Improved Convergence for ℓ_∞ and ℓ_1 Regression via Iteratively Reweighted Least Squares

Alina Ene [*1] **Adrian Vladu** [*1]

Abstract

The iteratively reweighted least squares method (IRLS) is a popular technique used in practice for solving regression problems. Various versions of this method have been proposed, but their theoretical analyses failed to capture the good practical performance.

In this paper we propose a simple and natural version of IRLS for solving ℓ_∞ and ℓ_1 regression, which provably converges to a $(1+\varepsilon)$-approximate solution in $O(m^{1/3}\log(1/\varepsilon)/\varepsilon^{2/3} + \log m/\varepsilon^2)$ iterations, where m is the number of rows of the input matrix. Interestingly, this running time is independent of the conditioning of the input, and the dominant term of the running time depends sublinearly in ε^{-1}, which is atypical for the optimization of non-smooth functions.

This improves upon the more complex algorithms of Chin et al. (ITCS '12), and Christiano et al. (STOC '11) by a factor of at least $1/\varepsilon^2$, and yields a truly efficient natural algorithm for the slime mold dynamics (Straszak-Vishnoi, SODA '16, ITCS '16, ITCS '17).

1. Introduction

Regression problems are fundamental primitives in scientific computing. Among these, ℓ_∞- and ℓ_1-regression are their hardest instantiations, since through standard reductions they can be shown to be equivalent to linear programming.

While the series of works on these topics is truly extensive and diverse, the simpler methods have pervaded into the realm of practical applications. Among these, an extremely popular scheme known for its simplicity is the iteratively re-weighted least squares (IRLS) method. The idea behind it is to reduce optimization problems to iteratively solving a series of weighted ℓ_2-minimization problems, where the weights are adaptively chosen in such a way that the resulting solutions from the sequence of least-squares problems converge to the sought optimal point. In particular, due to its relevance in signal processing, ℓ_1 regression is a very important application of IRLS (Candès et al., 2006; Chartrand & Yin, 2008).

Despite the fact that various versions of this method have been studied ever since the 60's (Lawson, 1961; Osborne, 1985) theoretical understanding of their convergence has lacked. Recent works have attempted to fill this gap, and offer provable guarantees (Daubechies et al., 2010; Straszak & Vishnoi, 2016a;b;c), some of them inspired from the interpretation of this method as a dynamical system. In particular, we note the *Physarum* dynamics, which have been studied in a completely different context (Ito et al., 2011; Johannson & Zou, 2012; Tero et al., 2007; Bonifaci et al., 2012; Becchetti et al., 2013) in order to justify an experiment which revealed that a unicellular organism, the slime mold, could solve the shortest path problem in a maze (Nakagaki et al., 2000). The fact that these dynamics are essentially just a version of the IRLS method was observed in (Straszak & Vishnoi, 2016a).

Returning to the more classical world of algorithm design and analysis, it is worth observing that existing analyses of IRLS methods fall into one of the following two categories: (i) they show convergence only when the problem is properly initialized, or (ii) the guaranteed running time is prohibitive in the sense that it is highly dependent on how the input is conditioned, or it has a high polynomial dependency on the desired solution accuracy.

In this paper, we focus on analyzing simple versions of IRLS which overcome both aforementioned obstacles. In particular, our methods always converge to $1+\varepsilon$ multiplicative approximation for the objectives $\min_{\boldsymbol{x}:\boldsymbol{A}\boldsymbol{x}=\boldsymbol{b}} \|\boldsymbol{x}\|_p$, $p \in \{1,\infty\}$, in $\widetilde{O}(m^{1/3}/\varepsilon^{2/3} + 1/\varepsilon^2)$ iterations[1] of solving a weighted least squares problem, where m is the dimension of the sought vector $\boldsymbol{x}$.

[*]Equal contribution [1]Boston University, Boston, USA. Correspondence to: Alina Ene <aene@bu.edu>, Adrian Vladu <avladu@mit.edu>.

Proceedings of the 36^{th} International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019.

[1]We use $\widetilde{O}$ notation to suppress polylogarithmic factors in m/ε.

Inspiration for our methods is heavily drawn from the work of (Christiano et al., 2011), which offered a ground-breaking result by showing that in undirected graphs, a $(1+\varepsilon)$-approximate maximum flow can be found in $\widetilde{O}(m^{1/3}/\varepsilon^{11/3})$ iterations (subsequently the ε dependence was improved to $1/\varepsilon^{8/3}$, see (Chin et al., 2013)) of solving a weighted least squares problem – which in conjunction with efficient Laplacian system solvers, broke a longstanding barrier for fast graph algorithms. While these algorithms generalize to arbitrary ℓ_1 and ℓ_∞ regression problems, they are somewhat involved, in particular due to the fact that they are the product of combining the multiplicative weights update method with a regularization technique, and a second potential function.[2]

Instead, our method attempts to directly solve the non-smooth objective while tracking a single potential function. The number of iterations looks surprising, since the dominant term is $\widetilde{O}(m^{1/3}/\varepsilon^{2/3})$, whenever $\varepsilon \geq m^{-1/4}$, while classical techniques for optimizing non-smooth functions require a number of iterations that depends on the product between the function's parameters (such as Lipschitz constant of the gradient or radius of the domain), and $1/\varepsilon$ in the best case, when accelerated methods are used; see (Nesterov, 2005) for more details. [3]

Interestingly, a line of work that yielded results very similar in spirit to ours is that of approximately solving positive linear and semidefinite programs (Young, 2001; Allen-Zhu & Orecchia, 2015; Allen-Zhu et al., 2016), where the goal was to produce a first order optimization method that can be implemented in a number of iterations independent of the conditioning of the input. Improving the ε dependence to $o(1/\varepsilon^2)$ is an important open problem in this subfield.

We believe that understanding the connection between these results can pave the way for designing new efficient optimization primitives.

[2] To be more specific, Christiano et al. solve the approximate maximum flow problem, which is a specific instance of ℓ_∞ regression. Chin et al. build on this work to solve ℓ_1 regression with block structure; the block structure is relevant for their specific applications, but is a direct extension of the method, so solving vanilla ℓ_1 regression is still the main problem tackled there.

[3] We emphasize that using off-the-shelf methods, without further assumptions on the input, the number of iterations of any standard optimization method would be $\Omega(\sqrt{m})$ even for the very special instances where the affine constraint corresponds to a flow satisfying a given demand in unweighted graphs, and in general will depend on how the input matrix is conditioned, since this conditioning determines the magnitude of the subgradients or the diameter of the domain we are optimizing over. The breakthrough of Christiano et al. was the first work that managed to reduce this dependence for maximum flow, which is a specific instance of the ℓ_∞ regression problem.

