Algorithms for Lipschitz Learning on Graphs [*]

Rasmus Kyng   RASMUS.KYNG@YALE.EDU
Yale University

Anup Rao   ANUP.RAO@YALE.EDU
Yale University

Sushant Sachdeva   SUSHANT.SACHDEVA@YALE.EDU
Yale University

Daniel A. Spielman   SPIELMAN@CS.YALE.EDU
Yale University

Abstract

We develop fast algorithms for solving regression problems on graphs where one is given the value of a function at some vertices, and must find its smoothest possible extension to all vertices. The extension we compute is the absolutely minimal Lipschitz extension, and is the limit for large p of p-Laplacian regularization. We present an algorithm that computes a minimal Lipschitz extension in expected linear time, and an algorithm that computes an absolutely minimal Lipschitz extension in expected time $\widetilde{O}(mn)$. The latter algorithm has variants that seem to run much faster in practice. These extensions are particularly amenable to regularization: we can perform l_0-regularization on the given values in polynomial time and l_1-regularization on the initial function values and on graph edge weights in time $\widetilde{O}(m^{3/2})$.

Our definitions and algorithms naturally extend to directed graphs.

1. Introduction

We consider a problem where we are given a weighted undirected graph $G = (V, E, \ell)$ and values $v_0 : T \to \mathbb{R}$ on a subset T of its vertices. We view the weights ℓ as indicating the lengths of edges, with shorter length indicating greater similarity. Our goal it to assign values to every vertex $v \in V \backslash T$ so that the values assigned are as *smooth* as possible across edges. A minimal Lipschitz extension of v_0 is a vector v that minimizes

$$\max_{(x,y)\in E} (\ell(x,y))^{-1} \left| v(x) - v(y) \right|, \tag{1}$$

subject to $v(x) = v_0(x)$ for all $x \in T$. We call such a vector an inf-minimizer. Inf-minimizers are not unique. So, among inf-minimizers we seek vectors that minimize the second-largest absolute value of $\ell(x,y)^{-1} \left| v(x) - v(y) \right|$ across edges, and then the third-largest given that, and so on. We call such a vector v a lex-minimizer. It is also known as an absolutely minimal Lipschitz extension of v_0.

These are the limit of the solution to p-Laplacian minimization problems for large p, *i.e.*,

$$\operatorname*{arg\,min}_{\substack{v \in \mathbb{R}^n \\ v|_T = v_0|_T}} \sum_{(x,y)\in E} (\ell(x,y))^{-p} |v(x) - v(y)|^p. \tag{2}$$

The use of $p = 2$ was suggested in the foundational paper of Zhu et al. (2003), and is particularly nice because it can be obtained by solving a system of linear equations in a symmetric diagonally dominant matrix, which can be done very quickly (Spielman and Teng (2004); Cohen et al. (2014)). The use of larger values of p has been discussed by Alamgir and Luxburg (2011), and by Bridle and Zhu (2013), but it is much more complicated to compute. The fastest algorithms we know for this problem require convex programming, and then require very high accuracy to obtain the values at most vertices. By taking the limit as p goes to infinity, we recover the lex-minimizer, which we will show can be computed quickly.

The lex-minimization problem has a remarkable amount of structure. For example, in uniformly weighted graphs the value of the lex-minimizer at every vertex not in T is equal to the average of the minimum and maximum of the values at its neighbors. This is analogous to the property of the 2-Laplacian minimizer that the value at every vertex not in T equals the average of the values at its neighbors.

Contributions. We first present several important structural properties of lex-minimizers in Section 3.1. As we shall point out, some of these were known from previous work, sometimes in restricted settings. We state them generally and prove them for completeness. We also prove that the lex-minimizer is as stable as possible under perturbations of v_0 (Theorem 7).

The structure of the lex-minimization problem has led us to develop elegant algorithms for its solution. Both the algorithms and their analyses could be taught to undergraduates. We believe that these algorithms could be used in place of 2-Laplacian minimization in many applications.

We present algorithms for the following problems. Throughout, $m = |E|$ and $n = |V|$.

Inf-minimization: An algorithm that runs in expected time $O(m + n \log n)$ (Section 4.3).

Lex-minimization: An algorithm that runs in expected time $O(n(m + n \log n))$ (Section 4), along with a variant that runs quickly in practice (Section 4.4).

l_1**-regularization of edge lengths for inf-minimization:** The problem of minimizing (1) given a limited budget with which one can increase edge lengths is a linear programming problem. We show how to solve it in time $\widetilde{O}(m^{3/2})$ with an interior point method by using fast Laplacian solvers (Section 8). The same algorithm can accommodate l_1-regularization of the values given in v_0.

l_0**-regularization of vertex values for inf-minimization:** We give a polynomial time algorithm for l_0-regularization of the values at vertices. That is, we minimize (1) given a budget of a number of vertices that can be proclaimed outliers and removed from T (Section 7.1). We solve this problem by reducing it to the problem of computing minimum vertex covers on transitively closed directed acyclic graphs, a special case of minimum vertex cover that can be solved in polynomial time.

After any regularization for inf-minimization, we suggest computing the lex-minimizer. We find the result for l_0-regularization of vertex values to be particularly surprising, especially because we prove that the analogous problem for 2-Laplacian minimization is NP-Hard (Section 7.2).

All of our algorithms extend naturally to **directed graphs** (Section 5). This is in contrast with the problem of minimizing 2-Laplacians on directed graphs, which corresponds to computing electrical flows in networks of resistors and diodes, for which fast algorithms are not presently known.

We present a few **experiments** on examples demonstrating that the lex-minimizer can overcome known deficiencies of the 2-Laplacian minimizer (Section 1.1, Figures 1,2), as well as a demonstration of the performance of the directed analog of our algorithms on the WebSpam dataset of Castillo et al. (2006) (Section 6). In the WebSpam problem we use the link structure of a collection of web sites to flag some sites as spam, given a small number of labeled sites known to be spam or normal.
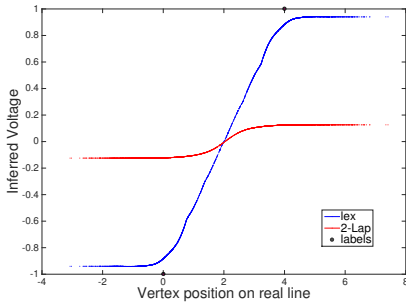
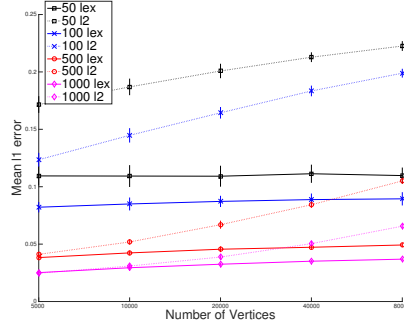Figure 1: Lex vs 2-Laplacian on 1D gaussian clusters.

Figure 2: kNN graphs on samples from 4D cube.

1.1. Relation to Prior Work

We first encountered the idea of using the minimizer of the 2-Laplacian given by (2) for regression and classification on graphs in the work of Zhu et al. (2003) and Belkin et al. (2004) on semi-supervised learning. These works transformed learning problems on sets of vectors into problems on graphs by identifying vectors with vertices and constructing graphs with edges between nearby vectors. One shortcoming of this approach (see Nadler et al. (2009), Alamgir and Luxburg (2011), Bridle and Zhu (2013)) is that if the number of vectors grows while the number of labeled vectors remains fixed, then almost all the values of the 2-Laplacian minimizer converge to the mean of the labels on most natural examples. For example, Nadler et al. (2009) consider sampling points from two Gaussian distributions centered at 0 and 4 on the real line. They place edges between every pair of points (x, y) with length $\exp(|x - y|^2 / 2\sigma^2)$ for $\sigma = 0.4$, and provide only the labels $v_0(0) = -1$ and $v_0(4) = 1$. Figure 1 shows the values of the 2-Laplacian minimizer in red, which are all approximately zero. In contrast, the values of the lex-minimizer in blue, which are smoothly distributed between the labeled points, are shown.

The "manifold hypothesis" (see Chapelle et al. (2010), Ma and Fu (2011)) holds that much natural data lies near a low-dimensional manifold and that natural functions we would like to learn on this data are smooth functions on the manifold. Under this assumption, one should expect lex-minimizers to interpolate well. In contrast, the 2-Laplacian minimizers degrade (dotted lines) if the number of labeled points remains fixed while the total number of points grows. In Figure 2, we demonstrate this by sampling many points uniformly from the unit cube in 4 dimensions, form their 8-nearest neighbor graph, and consider the problem of regressing the first coordinate. We performed 8 experiments, varying the number of labeled points in $\{50, 100, 500, 1000\}$. Each data point is the mean average l_1 error over 100 experiments. The plots for root mean squared error are similar. The standard deviation of the estimations of the mean are within one pixel, and so are not displayed. The performance of the lex-minimizer (solid lines) does not degrade as the number of unlabeled points grows.

Analogous to our inf-minimizers, minimal Lipschitz extensions of functions in Euclidean space and over more general metric spaces have been studied extensively (Kirszbraun (1934), McShane (1934), Whitney (1934)). von Luxburg and Bousquet (2003) employ Lipschitz extensions on metric spaces for classification and relate these to Support Vector Machines. Their work inspired improvements in classification and regression in metric spaces with low doubling dimension (Gottlieb et al. (2013), Gottlieb et al. (2013b)). Theoretically fast, although not actually practical, algorithms have been given for constructing minimal Lipschitz extensions of functions on low-dimensional Euclidean spaces (Fefferman (2009a), Fefferman and Klartag (2009), Fefferman (2009b)). Sinop and Grady (2007) suggest using inf-minimizers for binary clas-

sification problems on graphs. For this special case, where all of the given values are either 0 or 1, they present an $O(m + n \log n)$ time algorithm for computing an inf-minimizer. The case of general given values, which we solve in this paper, is much more complicated. To compensate for the non-uniqueness of inf-minimizers, they suggest choosing the inf-minimizer that minimizes (2) with $p = 2$. We believe that the lex-minimizer is a more natural choice.

The analog of our lex-minimizer over continuous spaces is called the absolutely minimal Lipschitz extension (AMLE). Starting with the work of Aronsson (1967), there have been several characterizations and proofs of the existence and uniqueness of the AMLE (Jensen (1993), Crandall et al. (2001), Barles and Busca (2001), Aronsson et al. (2004)). Many of these results were later extended to general metric spaces, including graphs (Milman (1999), Peres et al. (2011), Naor and Sheffield (2010), Sheffield and Smart (2010)). However, to the best of our knowledge, fast algorithms for computing lex-minimizers on graphs were not known. For the special case of undirected, unweighted graphs, Lazarus et al. (1999) presented both a polynomial-time algorithm and an iterative method. Oberman (2011) suggested computing the AMLE in Euclidean space by first discretizing the problem and then solving the corresponding graph problem by an iterative method. However, no run-time guarantees were obtained for either iterative method.

2. Notation and Basic Definitions

Lexicographic Ordering. Given a vector $r \in \mathbb{R}^m$, let π_r denote a permutation that sorts r in non-increasing order by absolute value, *i.e.*, $\forall i \in [m-1], |r(\pi_r(i))| \geq |r(\pi_r(i+1))|$. Given two vectors $r, s \in \mathbb{R}^m$, we write $r \preceq s$ to indicate that r is smaller than s in the *lexicographic ordering* on sorted absolute values, *i.e.*

$$\exists j \in [m], \big|r(\pi_r(j))\big| < \big|s(\pi_s(j))\big| \text{ and } \forall i \in [j-1], \big|r(\pi_r(i))\big| = \big|s(\pi_s(i))\big|$$
$$\text{or } \forall i \in [m], \big|r(\pi_r(i))\big| = \big|s(\pi_s(i))\big|.$$

Note that it is possible that $r \preceq s$ and $s \preceq r$ while $r \neq s$. It is a total relation: for every r and s at least one of $r \preceq s$ or $s \preceq r$ is true.

Graphs and Matrices. We will work with weighted graphs. Unless explicitly stated, we will assume that they are undirected. For a graph G, we let V_G be its set of vertices, E_G be its set of edges, and $\ell_G : E_G \to \mathbb{R}_+$ be the assignment of positive lengths to the edges. We let $|V_G| = n$, and $|E_G| = m$. We assume ℓ_G is symmetric, *i.e.*, $\ell_G(x, y) = \ell_G(y, x)$. When G is clear from the context, we drop the subscript.

A **path** P in G is an ordered sequence of (not necessarily distinct) vertices $P = (x_0, x_1, \ldots, x_k)$, such that $(x_{i-1}, x_i) \in E$ for $i \in [k]$. The **endpoints** of P are denoted by $\partial_0 P = x_0, \partial_1 P = x_k$. The set of **interior** vertices of P is defined to be $\text{int}(P) = \{x_i : 0 < i < k\}$. For $0 \leq i < j \leq k$, we use the notation $P[x_i : x_j]$ to denote the subpath $(x_i, \ldots, x_j)$. The length of P is $\ell(P) = \sum_{i=1}^{k} \ell(x_{i-1}, x_i)$.

A function $v_0 : V \to \mathbb{R} \cup \{*\}$ is called a **voltage assignment** (to G). A vertex $x \in V$ is a **terminal** with respect to v_0 iff $v_0(x) \neq *$. The other vertices, for which $v_0(x) = *$, are **non-terminals**. We let $T(v_0)$ denote the set of terminals with respect to v_0. If $T(v_0) = V$, we call v_0 a **complete voltage assignment** (to G). We say that an assignment $v : V \to \mathbb{R} \cup \{*\}$ **extends** v_0 if $v(x) = v_0(x)$ for all x such that $v_0(x) \neq *$.

