N$, and let $I_* := \{i \in N \mid u(i) < \infty\}$. For any $(I_0, I_1) \in 3^N$ with $I_1 \subseteq I_*$, we let

$$\mathbf{L}(I_0, I_1) := \{x \in \mathbb{R}^N \mid \forall i \in I_0 : x(i) = 0,\ \forall i \in I_1 : x(i) = u(i)\,\}\,. \tag{3}$$

We call $\{Ax \mid x \in \mathbf{B}(u) \cap \mathbf{L}(I_0, I_1)\} \subseteq Z(A, u)$ a *pseudoface* of $Z(A, u)$. We note that every face of $Z(A, u)$ is a pseudoface, but there might be pseudofaces that do not correspond to any face.

We define a *centroid set* for (I_0, I_1) as

$$\mathcal{C}(I_0, I_1) := \arg\min_y \{\|Ay - b\| \mid y \in \mathbf{L}(I_0, I_1))\} \ . \tag{4}$$

Proposition 2.8 *For $(I_0, I_1) \in 3^N$ with $I_1 \subseteq I_*$, $\mathcal{C}(I_0, I_1)$ is an affine subspace of $\mathbb{R}^N$, and for some $w \in \mathbb{R}^M$, it holds that $Ay = w$ for every $y \in \mathcal{C}(I_0, I_1)$.*

The *centroid mapping* $\Psi : \mathbf{B}(u) \to \mathbb{R}^N$ is a mapping that satisfies

$$\Psi(\Psi(x)) = \Psi(x) \quad \text{and} \quad \Psi(x) \in \mathcal{C}(I_0(x), I_1(x)) \quad \forall x \in \mathbf{B}(u) \, .$$

We say that $x \in \mathbf{B}(u)$ is a *stable point* if $\Psi(x) = x$. A simple, 'oblivious' centroid mapping arises by taking a minimum-norm point of the centroid set:

$$\Psi(x) := \arg\min\{\|y\| \mid y \in \mathcal{C}(I_0(x), I_1(x))\} \, . \tag{5}$$

However, this mapping has some undesirable properties. For example, we may have an iterate x that is already in $\mathcal{C}(I_0(x), I_1(x))$, but $\Psi(x) \neq x$. Instead, we aim for centroid mappings that move the current point 'as little as possible'. This can be formalized as follows. The centroid mapping Ψ is called *cycle-free*, if the vector $\Psi(x) - x$ is cycle-free w.r.t. A for every $x \in \mathbf{B}(u)$. The next claim describes a general class of cycle-free centroid mappings.

Lemma 2.9 *For every $x \in \mathbf{B}(u)$, let $D(x) \in \mathbb{R}^{N \times N}_{>0}$ be a positive diagonal matrix. Then,*

$$\Psi(x) := \arg\min\{\|D(x)(y - x)\| \mid y \in \mathcal{C}(I_0(x), I_1(x))\} \tag{6}$$

defines a cycle-free centroid mapping.

Proof For a contradiction, assume $y - x$ is not cycle-free for $y = \Psi(x)$, that is, a generalized path-circuit decomposition contains an inner vector z. For $y' = y - z$ we have $Ay' = Ay$, meaning that $y' \in \mathcal{C}(I_0(x), I_1(x))$. This is a contradiction, since $\|D(x)(y' - x)\| < \|D(x)(y - x)\|$ for any positive diagonal matrix $D(x)$. □

We emphasize that $D(x)$ in the above statement is a function of x and can be any positive diagonal matrix. Note also that the diagonal entries for indices in $I_0(x) \cup I_1(x)$ do not matter. In our experiments, defining $D(x)$ with diagonal entries $1/x(i) + 1/(u(i) - x(i))$ for $i \in J(x)$ performs particularly well. Intuitively, this choice aims to move less the coordinates close to the boundary.[3] The next proposition follows from Lagrangian duality, and provides a way to compute $\Psi(x)$ as in (6) by solving a system of linear equations.

[3] Note that the weights for $i \in I_0(x) \cup I_1(x)$ do not matter, since we force $y(i) = x(i)$ on these coordinates. The choice $1/x(i) + 1/(u(i) - x(i))$ would set ∞ on these coordinates.

Proposition 2.10 *For a partition $N = I_0 \cup I_1 \cup J$, the centroid set can be written as*

$$\mathcal{C}(I_0, I_1) = \left\{ y \in \mathbf{L}(I_0, I_1) \mid (A^J)^\top (Ay - b) = \mathbf{0} \right\} .$$

For $(I_0, I_1, J) = (I_0(x), I_1(x), J(x))$ and $D = D(x)$, the point $y = \Psi(x)$ as in (6) can be obtained as the unique solution to the system of linear equations

$$\begin{aligned} (A^J)^\top Ay &= (A^J)^\top b & \\ y_J + (D_J^J)^{-1}(A^J)^\top A^J \lambda &= x_J & \\ y(i) &= 0 & \forall i \in I_0 \\ y(i) &= u(i) & \forall i \in I_1 \\ \lambda &\in \mathbb{R}^J & . \end{aligned}$$

3 The update-and-stabilize framework

Now we describe a general algorithmic framework MNPZ(A, b, u) for solving (P), shown in Algorithm 1. Similarly to Wolfe's MNP algorithm, the algorithm comprises major and minor cycles. We maintain a point $x \in \mathbf{B}(u)$, and x is stable at the end of every major cycle. Each major cycle starts by calling the subroutine `Update`(x); the only general requirement on this subroutine is as follows:

(U1) for $y =$ `Update`(x), $y = x$ if and only if x is optimal to (P), and $\|Ay - b\| < \|Ax - b\|$ otherwise, and

(U2) if $y \neq x$, then for any $\lambda \in [0, 1)$, $z = \lambda y + (1 - \lambda)x$ satisfies $\|Ay - b\| < \|Az - b\|$.

Property (U1) can be obtained from any first order algorithm; we introduce some important examples in Sect. 3.1. Property (U2) might be violated if using a fixed step-length, which is a common choice. In order to guarantee (U2), we can post-process the first order update: choose y as the optimal point on the line segment $[x, y']$, where y' is the update found by the fixed-step update.

The algorithm terminates in the first major cycle when $x =$ `Update`(x). Within each major cycle, the minor cycles repeatedly use the centroid mapping Ψ. As long as $w := \Psi(x) \neq x$, i.e., x is not stable, we set $x := w$ if $w \in \mathbf{B}(u)$; otherwise, we set the next x as the intersection of the line segment $[x, w]$ and the boundary of $\mathbf{B}(u)$. The requirement (U1) is already sufficient to show finite termination.