1.1. Main Theorem

We state the main theorem of this paper. It follows from the correctness proofs described in Sections 3.1 and 3.2, and the convergence proofs from Lemmas A.6 and A.8.

Theorem 1.1. *There exist algorithms ℓ_∞-MINIMIZATION and ℓ_1-MINIMIZATION such that, on input $(\boldsymbol{A}, \boldsymbol{b}, \varepsilon, M)$, where $\boldsymbol{A} \in \mathbb{R}^{n\times m}$ is a matrix, $\boldsymbol{b} \in \mathbb{R}^n$ is a vector which lies in the span of $\boldsymbol{A}$'s columns, ε is an accuracy parameter, and M is a target value:*

1. *ℓ_∞-MINIMIZATION returns a solution $\boldsymbol{x}$ such that $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$, and $\|\boldsymbol{x}\|_\infty \leq (1+\varepsilon)M$, or certifies that $\min_{\boldsymbol{x}:\boldsymbol{A}\boldsymbol{x}=\boldsymbol{b}} \|\boldsymbol{x}\|_\infty \geq (1-\varepsilon)M$.*
2. *ℓ_1-MINIMIZATION returns a solution $\boldsymbol{x}$ such that $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$, and $\|\boldsymbol{x}\|_1 \leq (1+\varepsilon)M$, or certifies that $\min_{\boldsymbol{x}:\boldsymbol{A}\boldsymbol{x}=\boldsymbol{b}} \|\boldsymbol{x}\|_1 \geq (1-\varepsilon)M$.*

Furthermore both algorithms finish in

$$O\left(\frac{m^{1/3}\log(1/\varepsilon)}{\varepsilon^{2/3}} + \frac{\log m}{\varepsilon^2}\right)$$

iterations, each of which can be implemented in the time required to solve a linear system of the form $\boldsymbol{A}\boldsymbol{D}\boldsymbol{A}^\top\boldsymbol{\phi} = \boldsymbol{b}$, where $\boldsymbol{D} \in \mathbb{R}^{m\times m}$ is an arbitrary nonnegative diagonal matrix.

While our theorem statements are concerned with approximately solving a decision problem which requires a guess M on the value of the objective, it follows from standard techniques that this can be used to find a good approximation to the optimal solution without paying more than a $\widetilde{O}(1)$ overhead in the number of iterations. For completeness, we provide the details in Section D.

1.2. Relation to Previous IRLS Methods and Slime-Mold Dynamics

A popular method for solving ℓ_1 minimization is the iteratively re-weighted least squares method (IRLS). This is essentially based on the observation that whenever $\boldsymbol{x}^* = \arg\min_{\boldsymbol{A}\boldsymbol{x}=\boldsymbol{b}} \|\boldsymbol{x}\|_1$, one also has that this is the minimizer of the least squares problem $\arg\min_{\boldsymbol{x}:\boldsymbol{A}\boldsymbol{x}=\boldsymbol{b}} \langle 1/\boldsymbol{x}^*, \boldsymbol{x}^2\rangle$.[4] Hence one approach that has been employed ever since the 60's (Lawson, 1961; Osborne, 1985; Daubechies et al., 2010) is to iteratively adjust the weighting of the coordinates and re-solve the least squares problem, until $\boldsymbol{x}$ converges to a stationary point. This is rigorously described by the iteration

$$\boldsymbol{x}^{(t+1)} = \arg\min_{\boldsymbol{A}\boldsymbol{x}=\boldsymbol{b}} \left\langle \frac{1}{|\boldsymbol{x}^{(t)}|}, \boldsymbol{x}^2 \right\rangle,$$

[4] Throughout the paper we use the convention that $0/0 = 0$.

We abuse notation by applying scalar operations to vectors, with the meaning that they are applied element-wise.

Subsequent works attempted to rigorously analyze this iteration and prove convergence bounds. Oftentimes this relied on specific structure, such as $\boldsymbol{x}$ being sparse (Daubechies et al., 2010). A recent series of works drew inspiration from convergence proofs for the slime-mold dynamics – a method which essentially solves ℓ_1 minimization, based on a model used to describe the evolution of a slime mold (*Physarum polycephalum*) as it spreads through its environment in order to optimize its access to food sources (Nakagaki et al., 2000; Tero et al., 2007). Based on the intuition that these dynamics yield a method for solving the transportation problem, Straszak and Vishnoi proved in a series of works (Straszak & Vishnoi, 2016a;b;c) that this is as a matter of fact equivalent to the IRLS method, and provided a rigorous convergence analysis for a damped version of it:

$$\boldsymbol{x}^{(t+1)} = \arg\min_{\boldsymbol{A}\boldsymbol{x}=\boldsymbol{b}} \left\langle \frac{1}{\sqrt{(\boldsymbol{x}^{(t)})^2 + \eta^2}}, \boldsymbol{x}^2 \right\rangle .$$

Unfortunately their convergence proof shows that this method is highly inefficient, and the time to convergence has a high polynomial dependence in the desired accuracy, and the structure of the linear constraint.

By comparison, what we describe in this work is an IRLS method where the weights are updated according to a thresholding rule. Given a guess M for the optimal value, we perform an iteration equivalent to:

$$c_i^{(t+1)} = c_i^{(t)} \cdot \psi_{1/(1-\varepsilon)} \left(\frac{x_i^{(t)}/c_i^{(t)}}{\left\langle \frac{1}{\boldsymbol{c}^{(t)}}, \left(\boldsymbol{x}^{(t)}\right)^2 \right\rangle} \cdot M \right)^2 ,$$

$$\boldsymbol{x}^{(t+1)} = \arg\min_{\boldsymbol{A}\boldsymbol{x}=\boldsymbol{b}} \left\langle \frac{1}{\boldsymbol{c}^{(t+1)}}, \boldsymbol{x}^2 \right\rangle ,$$

where ψ is a thresholding operator i.e. $\psi_b(u) = u$, if $u \geq b$, and $\psi_b(u) = 1$ otherwise. Intuitively, this increases the weights c_i only for the elements where the corresponding component x_i^2/c_i of the quadratic objective contributes significantly, therefore we want to favor increasing it even more in the future by decreasing the weight $1/c_i$ we place on this coordinate.[5]

[5] Another way to think of this is that, ignoring the thresholding operator, the update would simply be $c_i^{(t+1)} = (x_i^{(t)})^2/c_i^{(t)} \cdot \gamma$, where γ is some normalization factor. What thresholding achieves here is to decide whether the contribution of a particular coordinate to the energy of the system is sufficiently large compared to the contributions of the entire vector $\boldsymbol{x}$.