Given an assignment $v_0 : V \to \mathbb{R} \cup \{*\}$, and two terminals $x, y \in T(v_0)$ for which $(x, y) \in E$, we define the **gradient** on (x, y) due to v_0 to be

$$\text{grad}_G[v_0](x, y) = \frac{v_0(x) - v_0(y)}{\ell(x, y)} .$$

We can view $\text{grad}_G[v_0](x, y)$ as the *current* in the edge (x, y) induced by voltages v_0. When v_0 is a complete voltage assignment, we interpret $\text{grad}_G[v_0]$ as a vector in $\mathbb{R}^m$, with an entry per edge. For convenience, we define $\text{grad}_G[v_0](x, y) = -\text{grad}_G[v_0](y, x)$. When G is clear from the context, we drop the subscript.

4

A graph G along with a voltage assignment v to G is called a **partially-labeled graph**, denoted (G, v). We say that a partially-labeled graph (G, v_0) is a **well-posed instance** if for every maximal connected component H of G, we have $T(v_0) \cap V_H \neq \emptyset$.

A path P in a partially-labeled graph (G, v_0) is called a **terminal path** if both endpoints are terminals. We define $\nabla P(v_0)$ to be its gradient:

$$\nabla P(v_0) = \frac{v_0(\partial_0 P) - v_0(\partial_1 P)}{\ell(P)} \ .$$

If P contains no terminal-terminal edges (and thus contains a non-terminal), it is a **free terminal path**.

Lex-Minimization. An instance of the LEX-MINIMIZATION problem is described by a partially-labeled graph (G, v_0). The objective is to compute a complete voltage assignment $v : V_G \to \mathbb{R}$ extending v_0 that lex-minimizes $\mathsf{grad}[v]$.

Definition 1 (Lex-minimizer) *Given a partially-labeled graph (G, v_0), we define $\mathsf{lex}_G[v_0]$ to be a complete voltage assignment to V that extends v_0, and such that for every other complete assignment $v' : V_G \to \mathbb{R}$ that extends v_0, we have $\mathsf{grad}_G[\mathsf{lex}_G[v_0]] \preceq \mathsf{grad}_G[v']$. That is, $\mathsf{lex}_G[v_0]$ achieves a lexicographically-minimal gradient assignment to the edges.*

We call $\mathsf{lex}_G[v_0]$ the lex-minimizer for (G, v_0). Note that if $T(v_0) = V_G$, then trivially, $\mathsf{lex}_G[v_0] = v_0$.

3. Basic Properties of Lex-Minimizers

Lazarus et al. (1999) established that lex-minimizers in unweighted and undirected graphs exist, are unique, and may be computed by an elementary meta-algorithm. We state and prove these facts for undirected weighted graphs, and defer the discussion of the directed case to Section 5. We also state for directed and weighted graphs characterizations of lex-minimizers that were established by Peres et al. (2011), Naor and Sheffield (2010) and Sheffield and Smart (2010) for unweighted graphs. These results are essential for the analyses of our algorithms. We defer most proofs to Appendix A.

Definition 2 *A steepest fixable path in an instance (G, v_0) is a free terminal path P that has the largest gradient $\nabla P(v_0)$ amongst such paths.*

Observe that a steepest fixable path with $\nabla P(v_0) \neq 0$ must be a simple path.

Definition 3 *Given a steepest fixable path P in an instance (G, v_0), we define $\mathsf{fix}_G[v_0, P] : V_G \to \mathbb{R} \cup \{*\}$ to be the voltage assignment defined as follows*

$$\mathsf{fix}_G[v_0, P](x) = \begin{cases} v_0(\partial_0 P) - \nabla P(v_0) \cdot \ell_G(P[\partial_0 P : x]) & x \in \mathrm{int}(P) \setminus T(v_0), \\ v_0(x) & \textit{otherwise.} \end{cases}$$

We say that the vertices $x \in \mathrm{int}(P)$ are fixed by the operation $\mathsf{fix}[v_0, P]$. If we define $v_1 = \mathsf{fix}_G[v_0, P]$, where $P = (x_0, \ldots, x_r)$ is the steepest fixable path in (G, v_0), then it is easy to argue that for every $i \in [r]$, we have $\mathsf{grad}[v_1](x_{i-1}, x_i) = \nabla P$ (see Lemma 26). The meta-algorithm META-LEX, spelled out as Algorithm 1, entails repeatedly fixing steepest fixable paths. While it is possible to have multiple steepest fixable paths, the result of fixing all of them does not depend on the order in which they are fixed.

Theorem 4 *Given a well-posed instance (G, v_0), the meta-algorithm META-LEX, which repeatedly fixes steepest fixable paths, produces the unique lex-minimizer extending v_0.*

Corollary 5 *Given a well-posed instance (G, v_0) such that $T(v_0) \neq V_G$, let P be a steepest fixable path in (G, v_0). Then, $(G, \mathsf{fix}[v_0, P])$ is also a well-posed instance, and $\mathsf{lex}_G[\mathsf{fix}[v_0, P]] = \mathsf{lex}_G[v_0]$.*

Since a lex-minimal element must be an inf-minimizer, we also obtain the following corollary, that can also be proved using LP duality.

Lemma 6 *Suppose we have a well-posed instance (G, v_0). Then, there exists a complete voltage assignment v extending v_0 such that $\left\|\mathsf{grad}[v]\right\|_\infty \leq \alpha$, iff every terminal path P in (G, v_0) satisfies $\nabla P(v_0) \leq \alpha$.*

We also prove the following theorem that shows that $\mathsf{lex}[v_0]$ is stable under changes in v_0.

Theorem 7 *Given well-posed instances (G, v_0), (G, v_1) such that $T := T(v_0) = T(v_1)$. Then $|\mathsf{lex}_G[v_0](i) - \mathsf{lex}_G[v_1](i)| \leq \max_{t \in T} |v_0(t) - v_1(t)|$ for all $i \in V_G$.*

3.1. Alternate Characterizations

There are at least two other seemingly disparate definitions that are equivalent to lex-minimal voltages.

l_p**-norm Minimizers.** As mentioned in the introduction, for a well-posed instance (G, v_0) the lex-minimizer is also the limit of l_p minimizers. This follows from existing results about the limit of l_p-minimizers (Egger and Huotari (1990)) in affine spaces, since $\{\mathsf{grad}[v] \mid v$ is complete, v extends $v_0\}$ forms an affine subspace of $\mathbb{R}^m$. Thus, we have the following theorem:

Theorem 8 (follows from Egger and Huotari (1990)) *For any $p \in (1, \infty)$, given a well-posed instance (G, v_0) define v_p to be the unique complete voltage assignment extending v_0 and minimizing $\left\|\mathsf{grad}[v]\right\|_p$, i.e.*

$$v_p = \underset{\substack{v \text{ is complete} \\ v \text{ extends } v_0}}{\arg\min} \left\|\mathsf{grad}[v]\right\|_p .$$

Then, $\lim_{p \to \infty} v_p = \mathsf{lex}_G[v_0]$.

Max-Min Gradient Averaging. Consider a well-posed instance (G, v_0), and a complete voltage assignment v extending v_0. If G is such that $\ell(e) = 1$ for all $e \in E_G$, it is easy to see that $\mathsf{lex} = \mathsf{lex}_G[v_0]$ satisfies the following simple condition for all $x \in V_G \setminus T(v_0)$,

$$\mathsf{lex}(x) = \frac{1}{2} \left(\max_{(x,y) \in E_G} \mathsf{lex}(y) + \min_{(x,z) \in E_G} \mathsf{lex}(z) \right).$$

This condition should be contrasted to the optimality condition for l_2-regularization on these instances, which gives for all non-terminals x, the optimal voltage v satisfies $v(x) = \frac{1}{\deg(x)} \sum_{y:(x,y) \in E_G} v(y)$.

To prove the above claim, consider locally changing lex at x and observe that the gradients of edges not incident at x remain unchanged, and at least one of edges incident at x will have a strictly larger gradient, contradicting lex-minimality. For general graphs, this condition of local optimality can still be characterized by a simple *max-min gradient averaging property* as described below.

Definition 9 (Max-Min Gradient Averaging) *Given a well-posed instance (G, v_0), and a complete voltage assignment v extending v_0, we say that v satisfies the* max-min gradient averaging *property (w.r.t. (G, v_0)) if for every $x \in V_G \setminus T(v_0)$, we have*

$$\max_{y:(x,y) \in E_G} \mathsf{grad}[v](x, y) = - \min_{y:(x,y) \in E_G} \mathsf{grad}[v](x, y).$$

6

As stated in the theorem below, $\mathsf{lex}_G[v_0]$ is the unique assignment satisfying max-min gradient averaging property. Sheffield and Smart (2010) proved a variant of this statement for weighted graphs. For completeness, we present a proof in the appendix.

Theorem 10 *Given a well-posed instance (G, v_0), $\mathsf{lex}_G[v_0]$ satisfies max-min gradient averaging property. Moreover, it is the unique complete voltage assignment extending v_0 that satisfies this property w.r.t. (G, v_0).*

This characterization is particularly useful for implementations since it can be verified quickly.

4. Algorithms

We now sketch the ideas behind our algorithms and give precise statements of our results. A full description of all the algorithms is included in the appendix.

We define the **pressure** of a vertex to be the gradient of the steepest terminal path through it:

$$\mathsf{pressure}[v_0](x) = \max\{\nabla P(v_0) \mid P \text{ is a terminal path in } (G, v_0) \text{ and } x \in P\}.$$

Observe that in a graph with no terminal-terminal edges, a free terminal path is a steepest fixable path iff its gradient is equal to the highest pressure amongst all vertices. Moreover, vertices that lie on steepest fixable paths are exactly the vertices with the highest pressure. For a given $\alpha > 0$, in order to identify vertices with pressure exceeding α, we compute vectors $\mathsf{vHigh}[\alpha](x)$ and $\mathsf{vLow}[\alpha](x)$ defined as follows in terms of dist, the metric on V induced by ℓ:

$$\mathsf{vLow}[\alpha](x) = \min_{t \in T(v_0)} \{v_0(t) + \alpha \cdot \mathsf{dist}(x, t)\} \qquad \mathsf{vHigh}[\alpha](x) = \max_{t \in T(v_0)} \{v_0(t) - \alpha \cdot \mathsf{dist}(t, x)\}.$$

4.1. Lex-minimization on Star Graphs

We first consider the problem of computing the lex-minimizer on a star graph where every vertex but the center is a terminal. This special case is a subroutine in the general algorithm, and also motivates some of our techniques.

Let x be the center vertex, T be the set of terminals, and all edges be of the form (x, t) with $t \in T$. The initial voltage assignment is given by $v : T \to \mathbb{R}$, and we abbreviate $\mathsf{dist}(x, t)$ by $\mathsf{d}(t) = \ell(x, t)$. From Corollary 5 we know that we can determine the value of the lex minimizer at x by finding a steepest fixable path. By definition, we need to find $t_1, t_2 \in T$ that maximize the gradient of the path from t_1 to t_2, $\nabla(t_1, t_2) = \frac{v(t_1) - v(t_2)}{\mathsf{d}(t_2) + \mathsf{d}(t_2)}$. As observed above, this is equivalent to finding a terminal with the highest pressure. We now present a simple randomized algorithm for this problem that runs in expected linear time.

Given a terminal t_1, we can compute its pressure α along with the terminal t_2 such that $|\nabla(t_1, t_2)| = \alpha$ in time $O(|T|)$ by scanning over the terminals in T. Consider doing this for a random terminal t_1. We will show that in linear time one can then find the subset of terminals $T' \subset T$ whose pressure is greater than α. Assuming this, we complete the analysis of the algorithm. If $T' = \emptyset$, t_1 is a vertex with highest pressure. Hence the path from t_1 to t_2 is a steepest fixable path, and we return (t_1, t_2). If $T' \neq \emptyset$, the terminal with the highest pressure must be in T', and we recurse by picking a new random $t_1 \in T'$. As the size of T' will halve in expectation at each iteration, the expected time of the algorithm on the star is $O(|T|)$.

To determine which terminals have pressure exceeding α, we observe that $\exists t_2 : \alpha < \nabla(t_1, t_2) = \frac{v(t_1) - v(t_2)}{\mathsf{d}(t_1) + \mathsf{d}(t_2)}$, is equivalent to $\exists t_2 : v(t_2) + \alpha \mathsf{d}(t_2) < v(t_1) - \alpha \mathsf{d}(t_1)$. This, in turn, is equivalent to $\mathsf{vLow}[\alpha](x) < v(t_1) - \alpha \mathsf{d}(t_1)$. We can compute $\mathsf{vLow}[\alpha](x)$ in deterministic $O(|T|)$ time. Similarly, we can check if $\exists t_2 : \alpha < \nabla(t_2, t_1)$ by checking if $\mathsf{vHigh}[\alpha](x) > v_{t_1} + \alpha \mathsf{d}(t_1)$. Thus, in linear time, we can compute the set T' of terminals with pressure exceeding α. The above algorithm is described in Algorithm 10.