Theorem 3.1 *Consider any `Update`(x) subroutine that satisfies (U1) and any centroid mapping Ψ. The algorithm* MNPZ(A, b, u) *finds an optimal solution to (P) within 3^n major cycles. Every major cycle contains at most n minor cycles.*

Proof Requirement (U1) guarantees that if the algorithm terminates, it returns an optimal solution. We claim that the same sets (I_0, I_1) cannot appear as $(I_0(x), I_1(x))$ at the end of two different major cycles; this implies the bound on the number of major cycles. To see this, we note that for $x = \Psi(x)$, $x \in \mathcal{C}(I_0(x), I_1(x)) = \mathcal{C}(I_0, I_1)$; thus,

Algorithm 1: MNPZ(A, b, u)

```
Input  : $A \in \mathbb{R}^{M \times N}$, $b \in \mathbb{R}^M$, $u \in (\mathbb{R} \cup \{\infty\})^N$
Output: An optimal solution $x$ to (P)
x ←initial point from $\mathbf{B}(u)$ ;
repeat
    $x \leftarrow$ Update($x$) ;                                    // Major cycle
    $w \leftarrow \Psi(x)$ ;
    while $\Psi(x) \neq x$                                         // Minor cycle
     do
        $\alpha^* \leftarrow \arg\max\{\alpha \in [0,1] \mid x + \alpha(w - x) \in \mathbf{B}(u)\}$ ;
        $x \leftarrow x + \alpha^*(w - x)$ ;
        $w \leftarrow \Psi(x)$ ;
    $x \leftarrow w$ ;
until $x =$ Update($x$)
return $x$
```

$\|Ax - b\| = \min\{\|Az - b\| \mid z \in \mathbf{L}(I_0, I_1)\}$. By (U1), $\|Ay - b\| < \|Ax - b\|$ at the beginning of every major cycle. Moreover, it follows from the definition of the centroid mapping that $\|Ax - b\|$ is non-increasing in every minor cycle. To bound the number of minor cycles in a major cycle, note that the set $I_0(x) \cup I_1(x) \subseteq N$ is extended in every minor cycle. □

3.1 The update subroutine

We can implement the Update(x) subroutine satisfying (U1) and (U2) using various first order methods for constrained optimization.

Recall the gradient g^x from (2); we use $g = g^x$ when x is clear from the context. The following property of stable points can be compared to the optimality condition in Lemma 2.6.

Lemma 3.2 *If $x (= \Psi(x))$ is a stable point, then $g^x(j) = 0$ for all $j \in J(x)$.*

Proof This directly follows from Proposition 2.10 that asserts $(A^{J(x)})^\top (Ax - b) = \mathbf{0}$. □

We now describe three classical options. We stress that the centroid mapping Ψ can be chosen independently from the update step.

The Frank–Wolfe update The Frank–Wolfe or *conditional gradient* method is applicable only in the case when $u(i)$ is finite for every $i \in N$. In every update step, we start by computing $\bar{y}$ as a minimizer of the linear objective $\langle g, y\rangle$ over $\mathbf{B}(u)$, that is,

$$\bar{y} \in \arg\min\{\langle g, y\rangle \mid y \in \mathbf{B}(u)\} . \tag{7}$$

We set Update(x) $:= x$ if $\langle g, \bar{y}\rangle = \langle g, x\rangle$, or $y =$ Update(x) is selected so that y minimizes $\frac{1}{2}\|Ay - b\|^2$ on the line segment $[x, \bar{y}]$.

Clearly, $\bar{y}(i) = 0$ whenever $g(i) > 0$, and $\bar{y}(i) = u(i)$ whenever $g(i) < 0$. However, $\bar{y}(i)$ can be chosen arbitrarily if $g(i) = 0$. In this case, we keep $\bar{y}(i) = x(i)$; this will be significant to guarantee stability of solutions in the analysis.

The projected gradient update The projected gradient update moves in the opposite gradient direction to $\bar{y} := x - \lambda g$ for some step-length $\lambda > 0$, and obtains the output $y = \texttt{Update}(x)$ as the projection y of $\bar{y}$ to the box $\mathbf{B}(u)$. This projection simply changes every negative coordinate to 0 and every $\bar{y}(i) > u(i)$ to $y(i) = u(i)$. To ensure (U2), we can perform an additional step that replaces y by the point $y' \in [x, y]$ that minimizes $\frac{1}{2}\|Ay' - b\|^2$.

Consider now an NNLS instance (i.e., $u(i) = \infty$ for all $i \in N$), and let x be a stable point. Recall $I_1(x) = \emptyset$ in the NNLS setting. Lemma 3.2 allows us to write the projected gradient update in the following simple form that also enables to use optimal line search. Define

$$z^x(i) := \max\{-g^x(i), 0\}, \tag{8}$$

and use $z = z^x$ when clear from the context. According to Lemma 2.6, x is optimal to (P) if and only if $z = \mathbf{0}$. We use the optimal line search

$$y := \arg\min_y \left\{ \tfrac{1}{2}\|Ay - b\|^2 \mid y = x + \lambda z, \lambda \geq 0 \right\} .$$

If $z \neq \mathbf{0}$, this can be written explicitly as

$$y := x + \frac{\|z\|^2}{\|Az\|^2} z . \tag{9}$$

To verify this formula, we note that $\|z\|^2 = -\langle g, z\rangle$, since for every $i \in N$ either $z(i) = 0$ or $z(i) = -g(i)$.

Coordinate update Our third update rule is the one used in the Lawson–Hanson algorithm. Given a stable point $x \in \mathbf{B}(u)$, we select a coordinate $j \in N$ where either $j \in I_0(x)$ and $g(j) < 0$ or $j \in I_1(x)$ and $g(j) > 0$, and set y such that $y(i) = x(i)$ if $i \neq j$, and $y(j)$ is chosen in $[0, u(j)]$ so that $\frac{1}{2}\|Ay - b\|^2$ is minimized. As in the Lawson–Hanson algorithm, we can maintain basic solutions throughout.

Lemma 3.3 *Assume A^J is linearly independent for $J = J(x)$. Then, $A^{J'}$ is also linearly independent for $J' = J(y) = J \cup \{j\}$, where $y = \texttt{Update}(x)$ using a coordinate update.*

Proof For a contradiction, assume $A^j = A^J w$ for some $w \in \mathbb{R}^J$. Then,

$$g(j) = (A^j)^\top (Ax - b) = w^\top (A^J)^\top (Ax - b) = 0 ,$$

a contradiction. □

Let us start with $x = \mathbf{0}$, i.e., $J(x) = I_1(x) = \emptyset$, $I_0(x) = N$. Then, $A^{J(x)}$ remains linearly independent throughout. Hence, every stable solution x is a basic solution to (P). Note that whenever $A^{J(x)}$ is linearly independent, $\mathcal{C}(I_0(x), I_1(x))$ contains a single point, hence, $\Psi(x)$ is uniquely defined.