2. Preliminaries

2.1. Basic Notation

Sets. We let $\mathbb{R}$ be the set of real numbers. For any natural number n, we write $[n] := \{1, \ldots, n\}$. We denote by Δ_m the m-dimensional simplex i.e. $\Delta_m = \{\boldsymbol{p} \in \mathbb{R}^m : \sum_{i=1}^m \boldsymbol{p}_i = 1, \boldsymbol{p}_i \geq 0 \text{ for all } i\}$.

Vectors. We let $\mathbf{0}, \mathbf{1} \in \mathbb{R}^n$ denote the all zeros and all ones vectors, respectively. When it is clear from the context, we apply scalar operations to vectors with the interpretation that they are applied coordinate-wise.

Matrices. We write matrices in bold. We use $\boldsymbol{I}$ to denote the identity matrix. Given a vector $\boldsymbol{x}$ we let $\mathbf{D}(\boldsymbol{x})$ be the diagonal matrix whose entries are given by $\boldsymbol{x}$. For a symmetric matrix $\boldsymbol{A}$, we let $\boldsymbol{A}^+$ be its Moore-Penrose pseudoinverse, i.e. $\boldsymbol{A}\boldsymbol{A}^+ = \boldsymbol{A}^+\boldsymbol{A} = \boldsymbol{I}_{\mathrm{Im}(\boldsymbol{A})}$. The pseudoinverse can be thought of as replacing all the nonzero eigenvalues of $\boldsymbol{A}$ with their reciprocals.

Inner products. When it is convenient, we use $\langle \cdot, \cdot \rangle$ notation to denote inner products. Given two vectors $\boldsymbol{x}, \boldsymbol{y}$ of equal dimensions, we let $\langle \boldsymbol{x}, \boldsymbol{y} \rangle = \boldsymbol{x}^\top \boldsymbol{y}$.

Norms. Given a vector $\boldsymbol{x}$, we denote the ℓ_p norm of $\boldsymbol{x}$ by $\|\boldsymbol{x}\|_p = (\sum x_i^p)^{1/p}$. When the subscript is dropped, we refer to the ℓ_2 norm. From this definition, we can also see that $\|\boldsymbol{x}\|_\infty = \max_i |x_i|$.

2.2. Proof Technique

Let us first understand the idea behind our ℓ_∞ minimization algorithm. The problem we aim to solve is $\min_{\boldsymbol{x}:\boldsymbol{A}\boldsymbol{x}=\boldsymbol{b}} \|\boldsymbol{x}\|_\infty$. Letting Δ_m be the m-dimensional unit simplex, we can write our objective equivalently as

$$\min_{\boldsymbol{x}:\boldsymbol{A}\boldsymbol{x}=\boldsymbol{b}} \|\boldsymbol{x}^2\|_\infty = \min_{\boldsymbol{x}:\boldsymbol{A}\boldsymbol{x}=\boldsymbol{b}} \max_{\boldsymbol{r}\in\Delta_m} \langle \boldsymbol{r}, \boldsymbol{x}^2 \rangle$$
$$= \max_{\boldsymbol{r}\in\Delta_m} \left(\min_{\boldsymbol{x}:\boldsymbol{A}\boldsymbol{x}=\boldsymbol{b}} \langle \boldsymbol{r}, \boldsymbol{x}^2 \rangle \right) := \max_{\boldsymbol{r}\in\Delta_m} \mathcal{E}_{\boldsymbol{r}}(\boldsymbol{b}) ,$$

where the second identity follows from Sion's theorem (Sion, 1958), which allows us to interchange min and max. The quantity between the parentheses has a very natural interpretation, in the case of electrical networks: it is precisely the electrical energy required to route a demand $\boldsymbol{b}$ through an electrical network encoded in $\boldsymbol{A}$. Furthermore, we have an easy way to lower bound how this energy increases whenever resistances are increased, which is a finer quantitative version of Rayleigh's monotonicity principle. More precisely, we can easily certify a lower bound on the increase in energy determined by increasing a single coordinate of $\boldsymbol{r}$. Using this observation, which we make more precise in Section A.2, we can identify a set of coordinates

of $\boldsymbol{r}$ to increase, guaranteeing that if $\boldsymbol{r}'$ is the new vector with perturbed resistances, we have

$$\frac{\mathcal{E}_{\boldsymbol{r}'}(\boldsymbol{b}) - \mathcal{E}_{\boldsymbol{r}}(\boldsymbol{b})}{\|\boldsymbol{r}' - \boldsymbol{r}\|_1} \geq M^2\,, \tag{2.1}$$

for a fixed parameter M. In the case when no coordinates of $\boldsymbol{r}$ can be increased, while preserving this property, this yields a certificate that $\boldsymbol{r}$ is as a matter of fact (close to) optimal, and thus we are done (Lemma A.5). Hence our goal becomes that of guaranteeing that $\|\boldsymbol{r}\|_1$ increases very fast. Indeed, since the "electrical energy" increases at the right rate relative to $\|\boldsymbol{r}\|_1$, after the latter has increased sufficiently, we can safely guarantee that $\mathcal{E}_{\boldsymbol{r}}(\boldsymbol{b})/\|\boldsymbol{r}\|_1 \geq (1-\varepsilon)M$, since the increase in $\|\boldsymbol{r}\|_1$ cancels out most of the initial error introduced by starting with a potentially poor solution.

The ℓ_1 minimization algorithm relies on squaring the objective, and then writing it equivalently as

$$\min_{\boldsymbol{x}:\boldsymbol{A}\boldsymbol{x}=\boldsymbol{b}} \|\boldsymbol{x}\|_1^2 = \min_{\boldsymbol{x}:\boldsymbol{A}\boldsymbol{x}=\boldsymbol{b}} \left(\min_{\boldsymbol{c}\in\Delta_m} \left\langle \frac{1}{\boldsymbol{c}}, \boldsymbol{x}^2 \right\rangle \right)$$
$$= \min_{\boldsymbol{c}\in\Delta_m} \left(\min_{\boldsymbol{x}:\boldsymbol{A}\boldsymbol{x}=\boldsymbol{b}} \left\langle \frac{1}{\boldsymbol{c}}, \boldsymbol{x}^2 \right\rangle \right) = \min_{\boldsymbol{c}\in\Delta_m} \mathcal{E}_{1/\boldsymbol{c}}(\boldsymbol{b})\,.$$