Theorem 11 *Given a set of terminals T, initial voltages $v : T \to \mathbb{R}$, and distances $\mathsf{d} : T \to \mathbb{R}_+$, STARSTEEPESTPATH$(T, v, \mathsf{d})$ returns (t_1, t_2) maximizing $\frac{v(t_1) - v(t_2)}{\mathsf{d}(t_1) + \mathsf{d}(t_2)}$, and runs in expected time $O(|T|)$.*

4.2. Lex-minimization on General Graphs

We know that META-LEX computes lex-minimizers given an algorithm for finding a steepest fixable path in (G, v_0) (Theorem 4). Recall that finding a steepest fixable path is equivalent to finding a path with gradient equal to the highest pressure amongst all vertices. We show how to do this in expected time $O(m + n \log n)$.

We describe an algorithm VERTEXSTEEPESTPATH that finds a terminal path P through any vertex x such that $\nabla P(v_0) = \mathsf{pressure}[v_0](x)$ in expected $O(m + n \log n)$ time. Using Dijkstra's algorithm, we compute $\mathsf{dist}(x, t)$ for all $t \in T$. If $x \in T(v_0)$, then there must be a terminal path P that starts at x that has $\nabla P(v_0) = \mathsf{pressure}[v_0](x)$. To compute such a P we examine all $t \in T(v_0)$ in $O(|T|)$ time to find the t that maximizes $|\nabla(x, t)| = \frac{|v(x) - v(t)|}{\mathsf{dist}(x, t)}$, and then return a shortest path between x and that t.

If $x \notin T(v_0)$, then the steepest path through x between terminals t_1 and t_2 must consist of shortest paths between x and t_1 and between x and t_2. Thus, we can reduce the problem to that of finding the steepest path in a star graph where x is the only non-terminal and is connected to each terminal t by an edge of length $\mathsf{dist}(x, t)$. By Theorem 11, we can find this steepest path in $O(|T|)$ expected time. The above algorithm is formally described as Algorithm 9.

Theorem 12 *Given a well-posed instance (G, v_0), and a vertex $x \in V_G$, VERTEXSTEEPESTPATH(G, v_0, x) returns a terminal path P through x such that $\nabla P(v_0) = \mathsf{pressure}[v_0](x)$, in $O(m + n \log n)$ expected time.*

As in the algorithm for the star graph, we need to identify the vertices whose pressure exceeds a given α. For a fixed α, we can compute $\mathsf{vLow}[\alpha](x)$ and $\mathsf{vHigh}[\alpha](x)$ for all $x \in V_G$ using a simple modification of Dijkstra's algorithm in $O(m + n \log n)$ time. We describe the algorithms COMPVHIGH, COMPVLOW for these tasks in Algorithms 3 and 4. The following lemma encapsulates the usefulness of vLow and vHigh.

Lemma 13 *For every $x \in V_G$, $\mathsf{pressure}[v_0](x) > \alpha$ iff $\mathsf{vHigh}[\alpha](x) > \mathsf{vLow}[\alpha](x)$.*

It immediately follows that the algorithm COMPHIGHPRESSGRAPH(G, v_0, α) described in Algorithm 6 computes the vertex induced subgraph on the vertex set $\{x \in V_G | \mathsf{pressure}[v_0](x) > \alpha\}$.

We can combine these algorithms into an algorithm STEEPESTPATH that finds the steepest fixable path in (G, v_0) in $O(m + n \log n)$ expected time. We may assume that there are no terminal-terminal edges in G. We sample an edge (x_1, x_2) uniformly at random from E_G, and a terminal x_3 uniformly at random from V_G. For $i = 1, 2, 3$, we compute the steepest terminal path P_i containing x_i. By Theorem 12, this can be done in $O(m + n \log n)$ expected time. Let α be the largest gradient $\max_i \nabla P_i$. As mentioned above, we can identify G', the induced subgraph on vertices x with pressure exceeding α, in $O(m + n \log n)$ time. If G' is empty, we know that the path P_i with largest gradient is a steepest fixable path. If not, a steepest fixable path in (G, v_0) must be in G', and hence we can recurse on G'. Since we picked a uniformly random edge, and a uniformly random vertex, the expected size of G' is at most half that of G. Thus, we obtain an expected running time of $O(m + n \log n)$. This algorithm is described in detail in Algorithm 7.

Theorem 14 *Given a well-posed instance (G, v_0) with $E_G \cap (T(v_0) \times T(v_0)) = \emptyset$, STEEPESTPATH$(G, v_0)$ returns a steepest fixable path in (G, v_0), and runs in $O(m + n \log n)$ expected time.*

By using STEEPESTPATH in META-LEX, we get the COMPLEXMIN, shown in Algorithm 1. From Theorem 4 and Theorem 14, we immediately get the following corollary.

Corollary 15 *Given a well-posed instance (G, v_0) as input, algorithm COMPLEXMIN computes a lex-minimizing assignment that extends v_0 in $O(n(m + n \log n))$ expected time.*

4.3. Linear-time Algorithm for Inf-minimization

Given the algorithms in the previous section, it is straightforward to construct an infinity minimizer. Let $\alpha^\star$ be the gradient of the steepest terminal path. From Lemma 6, we know that the norm of the inf minimizer is $\alpha^\star$. Considering all trivial terminal paths (terminal-terminal edges), and using STEEPESTPATH, we can compute $\alpha^\star$ in randomized $O(m + n \log n)$ time. It is well known (McShane (1934); Whitney (1934)) that $v_1 = \mathsf{vLow}[\alpha^\star]$ and $v_2 = \mathsf{vHigh}[\alpha^\star]$ are inf-minimizers. It is also known that $\frac{1}{2}(v_1 + v_2)$ is the inf-minimizer that minimizes the maximum ℓ_∞-norm distance to all inf-minimizers. In the case of path graphs, this was observed by Gaffney and Powell (1976) and independently by Micchelli et al. (1976). For completeness, the algorithm is presented as Algorithm 5, and we have the following result.

Theorem 16 *Given a well-posed instance (G, v_0), COMPINFMIN(G, v_0) returns a complete voltage assignment v for G extending v_0 that minimizes $\left\| \mathsf{grad}[v] \right\|_\infty$, and runs in randomized $O(m + n \log n)$ time.*

4.4. Faster Algorithms for Lex-minimization

The lex-minimizer has additional structure that allows one to compute it more efficiently. One observation that leads to faster implementations is that fixing a steepest fixable path does not increase the pressure at vertices, provided one appropriately ignores terminal-terminal edges. Thus, if $G^{(\alpha)}$ is the subgraph with pressure greater than α, we can iteratively fix all steepest fixable paths P in $G^{(\alpha)}$ with $\nabla P > \alpha$. Another simple observation is that if $G^{(\alpha)}$ is disconnected, we can recurse on each of the connected components. A complete description of an the algorithm COMPFASTLEXMIN based on these idea is given in Algorithm 11. The algorithm provably computes $\mathsf{lex}_G(v_0)$, and it is possible to implement it so that the space requirement is only $O(m + n)$. Although, we are unable to prove theoretical bounds on the running time that are better than $O(n(m + n \log n))$, it runs extremely quickly in practice. We used it to perform the experiments in this paper. For random regular graphs and Delaunay graphs, with $n = 0.5 \times 10^6$ vertices and around 2 million edges $m \sim 1.5 - 2 \times 10^6$, it takes a couple of minutes on a 2009 MacBook Pro. Similar times are observed for other model graphs of this size such as random regular graphs and real world networks. An implementation of this algorithm may be found at `https://github.com/danspielman/YINSlex`.

5. Directed Graphs

Our definitions and algorithms, including those for regularization, extend to directed graphs with small modifications. We view directed edges as diodes and only consider potential differences in the direction of the edge. For a complete voltage assignment v for a directed graph G, we define the directed gradient on (x, y) due to v to be $\mathsf{grad}_G^+[v](x, y) = \max\left\{ \frac{v(x) - v(y)}{\ell(x, y)}, 0 \right\}$. Given a partially-labelled directed graph (G, v_0), we say that a a complete voltage assignment v is a **lex-minimizer** if it extends v_0 and for every complete voltage assignment v' that extends v_0 we have $\mathsf{grad}_G^+[v] \preceq \mathsf{grad}_G^+[v']$. We say that (G, v_0) is a **well-posed** directed instance if every free vertex appears in a directed path between two terminals.

The main difference is that the directed lex-minimizer isn't necessarily unique. We focussed on undirected graphs for clarity of exposition. We have the following structural results for directed graphs.

Theorem 17 *Given a well-posed directed instance (G, v_0), there exists a lex-minimizer, and the set of all lex-minimizers is convex. Moreover, for every two lex-minimizers v and v', we have $\mathsf{grad}_G^+[v] = \mathsf{grad}_G^+[v']$.*

The directed lex-minimizer need not be unique. However, a weaker version of Theorem 4 still holds.

Theorem 18 *Given a well-posed instance* (G, v_0) *on a directed graph* G, *let* v_1 *be the partial voltage assignment extending* v_0 *obtained by repeatedly fixing steepest fixable (directed) paths* P *with* $\nabla P > 0$. *Then, any lex-minimizer of* (G, v_0) *must extend* v_1. *Moreover, for every edge* $e \in E_G \setminus (T(V_1) \times T(V_1))$, *any lex-minimizer* v *of* (G, v_0) *must satisfy* $\mathsf{grad}^+[v](e) = 0$.

When the value of the lex-minimizer at a vertex is not uniquely determined, it is constrained to an interval. In our experiments, we pick the convention that when the voltage at a vertex is constrained to $(-\infty, a]$ or $[a, \infty)$, we assign a to the terminal. When it is constrained to a finite interval, we assign a voltage closest to the median of the original voltages.

6. Experiments on WebSpam

We demonstrate the performance of our lex-minimization algorithms on directed graphs by using them to detect spam webpages as in Zhou et al. (2007). We use the dataset `webspam-uk2006-2.0` described in Castillo et al. (2006). This collection includes 11,402 hosts, out of which 7,473 (65.5 %) are labeled, either as `spam` or `normal`. Each host corresponds to the collection of web pages it serves. Of the hosts, 1924 are labeled spam (25.7 % of all labels). We consider the problem of flagging some hosts as spam, given only a small fraction of the labels for training. We assign a value of 1 to the spam hosts, and a value of 0 to the normal ones. We then compute a lex minimizer and examine the effect of flagging as spam all hosts with a value greater than some threshold.

Following Zhou et al. (2007), we create edges between hosts with lengths equal to the reciprocal of the number of links from one to the other. We run our experiments only on the largest strongly connected component of the graph, which contains 7945 hosts of which 5552 are labeled. 16 % of the nodes in this subgraph are labeled `spam`. To create training and test data, for a given value p, we select a random subset of p % of the `spam` labels and a random subset of p % of the `normal` labels to use for training. The remaining labels are used for testing. We report results for $p = 5$ and $p = 20$.

Again following Zhou et al. (2007), we plot the precision and recall of different choices of threshold for flagging pages as spam. *Recall* is the fraction of spam pages our algorithm flags as spam, and *precision* is the fraction of pages our algorithm flags as spam that actually are spam. Amongst the algorithms studied by Zhou et al. (2007), the top performer was their algorithm based on sampling according to a random-walk that follows in-links from other hosts. We compare their algorithm with the classification we get by directing edges in the opposite directions of links. This has the effect that a link to a spam host is evidence of spamminess, and a link from a normal host is evidence of normality.

Results are shown in Figure 3. While we are not able to reliably flag all spam hosts, we see that in the range of 10-50 % recall, we are able to flag spam with precision above 82 %. We see that the performance of directed lex-minimization does not degrade rapidly when from the "large training set" regime of $p = 20$, to the "small training set" regime of $p = 5$. For comparison, in Appendix C, we show the performance of our algorithm and that of Zhou et al. (2007) both with link directions reversed, as well as the performance of undirected lex-minimization and Laplacian inference, all of which are significantly worse.

7. l_0-Regularization of Vertex Values

We now explain how we can accommodate noise in both the given voltages and in the given lengths of edges. We can find the minimum number of labels to ignore, or the minimum increase in edges lengths needed so that there exists an extension whose gradients have l_∞-norm lower than a given target. After determining which labels to ignore or the needed increment in edge lengths, we recommend computing a lex minimizer.
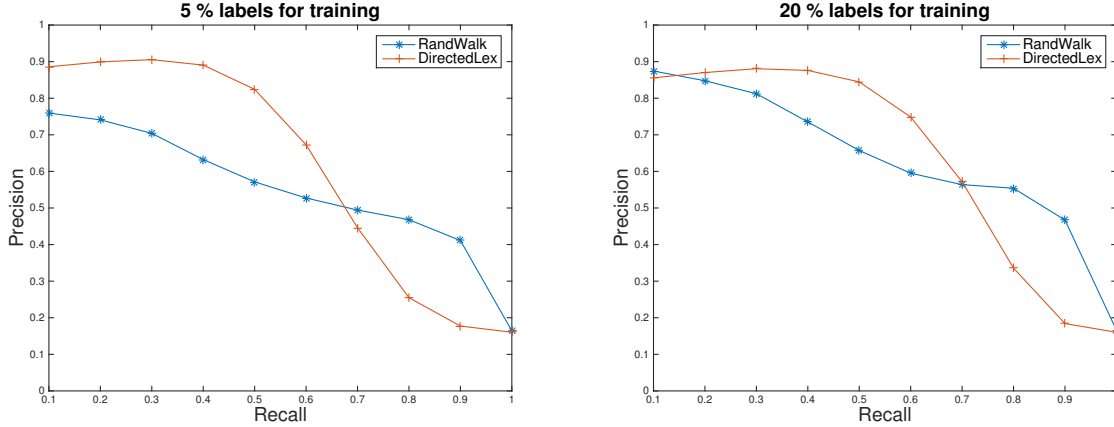
Figure 3: Recall and precision in the web spam classification experiment. Each data point is an average over 100 runs. The largest standard deviation of the mean precision across the plotted recall values was less than 1.3 %. The algorithm of Zhou et al. (2007) appears as RANDWALK. Our directed lex-minimization algorithm appears as DIRECTEDLEX.