For the special case of NNLS, i.e., for the case with no upper bounds, one can obtain simple explicit formulas for the coordinate update y. For z as in (8), let us return $y = x$ if $z = \mathbf{0}$. Otherwise, let $j \in \arg\max_k z(k)$; note that $j \in I_0(x)$. Let

$$y(i) := \begin{cases} x(i) & \text{if } i \in N \setminus \{j\}\,, \\ \frac{z(i)}{\|A^i\|^2} & \text{if } i = j\,. \end{cases} \tag{10}$$

The following lemma is immediate. In the NNLS setting, (U2) is guaranteed for the updates described above. For the general form with upper bounds, we can post-process as noted above to ensure (U2).

Lemma 3.4 *The Frank–Wolfe, projected gradient, and coordinate update rules all satisfy* (U1) *and* (U2).

Cycle-free update rules

Definition 3.5 We say that $\texttt{Update}(x)$ is a *cycle-free update rule*, if for every $x \in \mathbf{B}(u)$ and $y = \texttt{Update}(x)$, $x - y$ is cycle-free w.r.t. A.

Lemma 3.6 *The Frank–Wolfe, projected gradient, and coordinate updates are all cycle-free.*

Proof Each of the three rules satisfies that for any $x \in \mathbf{B}(u)$ with gradient g and $y = \texttt{Update}(x)$, $y - x$ conforms to $-g$. We show that this implies the required property.

For a contradiction, assume that a generalized path-cycle decomposition of $y - x$ contains an inner vector h. Thus, $h \neq \mathbf{0}$, $Ah = \mathbf{0}$, and h conforms to $-g$. Consequently, $\langle g, h\rangle < 0$. Recalling the form of g from (2), we get

$$0 > \langle g, h\rangle = \left\langle A^\top(Ax - b), h\right\rangle = \langle Ax - b, Ah\rangle = 0\,,$$

a contradiction. □

4 Analysis

Our main goal is to show the following convergence bound. The proof will be given in Sect. 4.3. Recall that in an NNLS instance, all upper capacities are infinite.

Theorem 4.1 *Consider an NNLS instance of* (P)*, and assume we use a cycle-free centroid mapping. Algorithm 1 terminates with an optimal solution in* $O(n \cdot m^2 \cdot \kappa^2(\mathcal{X}_A) \cdot \|A\|^2 \cdot \log(n + \kappa(\mathcal{X}_A)))$ *major cycles using projected gradient updates* (9)*, and in* $O(n^2 m^2 \cdot \kappa^2(\mathcal{X}_A) \cdot \|A\|^2 \cdot \log(n + \kappa(\mathcal{X}_A)))$ *major cycles using coordinate updates* (9)*, when initialized with* $x = \mathbf{0}$*. In both cases, the total number of minor cycles is* $O(n^2 m^2 \cdot \kappa^2(\mathcal{X}_A) \cdot \|A\|^2 \cdot \log(n + \kappa(\mathcal{X}_A)))$.

4.1 Proximity bounds

We show that if using a cycle-free update rule and a cycle-free centroid mapping, the movement of the iterates in Algorithm 1 can be bounded by the change in the objective value. First, a nice property of the centroid set is that the movement of Ax directly relates to the decrease in the objective value. Namely,

Lemma 4.2 *For $x \in \mathbf{B}(u)$, let $y \in \mathcal{C}(I_0(x), I_1(x))$. Then,*

$$\|Ax - Ay\|^2 = \|Ax - b\|^2 - \|Ay - b\|^2 .$$

Consequently, if Ψ is a cycle-free centroid mapping and $y = \Psi(x)$, then

$$\|x - y\|^2 \leq m^2 \cdot \kappa^2(\mathcal{X}_A) \cdot \left(\|Ax - b\|^2 - \|Ay - b\|^2\right) .$$

Proof Let $J := J(x)$. Since $Ax - b = (Ax - Ay) + (Ay - b)$, the claim is equivalent to showing that

$$\langle Ax - Ay, Ay - b\rangle = 0 .$$

Noting that $Ax - Ay = A^J x_J - A^J y_J$, we can write

$$\langle Ax - Ay, Ay - b\rangle = (x_J - y_J)^\top (A^J)^\top (Ay - b) = 0 ,$$

where the equality follows since $(A^J)^\top (Ay - b) = \mathbf{0}$ by Proposition 2.10. The second part follows from Lemma 2.3. □

Next, let us consider the movement of x during a call to `Update`(x).

Lemma 4.3 *Let $x \in \mathbf{B}(u)$ and $y =$ `Update`(x). Then,*

$$\|Ax - Ay\|^2 \leq \|Ax - b\|^2 - \|Ay - b\|^2 .$$

If using a cycle-free update rule, we also have

$$\|x - y\|^2 \leq m^2 \cdot \kappa^2(\mathcal{X}_A) \cdot \left(\|Ax - b\|^2 - \|Ay - b\|^2\right) .$$

Proof From property (U2), it is immediate to see that $\langle Ay - b, Ax - Ay\rangle \geq 0$. This implies the first claim. The second claim follows from the definition of a cycle-free update rule and Lemma 2.3. □

Lemma 4.4 *Let $x \in \mathbf{B}(u)$, and let x' be an iterate obtained by consecutive t major or minor updates of Algorithm 1 using a cycle-free update rule and a cycle-free centroid mapping, starting from x. Then,*

$$\|x - x'\| \leq m \cdot \kappa(\mathcal{X}_A) \cdot \sqrt{t\left(\|Ax - b\|^2 - \|Ax' - b\|^2\right)} .$$

Proof Let us consider the (major and minor cycle) iterates $x = x^{(k)}, x^{(k+1)}, \ldots, x^{(k+t)} = x'$. From the triangle inequality, and the arithmetic-quadratic means inequality,

$$\|x - x'\| \le \sum_{j=1}^{t} \|x^{(k+j)} - x^{(k+j-1)}\| \le \sqrt{t \sum_{j=1}^{t} \|x^{(k+j)} - x^{(k+j-1)}\|^2}$$

The statement then follows using the bounds in Lemma 4.2 and Lemma 4.3. □

4.2 Geometric convergence of the projected gradient and coordinate updates

We present a simple convergence analysis for the NNLS setting. For the general capacitated setting, similar bounds should follow from [22]. Recall that $\eta(x)$ denotes the optimality gap at x.

Theorem 4.5 *Consider an NNLS instance of* (P)*, and let $x \ge \mathbf{0}$ be a stable point. Then for $y = \texttt{Update}(x)$ using the projected gradient update* (9) *we have*

$$\eta(y) \le \left(1 - \frac{1}{2m^2 \cdot \kappa^2(\mathcal{X}_A) \cdot \|A\|^2}\right) \eta(x)\,.$$

Using coordinate updates as in (10)*, we have*

$$\eta(y) \le \left(1 - \frac{1}{2nm^2 \cdot \kappa^2(\mathcal{X}_A) \cdot \|A\|^2}\right) \eta(x)\,.$$

Consequently, either with projected gradient or with coordinate updates, after performing $O(nm^2 \cdot \kappa^2(\mathcal{X}_A) \cdot \|A\|^2)$ minor and major cycles from an iterate x, we obtain an iterate x' with $\eta(x') \le \eta(x)/2$.