For the first identity we used the fact that $\|\boldsymbol{x}\|_1^2 = \min_{\boldsymbol{c}\in\Delta_m} \langle 1/\boldsymbol{c}, \boldsymbol{x}^2\rangle$, achieved at $\boldsymbol{c} = \boldsymbol{x}/\|\boldsymbol{x}\|_1$; see (Owen, 2007; Sun & Zhang, 2012) for further use of this trick.[6] The second identity follows from joint convexity w.r.t. $\boldsymbol{c}$ and $\boldsymbol{x}$, which can be verified by computing the Hessian of the function in $(\boldsymbol{x}, \boldsymbol{c})$. So completely oppositely from the previous case, the objective of our problem becomes minimizing electrical energy with respect to a set of inverse resistances, which we will call conductances. Note that in this case the quantity that is invariant under scaling $\boldsymbol{c}$ by a constant is $\mathcal{E}_{1/\boldsymbol{c}} \cdot \|\boldsymbol{c}\|_1$. Therefore, equivalently, our goal will be to find the set of conductances $\boldsymbol{c} \geq 0$ for which $\left(\mathcal{E}_{1/\boldsymbol{c}}\right)^{-1}/\|\boldsymbol{c}\|_1 \geq \frac{1}{(1+\varepsilon)M}$. Similarly to the ℓ_∞ case, in this case we make progress by iteratively increasing conductances from $\boldsymbol{c}$ to $\boldsymbol{c}'$ in such a way that

$$\frac{\frac{1}{\mathcal{E}_{\boldsymbol{c}'}(\boldsymbol{b})} - \frac{1}{\mathcal{E}_{\boldsymbol{c}}(\boldsymbol{b})}}{\|\boldsymbol{c}' - \boldsymbol{c}\|_1} \geq \frac{1}{M^2}\,. \tag{2.2}$$

Just as before, we can prove that unless the value of the objective can not be made smaller than M, then $\boldsymbol{c}$ can be increased while enforcing this invariant (Lemma A.7). Hence we can prove fast convergence by arguing that $\|\boldsymbol{c}\|_1$ increases very fast.

[6] Interestingly, this can also be thought of as achieving tightness for reverse Hölder's inequality whenever we are considering the dual 'norms' ℓ_{-1} and $\ell_{1/2}$.

2.3. Approximate Solutions and Infeasibility Certificates

ℓ_∞ minimization We consider the formulation

$$\min_{\boldsymbol{A}\boldsymbol{x}=\boldsymbol{b}} \|\boldsymbol{x}\|_\infty\,, \tag{2.3}$$

for which we seek an approximate solution in the following sense. Given a target value M, we aim to find one of the following:

1. an approximate solution $\boldsymbol{x}$ in the sense that $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ and $\|\boldsymbol{x}\|_\infty \leq (1+\varepsilon)M$,
2. an approximate infeasibility certificate $\boldsymbol{r}$ in the sense that $\boldsymbol{r} \in \Delta_m$ and $\boldsymbol{b}^\top(\boldsymbol{A}\mathbf{D}(\boldsymbol{r})^{-1}\boldsymbol{A}^\top)^+\boldsymbol{b} \geq (1-\varepsilon)^2M^2$.

We prove in Lemma 2.1 that the latter is indeed an infeasibility certificate.

Lemma 2.1. *Let $\boldsymbol{x}^*$ be the solution to the problem defined in Equation 2.3, and let $\boldsymbol{r} \in \Delta_m$. Then $\|\boldsymbol{x}^*\|_\infty^2 \geq \boldsymbol{b}^\top(\boldsymbol{A}\mathbf{D}(\boldsymbol{r})^{-1}\boldsymbol{A}^\top)^+\boldsymbol{b}$.*

Proof. Using Lemma A.2 we can write

$$\boldsymbol{b}^\top(\boldsymbol{A}\boldsymbol{D}(\boldsymbol{r})^{-1}\boldsymbol{A}^\top)^+\boldsymbol{b} = \min_{\boldsymbol{x}:\boldsymbol{A}\boldsymbol{x}=\boldsymbol{b}} \langle \boldsymbol{r}, \boldsymbol{x}^2\rangle \leq \langle \boldsymbol{r}, (\boldsymbol{x}^*)^2\rangle$$
$$\leq \|\boldsymbol{r}\|_1 \|\boldsymbol{x}^*\|_\infty^2 = \|\boldsymbol{x}^*\|_\infty^2\,,$$

which gives us what we needed. □

ℓ_1 minimization We consider the formulation

$$\min_{\boldsymbol{A}\boldsymbol{x}=\boldsymbol{b}} \|\boldsymbol{x}\|_1\,, \tag{2.4}$$

for which seek an approximate solution in the following sense. Given a target value M, we seek one of the following:

1. an approximate infeasibility certificate $\boldsymbol{\phi} \in \mathbb{R}^n$ in the sense that $\frac{\boldsymbol{b}^\top\boldsymbol{\phi}}{\|\boldsymbol{A}^\top\boldsymbol{\phi}\|_\infty} \geq (1-\varepsilon)M$,
2. an approximate feasibility certificate $\boldsymbol{c}$ in the sense that $\boldsymbol{c} \in \Delta_m$ and $\boldsymbol{b}^\top(\boldsymbol{A}\mathbf{D}(\boldsymbol{c})\boldsymbol{A}^\top)^+\boldsymbol{b} \leq (1+\varepsilon)^2M^2$, which yields an approximately feasible solution $\boldsymbol{x} = \mathbf{D}(\boldsymbol{c})\boldsymbol{A}^\top(\boldsymbol{A}\mathbf{D}(\boldsymbol{c})\boldsymbol{A}^\top)^+\boldsymbol{b}$ in the sense that $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ and $\|\boldsymbol{x}\|_1 \leq (1+\varepsilon)M$.

The fact that the former is an approximate infeasibility certificate follows from convex duality. Indeed, one can see that the dual of the minimization problem is $\max_{\boldsymbol{\phi}:\|\boldsymbol{A}^\top\boldsymbol{\phi}\|_\infty\leq 1} \boldsymbol{b}^\top\boldsymbol{\phi}$, so exhibiting a solution as above implies that the value of this objective is at least $(1-\varepsilon)M$. A proof for the fact that the latter is indeed an approximate feasibility certificate, and that it yields an approximately feasible solution can be found in Lemma 2.2.