The algorithms we present in this section are essentially the same for directed and undirected graphs.

7.1. l_0-Vertex Regularization for Inf-minimization

The l_0-regularization of vertex labels can be viewed as outlier removal: the vector we compute may disagree with v_0 on up to k terminals. Given a voltage assignment v and a subset $T \subset V$ of the vertices, by $v(T)$ we mean the vector obtained by restricting v to T. We define the l_0-*Vertex Regularization for* l_∞ problem to be

$$\min_{v \in \mathbb{R}^n} \ \big\|\mathsf{grad}_G[v]\big\|_\infty \qquad \text{subject to } \big\|v(T) - v_0(T)\big\|_0 \le k, \tag{3}$$

where $v(T)$ is the vector of values of v on the terminals T.

In Appendix D, we describe an approximation algorithm APPROX-OUTLIER that approximately solves program (3). The precise statement we prove in Appendix D is given in the following theorem.

Theorem 19 (Approximate l_0-vertex regularization) *The algorithm* APPROX-OUTLIER *takes a positive integer k and a partially-labeled graph (G, v_0), and outputs an assignment v with* $\big\|v(T) - v_0(T)\big\|_0 \le 2k$, *and* $\big\|\mathsf{grad}_G[v]\big\|_\infty \le \alpha^*$, *where α^* is the optimum value of program* (3), *in time $O(k(m + n \log n))$.*

In Appendix D, we describe and analyze an algorithm OUTLIER that solves program (3) in poly-time.

Theorem 20 (Exact l_0-vertex regularization) *The algorithm* OUTLIER *takes a positive integer k and a partially-labeled graph (G, v_0) solves program* (3) *exactly. The algorithm runs in polynomial time.*

We give a proof of Theorem 20 in Appendix D. To do this, we reduce the program (3) to the problem of minimizing the required l_0-budget needed to achieve a fixed gradient α using a binary search over a set of $O(n^2)$ gradients. This latter problem we reduce in polynomial time to Minimum Vertex Cover (VC) on a transitively closed, directed acyclic graph (a TC-DAG). VC on a TC-DAG can be solved exactly in polynomial time by a reduction to the Maximum Bipartite Matching Problem (Fulkerson (1956)). The problem was phrased by Fulkerson as one of finding a maximum antichain of a finite poset. Any transitively closed DAG corresponds directly to the comparability graph of a poset. A maximum antichain of a poset is a maximum independent set of a the comparability graph of the poset, and hence its complement is a minimum vertex cover of the comparability graph. We refer to Fulkerson's algorithm as KONIG-COVER.

7.2. Hardness of l_0 regularization for l_2

The result that l_0-regularized inf-minimization can be solved exactly in polynomial time is surprising, especially because the analogous problem for 2-Laplacian minimization turns out to be NP-Hard.

We define the the l_0 *vertex regularization for l_2* for a partially-labeled graph (G, v_0) and an integer k by

$$\min_{v \in \mathbb{R}^n : \left\| v(T) - v_0(T) \right\|_0 \leq k} v^T L v,$$

where L is the Laplacian of G.

Theorem 21 l_0 *vertex regularization for l_2 is NP-Hard.*

We prove this in Appendix E by giving a polynomial time reduction from the minimum bisection problem.

8. l_1-Edge and Vertex Regularization of Inf-minimizers

Consider a partially-labeled graph (G, v_0) and an $\alpha > 0$. The set of voltage assignments given by

$$\left\{ v : v \text{ extends } v_0 \text{ and } \left\| \mathsf{grad}_G[v] \right\|_\infty \leq \alpha \right\}$$

is convex. Going further, let us consider the edge lengths in a graph to be specified by a vector $\ell \in \mathbb{R}^E$. Now the set of voltages v and and lengths ℓ which achieve $\|\mathsf{grad}_{G(\ell)}[v]\|_\infty \leq \alpha$ is jointly convex in v and ℓ. To see this, observe that

$$\left\| \mathsf{grad}_{G(\ell)}[v] \right\|_\infty \leq \alpha \Leftrightarrow \forall (u, v) \in E : -\alpha \ell(u, v) \leq v(u) - v(v) \leq \alpha \ell(u, v). \tag{4}$$

Furthermore, the condition "v extends v_0" is a linear constraint on v, which we express as $v(T) = v_0(T)$. From the above, it is clear that the gradient condition corresponds to a convex set, as it is an intersection of half-spaces. These half-spaces are given by $O(m)$ linear inequalities. We can leverage this to phrase many regularized variants of inf-minimization as convex programs, and in some cases linear programs.

For example, we may consider a variant of inf-minimization combined with an l_1-budget for changing lengths of edges and values on terminals. Given a parameter $\gamma > 0$ which specifies the relative cost of regularizing terminals to regularizing edges, the problem is as follows

$$\underset{v \in \mathbb{R}^n, s \in \mathbb{R}^m, s \geq 0}{\arg \min} \|s\|_1 + \gamma \left\| v(T) - v_0(T) \right\|_1 \qquad \text{subject to } \left\| \mathsf{grad}_{G(\ell+s)}[v] \right\|_\infty \leq \alpha. \tag{5}$$

From our observation (4), it follows that problem (5) may be expressed as a linear program with $O(n)$ variables and $O(m)$ constraints. We can use ideas from Daitch and Spielman (2008) to solve the resulting linear program in time $\widetilde{O}(m^{1.5})$ by an interior point method. The reason is that the linear equations the IPM must solve at each iteration may be reduced to linear equations in symmetric, diagonally dominant matrices, and these may be solved in nearly-linear time (Cohen et al. (2014)).

Conclusion. The initial experiments reported in the paper indicate that our algorithms give good results on real and synthetic datasets, in particular for tiny labeled sets. We believe that inf and lex minimizers, and the associated ideas presented in the paper, should be useful primitives that can be profitably combined with other approaches to learning on graphs.

References

Morteza Alamgir and Ulrike V. Luxburg. Phase transition in the family of p-resistances. In *Advances in Neural Information Processing Systems 24*, pages 379–387. 2011. URL http://books.nips.cc/papers/files/nips24/NIPS2011_0278.pdf.

Gunnar Aronsson. Extension of functions satisfying lipschitz conditions. *Arkiv fr Matematik*, 6(6):551–561, 1967. ISSN 0004-2080. doi: 10.1007/BF02591928. URL http://dx.doi.org/10.1007/BF02591928.

Gunnar Aronsson, Michael G. Crandall, and Petri Juutinen. A tour of the theory of absolutely minimizing functions. *Bull. Amer. Math. Soc. (N.S.)*, 41(4):439–505, 2004. ISSN 0273-0979. doi: 10.1090/S0273-0979-04-01035-3. URL http://dx.doi.org/10.1090/S0273-0979-04-01035-3.

Guy Barles and Jérôme Busca. Existence and comparison results for fully nonlinear degenerate elliptic equations without zeroth-order term. *Comm. Partial Differential Equations*, 26:2323–2337, 2001.

Mikhail Belkin, Irina Matveeva, and Partha Niyogi. Regularization and semi-supervised learning on large graphs. In *Learning Theory*, volume 3120 of *Lecture Notes in Computer Science*, pages 624–638. Springer Berlin Heidelberg, 2004. ISBN 978-3-540-22282-8. doi: 10.1007/978-3-540-27819-1_43. URL http://dx.doi.org/10.1007/978-3-540-27819-1_43.

Nick Bridle and Xiaojin Zhu. p-voltages: Laplacian regularization for semi-supervised learning on high-dimensional data. In *Eleventh Workshop on Mining and Learning with Graphs (MLG2013)*, 2013.

Carlos Castillo, Debora Donato, Luca Becchetti, Paolo Boldi, Stefano Leonardi, Massimo Santini, and Sebastiano Vigna. A reference collection for web spam. *SIGIR Forum*, 40(2):11–24, December 2006. ISSN 0163-5840. doi: 10.1145/1189702.1189703. URL http://doi.acm.org/10.1145/1189702.1189703.

Olivier Chapelle, Bernhard Schlkopf, and Alexander Zien. *Semi-Supervised Learning*. The MIT Press, 1st edition, 2010. ISBN 0262514125, 9780262514125.

Michael B Cohen, Rasmus Kyng, Gary L Miller, Jakub W Pachocki, Richard Peng, Anup B Rao, and Shen Chen Xu. Solving SDD linear systems in nearly $m \log^{1/2} n$ time. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, pages 343–352. ACM, 2014.

M.G. Crandall, L.C. Evans, and R.F. Gariepy. Optimal lipschitz extensions and the infinity laplacian. *Calculus of Variations and Partial Differential Equations*, 13(2):123–139, 2001. ISSN 0944-2669. doi: 10.1007/s005260000065. URL http://dx.doi.org/10.1007/s005260000065.

Samuel I. Daitch and Daniel A. Spielman. Faster approximate lossy generalized flow via interior point algorithms. In *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing*, STOC '08, pages 451–460, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-047-0. doi: 10.1145/1374376.1374441. URL http://doi.acm.org/10.1145/1374376.1374441.

Alan Egger and Robert Huotari. Rate of convergence of the discrete polya algorithm. *Journal of Approximation Theory*, 60(1):24 – 30, 1990. ISSN 0021-9045. doi: http://dx.doi.org/10.1016/0021-9045(90)90070-7. URL http://www.sciencedirect.com/science/article/pii/0021904590900707.

Charles Fefferman. Whitney's extension problems and interpolation of data. *Bull. Amer. Math. Soc. (N.S.)*, 46(2):207–220, 2009a. ISSN 0273-0979. doi: 10.1090/S0273-0979-08-01240-8. URL http://dx.doi.org/10.1090/S0273-0979-08-01240-8.

Charles Fefferman. Fitting a [image] -smooth function to data, iii. *Annals of Mathematics*, 170(1):pp. 427–441, 2009b. ISSN 0003486X. URL http://www.jstor.org/stable/40345469.

Charles Fefferman and Bo'az Klartag. Fitting a cm -smooth function to data i. *Annals of Mathematics*, 169 (1):pp. 315–346, 2009. ISSN 0003486X. URL http://www.jstor.org/stable/40345445.

D. R. Fulkerson. Note on dilworths decomposition theorem for partially ordered sets. *Proc. Amer. Math. Soc*, 1956.

P.W. Gaffney and M.J.D. Powell. Optimal interpolation. In *Numerical Analysis*, volume 506 of *Lecture Notes in Mathematics*, pages 90–99. Springer Berlin Heidelberg, 1976. ISBN 978-3-540-07610-0. doi: 10.1007/BFb0080117. URL http://dx.doi.org/10.1007/BFb0080117.

L.-A. Gottlieb, A. Kontorovich, and R. Krauthgamer. Efficient classification for metric data. *CoRR*, abs/1306.2547, 2013. URL http://arxiv.org/abs/1306.2547.

L.-A. Gottlieb, A. Kontorovich, and R. Krauthgamer. Efficient regression in metric spaces via approximate lipschitz extension. In *Similarity-Based Pattern Recognition*, volume 7953 of *Lecture Notes in Computer Science*, pages 43–58. Springer Berlin Heidelberg, 2013b. ISBN 978-3-642-39139-2. doi: 10.1007/978-3-642-39140-8_3. URL http://dx.doi.org/10.1007/978-3-642-39140-8_3.

Robert Jensen. Uniqueness of lipschitz extensions: Minimizing the sup norm of the gradient. *Archive for Rational Mechanics and Analysis*, 123(1):51–74, 1993. ISSN 0003-9527. doi: 10.1007/BF00386368. URL http://dx.doi.org/10.1007/BF00386368.

M. Kirszbraun. ber die zusammenziehende und lipschitzsche transformationen. *Fundamenta Mathematicae*, 22(1):77–108, 1934. URL http://eudml.org/doc/212681.

Andrew J. Lazarus, Daniel E. Loeb, James G. Propp, Walter R. Stromquist, and Daniel H. Ullman. Combinatorial games under auction play. *Games and Economic Behavior*, 27(2):229 – 264, 1999. ISSN 0899-8256. doi: http://dx.doi.org/10.1006/game.1998.0676. URL http://www.sciencedirect.com/science/article/pii/S0899825698906765.

Yunqian Ma and Yun Fu. *Manifold Learning Theory and Applications*. CRC Press, Inc., Boca Raton, FL, USA, 1st edition, 2011. ISBN 1439871094, 9781439871096.

E. J. McShane. Extension of range of functions. *Bull. Amer. Math. Soc.*, 40(12):837–842, 12 1934. URL http://projecteuclid.org/euclid.bams/1183497871.

C.A. Micchelli, T.J. Rivlin, and S. Winograd. The optimal recovery of smooth functions. *Numerische Mathematik*, 26(2):191–200, 1976. ISSN 0029-599X. doi: 10.1007/BF01395972. URL http://dx.doi.org/10.1007/BF01395972.

V. A. Milman. Absolutely minimal extensions of functions on metric spaces. 1999. URL http://iopscience.iop.org/1064-5616/190/6/A05/pdf/MSB_190_6_A05.pdf.