Let us formulate the update progress using optimal line search.

Lemma 4.6 *For a stable point $x \ge \mathbf{0}$, the update* (9) *satisfies*

$$\|Ax - b\|^2 - \|Ay - b\|^2 \ge \frac{\|z\|^2}{\|A\|^2}\,,$$

and the update (10) *satisfies*

$$\|Ax - b\|^2 - \|Ay - b\|^2 = \frac{z(j)^2}{\|A^j\|^2}\,.$$

Proof For the update (9) with stepsize $\lambda = \|z\|^2/\|Az\|^2$, we have

$$\begin{aligned}\|Ay - b\|^2 &= \|Ax - b\|^2 + \lambda^2\|Az\|^2 + 2\lambda\langle Ax - b, Az\rangle \\ &= \|Ax - b\|^2 + \lambda^2\|Az\|^2 + 2\lambda\langle g, z\rangle \\ &= \|Ax - b\|^2 + \lambda^2\|Az\|^2 - 2\lambda\|z\|^2 \\ &= \|Ax - b\|^2 - \frac{\|z\|^4}{\|Az\|^2}\,,\end{aligned}$$

where the third equality uses $\langle g, z\rangle = -\|z\|^2$ noted previously. The statement follows by using $\|Az\| \le \|A\| \cdot \|z\|$.

The proof is similar for the update (10). Here, $y = x + \lambda e_j$, where e_j is the jth unit vector, and $\lambda = z(j)/\|A^j\|^2$. The bound follows by noting that $\langle Ax - b, Ae_j\rangle = \langle g, e_j\rangle = -z(j)$. □

We now use Lemma 2.7 to bound $\|z\|$.

Lemma 4.7 *For a stable point $x \ge \mathbf{0}$ and the update direction $z = z^x$ as in* (8)*, we have*

$$\|z\| \ge \frac{\sqrt{\eta(x)}}{\sqrt{2}m \cdot \kappa(\mathcal{X}_A)}\,.$$

Proof Let $x^* \ge \mathbf{0}$ be an optimal solution to (P) as in Lemma 2.7, and $b^* = Ax^*$. Using convexity of $f(x) := \frac{1}{2}\|Ax - b\|^2$,

$$p^* = f(x^*) \ge f(x) + \langle g, x^* - x\rangle \ge f(x) - \langle z, x^* - x\rangle\,,$$

where the second inequality follows by noting that for each $i \in N$, either $z(i) = -g(i)$, or $z(i) = 0$ and $g(i)(x^*(i) - x(i)) \ge 0$. From the Cauchy-Schwarz inequality and Lemma 2.7, we get

$$p^* \ge f(x) - \|z\| \cdot \|x^* - x\| \ge f(x) - m \cdot \kappa(\mathcal{X}_A) \cdot \|Ax - b^*\| \cdot \|z\|\,,$$

that is,

$$\|z\| \ge \frac{\eta(x)}{m \cdot \kappa(\mathcal{X}_A) \cdot \|Ax - b^*\|}\,.$$

The proof is complete by showing

$$2\eta(x) \ge \|Ax - b^*\|^2\,. \tag{11}$$

Recalling that $\eta(x) = \frac{1}{2}\|Ax - b\|^2 - \frac{1}{2}\|Ax^* - b\|^2$ and that $b^* = Ax^*$, this is equivalent to

$$\langle Ax - Ax^*, Ax^* - b\rangle \ge 0\,.$$

This can be further written as

$$\langle x - x^*, g^{x^*} \rangle \geq 0 \,,$$

which is implied by the first order optimality condition at x^*. This proves (11), and hence the lemma follows. □

Proof *(Proof of Theorem 4.5)* The proof for the bound in projected gradient updates is immediate from Lemma 4.6 and Lemma 4.7. For coordinate updates, recall that j is selected as the index of the largest component $z(j)$. Thus, $z(j)^2 \geq \|z\|^2/n$, and $\|A^j\| \leq \|A\|$.

For the second part, the statement follows for projected gradient updates by the first part and by noting that there are at most n minor cycles in every major cycle. For coordinate updates, every major cycle adds one component to $J(x)$ whereas every minor cycle removes at least one. Hence, the total number of minor cycles is at most m plus the total number of major cycles. □

4.3 Overall convergence bounds

In this subsection, we prove Theorem 4.1. Using Lemma 4.4 and Theorem 4.5, we can derive the following stronger proximity bound:

Lemma 4.8 *Consider an NNLS instance of* (P). *Let $x \geq \mathbf{0}$ be an iterate of Algorithm 1 using projected gradient or coordinate updates, and let $x' \geq \mathbf{0}$ be any later iterate. Then, for a value*

$$\Theta := O(\sqrt{n}m^2 \cdot \kappa^2(\mathcal{X}_A) \cdot \|A\|) \,,$$

we have

$$\|x - x'\| \leq \Theta\sqrt{\eta(x)} \,.$$

Proof According to Theorem 4.5, after $T := O(nm^2 \cdot \kappa^2(\mathcal{X}_A) \cdot \|A\|^2)$ major and minor cycles, we get to an iterate x'' with $\eta(x'') \leq \eta(x)/4$. Thus, Lemma 4.4 gives

$$\|x - x''\| \leq m \cdot \sqrt{2T} \cdot \kappa(\mathcal{X}_A) \cdot \sqrt{\eta(x)} \,.$$

Let us now define $x^{(k)}$ as the iterate following x after Tk major and minor cycles; we let $x^{(0)} := x$. By Theorem 4.5, $x^{(k)} \leq \eta(x)/4^k$, and similarly as above, for each $k = 0, 1, 2, \ldots$ we get

$$\|x^{(k)} - x^{(k+1)}\| \leq m \cdot \sqrt{2T} \cdot \kappa(\mathcal{X}_A) \cdot \frac{\sqrt{\eta(x)}}{2^k} \,.$$

The above bound also holds for any iterate x' between $x^{(k)}$ an $x^{(k+1)}$. Using these bounds and the triangle inequality, for any iterate x' after x, we obtain

$$\|x - x'\| \leq 2m \cdot \sqrt{2T} \cdot \kappa(\mathcal{X}_A) \cdot \sqrt{\eta(x)}\,.$$

This completes the proof. □

We need one more auxiliary lemma.