Lemma 2.2. *Given $\boldsymbol{c} \in \Delta_m$, the vector $\boldsymbol{x} = \mathbf{D}(\boldsymbol{c})\boldsymbol{A}^\top(\boldsymbol{A}\mathbf{D}(\boldsymbol{c})\boldsymbol{A}^\top)^+\boldsymbol{b}$ satisfies $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$, and $\|\boldsymbol{x}\|_1^2 \leq \boldsymbol{b}^\top(\boldsymbol{A}\mathbf{D}(\boldsymbol{c})\boldsymbol{A}^\top)^+\boldsymbol{b}$.*

Proof. The fact that $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ follows directly by substitution, and using the fact that $\boldsymbol{b} \in \mathrm{Im}(\boldsymbol{A})$. Using Lemma A.2 and the definition in (A.1) we write

$$\begin{aligned}
&\boldsymbol{b}^\top(\boldsymbol{A}\mathbf{D}(\boldsymbol{c})\boldsymbol{A}^\top)^+\boldsymbol{b} \\
&= \boldsymbol{b}^\top(\boldsymbol{A}\mathbf{D}(\boldsymbol{c})\boldsymbol{A}^\top)^+(\boldsymbol{A}\mathbf{D}(\boldsymbol{c})\boldsymbol{A}^\top)(\boldsymbol{A}\mathbf{D}(\boldsymbol{c})\boldsymbol{A}^\top)^+\boldsymbol{b} \\
&= \sum_{i=1}^m \frac{1}{c_i} \cdot \left(\mathbf{D}(\boldsymbol{c})\boldsymbol{A}^\top(\boldsymbol{A}\mathbf{D}(\boldsymbol{c})\boldsymbol{A}^\top)^+\boldsymbol{b}\right)^2 = \sum_{i=1}^m \frac{1}{c_i} \cdot x_i^2 \,.
\end{aligned}$$

We can use this identity inside the following upper bound, which we obtain by applying Cauchy-Schwarz:

$$\begin{aligned}
\|\boldsymbol{x}\|_1 = \sum_{i=1}^m \frac{|x_i|}{\sqrt{c_i}} \cdot \sqrt{c_i} &\leq \sqrt{\left(\sum_{i=1}^m \frac{x_i^2}{c_i}\right)\left(\sum_{i=1}^m c_i\right)} \\
&= \sqrt{\boldsymbol{b}^\top(\boldsymbol{A}\mathbf{D}(\boldsymbol{c})\boldsymbol{A}^\top)^+\boldsymbol{b}} \,.
\end{aligned}$$

This yields our claim. □

3. The Algorithms

Having introduced the necessary notation, we can describe our simple IRLS routine. We prove convergence in Section A.

3.1. The ℓ_∞ Minimization Algorithm

We first present the algorithm for the ℓ_∞ version of the problem, since it is the most intuitive. The method attempts to find a weighting of the columns of $\boldsymbol{A}$ i.e. a vector $\boldsymbol{r} \in \mathbb{R}^m$ for which the corresponding least squares solution has a small ℓ_∞ norm; more precisely $\|\boldsymbol{x}\|_\infty / \|\boldsymbol{r}\|_1 \leq (1+\varepsilon)M$ for some chosen target value M.

Then the weighting is updated via the following simple thresholding rule. Elements for which the corresponding coordinate of the least squares solution x_i is below the desired target value are left unchanged. The others are scaled exactly by the amount by which the square of the corresponding coordinate x_i violates the desired threshold i.e. x_i^2/M^2.

Note that the iteration defined here simply attempts to construct an infeasibility certificate for the problem defined in Equation 2.3. Building the feasible solution involves maintains a solution obtained by uniformly averaging a subset of the iterates $\boldsymbol{x}$ witnessed so far. These are used to return the approximately feasible solution in case the algorithm fails to quickly produce an (approximate) infeasibility certificate. The details referring to how and why we perform this specific set of updates are explained in the convergence proof. The steps involved in building this feasible solution are written in blue. They can be ignored if the goal is simply that of returning a yes/no answer.

Algorithm 1 ℓ_∞-MINIMIZATION$(\boldsymbol{A}, \boldsymbol{b}, \varepsilon, M)$

1: **Input:** Matrix $\boldsymbol{A} \in \mathbb{R}^{n\times m}$, vector $\boldsymbol{b} \in \mathbb{R}^n$, accuracy ε, target value M.
2: **Output:** Vector $\boldsymbol{x}$ such that $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ and $\|\boldsymbol{x}\|_\infty \leq (1+\varepsilon)M$, or approximate infeasibility certificate $\boldsymbol{r} \in \Delta_m$.
3: $t = 0$, $\boldsymbol{r}^{(0)} = \mathbf{1}/m$.
4: $t' = 0$, $\boldsymbol{s}^{(t')} = \vec{0}$.
5: **while** $\|\boldsymbol{r}^{(t)}\|_1 \leq 1/\varepsilon$ **do**
6: $\quad \boldsymbol{x}^{(t)} = \arg\min_{\boldsymbol{x}:\boldsymbol{A}\boldsymbol{x}=\boldsymbol{b}} \langle \boldsymbol{r}, \boldsymbol{x}^2 \rangle$.
$\quad$ // Equivalently,
$\quad x^{(t)} = \mathbf{D}(r)^{-1}\boldsymbol{A}^\top\left(\boldsymbol{A}\mathbf{D}(r)^{-1}\boldsymbol{A}^\top\right)^+ b\,.$
7: $\quad$ **if** $\|\boldsymbol{x}^{(t)}\|_\infty \leq m^{1/3} \cdot M$ **then**
8: $\quad\quad t' = t'+1$, $\boldsymbol{s}^{(t')} = \boldsymbol{s}^{(t'-1)} + \boldsymbol{x}^{(t)}$.
9: $\quad$ **end if**
10: $\quad$ **if** $\|\boldsymbol{s}^{(t')}\|_\infty / t' \leq (1+\varepsilon)M$ **then**
11: $\quad\quad$ **return** $\boldsymbol{s}^{(t')}/t'$.
12: $\quad$ **end if**
13: $\quad \alpha_i^{(t)} = \begin{cases} 1 & \text{if } |x_i^{(t)}| < (1+\varepsilon)M, \\ \frac{(x_i^{(t)})^2}{M^2} & \text{otherwise.} \end{cases}$
14: $\quad$ **if** $\boldsymbol{\alpha}^{(t)} = \mathbf{1}$ **then**
15: $\quad\quad$ **return** $\boldsymbol{x}^{(t)}$.
16: $\quad$ **end if**
17: $\quad \boldsymbol{r}^{(t+1)} = \boldsymbol{r}^{(t)} \cdot \boldsymbol{\alpha}^{(t)}$.
18: $\quad t = t+1$.
19: **end while**
20: **return** $\boldsymbol{r}/\|\boldsymbol{r}\|_1$.