Boaz Nadler, Nathan Srebro, and Xueyuan Zhou. Statistical analysis of semi-supervised learning: The limit of infinite unlabelled data. 2009. URL http://ttic.uchicago.edu/~nati/Publications/NSZnips09.pdf.

A. Naor and S. Sheffield. Absolutely minimal Lipschitz extension of tree-valued mappings. *CoRR*, abs/1005.2535, May 2010. URL http://arxiv.org/abs/1005.2535.

A. M. Oberman. Finite difference methods for the Infinity Laplace and p-Laplace equations. *CoRR*, abs/1107.5278, July 2011. URL http://arxiv.org/abs/1107.5278.

Yuval Peres, Oded Schramm, Scott Sheffield, and DavidB. Wilson. Tug-of-war and the infinity laplacian. In *Selected Works of Oded Schramm*, Selected Works in Probability and Statistics, pages 595–638. Springer New York, 2011. ISBN 978-1-4419-9674-9. doi: 10.1007/978-1-4419-9675-6_18. URL http://dx.doi.org/10.1007/978-1-4419-9675-6_18.

S. Sheffield and C. K. Smart. Vector-valued optimal Lipschitz extensions. *CoRR*, abs/1006.1741, June 2010. URL http://arxiv.org/abs/1006.1741.

Ali Kemal Sinop and Leo Grady. A seeded image segmentation framework unifying graph cuts and random walker which yields a new algorithm. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE, 2007.

Daniel A. Spielman and Shang-Hua Teng. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing (STOC)*, pages 81–90, June 2004.

Vijay V. Vazirani. *Approximation Algorithms*. Springer-Verlag New York, Inc., New York, NY, USA, 2001. ISBN 3-540-65367-8.

Ulrike von Luxburg and Olivier Bousquet. Distance-based classification with lipschitz functions. In *Learning Theory and Kernel Machines*, volume 2777 of *Lecture Notes in Computer Science*, pages 314–328. Springer Berlin Heidelberg, 2003. ISBN 978-3-540-40720-1. doi: 10.1007/978-3-540-45167-9_24. URL http://dx.doi.org/10.1007/978-3-540-45167-9_24.

Hassler Whitney. Analytic extensions of differentiable functions defined in closed sets. *Transactions of the American Mathematical Society*, 36(1):pp. 63–89, 1934. ISSN 00029947. URL http://www.jstor.org/stable/1989708.

Dengyong Zhou, Christopher J. C. Burges, and Tao Tao. Transductive link spam detection. In *Proceedings of the 3rd International Workshop on Adversarial Information Retrieval on the Web*, AIRWeb '07, pages 21–28, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-732-2. doi: 10.1145/1244408.1244413. URL http://doi.acm.org/10.1145/1244408.1244413.

Xiaojin Zhu, Zoubin Ghahramani, and John Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *IN ICML*, pages 912–919, 2003.

Appendix A. Basic Properties of Lex-Minimizers

A.1. Meta Algorithm

Algorithm 1: Algorithm META-LEX: Given a well-posed instance (G, v_0), outputs $\mathsf{lex}_G[v_0]$.

for $i = 1, 2, \ldots$:

    1. **if** $T(v_{i-1}) = V_G$, **then** return v_{i-1}.
    2. $E' = E_G \setminus (T(v_{i-1}) \times T(v_{i-1}))$, $G' := (V_G, E')$.
    3. Let $P_i^\star$ be a steepest fixable path in (G', v_{i-1}). Let $\alpha_i^\star \leftarrow \nabla P^\star(v_{i-1})$.
    4. $v_i \leftarrow \mathsf{fix}[v_{i-1}, P_i^\star]$.

In this subsection, we prove the results that appeared in section 2. We start with a simple observation.

Proposition 22 *Given a well-posed instance (G, v_0) such that $T(v_0) \neq V$, let P be a steepest fixable path in (G, v_0). Then, $\mathsf{fix}[v_0, P]$ extends v_0, and $(G, \mathsf{fix}[v_0, P])$ is also a well-posed instance.*

The properties we prove below do not depend on the choice of the steepest fixable path.

Proposition 23 *For any well-posed instance (G, v_0), with $|V_G| = n$, META-LEX(G, v_0) terminates in at most n iterations, and outputs a complete voltage assignment v that extends v_0.*

Proof of Proposition 23: By Proposition 22, at any iteration i, v_{i-1} extends v_0 and (G', v_{i-1}) is a well-posed instance. META-LEX only outputs v_{i-1} iff $T(v_{i-1}) = V$, which means v_{i-1} is a complete voltage assignment. For any v_{i-1} that is not complete, for any $x \in V \setminus T(v_{i-1})$, we must have a free terminal path in (G', v_{i-1}) that contains x. Hence, a steepest fixable path $P_i^\star$ exists in (G', v_{i-1}). Since $P_i^\star$ is a free terminal path, $\mathsf{fix}[v_{i-1}, P_i^\star]$ fixes the voltage for at least one non-terminal. Thus, META-LEX(G, v_0) must complete in at most n iterations. ∎

For the following lemmas, consider a run of META-LEX with well-posed instance (G, v_0) as input. Let v_{out} be the complete voltage assignment output by META-LEX. Let E_i be the set of edges E' and G_i be the graph G' constructed in iteration i of META-LEX.

Lemma 24 *For every edge $e \in E_{i-1} \setminus E_i$, we have $\big|\mathsf{grad}[v_{\mathsf{out}}](e)\big| \leq \alpha_i^\star$. Moreover, $\alpha_i^\star$ is non-increasing with i.*

Proof of Lemma 24: Let $P_i^\star = (x_0, \ldots, x_r)$ be a steepest fixable path in iteration i (when we deal with instance (G_{i-1}, v_{i-1})). Consider a terminal path P_{i+1} in (G_i, v_i) such that $\{\partial_0 P_{i+1}, \partial_1 P_{i+1}\} \cap (T(v_i) \setminus T(v_{i-1})) \neq \emptyset$. We claim that $\nabla P_{i+1}(v_i) \leq \alpha_i^\star$. On the contrary, assume that $\nabla P_{i+1}(v_i) > \alpha_i^\star$. Consider the case $\partial_0 P_{i+1} \in T(v_i) \setminus T(v_{i-1}), \partial_1 P_1 \in T(v_{i-1})$. By the definition of v_i, we must have $\partial_0 P_{i+1} = x_j$ for some $j \in [r-1]$. Let P_{i+1}' be the path formed by joining paths $P_i^\star[x_0 : x_j]$ and P_{i+1}. P_{i+1}' is a free terminal path in (G_{i-1}, v_{i-1}). We have,

$$
\begin{aligned}
v_{i-1}(x_0) - v_{i-1}(\partial_1 P_{i+1}) &= (v_i(x_0) - v_i(x_j)) + (v_i(\partial_0 P_{i+1}) - v_i(\partial_1 P_{i+1})) \\
&> \alpha_i^\star \cdot \ell(P_i^\star[x_0 : x_j]) + \alpha_i^\star \cdot \ell(P_{i+1}) = \alpha_i^\star \cdot \ell(P_{i+1}'),
\end{aligned}
$$

giving $\nabla P_{i+1}'(v_i) > \alpha_i^\star$, which is a contradiction since the steepest fixable path $P_i^\star$ in (G_{i-1}, v_{i-1}) has gradient $\alpha_i^\star$. The other cases can be handled similarly.

16

Applying the above claim to an edge $e \in E_{i-1} \setminus E_i$, whose gradient is fixed for the first time in iteration i, we obtain that $\mathsf{grad}[v_{i+1}](e) \leq \alpha_i^\star$. If v is the complete voltage assignment output by META-LEX, since v extends v_{i+1}, we get $\mathsf{grad}[v_{\mathsf{out}}](e) \leq \alpha_i^\star$. Applying the claim to the symmetric edge, we obtain $-\mathsf{grad}[v_{\mathsf{out}}](e) \leq \alpha_i^\star$, implying $|\mathsf{grad}[v_{\mathsf{out}}](e)| \leq \alpha_i^\star$.

Consider any free terminal path P_{i+1} in (G_i, v_i). If P_{i+1} is also a terminal path in (G_{i-1}, v_{i-1}), it is a free terminal path in (G_{i-1}, v_{i-1}). In addition, since a steepest fixable path $P_i^\star$ in (G_{i-1}, v_{i-1}) has $\nabla P_i^\star = \alpha_i^\star$, we get $\nabla P_{i+1}(v_i) = \nabla P_{i+1}(v_{i-1}) \leq \alpha_i^\star$. Otherwise, we must have $\{\partial_0 P_{i+1}, \partial_1 P_{i+1}\} \cap (T(v_i) \setminus T(v_{i-1})) \neq \emptyset$, and we can deduce $\nabla P_{i+1}(v_i) \leq \alpha_i^\star$ using the above claim. Thus, all free terminal paths P_{i+1} in (G_i, v_i) satisfy $\nabla P_{i+1}(v_i) \leq \alpha_i^\star$. In particular, $\alpha_{i+1}^\star = \nabla P_{i+1}^\star(v_i) \leq \alpha_i^\star$. Thus, $\alpha_i^\star$ is non-increasing with i. ∎

Lemma 25 *For any complete voltage assignment v for G that extends v_0, if $v \neq v_{\mathsf{out}}$, we have $\mathsf{grad}[v] \not\preceq \mathsf{grad}[v_{\mathsf{out}}]$, and hence $\mathsf{grad}[v_{\mathsf{out}}] \preceq \mathsf{grad}[v]$.*

Proof of Lemma 25: Consider any complete voltage assignment v for G that extends v_0, such that $v \neq v_{\mathsf{out}}$. Thus, there exists a unique i such that v extends v_{i-1} but does not extend v_i. We will argue that $\mathsf{grad}[v] \not\preceq \mathsf{grad}[v_{\mathsf{out}}]$, and hence $\mathsf{grad}[v_{\mathsf{out}}] \preceq \mathsf{grad}[v]$. For every edge $e \in E \setminus E_{i-1}$ that has been fixed so far, $\mathsf{grad}[v](e) = \mathsf{grad}[v_{i-1}](e) = \mathsf{grad}[v_{\mathsf{out}}](e)$, and hence we can ignore these edges.

Since v extends v_{i-1} but not v_i, there exists an $x \in T(v_i) \setminus T(v_{i-1})$ such that $v(x) \neq v_i(x) = v_{\mathsf{out}}(x)$. Assume $v(x) < v_i(x)$ (the other case is symmetric). If $P_i^\star = (x_0, \ldots, x_r)$ is the steepest fixable path with gradient $\alpha_i^\star$ picked in iteration i, we must have $x = x_j$ for some $j \in [r-1]$. Thus,

$$\sum_{k=1}^{j}(v(x_{k-1}) - v(x_k)) = v(x_0) - v(x_j) > v_i(x_0) - v_i(x_j) = \alpha_i^\star \cdot \ell(P_i^\star[x_0 : x_j]) = \alpha_i^\star \cdot \sum_{k=1}^{j} \ell(x_{k-1}, x_k).$$

Thus, for some $k \in [j]$, we must have $\mathsf{grad}[v](x_{k-1}, x_k) > \alpha_i^\star$. Since P_i^* is a path in G_{i-1}, we have $\{x_{k-1}, x_k\} \not\subseteq T(v_{i-1})$. This gives $(x_{k-1}, x_k) \in (E_{i-1} \setminus E_i)$. But then, from Lemma 24, it follows that for all $e \in (E_{i-1} \setminus E_i)$, we have $|\mathsf{grad}[v_{\mathsf{out}}](e)| \leq \alpha_i^\star$. Thus, we have $\mathsf{grad}[v] \not\preceq \mathsf{grad}[v_{\mathsf{out}}]$. ∎

Lemma 26 *Let $P = (x_0, \ldots, x_r)$ be a steepest fixable path such that it does not have any edges in $T(v_0) \times T(v_0)$ and $v_1 = \mathsf{fix}_G[v_0, P]$. Then for every $i \in [r]$, we have $\mathsf{grad}[v_1](x_{i-1}, x_i) = \nabla P$.*

Proof of Lemma 26: Suppose this is not true and let $j \in [r]$ be the minimum number such that $\mathsf{grad}[v_1](x_{j-1}, x_j) \neq \nabla P$. By definition of v_1 we would necessarily have $j < r$ and $v_j \in T(v_0)$. Suppose $\mathsf{grad}[v_1](x_{j-1}, x_j) < \nabla P$. We would then have $v_1(x_0) - v_1(x_j) < \nabla P * \ell(P[x_0 : x_j])$. Since P does not have any edges in $T(v_0) \times T(v_0)$, $P_1 := (x_j, \ldots, x_r)$ would be a free terminal path with $\nabla P_1 > \nabla P$. This is a contradiction. Other cases can be ruled out similarly.

∎

Proof of Theorem 4: Consider an arbitrary run of META-LEX on (G, v_0). Let v_{out} be the complete voltage assignment output by META-LEX. Proposition 22 implies that v_{out} extends v_0. Lemma 25 implies that for any complete voltage assignment $v \neq v_{\mathsf{out}}$ that extends v_0, we have $\mathsf{grad}[v_{\mathsf{out}}] \preceq \mathsf{grad}[v]$. Thus, v_{out} is a lex-minimizer. Moreover, the lemma also gives that for any such v, $\mathsf{grad}[v] \not\preceq \mathsf{grad}[v_{\mathsf{out}}]$. and hence

v_{out} is a unique lex-minimizer. Thus, v_{out} is the unique voltage assignment satisfying Def. 1, and we denote it as $\text{lex}_G[v_0]$. Since we started with an arbitrary run of META-LEX, uniqueness implies that every run of META-LEX on (G, v_0) must output $\text{lex}_G[v_0]$. ∎

Proof of Lemma 6: Suppose we have a complete voltage assignment v extending v_0, such that $\left\|\text{grad}[v]\right\|_\infty \leq \alpha$. For any terminal path $P = (x_0, \ldots, x_r)$, we get,

$$\nabla P(v_0) = v_0(\partial_0 P) - v_0(\partial_1 P) = v(\partial_0 P) - v(\partial_1 P) = \sum_{i=1}^{r} \text{grad}[v](x_{i-1}, x_i) \leq \alpha \cdot \sum_{i=1}^{r} \ell(x_{i-1}, x_i) = \alpha \cdot \ell(P),$$

giving $\nabla P(v_0) \leq \alpha$.