Lemma 4.9 *Consider an NNLS instance of* (P)*, and let $x \geq \mathbf{0}$ be a stable point. Let $\hat{x} \geq \mathbf{0}$ such that for each $i \in N$, either $\hat{x}(i) = x(i)$, or $\hat{x}(i) = 0 < x(i)$. Then,*

$$\|A\hat{x} - b\|^2 = \|Ax - b\|^2 + \|A\hat{x} - Ax\|^2\,.$$

Proof The claim is equivalent to showing

$$\langle A\hat{x} - Ax, Ax - b\rangle = 0\,.$$

We can write $\langle A\hat{x} - Ax, Ax - b\rangle = \langle g^x, \hat{x} - x\rangle$. By assumption, $\hat{x}(i) - x(i) \neq 0$ only if $x(i) > 0$, but in this case $g^x(i) = 0$ by Lemma 3.2. □

For the threshold Θ as in Lemma 4.8 and for any $x \geq \mathbf{0}$, let us define

$$J^{\star(x)} := \left\{i \mid x(i) > \Theta\sqrt{\eta(x)}\right\}\,.$$

The following is immediate from Lemma 4.8.

Lemma 4.10 *Consider an NNLS instance of* (P)*. Let $x \geq \mathbf{0}$ be an iterate of Algorithm 1 using projected gradient updates, and $x' \geq \mathbf{0}$ be any later iterate. Then,*

$$J^{\star(x)} \subseteq J(x')\,.$$

We are ready to prove Theorem 4.1.

Proof *(Proof of Theorem 4.1)* At any point of the algorithm, let $J^\star$ denote the union of the sets $J^{\star(x)}$ for all iterations thus far. Consider a stable iterate x at the beginning of any major cycle, and let

$$\varepsilon := \frac{\sqrt{\eta(x)}}{4n \cdot \Theta \cdot \|A\|}\,.$$

Theorem 4.5 guarantees that within $O(nm^2 \cdot \kappa^2(\mathcal{X}_A) \cdot \|A\|^2 \cdot \log(n + \kappa(\mathcal{X}_A)))$ major and minor cycles we arrive at an iterate x' such that $\sqrt{\eta(x')} < \varepsilon$. We note that $\log(n + \kappa(\mathcal{X}_A) + \|A\|) = O(\log(n + \kappa(\mathcal{X}_A)))$ according to Remark 2.4. We show that

$$J^{\star(x')} \cap I_0(x) \neq \emptyset\,. \tag{12}$$

From here, we can conclude that $J^\star$ was extended between iterates x and x'. This may happen at most n times, leading to the claimed bound on the total number of major and minor cycles. Using Theorem 4.5 we also obtain the respective bounds on the number of major cycles for the two different updates.

For a contradiction, assume that (12) does not hold. Thus, for every $i \in I_0(x)$, we have $x'(i) \le \Theta\varepsilon$. Let us define $\hat{x} \in \mathbb{R}^N$ as

$$\hat{x}(i) := \begin{cases} 0 & \text{if } i \in I_0(x)\,, \\ x'(i) & \text{if } i \in J(x)\,. \end{cases}$$

By the above assumption, $\|\hat{x} - x'\|_\infty \le \Theta\varepsilon$, and therefore $\|A\hat{x} - Ax'\| \le \sqrt{n}\Theta\|A\|\varepsilon$. From Lemma 4.9, we can bound

$$\|A\hat{x} - b\|^2 = \|Ax' - b\|^2 + \|A\hat{x} - Ax'\|^2 \le 2p^* + (n\Theta^2\|A\|^2 + 2)\varepsilon^2\,. \tag{13}$$

Recall that since x is a stable solution,

$$\|Ax - b\| = \min\left\{\|Ay - b\| \,:\, y \in \mathbf{L}(I_0(x), \emptyset)\right\}\,.$$

Since $\hat{x}$ is a feasible solution to this program, it follows that $\|A\hat{x} - b\|^2 \ge \|Ax - b\|^2$. We get that

$$2\eta(x) = \|Ax - b\|^2 - 2p^* \le \|A\hat{x} - b\|^2 - 2p^* \le (n\Theta^2\|A\|^2 + 2)\varepsilon^2\,,$$

in contradiction with the choice of ε. □

5 Computational experiments

We give preliminary computational experiments of different versions of our algorithm, and compare them to standard gradient methods and existing NNLS implementations. The experiments were programmed and executed by MATLAB version R2023a on a personal computer having 11th Gen Intel(R) Core(TM) i7-11370H @ 3.30GHz and 16GB of memory.

We considered two families of randomly generated NNLS instances. In Appendix A, we also present experiments for capacitated instances (finite $u(i)$ values).

We tested each combination of two update methods: Projected Gradient (PG), and coordinate (C); and two centroid mappings, the 'oblivious' mapping (5) and the 'local norm' mapping (6) with diagonal entries $1/x(i)$, $i \in N$. Recall that for coordinate updates and starting from $x = \mathbf{0}$, there is a unique centroid mapping by Lemma 3.3.

Our first benchmarks are the projected gradient (PG) and the projected fast (accelerated) gradient (PFG) methods. In contrast to our algorithms, these do not finitely terminate. We stopped the algorithms once they found a near-optimal solution within a certain accuracy threshold.

Further, we also compare our algorithms against the standard MATLAB implementation of the Lawson–Hanson algorithm called `lsqnonneg`, and against the

Table 1 Computation time (in sec) for uncapacitated rectangular instances

m	100	200	300	400	500	500	500
n	200	400	600	800	1000	2000	3000
PG+(5)	0.06	0.50	1.77	5.09	11.92	20.07	50.23
PG+(6)	0.01	0.06	0.18	0.28	0.29	0.17	0.22
C	0.05	0.31	1.02	2.52	4.84	2.98	2.90
PG	0.84	5.34	16.16 (1)	21.85 (1)	24.93 (2)	0.07	0.05
PFG	0.06	0.53	1.52	2.06	4.22	0.09	0.07
lsqnonneg	0.01	0.16	0.54	1.38	2.79	1.51	1.55
TNT-NN	0.01	0.03	0.08	0.17	0.33	0.22	0.61

Table 2 # of major cycles for uncapacitated rectangular instances

m	100	200	300	400	500	500	500
n	200	400	600	800	1000	2000	3000
PG+(5)	6.4	9.4	13.4	11.6	13.4	1.0	1.0
PG+(6)	2.0	3.0	3.0	2.4	1.6	1.0	1.0
C	128.8	267.4	404.0	536.6	664.4	520.8	501.8

Table 3 The total # of minor cycles for uncapacitated rectangular instances

m	100	200	300	400	500	500	500
n	200	400	600	800	1000	2000	3000
PG+(5)	201.8	462.6	727.8	1129.8	1619.4	1029.6	1517.4
PG+(6)	17.4	32.6	42.0	32.0	18.8	2.0	1.0
C	157.6	336.6	512.2	675.8	828.4	542.4	509.2

implementation `TNT-NN` from [21]. We note that `lsqnonneg` and the coordinate update version of our algorithms are essentially the same.