Correctness. We notice that Algorithm 1 has two possible outcomes. Either it returns a primal approximately feasible vector (lines 11and 15), or returns a dual certificate (line 20). In the former case, it is clear from the description of the algorithm that the returned vector is indeed approximately feasible: line 11 returns a uniform average of vectors satisfying the linear constraint with small ℓ_∞ norm; line 20 returns the $\boldsymbol{x}^{(t)}$ computed within the corresponding iteration, whenever $\boldsymbol{\alpha}^{(t)} = \vec{1}$, i.e. $\|\boldsymbol{x}^{(t)}\|_\infty < (1+\varepsilon)M$.

Also, note that in case none of these stopping conditions is triggered, the algorithm returns a dual certificate on line 20 after a finite number of iterations. Indeed, note that every iteration where $\alpha_i^{(t)} \neq \vec{1}$, at least one element of $\boldsymbol{r}^{(t)}$ gets increased by a factor of at least $(1+\varepsilon)^2$, due to way $\boldsymbol{\alpha}^{(t)}$ is defined. Since the algorithm stops when $\|\boldsymbol{r}^{(t)}\|_1 = 1/\varepsilon$, no element of $\boldsymbol{r}$ can be scaled more than $O(\log_{(1+\varepsilon)}(m/\varepsilon))$ times, hence the total number of iterations is very roughly upper bounded by $O(m\log(m/\varepsilon)/\varepsilon)$. We will see in Sec-

Algorithm 2 ℓ_1-MINIMIZATION$(\boldsymbol{A}, \boldsymbol{b}, \varepsilon, M)$

1: **Input:** Matrix $\boldsymbol{A} \in \mathbb{R}^{n\times m}$, vector $\boldsymbol{b} \in \mathbb{R}^n$, accuracy ε, target value M.
2: **Output:** Vector $\boldsymbol{x}$ such that $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ and $\|\boldsymbol{x}\|_1 \leq (1+\varepsilon)M$, or approximate infeasibility certificate $\boldsymbol{\phi} \in \Delta_n$.
3: $t = 0$, $\boldsymbol{c}^{(0)} = \mathbf{1}/m$.
4: $t' = 0$, $\boldsymbol{s}^{(t')} = \mathbf{0}$, $\boldsymbol{\Phi}^{(0)} = \mathbf{0}$.
5: **while** $\|\boldsymbol{c}^{(t)}\|_1 \leq 1 + \frac{1}{(1+\varepsilon)^2-1}$ **do**
6: $\quad \boldsymbol{\phi}^{(t)} = \left(\boldsymbol{A}\mathbf{D}(\boldsymbol{c})\boldsymbol{A}^\top\right)^+ \boldsymbol{b}$. // Equivalently, $\boldsymbol{\phi}^{(t)}$ is the vector of potentials which induce the electrical flow $\boldsymbol{x} = \arg\min_{\boldsymbol{A}x=\boldsymbol{b}}\langle 1/\boldsymbol{c}, \boldsymbol{x}^2\rangle$ via $\boldsymbol{x} = \mathbf{D}(\boldsymbol{c})\boldsymbol{A}^\top\boldsymbol{\phi}$.
7: $\quad$ **if** $\left\|\frac{\boldsymbol{A}^\top\boldsymbol{\phi}^{(t)}}{\boldsymbol{b}^\top\boldsymbol{\phi}^{(t)}}\right\|_\infty \leq m^{1/3}\cdot\frac{1}{M}$ **then**
8: $\quad\quad t' = t'+1$, $\boldsymbol{s}^{(t')} = \boldsymbol{s}^{(t'-1)} + \left|\frac{\boldsymbol{A}^\top\boldsymbol{\phi}^{(t)}}{\boldsymbol{b}^\top\boldsymbol{\phi}^{(t)}}\right|$, $\boldsymbol{\Phi}^{(t')} = \boldsymbol{\Phi}^{(t'-1)} + \frac{\boldsymbol{\phi}^{(t)}}{\boldsymbol{b}^\top\boldsymbol{\phi}^{(t)}}$.
9: $\quad$ **end if**
10: $\quad$ **if** $\|\boldsymbol{s}^{(t')}\|_\infty/t' \leq \frac{1}{(1-\varepsilon)M}$ **then**
11: $\quad\quad$ **return** $\boldsymbol{\Phi}^{(t')}/t'$.
12: $\quad$ **end if**
13: $\quad \alpha_i^{(t)} = \begin{cases} 1 & \text{if } \frac{|\boldsymbol{A}^\top\boldsymbol{\phi}^{(t)}|_i}{\boldsymbol{b}^\top\boldsymbol{\phi}^{(t)}} \leq \frac{1}{(1-\varepsilon)M}, \\ \left(\frac{(\boldsymbol{A}^\top\boldsymbol{\phi}^{(t)})_i}{\boldsymbol{b}^\top\boldsymbol{\phi}^{(t)}}\right)^2 \cdot M^2 & \text{otherwise.} \end{cases}$
14: $\quad$ **if** $\boldsymbol{\alpha}^{(t)} = \mathbf{1}$ **then**
15: $\quad\quad$ **return** $\boldsymbol{\phi}^{(t)}$.
16: $\quad$ **end if**
17: $\quad \boldsymbol{c}^{(t+1)} = \boldsymbol{c}^{(t)}\cdot\boldsymbol{\alpha}^{(t)}$.
18: $\quad t = t+1$.
19: **end while**
20: **return** $\boldsymbol{x} = \mathbf{D}(\boldsymbol{c})\boldsymbol{A}^\top\boldsymbol{\phi}^{(t)}$.

tion A that we can prove a much finer upper bound.

Finally, we need to argue that whenever the algorithm returns on line 20, it returns an infeasibility certificate as per Lemma 2.1. We defer the proof to Lemma A.5 in Section A.

3.2. The ℓ_1 Minimization Algorithm

The ℓ_1 version is very similar. As a matter of fact, it can be re-derived simply by attempting to solve the convex dual of the problem from (2.3), which is an ℓ_∞ minimization problem, by using the routine from Figure 1. However, since the reduction requires several, and previous works attempted to solve this directly using various versions of IRLS, we provide a natural iteration which does not involve any reductions.

Correctness. We notice that Algorithm 2 has two possible outcomes. Either it returns an approximate infeasibility certificate (lines 11 and 15), or returns an approximately feasible solution (line 20).