On the other hand, suppose every terminal path P in (G, v_0) satisfies $\nabla P(v_0) \leq \alpha$. Consider $v = \text{lex}_G[v_0]$. We know that v extends v_0. For every edge $e \in E_G \cap T(v_0) \times T(v_0)$, e is a (trivial) terminal path in (G, v_0), and hence has satisfies $\text{grad}[v](e) = \text{grad}[v_0](e) = \nabla e(v_0) \leq \alpha$. Considering the reverse edge, we also obtain $-\text{grad}[v](e) \leq \alpha$. Thus, $|\text{grad}[v](e)| \leq \alpha$. Moreover, using Lemma 24, we know that for edge $e \in E_G \setminus T(v_0) \times T(v_0)$, $|\text{grad}[v](e)| \leq \alpha_1^\star = \nabla P_1^\star \leq \alpha$ since P_1 is a terminal path in (G, v_0). Thus, for every $e \in E_G$, $|\text{grad}[v](e)| \leq \alpha$, and hence $\left\|\text{grad}[v]\right\|_\infty \leq \alpha$. ∎

A.2. Stability

The following theorem states that $\text{lex}_G[v_0]$ is monotonic with respect to v_0 and it respects scaling and translation of v_0. Our main stability result, stated earlier as Theorem 7, is an immediate corollary.

Theorem 27 *Let (G, v_0) be a well-posed instance with $T := T(v_0)$ as the set of terminals. Then the following statements hold.*

1. *For any $c, d \in \mathbb{R}$, v_1 a partial assignment with terminals $T(v_1) = T$ and $v_1(t) = cv_0(t) + d$ for all $t \in T$. Then, $\text{lex}_G[v_1](i) = c \cdot \text{lex}_G[v_0](i) + d$ for all $i \in V_G$.*

2. *v_1 a partial assignment with terminals $T(v_1) = T$. Suppose further that $v_1(t) \geq v_0(t)$ for all $t \in T$. Then, $\text{lex}_G[v_1](i) \geq \text{lex}_G[v_0](i)$ for all $i \in V_G$.*

For any well-posed (G, v_0), there could be several possible executions of META-LEX, each characterized by the sequence of paths $P_i^\star$. We can apply Theorem 4 to deduce the following structural result about the lex-minimizer.

Corollary 28 *For any well-posed instance (G, v_0), consider a sequence of paths $(P_1, \ldots, P_r)$ and voltage assignments $(v_1, \ldots, v_r)$ for some positive integer r such that:*

1. *$P_i^\star$ is a steepest fixable path in (G_{i-1}, v_{i-1}) for $i = 1, \ldots, r$.*

2. *$v_i = \text{fix}[v_{i-1}, P_i^\star]$ for $i = 1, \ldots, r$.*

3. *$T(v_r) = V_G$.*

Then, we have $v_r = \text{lex}_G[v_0]$.

We call such a sequence of paths and voltages to be a decomposition of $\text{lex}_G[v_0]$. Again, note that $\text{lex}_G[v_0]$ can possibly have multiple decompositions. However, any two such decompositions are *consistent* in the sense that they produce the same voltage assignment.

Proof of Theorem 7: We first define some operations on partial assignments which simplifies the notation. Let v_0, v_1 be any two partial assignments with the same set of terminals $T := T(v_0) = T(v_1)$ and $c, d \in \mathbb{R}$. By $cv_0 + d$ we mean a partial assignment v with $T(v) = T$ satisfying $v(t) = cv_0(t) + d$ for all $t \in T$. Also, by $v_0 + v_1$ we mean a partial assignment v with $T(v) = T$ satisfying $v(t) = v_0(t) + v_1(t)$ for all $t \in T$. Also, we say $v_1 \geq v_0$ if $v_1(t) \geq v_0(t)$ for all $t \in T$.

Now we can show how Theorem 7 follows from Theorem 27. Let $v := v_1 - v_0$, and $\|v\|_\infty = \epsilon$, for some $\epsilon > 0$. Therefore, $v_0 + \epsilon \geq v_1 \geq v_0 - \epsilon$. Theorem 27 then implies that $\text{lex}_G[v_0] + \epsilon \geq \text{lex}[v_1] \geq \text{lex}[v_0] - \epsilon$, hence proving the corollary. ∎

Proof sketch of Theorem 27: It is easy to see that the first statement holds. For the second statement, we first observe that if there is a sequence of paths $P_1, ..., P_r$ that is simultaneously a decomposition of both $\text{lex}[v_0]$ and $\text{lex}[v_1]$, then this is easy to see. If such a path sequence doesn't exist, then we look at $v_t := v_0 + t(v_1 - v_0)$. We state here without a proof (though the proof is elementary) that we can then split the interval $[0, 1]$ into finitely many subintervals $[a_0, a_1], [a_1, a_2], .., [a_{k-1}, a_k]$, with $a_0 = 0, a_k = 1$, such that for any i, there is a path sequence $P_1, ..., P_r$ which is a decomposition of $\text{lex}[v_t]$ for all $t \in [a_i, a_{i+1}]$. We then observe that $v_0 = v_{a_0} \leq v_{a_1} \leq ... v_{a_k} = v_1$. Since for every a_i, a_{i+1}, there is a path sequence which is simultaneously a decomposition of both $\text{lex}[v_{a_i}]$ and $\text{lex}[v_{a_{i+1}}]$, we immediately get

$$\text{lex}[v_0] = \text{lex}[v_{a_0}] \leq \text{lex}[v_{a_1}] \leq ... \leq \text{lex}[v_{a_k}] = \text{lex}[v_1].$$

∎

A.3. Alternate Characterizations

Proof of Theorem 10: We know that $\text{lex}_G[v_0]$ extends v_0. We first prove that $v = \text{lex}_G[v_0]$ satisfies the max-min gradient averaging property. Assume to the contrary. Thus, there exists $x \in V_G \setminus T(v_0)$ such that

$$\max_{y:(x,y)\in E_G} \text{grad}[v](x, y) \neq - \min_{y:(x,y)\in E_G} \text{grad}[v](x, y).$$

Assume that $\max_{(x,y)\in E_G} \text{grad}[v](x, y) \geq - \min_{(x,y)\in E_G} \text{grad}[v](x, y)$. Then, consider v' extending v_0 that is identical to v except for $v'(x) = v(x) - \epsilon$ for $\epsilon > 0$. For ϵ small enough, we get that

$$\max_{y:(x,y)\in E_G} \text{grad}[v'](x, y) < \max_{y:(x,y)\in E_G} \text{grad}[v](x, y)$$

and

$$- \min_{y:(x,y)\in E_G} \text{grad}[v'](x, y) < \max_{y:(x,y)\in E_G} \text{grad}[v](x, y).$$

The gradient of edges not incident on the vertex x is left unchanged. This implies that $\text{grad}[v] \not\preceq \text{grad}[v']$, contradicting the assumption that v is the lex-minimizer. (The other case is similar).

For the other direction. Consider a complete voltage assignment v extending v_0 that satisfies the max-min gradient averaging property w.r.t. (G, v_0). Let

$$\alpha = \max_{\substack{(x,y) \in E_G \\ x \in V \setminus T(v_0)}} \text{grad}[v](x, y) \geq 0$$

be the maximum edge gradient, and consider any edge $(x_0, x_1) \in E_G$ such that $\text{grad}[v](x_1, x_0) = \alpha$, with $x_1 \in V \setminus T(v_0)$. If $\alpha = 0$, $\text{grad}[v]$ is identically zero, and is trivially the lex-minimal gradient assignment. Thus, both v and $\text{lex}_G[v_0]$ are constant on each connected component. Since (G, v_0) is well-posed, there is at least one terminal in each component, and hence v and $\text{lex}_G[v_0]$ must be identical.

Now assume $\alpha > 0$. By the max-min gradient averaging property, $\exists x_2 \in V_G$ such that $(x_1, x_2) \in E_G$ and

$$\text{grad}[v](x_1, x_2) = \min_{y:(x_1,y) \in E_G} \text{grad}[v](x_1, y) = - \max_{y:(x_1,y) \in E_G} \text{grad}[v](x_1, y)$$

$$\leq -\text{grad}[v](x_1, x_0) = -\alpha.$$

Thus, $\text{grad}[v](x_2, x_1) \geq \alpha$. Since α is the maximum edge gradient, we must have $\text{grad}[v](x_2, x_1) = \alpha$. Moreover, $v(x_2) > v(x_1) > v(x_0)$, thus $x_2 \neq x_0$. We can inductively apply this argument at x_2 until we hit a terminal. Similarly, if $x_0 \notin T(v_0)$ we can extend the path in the other direction. Consequently, we obtain a path $P = (x_j, \ldots, x_2, x_1, x_0, x_{-1}, \ldots, x_k)$ with all vertices as distinct, such that $x_j, x_k \in T(v_0)$, and $x_i \in V \setminus T(v_0)$ for all $i \in [j+1, k-1]$. Moreover, $\text{grad}[v](x_i, x_{i-1}) = \alpha$ for all $j < i \leq k$. Thus, P is a free terminal path with $\nabla P[v_0] = \alpha$.

Moreover, since v is a voltage assignment extending v_0 with $\big\|\text{grad}[v]\big\|_\infty = \alpha$, using Lemma 6, we know that every terminal path P' in (G, v_0) must satisfy $\nabla P'(v_0) \leq \alpha$. Thus, P is a steepest fixable path in (G, v_0). Thus, letting $v_1 = \text{fix}[v_0, P]$, using Corollary 5, we obtain that $\text{lex}_G[v_1] = \text{lex}_G[v_0]$. Moreover, since $\alpha = \nabla P[v_0] = \text{grad}[v](x_i, x_{i-1})$ for all $i \in (j, k]$, we get $v_1(x_i) = v(x_i)$ for all $i \in (j, k)$. Thus, v extends v_1.

We can iterate this argument for r iterations until $T(v_r) = V_G$, giving $v = v_r$ and $v_r = \text{lex}_G[v_r] = \text{lex}_G[v_0]$. (Since we are fixing at least one terminal at each iteration, this procedure terminates). Thus, we get $v = \text{lex}_G[v_0]$. ∎

Appendix B. Description of the Algorithms

Algorithm 2: MODDIJKSTRA(G, v_0, α): Given a well-posed instance (G, v_0), a gradient value $\alpha \geq 0$, outputs a complete voltage assignment v for G, and an array parent $: V \to V \cup \{\text{null}\}$.

1. **for** $x \in V_G$,
2.   Add x to a fibonacci heap, with $\text{key}(x) = +\infty$.
3. $\text{finished}(x) \leftarrow$ **false**
4. **for** $x \in T(v_0)$
5.   Decrease $\text{key}(x)$ to $v_0(x)$.
6. $\text{parent}(x) \leftarrow$ null.
7. **while** heap is not empty
8. $x \leftarrow$ pop element with minimum key from heap
9. $v(x) \leftarrow \text{key}(x)$. $\text{finished}(x) \leftarrow$ **true** .

```
10.    for y : (x, y) ∈ E_G
11.      if finished(y) = false
12.        if key(y) > v(x) + α · ℓ(x, y)
13.          Decrease key(y) to v(x) + α · ℓ(x, y).
14.          parent(y) ← x.
15.  return (v, parent)
```

Theorem 29 *For a well-posed instance (G, V_0) and a gradient value $\alpha \geq 0$, let $(v, \mathsf{parent}) \leftarrow \text{MODDIJKSTRA}(G, v_0, \alpha)$. Then, v is a complete voltage assignment such that, $\forall x \in V_G$, $v(x) = \min_{t \in T(v_0)}\{v_0(t) + \alpha \mathsf{dist}(x, t)\}$. Moreover, the pointer array parent satisfies $\forall x \notin T(v_0)$, $\mathsf{parent}(x) \neq \mathsf{null}$ and $v(x) = v(\mathsf{parent}(x)) + \alpha \cdot \ell(x, \mathsf{parent}(x))$.*

Algorithm 3: Algorithm $\text{COMPVLOW}(G, v_0, \alpha)$: Given a well-posed instance (G, v_0), a gradient value $\alpha \geq 0$, outputs vLow, a complete voltage assignment for G, and an array $\mathsf{LParent} : V \to V \cup \{\mathsf{null}\}$.

1. $(\mathsf{vLow}, \mathsf{LParent}) \leftarrow \text{MODDIJKSTRA}(G, v_0, \alpha)$
2. **return** $(\mathsf{vLow}, \mathsf{LParent})$

Algorithm 4: Algorithm $\text{COMPVHIGH}(G, v_0, \alpha)$: Given a well-posed instance (G, v_0), a gradient value $\alpha \geq 0$, outputs vHigh, a complete voltage assignment for G, and an array $\mathsf{HParent} : V \to V \cup \{\mathsf{null}\}$.