Generating instances We generated two families of experiments. In the *rectangular* experiments $n \geq 2m$, and in the *near-square* experiments $m \leq n \leq 1.1m$. In both cases, the entries of the $m \times n$ matrix A were chosen independently uniformly at random from the interval $[-0.5, 0.5]$. In the rectangular experiments, the entries of b were also chosen independently uniformly at random from $[-0.5, 0.5]$. Thus, the underlying LP $Ax = b, x \geq \mathbf{0}$ may or may not be feasible.

For the near-square instances, such a random choice of b leads to infeasible instances with high probability. We used this method to generate infeasible instances. We also constructed families where the LP is feasible as follows. For a sparsity parameter $\chi \in (0, 1]$, we sampled a subset $J \subseteq N$, adding each variable independently with probability χ, and generated coefficients $\{z_i : i \in J\}$ independently at random from $[0, 1]$. We then set $b = \sum_{j \in J} A^j z_j$.

Table 4 Computation time (in sec) for uncapacitated near-square instances

m	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
n	1020	1020	1020	1020	1050	1050	1050	1050	1100	1100	1100	1100
Status	I	F (0.1)	F (0.5)	F (1)	I	F (0.1)	F (0.5)	F(1)	I	F (0.1)	F (0.5)	F (1)
PG+(5)	13.44	0.88	2.59	0.83	14.74	2.21	2.20	1.49	16.85	4.46	9.61	2.40
PG+(6)	18.67	0.83	3.39	0.24	18.99	0.93	0.95	0.25	18.81	1.03	1.05	0.25
C	5.02	0.13	2.92	34.75	5.33	0.14	3.24	36.28	6.13	0.15	3.68	35.88
PG	0.17	0.49	1.41	60.00 (5)	0.20	0.59	1.61	60.00 (5)	0.25	0.81	2.32	18.36
PFG	0.16	0.12	0.31	60.00 (5)	0.19	0.12	0.34	60.00 (5)	0.24	0.15	0.41	60.00 (5)
lsqnonneg	2.37	0.06	1.46	16.88	2.55	0.06	1.59	17.71	2.86	0.07	1.81	17.55
TNT-NN	0.14	0.08	0.18	0.38	0.15	0.09	0.21	0.45	0.16	0.10	0.26	0.47

Table 5 # of major cycles for uncapacitated near-square instances

m	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
n	1020	1020	1020	1020	1050	1050	1050	1050	1100	1100	1100	1100
Status	I	F (0.1)	F (0.5)	F (1)	I	F (0.1)	F (0.5)	F (1)	I	F (0.1)	F (0.5)	F (1)
PG+(5)	4.2	1.0	1.2	1.2	4.0	1.0	1.0	1.0	4.4	1.0	1.6	1.0
PG+(6)	4.2	1.0	1.2	1.0	4.2	1.0	1.0	1.0	4.2	1.0	1.0	1.0
C	511.2	98.4	406.0	1080.6	524.4	101.0	423.4	1096.2	553.6	105.8	446.6	1092.6

Table 6 Total # of minor cycles for uncapacitated near-square instances

m	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
n	1020	1020	1020	1020	1050	1050	1050	1050	1100	1100	1100	1100
Status	I	F (0.1)	F (0.5)	F (1)	I	F (0.1)	F (0.5)	F (1)	I	F (0.1)	F (0.5)	F (1)
PG+(5)	572.4	21.0	78.2	18.6	601.4	51.0	51.0	32.8	636.0	101.0	270.8	52.2
PG+(6)	571.6	14.6	76.8	3.2	570.8	15.8	16.0	3.2	557.4	16.4	17.0	3.0
C	512.4	97.4	411.4	1160.2	529.0	100.0	431.8	1191.4	557.2	104.8	459.2	1184.2

Computational results We stopped each algorithm when the computation time reached 60 s. For each (m, n), we test all the algorithms 5 times and the results shown here are the 5-run averaged figures.

Table 1 shows the overall computational times for rectangular instances; values in brackets show the number of trials whose computation time exceeded 60 s. Tables 2 and 3 show the number of major cycles, and the total number of minor cycles, respectively. Table 4 shows the overall computational times for near-square instances. The status 'I' denotes infeasible instances and 'F' feasible instances, with the sparsity parameter χ in brackets, with values 0.1, 0.5, and 1. Tables 5 and 6 show the number of major cycles, and the total number of minor cycles, respectively, for near-square instances.

Comparison of the results For rectangular instances, the 'local-norm' update (6) performs significantly better than the 'oblivious' update (5). The 'oblivious' updates are also outperformed by the coordinate updates, both in terms of running time as well as in the total number of minor cycles.

As noted above, while our algorithm with coordinate updates and `lsqnonneg` are basically the same, the running time of the latter algorithm is better by around factor two. This is since `lsqnonneg` might be using more efficient linear algebra operations, in contrast to our more basic implementation.

The algorithm `TNT-NN` from [21] is a fast practical algorithm using a number of heuristics, representing the state-of-the-art active set method for NNLS. Notably, our algorithm with 'local-norm' updates (6) is almost always within a factor two for rectangular instances, and performs better in some cases. This is despite the fact that we only use a basic implementation without using more efficient linear algebra methods or including further heuristics.

For rectangular instances, `TNT-NN` and 'local-norm' updates also outperform fast projected gradient in most cases.

The picture is more mixed for near-square instances. There is a marked difference between feasible and infeasible instances. The 'local-norm' and 'oblivious' update rules perform similarly, with a small number of major cycles. The number of minor cycles is much higher for infeasible instances. For infeasible instances, coordinate updates are faster than either variant of the PG update rule, while PG updates are faster for feasible instances.

The algorithm `TNT-NN` is consistently faster than our algorithm, with better running times for infeasible instances. For projected gradient and projected fast gradient, the running times are similar to `TNT-NN` except for feasible instances with sparsity parameter $\chi = 1$, where they do not terminate within the 60 s limit in most cases. In contrast, these appear to be the easiest instances to our method with PG updates with the 'local-norm' mapping.

6 Concluding remarks

We have proposed a new 'Update-and-Stabilize' framework for the minimum-norm-point problem (P). Our method combines classical first order methods with 'stabilizing' steps using the centroid mapping that amounts to computing a projection to an affine subspace. Our algorithm is always finite, and is strongly polynomial when the associated circuit imbalance measure is constant. In particular, this gives the first such convergence bound for the Lawson–Hanson algorithm.

There is scope for further improvements both in the theoretical analysis and in practical implementations. In this paper, we only analyzed the running time for uncapacitated instances. Combined with existing results from [22], we expect that similar bounds can be shown for capacitated instances. We note that for the analysis, it would suffice to run minor cycles only once in a while, say after every $O(n)$ gradient updates. From a practical perspective however, running minor cycles after every update appears to be highly beneficial in most cases. Rigorous computational experiments, using standard families of LP benchmarks, are left for future work.