Let us verify that in the former case the returned vector is indeed an approximate infeasibility certificate. Line 11 returns $\boldsymbol{\Phi}^{(t')} = \sum_{t\in S}\frac{\boldsymbol{\phi}^{(t)}}{\boldsymbol{b}^\top\boldsymbol{\phi}^{(t)}}$, where we know that S is a set for which

$$\left\|\boldsymbol{A}^\top\boldsymbol{\Phi}^{(t')}\right\|_\infty = \left\|\boldsymbol{A}^\top\cdot\sum_{t\in S}\frac{\boldsymbol{\phi}^{(t)}}{\boldsymbol{b}^\top\boldsymbol{\phi}^{(t)}}\right\|_\infty$$
$$= \left\|\sum_{t\in S}\frac{\boldsymbol{A}^\top\boldsymbol{\phi}^{(t)}}{\boldsymbol{b}^\top\boldsymbol{\phi}^{(t)}}\right\|_\infty \leq \left\|\sum_{t\in S}\left|\frac{\boldsymbol{A}^\top\boldsymbol{\phi}^{(t)}}{\boldsymbol{b}^\top\boldsymbol{\phi}^{(t)}}\right|\right\|_\infty \leq \frac{t'}{(1-\varepsilon)M}.$$

Since $\boldsymbol{b}^\top\boldsymbol{\Phi}^{(t')} = t'$, we see that returned vector $\boldsymbol{\Phi}^{(t')}/t'$ is an approximate infeasibility certificate, as defined in Section 2.3. If the algorithm returns on line 15, we get that $\left\|\frac{\boldsymbol{A}^\top\boldsymbol{\phi}^{(t)}}{\boldsymbol{b}^\top\boldsymbol{\phi}^{(t)}}\right\|_\infty \leq \frac{1}{(1-\varepsilon)M}$, hence $\boldsymbol{\phi}^{(t)}$ is an approximate infeasibility certificate.

Also, note that in case none of these stopping conditions is triggered, the algorithm returns a solution on line 20 after a finite number of iterations. Indeed, just as in the ℓ_∞ case, in every iteration some conductance gets increased by a factor of at least $\Omega(1+\varepsilon)$, hence the algorithm must stop in finite time. We provide a rigorous analysis of the time required for convergence in Section A.

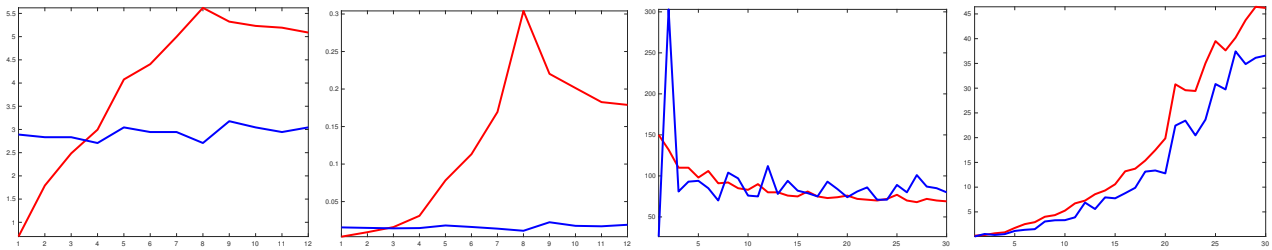
Finally, we need to argue that whenever the algorithm returns a solution on line 20, it is indeed an approximately feasible solution. We defer the proof to Lemma A.7 in Section A.

4. Experiments

We test both our resistance/conductance update schemes in order to verify that the resulting algorithms converge fast in practice. We slightly modify the schemes such that they always update their target value M depending on the value of the objective they have achieved so far. We stop when given the history of witnessed iterates, we can certify a sufficiently small duality gap. For solving linear systems, we used the conjugate gradient implementation from the ℓ_1-MAGIC optimization suite (Candès & Romberg).

We test both algorithms while varying ε, and varying m. We consider both the update scheme given by our algorithms from Section 3, and one where we attempt to double the length of the step for as long as the invariants from (2.1) and (2.2), respectively, are maintained. We notice that in general, using this long step strategy, we improve both the number of iterations and the running time.

The plots corresponding to the standard update scheme (short-steps) are drawn in red, those corresponding to the long-step version are drawn in blue.

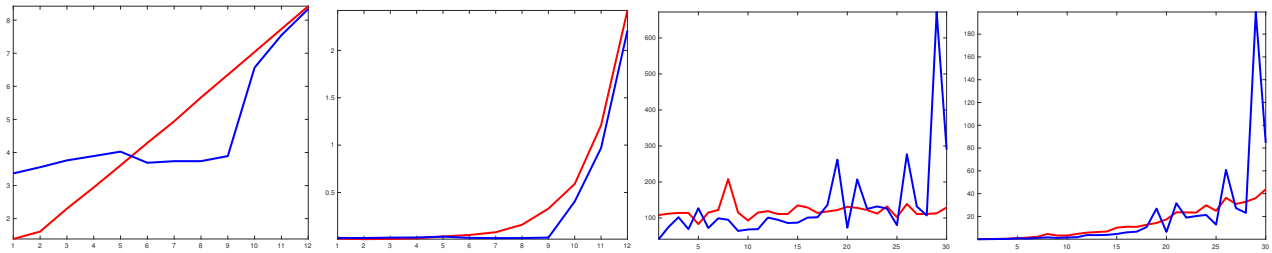


(a) $\log$(iterations) for ℓ_∞ minimization as function of $\log(\varepsilon^{-1})$.

(b) Time (sec) for ℓ_∞ minimization as function of $\log(\varepsilon^{-1})$.

(c) Number of iterations for ℓ_∞ minimization as function of $m/200$.

(d) Time (sec) for ℓ_∞ minimization as function of $m/200$.



(e) $\log$(iterations) for ℓ_1 minimization as function of $\log(\varepsilon^{-1})$.

(f) Time (sec) for ℓ_1 minimization as function of $\log(\varepsilon^{-1})$.

(g) Number of iterations for ℓ_1 minimization as function of $m/200$.

(h) Time (sec) for ℓ_1 minimization as function of $m/200$.

Figure 3.1: Experimental results.

The experiments where we varied ε are reported in figures 1(a), 1(b), 1(e), and 1(f). For all these experiments, the input consists a random 150×200 matrix $\boldsymbol{A}$ with orthogonal rows, and a vector $\boldsymbol{b}$ obtained from applying $\boldsymbol{A}$ to a ± 1-vector of sparsity 15. We plot the number of iterations/running time of the algorithm for $\varepsilon = 1/2^k$, where $k \in \{1, \ldots, 12\}$.