1. **for** $x \in V_G$
2.   **if** $x \in T(v_0)$ **then** $v_1(x) \leftarrow -v_0(x)$ **else** $v_1(x) \leftarrow v_1(x)$.
3. $(\mathsf{temp}, \mathsf{HParent}) \leftarrow \text{MODDIJKSTRA}(G, v_1, \alpha)$
4. **for** $x \in V_G : \mathsf{vHigh}(x) \leftarrow -\mathsf{temp}(x)$
5. **return** $(\mathsf{vHigh}, \mathsf{HParent})$

Corollary 30 *For a well-posed instance (G, V_0) and a gradient value $\alpha \geq 0$, let $(\mathsf{vLow}[\alpha], \mathsf{LParent}) \leftarrow \text{COMPVLOW}(G, v_0, \alpha)$ and $(\mathsf{vHigh}[\alpha], \mathsf{HParent}) \leftarrow \text{COMPVHIGH}(G, v_0, \alpha)$. Then, $\mathsf{vLow}[\alpha], \mathsf{vHigh}[\alpha]$ are complete voltage assignments for G such that, $\forall x \in V_G$,*

$$\mathsf{vLow}[\alpha](x) = \min_{t \in T(v_0)} \{v_0(t) + \alpha \cdot \mathsf{dist}(x, t)\} \qquad \mathsf{vHigh}[\alpha](x) = \max_{t \in T(v_0)} \{v_0(t) - \alpha \cdot \mathsf{dist}(t, x)\}.$$

Moreover, the pointer arrays $\mathsf{LParent}, \mathsf{HParent}$ satisfy $\forall x \notin T(v_0)$, $\mathsf{LParent}(x), \mathsf{HParent}(x) \neq \mathsf{null}$ and

$$\mathsf{vLow}[\alpha](x) = \mathsf{vLow}[\alpha](\mathsf{LParent}(x)) + \alpha \cdot \ell(x, \mathsf{LParent}(x)),$$
$$\mathsf{vHigh}[\alpha](x) = \mathsf{vHigh}[\alpha](\mathsf{HParent}(x)) - \alpha \cdot \ell(x, \mathsf{HParent}(x)).$$

Algorithm 5: Algorithm $\text{COMPINFMIN}(G, v_0)$: Given a well-posed instance (G, v_0), outputs a complete voltage assignment v for G, extending v_0 that minimizes $\left\|\mathsf{grad}[\mathsf{v}]\right\|_\infty$.

1. $\alpha \leftarrow \max\{|\mathsf{grad}[v_0](e)| \mid e \in E_G \cap (T(v_0) \times T(v_0))\}$.
2. $E_G \leftarrow E_G \setminus (T(v_0) \times T(v_0))$
3. $P \leftarrow \text{STEEPESTPATH}(G, v_0)$.
4. $\alpha \leftarrow \max\{\alpha, \nabla P(v_0)\}$
5. $(\mathsf{vLow}, \mathsf{LParent}) \leftarrow \text{COMPVLOW}(G, v_0, \alpha)$
6. $(\mathsf{vHigh}, \mathsf{HParent}) \leftarrow \text{COMPVHIGH}(G, v_0, \alpha)$

```
    7.  for $x \in V_G$
    8.      if $x \in T(v_0)$
    9.          then $v(x) \leftarrow v_0(x)$
   10.          else $v(x) \leftarrow \frac{1}{2} \cdot (vLow(x) + vHigh(x))$.
   11.  return $v$
```

Algorithm 6: Algorithm COMPHIGHPRESSGRAPH(G, v_0, α): Given a well-posed instance (G, v_0), a gradient value $\alpha \geq 0$, outputs a minimal induced subgraph G' of G where every vertex has $pressure[v_0](\cdot) > \alpha$.

```
    1.  (vLow, LParent) $\leftarrow$ COMPVLOW$(G, v_0, \alpha)$
    2.  (vHigh, HParent) $\leftarrow$ COMPVHIGH$(G, v_0, \alpha)$
    3.  $V_{G'} \leftarrow \{x \in V_G \mid vHigh(x) > vLow(x)\}$
    4.  $E_{G'} \leftarrow \{(x, y) \in E_G \mid x, y \in V_{G'}\}$.
    5.  $G' \leftarrow (V', E', \ell)$
    6.  return $G'$
```

Proof of Lemma 13:

$$vHigh[\alpha](x) > vLow[\alpha](x)$$

is equivalent to

$$\max_{t \in T(v_0)} \{v_0(t) - \alpha \cdot dist(t, x)\} > \min_{t \in T(v_0)} \{v_0(t) + \alpha \cdot dist(x, t)\},$$

which implies that there exists terminals $s, t \in T(v_0)$ such that

$$v_0(t) - \alpha \cdot dist(t, x) > v_0(s) + \alpha \cdot dist(x, s)$$

thus,

$$pressure[v_0](x) \geq \frac{v_0(t) - v_0(s)}{dist(t, x) + dist(x, s)} > \alpha.$$

So the inequality on vHigh and vLow implies that pressure is strictly greater than α. On the other hand, if $pressure[v_0](x) > \alpha$, there exists terminals $s, t \in T(v_0)$ such that

$$\frac{v_0(t) - v_0(s)}{dist(t, x) + dist(x, s)} = pressure[v_0](x) > \alpha.$$

Hence,

$$v_0(t) - \alpha \cdot dist(t, x) > v_0(s) + \alpha \cdot dist(x, s)$$

which implies $vHigh[\alpha](x) > vLow[\alpha](x)$. ∎

Algorithm 7: Algorithm STEEPESTPATH(G, v_0): Given a well-posed instance (G, v_0), with $T(v_0) \neq V_G$, outputs a steepest free terminal path P in (G, v_0).

```
    1.  Sample uniformly random $e \in E_G$. Let $e = (x_1, x_2)$.
    2.  Sample uniformly random $x_3 \in V_G$.
    3.  for $i = 1$ to $3$
    4.      $P \leftarrow$ VERTEXSTEEPESTPATH$(G, v_0, x_i)$
    5.  Let $j \in \arg\max_{j \in \{1,2,3\}} \nabla P_j(v_0)$
    6.  $G' \leftarrow$ COMPHIGHPRESSGRAPH$(G, v_0, \nabla P_j(v_0))$
```

7. **if** $E_{G'} = \emptyset$,
8.   **then return** P_j
9.   **else return** STEEPESTPATH$(G', v_0|_{V_{G'}})$

Algorithm 8: Algorithm COMPLEXMIN(G, v_0): Given a well-posed instance (G, v_0), with $T(v_0) \neq V_G$, outputs $\text{lex}_G[v_0]$.

1. **while** $T(v_0) \neq V_G$
2. $E_G \leftarrow E_G \setminus (T(v_0) \times T(v_0))$
3. $P \leftarrow$ STEEPESTPATH(G, v_0)
4. $v_0 \leftarrow \text{fix}[v_0, P]$
5. **return** v_0

Algorithm 9: Algorithm VERTEXSTEEPESTPATH(G, v_0, x): Given a well-posed instance (G, v_0), and a vertex $x \in V_G$, outputs a steepest terminal path in (G, v_0) through x.

1. Using Dijkstra's algorithm, compute $\text{dist}(x, t)$ for all $t \in T(v_0)$
2. **if** $x \in T(v_0)$
3. $y \leftarrow \arg\max_{y \in T(v_0)} \frac{|v_0(x) - v_0(y)|}{\text{dist}(x,y)}$
4.   **if** $v_0(x) \geq v_0(y)$
5.   **then return** a shortest path from x to y
6.   **else return** a shortest path from y to x
7. **else**
8.   **for** $t \notin T(v_0)$, $\text{d}(t) \leftarrow \text{dist}(x, t)$
9. $(t_1, t_2) \leftarrow$ STARSTEEPESTPATH$(T(v_0), v_0|_{T(v_0)}, \text{d})$
10.   Let P_1 be a shortest path from t_1 to x. Let P_2 be a shortest path from x to t_2.
11. $P \leftarrow (P_1, P_2)$. **return** P.

Algorithm 10: STARSTEEPESTPATH(T, v, d): Returns the steepest path in a star graph, with a single non-terminal connected to terminals in T, with lengths given by d, and voltages given by v.

1. Sample t_1 uniformly and randomly from T
2. Compute $t_2 \in \arg\max_{t \in T} \frac{|v(t_1) - v(t)|}{\text{d}(t_1) + \text{d}(t)}$
3. $\alpha \leftarrow \frac{|v(t_2) - v(t_1)|}{\text{d}(t_1) + \text{d}(t_2)}$
4. Compute $v_{\text{low}} \leftarrow \min_{t \in T}(v(t) + \alpha \cdot \text{d}(t))$
5. $T_{\text{low}} \leftarrow \{t \in T \mid v(t) > v_{\text{low}} + \alpha \cdot \text{d}(t)\}$
6. Compute $v_{\text{high}} \leftarrow \max_{t \in T}(v(t) - \alpha \cdot \text{d}(t))$
7. $T_{\text{high}} \leftarrow \{t \in T \mid v(t) < v_{\text{high}} - \alpha \cdot \text{d}(t)\}$
8. $T' \leftarrow T_{\text{low}} \cup T_{\text{high}}$.
9. **if** $T' = \emptyset$
10.   **then if** $v(t_1) \geq v(t_2)$ **then return** (t_1, t_2) **else return** (t_2, t_1)
11.   **else return** STARSTEEPESTPATH$(T', v|_{T'}, \text{d}_{T'})$

B.1. Faster Lex-minimization

Algorithm 11: Algorithm COMPFASTLEXMIN(G, v_0): Given a well-posed instance (G, v_0), with $T(v_0) \neq V_G$, outputs $\text{lex}_G[v_0]$.

1. **while** $T(v_0) \neq V_G$
2. $\quad$ $v_0 \leftarrow$ FIXPATHSABOVEPRESS$(G, v_0, 0)$
3. **return** v_0

Algorithm 12: Algorithm FIXPATHSABOVEPRESS(G, v_0, α): Given a well-posed instance (G, v_0), with $T(v_0) \neq V_G$, and a gradient value α, iteratively fixes all paths with gradient $> \alpha$.

1. **while** $T(v_0) \neq V_G$
2. $\quad$ $E_G \leftarrow E_G \setminus (T(v_0) \times T(v_0))$
3. $\quad$ Sample uniformly random $e \in E_G$. Let $e = (x_1, x_2)$.
4. $\quad$ Sample uniformly random $x_3 \in V_G$.
5. $\quad$ **for** $i = 1$ **to** 3
6. $\quad\quad$ $P_i \leftarrow$ VERTEXSTEEPESTPATH(G, v_0, x_i)
7. $\quad$ Let $j \in \arg\max_{j \in \{1,2,3\}} \nabla P_j(v_0)$
8. $\quad$ $G' \leftarrow$ COMPHIGHPRESSGRAPH$(G, v_0, \nabla P_j(v_0))$
9. $\quad$ **if** $E_{G'} = \emptyset$,
10. $\quad\quad$ **then** $v_0 \leftarrow \text{fix}[v_0, P]$
11. $\quad\quad$ **else** Let $G'_i, i = 1, \ldots, r$ be the connected components of G'.
12. $\quad\quad\quad$ **for** $i = 1, \ldots, r$
13. $\quad\quad\quad\quad$ $v_i \leftarrow$ FIXPATHSABOVEPRESS$(G'_i, v_0|_{V_{G'_i}}, \nabla P_j(v_0))$
14. $\quad\quad\quad\quad$ **for** $x \in V_{G'_i}$, set $v_0(x) \leftarrow v_i(x)$
15. $\quad$ **if** $\alpha > 0$ **then** $G \leftarrow$ COMPHIGHPRESSGRAPH(G, v_0, α)
16. **return** v_0

Appendix C. Experiments on WebSpam: Testing More Algorithms

For completeness, in this appendix we show how a number of algorithms perform on the web spam experiment of Section 6. We consider the following algorithms:

- RANDWALK along in-links. For a detailed description see Zhou et al. (2007). This algorithm essentially performs a Personalized PageRank random walk from each vertex x and computes a spam-value for the vertex x by taking a weighted average of the labels of the vertices where the random walk from x terminates. *Also shown in Section 6.*

- DIRECTEDLEX, with edges in the opposite directions of links. This has the effect that a link to a spam host is evidence of spam, and a link from a normal host is evidence of normality. *Also shown in Section 6.*

- RANDWALK along out-links.

- DIRECTEDLEX, with edges in the directions of links. This has the effect that a link from to a spam host is evidence of spam, and a link to a normal host is evidence of normality.

- UNDIRECTEDLEX: Lex-minimization with links treated as undirected edges.

- LAPLACIAN: l_2-regression with links treated as undirected edges.

- DIRECTED 1-NEAREST NEIGHBOR: Uses shortest distance along paths following out-links. *Spam-ratio* is defined distance from normal hosts, divided by distance to spam hosts. Sites are flagged as spam when spam-ratio exceeds some threshold. We also tried following paths along in-links instead, but that gave much worse results.