Future work should also compare the performance of our algorithms to the gradient projection method [5, 23], using techniques from that method to our algorithm and vice versa. We note that for NNLS instances, starting from a stable point our algorithm already finds the optimal gradient update. However, a similar search as in gradient projection methods may be useful in the capacitated case. In the other direction, we note that the conjugate gradient iterations used in gradient projection do not correspond to an explicit choice of a centroid mapping. A possible enhancement of gradient projection could come from approximating a 'local-norm' objective as in (6) in the second stage.

We also point out that the 'local-norm' selection rule (6) was inspired by the affine scaling method; the important difference is that our algorithm moves all the way to the boundary, whereas affine scaling stays in the interior throughout.

Acknowledgements We are grateful to Andreas Wächter for pointing us to the literature on the gradient projection method. The third author would like to thank Richard Cole, Daniel Dadush, Christoph Hertrich, Bento Natura, and Yixin Tao for discussions on first order methods and circuit imbalances.

Funding SF's research is supported by JSPS KAKENHI Grant Numbers JP19K11839 and 22K11922 and by the Research Institute for Mathematical Sciences, an International Joint Usage/Research Center located in Kyoto University. TK is supported by JSPS KAKENHI Grant Number JP19K11830. LAV's research is supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement no. 757481–ScaleOpt).

Declarations

Conflict of interests The authors have no conflicts of interest to declare that are relevant to the content of this article.

A Computational experiments for capacitated instances

Tables 7, 8, 9, 10, 11, and 12 show experimental results for capacitated instances. The instances were generated as in the NNLS case, with upper capacities $u(i) = 1$, $i \in N$. We did not use the benchmarks `lsqnonneg` and `TNT-NN` since these are not implemented for the capacitated setting. On the other hand, we also implemented our method with Frank–Wolfe updates, using both 'local-norm' and 'oblvious' centroid mappings. Among the first order benchmarks, we also included conditional gradient methods: the Frank–Wolfe and away-step Frank Wolfe (AFW) methods.

Table 7 Computation time (in sec) for capacitated rectangular instances

m	100	200	300	400	500	500	500
n	200	400	600	800	1000	2000	3000
FW+(5)	0.07	0.43	1.56	4.81	9.73	19.67	49.56
FW+(6)	0.02	0.07	0.22	0.49	0.63	0.17	0.31
PG+(5)	0.07	0.49	1.64	5.44	10.56	19.70	49.68
PG+(6)	0.01	0.07	0.20	0.44	0.55	0.17	0.32
C	0.05	0.28	0.94	2.19	5.35	2.88	2.88
FW	51.80 (4)	54.78 (4)	60.00 (5)	60.00 (5)	60.00 (5)	0.18	0.20
AFW	51.76 (4)	54.68 (4)	60.00 (5)	60.00 (5)	60.00 (5)	0.09	0.14
PG	0.31	2.64	5.25	5.44	33.33 (1)	0.04	0.04
PFG	0.02	0.12	0.24	0.42	0.84	0.05	0.06

Table 8 # of major cycles for capacitated rectangular instances

m	100	200	300	400	500	500	500
n	200	400	600	800	1000	2000	3000
FW+(5)	6.2	7.8	10.2	10.2	12.8	1.0	1.0
FW+(6)	2.6	3.4	3.6	3.4	3.2	1.0	1.0
PG+(5)	6.8	9.4	11.8	11.4	14.4	1.0	1.0
PG+(6)	2.6	2.8	3.8	3.2	3.6	1.0	1.0
C	127.0	253.6	389.0	507.6	697.6	514.8	502.2

Table 9 The total # of minor cycles for capacitated rectangular instances

m	100	200	300	400	500	500	500
n	200	400	600	800	1000	2000	3000
FW+(5)	186.6	378.6	629.2	1066.0	1356.4	1016.6	1498.8
FW+(6)	28.2	42.4	52.2	61.8	43.4	2.0	2.0
PG+(5)	212.4	442.4	668.6	1213.0	1465.4	1013.8	1498.4
PG+(6)	27.2	37.4	48.0	55.8	38.8	2.0	2.0
C	155.6	313.2	483.8	622.6	897.0	532.2	511.2

Table 10 Computation time (in sec) for capacitated near-square instances

m	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
n	1020	1020	1020	1020	1050	1050	1050	1050	1100	1100	1100	1100
Status	I	F (0.1)	F (0.5)	F (1)	I	F (0.1)	F (0.5)	F (1)	I	F (0.1)	F (0.5)	F (1)
FW+(5)	12.95	0.85	0.85	1.71	14.62	2.11	7.80	2.65	16.55	6.47	13.19	3.33
FW+(6)	18.10	0.78	0.77	0.26	18.69	0.86	0.88	0.25	18.44	0.97	1.04	0.26
PG+(5)	12.67	0.82	0.85	1.71	14.23	2.04	7.62	2.85	16.38	4.24	12.62	3.36
PG+(6)	17.80	0.77	0.80	0.25	18.15	0.89	3.26	0.24	18.39	0.97	1.04	0.25
C	4.78	0.13	3.04	39.38	5.29	0.14	3.34	37.89	5.66	0.15	3.84	40.21
FW	60.00 (5)	60.00 (5)	60.00 (5)	60.00 (5)	60.00 (5)	60.00 (5)	60.00 (5)	60.00 (5)	60.00 (5)	60.00 (5)	60.00 (5)	60.00 (5)
AFW	60.00 (5)	60.00 (5)	60.00 (5)	60.00 (5)	60.00 (5)	60.00 (5)	60.00 (5)	60.00 (5)	60.00 (5)	60.00 (5)	60.00 (5)	60.00 (5)
PG	0.11	0.49	1.28	60.00 (5)	0.14	0.52	1.84	60.00 (5)	0.17	0.69	2.22	17.30
PFG	0.10	0.11	0.28	60.00 (5)	0.11	0.11	0.37	60.00 (5)	0.15	0.14	0.36	60.00 (5)

Table 11 # of major cycles for capacitated near-square instances

m	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
n	1020	1020	1020	1020	1050	1050	1050	1050	1100	1100	1100	1100
Status	I	F (0.1)	F (0.5)	F (1)	I	F (0.1)	F (0.5)	F (1)	I	F (0.1)	F (0.5)	F (1)
FW+(5)	4.4	1.0	1.0	2.8	4.0	1.0	2.0	1.8	4.4	1.2	2.4	1.0
FW+(6)	3.8	1.0	1.0	1.0	4.2	1.0	1.0	1.0	4.6	1.0	1.0	1.0
PG+(5)	3.8	1.0	1.0	2.8	3.8	1.0	1.8	2.0	4.2	1.0	2.2	1.0
PG+(6)	3.6	1.0	1.0	1.0	4.0	1.0	1.2	1.0	4.2	1.0	1.0	1.0
C	508.6	108.2	487.2	1387.2	527.0	112.8	513.6	1388.4	542.4	115.2	538.2	1446.8