We notice that for these experiments, the number of iterations for the short-step version does indeed scale linearly with ε^{-1}; the long-step version makes significant gains in the ℓ_∞ case.

The experiments where we varied m are reported in figures 1(c), 1(d), 1(g), and 1(h). For all these experiments, the input consists of a random $150 \times (200 \cdot k)$ matrix $\boldsymbol{A}$ with orthogonal vectors, and a vector $\boldsymbol{b}$ obtained from applying $\boldsymbol{A}$ to a ± 1-vector of sparsity 15, and a fixed accuracy $\varepsilon = .01$. We plot the number of iterations required by the algorithm for $k \in \{1, \ldots, 30\}$.

We notice that for these experiments, both the number of iterations and the running time scale significantly better than by $m^{1/3}$, which suggests that this polynomial dependence in m depends on the input structure, and can be avoided in practice.

Acknowledgements

AE was partially supported by NSF CAREER grant CCF-1750333 and NSF grant CCF-1718342. AV was partially supported by NSF grant CCF-1718342.

References

Allen-Zhu, Z. and Orecchia, L. Using optimization to break the epsilon barrier: A faster and simpler width-independent algorithm for solving positive linear programs in parallel. In *Proceedings of the twenty-sixth annual ACM-SIAM symposium on Discrete algorithms*, pp. 1439–1456. Society for Industrial and Applied Mathematics, 2015.

Allen-Zhu, Z., Lee, Y. T., and Orecchia, L. Using optimization to obtain a width-independent, parallel, simpler, and faster positive sdp solver. In *Proceedings of the twenty-seventh annual ACM-SIAM symposium on Discrete algorithms*, pp. 1824–1831. Society for Industrial and Applied Mathematics, 2016.

Becchetti, L., Bonifaci, V., Dirnberger, M., Karrenbauer, A., and Mehlhorn, K. Physarum can compute shortest paths: Convergence proofs and complexity bounds. In *International Colloquium on Automata, Languages, and Programming*, pp. 472–483. Springer, 2013.

Bonifaci, V., Mehlhorn, K., and Varma, G. Physarum can compute shortest paths. *Journal of theoretical biology*, 309:121–133, 2012.

Candès, E. and Romberg, J. ℓ_1-MAGIC: Recovery of sparse signals via convex programming. URL `https://statweb.stanford.edu/~candes/l1magic/`.

Candès, E. J. et al. Compressive sampling. In *Proceedings of the international congress of mathematicians*, volume 3, pp. 1433–1452. Madrid, Spain, 2006.

Chartrand, R. and Yin, W. Iterative reweighted algorithms for compressive sensing. Technical report, 2008.

Chin, H. H., Madry, A., Miller, G. L., and Peng, R. Runtime guarantees for regression problems. In Kleinberg, R. D. (ed.), *Innovations in Theoretical Computer Science, ITCS '13, Berkeley, CA, USA, January 9-12, 2013*, pp. 269–282. ACM, 2013. ISBN 978-1-4503-1859-4. doi: 10.1145/2422436.2422469. URL https://doi.org/10.1145/2422436.2422469.

Christiano, P., Kelner, J. A., Madry, A., Spielman, D. A., and Teng, S. Electrical flows, laplacian systems, and faster approximation of maximum flow in undirected graphs. In Fortnow, L. and Vadhan, S. P. (eds.), *Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC 2011, San Jose, CA, USA, 6-8 June 2011*, pp. 273–282. ACM, 2011. ISBN 978-1-4503-0691-1. doi: 10.1145/1993636.1993674. URL https://doi.org/10.1145/1993636.1993674.

Daubechies, I., DeVore, R., Fornasier, M., and Güntürk, C. S. Iteratively reweighted least squares minimization for sparse recovery. *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences*, 63(1):1–38, 2010.

Ito, K., Johansson, A., Nakagaki, T., and Tero, A. Convergence properties for the physarum solver. *arXiv preprint arXiv:1101.5249*, 2011.

Johannson, A. and Zou, J. A slime mold solver for linear programming problems. In *Conference on Computability in Europe*, pp. 344–354. Springer, 2012.

Lawson, C. *Contributions to the Theory of Linear Least Maximum Approximation*. University of California, Los Angeles, 1961. URL https://books.google.at/books?id=b_CtbwAACAAJ.

Nakagaki, T., Yamada, H., and Tóth, Á. Intelligence: Maze-solving by an amoeboid organism. *Nature*, 407(6803): 470, 2000.

Nesterov, Y. Smooth minimization of non-smooth functions. *Mathematical programming*, 103(1):127–152, 2005.

Osborne, M. R. *Finite algorithms in optimization and data analysis*. 1985.

Owen, A. B. A robust hybrid of lasso and ridge regression. *Contemporary Mathematics*, 443(7):59–72, 2007.

Sherman, J. Area-convexity, l &infty; regularization, and undirected multicommodity flow. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pp. 452–460. ACM, 2017.

Sion, M. On general minimax theorems. *Pacific Journal of mathematics*, 8(1):171–176, 1958.

Straszak, D. and Vishnoi, N. K. On a natural dynamics for linear programming. In Sudan, M. (ed.), *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science, Cambridge, MA, USA, January 14-16, 2016*, pp. 291. ACM, 2016a. ISBN 978-1-4503-4057-1. doi: 10.1145/2840728.2840762. URL https://doi.org/10.1145/2840728.2840762.

Straszak, D. and Vishnoi, N. K. Natural algorithms for flow problems. In Krauthgamer, R. (ed.), *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pp. 1868–1883. SIAM, 2016b. ISBN 978-1-61197-433-1. doi: 10.1137/1.9781611974331.ch131. URL https://doi.org/10.1137/1.9781611974331.ch131.

Straszak, D. and Vishnoi, N. K. IRLS and slime mold: Equivalence and convergence. *CoRR*, abs/1601.02712, 2016c. URL http://arxiv.org/abs/1601.02712.

Sun, T. and Zhang, C.-H. Scaled sparse linear regression. *Biometrika*, 99(4):879–898, 2012.

Tero, A., Kobayashi, R., and Nakagaki, T. A mathematical model for adaptive transport network in path finding by true slime mold. *Journal of theoretical biology*, 244(4): 553–564, 2007.

Young, N. E. Sequential and parallel algorithms for mixed packing and covering. In *Foundations of Computer Science, 2001. Proceedings. 42nd IEEE Symposium on*, pp. 538–546. IEEE, 2001.