Table 12 Total # of minor cycles for capacitated near-square instances

m	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
n	1020	1020	1020	1020	1050	1050	1050	1050	1100	1100	1100	1100
Status	I	F (0.1)	F (0.5)	F (1)	I	F (0.1)	F (0.5)	F (1)	I	F (0.1)	F (0.5)	F (1)
FW+(5)	573.6	21.0	21.0	39.8	605.6	51.0	268.2	63.6	640.6	191.0	421.8	76.0
FW+(6)	573.6	14.2	14.0	3.4	576.6	15.0	15.2	3.2	561.4	16.0	17.2	3.2
PG+(5)	555.8	20.6	21.0	39.0	583.2	49.4	253.2	67.4	621.8	99.2	387.6	76.6
PG+(6)	553.2	14.6	14.6	3.4	554.0	15.8	72.2	3.2	548.8	16.4	17.0	3.2
C	510.4	106.6	500.2	1558.4	530.0	110.4	529.4	1571.8	545.8	113.2	558.2	1640.8

In our framework, the Frank–Wolfe and projected gradient update rules performed similarly. In contrast, among the benchmark experiments, projected gradient methods consistently outperformed conditional gradient methods: the latter methods did not terminate within the 60 s limit for most cases.

The overall experience is similar for uncapacitated (NNLS) and capacitated instances. Our method does well for rectangular instances, but is generally slower for infeasible near-square instances.

References

1. Bach, F.: Learning with submodular functions: a convex optimization perspective. Found. Trends Mach. Learn. **6**(2–3), 145–373 (2013)
2. Björck, Å.: A direct method for sparse least squares problems with lower and upper bounds. Numer. Math. **54**(1), 19–32 (1988)
3. Bro, R., De Jong, S.: A fast non-negativity-constrained least squares algorithm. J. Chemomet. J. Chemomet. Soc. **11**(5), 393–401 (1997)
4. Chakrabarty, D., Jain, P., Kothari, P.: Provable submodular minimization using Wolfe's algorithm. Adv. Neural Inf. Process. Syst. 27 (2014)
5. Conn, A.R., Gould, N.I., Toint, P.L.: Testing a class of methods for solving minimization problems with simple bounds on the variables. Math. Comput. **50**(182), 399–430 (1988)
6. Dadush, D., Huiberts, S., Natura, B., Végh, L.A.: A scaling-invariant algorithm for linear programming whose running time depends only on the constraint matrix. Math. Program. (2023)
7. Dadush, D., Natura, B., Végh, L.A.: Revisiting Tardos's framework for linear programming: Faster exact solutions using approximate solvers. In: Proceedings of the 61st Annual IEEE Symposium on Foundations of Computer Science (FOCS), pp. 931–942 (2020)
8. De Loera, J.A., Haddock, J., Rademacher, L.: The minimum Euclidean-norm point in a convex polytope: Wolfe's combinatorial algorithm is exponential. SIAM J. Comput. **49**(1), 138–169 (2020)
9. Ekbatani, F., Natura, B., Végh, A.L.: Circuit imbalance measures and linear programming. In: Surveys in combinatorics 2022, London Mathematical Society Lecture Note Series, pp. 64–114. Cambridge University Press (2022)
10. Ene, A., Vladu, A.: Improved convergence for ℓ_1 and ℓ_∞ regression via iteratively reweighted least squares. In: International Conference on Machine Learning, pp. 1794–1801. PMLR (2019)
11. Fujishige, S.: Lexicographically optimal base of a polymatroid with respect to a weight vector. Math. Oper. Res. **5**(2), 186–196 (1980)
12. Fujishige, S.: A capacity-rounding algorithm for the minimum-cost circulation problem: a dual framework of the Tardos algorithm. Math. Program. **35**(3), 298–308 (1986)
13. Fujishige, S., Hayashi, T., Yamashita, K., Zimmermann, U.: Zonotopes and the LP-Newton method. Optim. Eng. **10**(2), 193–205 (2009)
14. Fujishige, S., Isotani, S.: A submodular function minimization algorithm based on the minimum-norm base. Pac. J. Optim. **7**(1), 3–17 (2011)
15. Fulkerson, D.: Networks, frames, blocking systems. Math. Decis. Sci. Part I Lect. Appl. Math. **2**, 303–334 (1968)
16. Hoffman, A.J.: On approximate solutions of systems of linear inequalities. J. Res. Natl. Bur. Stand. **49**(4), 263–265 (1952)
17. Lacoste-Julien, S., Jaggi, M.: On the global linear convergence of Frank–Wolfe optimization variants. Adv. Neural Inf. Process. Syst. 28 (2015)
18. Lawson, C.L.: Contribution to the Theory of Linear Least Maximum Approximation. PhD thesis (1961)
19. Lawson, C.L., Hanson, R.J.: Solving least squares problems. SIAM (1995)
20. Leichner, S., Dantzig, G., Davis, J.: A strictly improving linear programming phase I algorithm. Ann. Oper. Res. **46**, 409–430 (1993)
21. Myre, J.M., Frahm, E., Lilja, D.J., Saar, M.O.: TNT-NN: a fast active set method for solving large non-negative least squares problems. Procedia Comput. Sci. **108**, 755–764 (2017)
22. Necoara, I., Nesterov, Y., Glineur, F.: Linear convergence of first order methods for non-strongly convex optimization. Math. Program. **175**(1), 69–107 (2019)

23. Nocedal, J., Wright, S.J.: Numerical Optimization. Springer (1999)
24. Orlin, J.B.: A faster strongly polynomial minimum cost flow algorithm. Oper. Res. **41**(2), 338–350 (1993)
25. Osborne, M.R.: Finite Algorithms in Optimization and Data Analysis. Wiley (1985)
26. Peña, J., Vera, J.C., Zuluaga, L.F.: New characterizations of Hoffman constants for systems of linear constraints. Math. Program. 1–31 (2020)
27. Rockafellar, R.T.: The elementary vectors of a subspace of R^N. In: Combinatorial Mathematics and Its Applications: Proceedings North Carolina Conference, Chapel Hill, 1967, pp. 104–127. The University of North Carolina Press (1969)
28. Stoer, J.: On the numerical solution of constrained least-squares problems. SIAM J. Numer. Anal. **8**(2), 382–411 (1971)
29. Tardos, É.: A strongly polynomial minimum cost circulation algorithm. Combinatorica **5**(3), 247–255 (1985)
30. Vavasis, S.A., Ye, Y.: A primal-dual interior point method whose running time depends only on the constraint matrix. Math. Program. **74**(1), 79–120 (1996)
31. Wilhelmsen, D.R.: A nearest point algorithm for convex polyhedral cones and applications to positive linear approximation. Math. Comput. **30**(133), 48–57 (1976)
32. Wolfe, P.: Finding the nearest point in a polytope. Math. Program. **11**(1), 128–149 (1976)