We use the experimental setup described in Section 6. Results are shown in Figure 4. The alternative convention for DIRECTEDLEX orients edges in the directions of links. This takes a link from a spam host to be evidence of spam, and a link to a normal host to be evidence of normality. This approach performs significantly worse than our preferred convention, as one would intuitively expect. UNDIRECTEDLEX and LAPLACIAN approaches also perform significantly worse. DIRECTED 1-NEAREST NEIGHBOR performs poorly, demonstrating that DIRECTEDLEX is very different from that approach. As observed by Zhou et al. (2007), sampling based on a random walk following out-links performs worse than following in-links. Up to 60 % recall, DIRECTEDLEX performs best, both in the regime of 5 % labels for training and in the regime of 20 % labels for training.
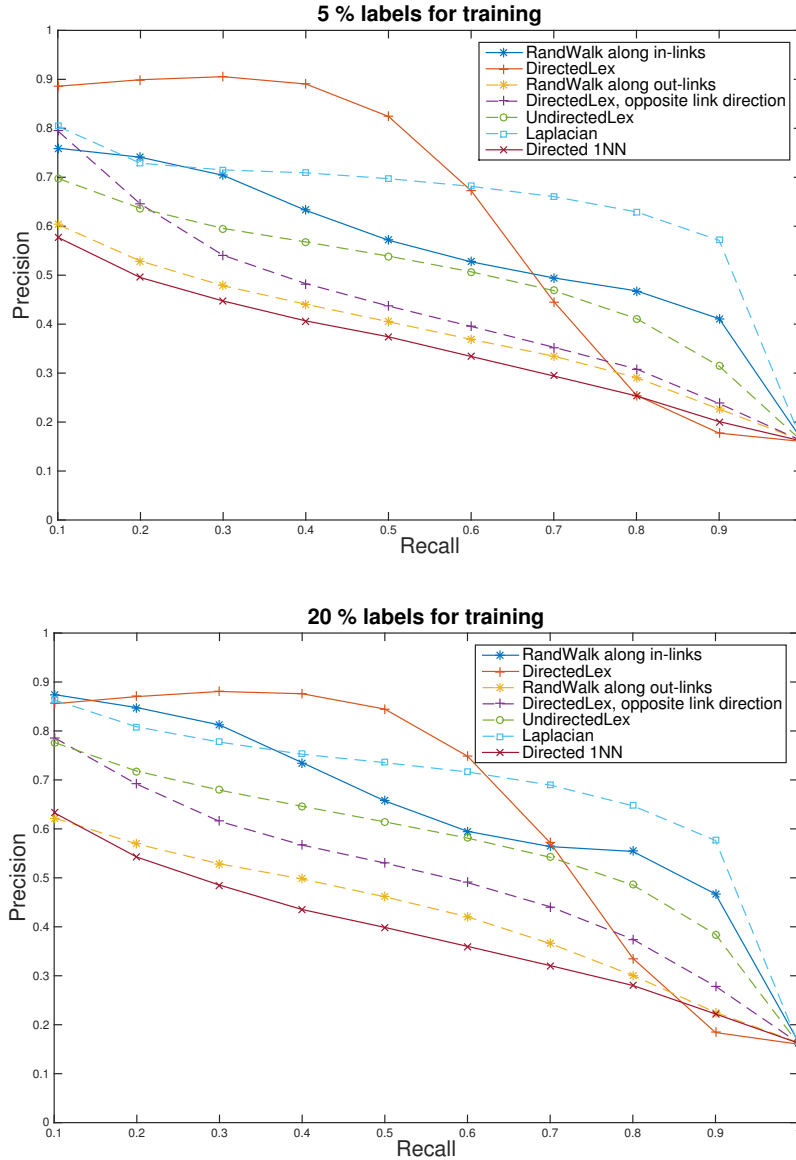
Figure 4: Recall and precision in the WebSpam classification experiment. Each data point shown was computed as an average over 100 runs. The largest standard deviation of the mean precision across the plotted recall values was less than 1.5 %. The algorithm of Zhou et al. (2007) appears as RANDWALK (along in-links). We also show RANDWALK along out-links. Our directed lex-minimization algorithm appears as DIRECTEDLEX. We also show DIRECTEDLEX with link directions reversed, along with UNDIRECTEDLEX and LAPLACIAN.

Appendix D. l_0-Vertex Regularization Proofs

In this appendix, we prove Theorem 19 and Theorem 20. For the purposes of proving the second theorem, we introduce an alternative version of problem (3). The optimization problem here requires us to minimize

l_0-regularization budget required to obtain an inf-minimizer with gradient below a given threshold:

$$\min_{v \in \mathbb{R}^n} \ \left\| v(T) - v_0(T) \right\|_0$$
$$\text{subject to } \left\| \text{grad}_G[v] \right\|_\infty \leq \alpha. \tag{6}$$

We will also need the following graph construction.

Definition 31 *The α-pressure terminal graph of a partially-labeled graph (G, v_0) is a directed unweighted graph $G_\alpha = (T(v_0), \widehat{E})$ such that $(s, t) \in \widehat{E}$ if and only if there is a terminal path P from s to t in G with*

$$\nabla P(v_0) > \alpha.$$

Note that the α-pressure terminal graph has $O(n)$ vertices but may be dense, even when G is not.

Algorithm 13: Algorithm TERM-PRESSURE: Given a well-posed instance (G, v_0) and $\alpha \geq 0$, outputs α pressure terminal graph G_α.

Initialize G_α with vertex set $V_\alpha = T(v_0)$ and edge set $\widehat{E} = \emptyset$.
for each terminal $s \in T(v_0)$

1. Compute the distances to every other terminal t by running Dijktra's algorithm, allowing shortest paths that run through other terminals.
2. Use the resulting distances to check for every other terminal t if there is a terminal path P from s to t with $\nabla P(v_0) > \alpha$. If there is, add edge (s, t) to $\widehat{E}$.

Lemma 32 *The α-pressure terminal graph of a voltage problem (G, v_0) can be computed in $O((m + n \log n)n)$ time using algorithm TERM-PRESSURE (Algorithm 13).*

Proof The correctness of the algorithm follows from the fact that Dijkstra's algorithm will identify all shortest distances between the terminals, and the pressure check will ensure that terminal pairs (s, t) are added to $\widehat{E}$ if and only if they are the endpoints of a terminal path P with $\nabla P(v_0) > \alpha$. The running time is dominated by performing Dijkstra's algorithm once for each terminal. A single run of Dijkstra's algorithm takes $O(m + n \log n)$ time, and this is performed at most n times, for a total running time of $O((m + n \log n)n)$. ∎

We make three observations that will turn out to be crucial for proving Theorems 19 and 20.

Observation 33 *G_α is a subgraph of G_β for $\alpha \geq \beta$.*

Proof Suppose edge (s, t) appears in G_α, then for some path P

$$\nabla P(v_0) > \alpha \geq \beta,$$

so the edge also appears in G_β. ∎

Observation 34 *G_α is transitively closed.*

Proof Suppose edges (s, t) and (t, r) appear in G_α. Let $P_{(s,t)}$, $P_{(t,r)}$, $P_{(s,r)}$ be the respective shortest paths in G between these terminal pairs. Then

$$\nabla P_{(s,r)}(v_0) = \frac{v_0(s) - v_0(r)}{\ell(P_{(s,r)})} \geq \frac{v_0(s) - v_0(r)}{\ell(P_{(s,t)}) + \ell(P_{(t,r)})} = \frac{v_0(s) - v_0(t) + v_0(t) - v_0(r)}{\ell(P_{(s,t)}) + \ell(P_{(t,r)})}$$

$$\geq \min\left(\frac{v_0(s) - v_0(t)}{\ell(P_{(s,t)})}, \frac{v_0(t) - v_0(r)}{\ell(P_{(t,r)})}\right) > \alpha. \tag{7}$$

So edge (s, r) also appears in G_α. This is sufficient for G_α to be transitively closed. ∎

Observation 35 G_α *is a directed acyclic graph.*

Proof Suppose for a contradiction that a directed cycle appears in G_α. Let s and t be two vertices in this cycle. Let $P_{(s,t)}$ and $P_{(t,s)}$ be the respective shortest paths in G between these terminal pairs. Because G_α is transitively closed, both edges (s, t) and (t, s) must appear in G_α. But $(s, t) \in \widehat{E}$ implies

$$v_0(s) - v_0(t) > \alpha \ell(P_{(s,t)}) > 0,$$

and similarly $(t, s) \in \widehat{E}$ implies

$$v_0(t) - v_0(s) > \alpha \ell(P_{(t,s)}) > 0.$$

This is a contradiction. ∎

The usefulness of the α-pressure terminal graph is captured in the following lemma. We define a vertex cover of a directed graph to be a vertex set that constitutes a vertex cover in the same graph with all edges taken to be undirected.

Lemma 36 *Given a partially-labeled graph (G, v_0) and a set $U \subseteq V$, there exists a voltage assignment $v \in \mathbb{R}^n$ that satisfies*

$$\{t \in T(v_0) : v(t) \neq v_0(t)\} \subseteq U \text{ and } \left\|\mathsf{grad}_G[v]\right\|_\infty \leq \alpha,$$

if and only if U is a vertex cover in the α-pressure terminal graph G_α of (G, v_0).

Proof We first show the "only if" direction. Suppose for a contradiction that there exists a voltage assignment v for which $\left\|\mathsf{grad}_G[v]\right\|_\infty \leq \alpha$, but U is not a vertex cover in G_α. Let (s, t) be an edge G_α which is not covered by U. The presence of this edge in G_α implies that there exists a terminal path P from s to t in G for which

$$\nabla P(v_0) > \alpha.$$

But, by Lemma 6 this means there is no assignment v for G which agrees with v_0 on s and t and has $\left\|\mathsf{grad}_G[v]\right\|_\infty \leq \alpha$. This contradicts our assumption.

Now we show the "if" direction. Consider an arbitrary vertex cover U of G_α. Suppose for a contradiction that there does not exist a voltage assignment v for G with $\left\|\mathsf{grad}_G[v]\right\|_\infty \leq \alpha$ and $\{t \in T(v_0) : v(t) \neq v_0(t)\} \subseteq U$. Define a partial voltage assignment v_U given by

$$v_U(t) = \begin{cases} v_0(t) & \text{if } t \in T(v_0) \setminus U \\ * & \text{o.w.} \end{cases}$$

28

The preceding statement is equivalent to saying that there is no v that extends v_U and has $\left\|\text{grad}_G[v]\right\|_\infty \leq \alpha$. By Lemma 6, this means there is terminal path between $s, t \in T(v_U)$ with gradient strictly larger than α. But this means an edge (s, t) is present in G_α and is not covered. This contradicts our assumption that U is a vertex cover. $\blacksquare$

Theorem 37 *The algorithm* KONIG-COVER *computes a minimum vertex cover for any transitively closed DAG G in polynomial time.*

We are now ready to prove Theorem 20.

Proof of Theorem 20: We describe and prove the algorithm OUTLIER. The algorithm will reduce problem (3) to problem (6): Suppose v^* is an optimal assignment for problem (3). It achieves a maximum gradient $\alpha^* = \left\|\text{grad}_G[v^*]\right\|_\infty$. Using Dijkstra's algorithm we compute the pairwise shortest distances between all terminals in G. From these distances and the terminal voltages, we compute the gradient on the shortest path between each terminal pair. By Lemma 6, α^* must equal one of these gradients. So we can solve problem (3) by iterating over the set of gradients between terminals and solving problem (6) for each of these $O(n^2)$ gradients. Among the assignments with $\left\|v(T) - v_0(T)\right\|_0 \leq k$, we then pick the solution that minimizes $\left\|\text{grad}_G[v]\right\|_\infty$.

In fact, we can do better. By Observation 33, G_α is a subgraph of G_β for $\alpha \geq \beta$. This means a vertex cover of G_α is also a vertex cover of G_β, and hence the minimum vertex cover for G_β is at least as large as the minimum vertex cover for G_α. This means we can do a binary search on the set of $O(n^2)$ terminal gradients to find the minimum gradient for which there exists an assignment with $\left\|v(T) - v_0(T)\right\|_0 \leq k$. This way, we only make $O(\log n)$ calls to problem (6), in order to solve problem (3).

We use the following algorithm to solve problem (6).

---

1. Compute the α-pressure terminal graph G_α of G using the algorithm TERM-PRESSURE.
2. Compute a minimum vertex cover U of G_α using the algorithm KONIG-COVER from Theorem 37.
3. Define a partial voltage assignment v_U given by

$$v_U(t) = \begin{cases} v_0(t) & \text{if } t \in T(v_0) \setminus U, \\ * & \text{otherwise.} \end{cases}$$

4. Using Algorithm 5, compute voltages v that extend v_U and output v.

From Lemma 32, it follows that step 1 computes the α-pressure terminal graph in polynomial time. From Theorem 37 it follows that step 2 computes the a minimum vertex cover of the α-pressure terminal graph in polynomial time, because our observations 34 and 35 establish that the graph is a TC-DAG. From Lemma 36 and Theorem 16, it follows that the output voltages solve program (6).

$\blacksquare$

To prove Theorem 19, we use the standard greedy approximation algorithm for MIN-VC (Vazirani (2001)).

Theorem 38 2-Approximation Algorithm for Vertex Cover. *The following algorithm gives a 2-approximation to the Minimum Vertex Cover problem on a graph $G = (V, E)$.*

> *0. Initialize $U = \emptyset$.*
> *1. Pick an edge $(u, v) \in E$ that is not covered by U.*
> *2. Add u and v to the set U.*
> *3. Repeat from step 1 if there are still edges not covered by U.*
> *4. Output U.*

We are now in a position to prove Theorem 19

Proof of Theorem 19: Given an arbitrary k and a partially-labeled graph (G, v_0), let α^* be the optimum value of program (3). Observe that by Lemma 36, this implies that G_{α^*} has a vertex cover of size k. Given the partial assignment v_0, for every vertex set U, we define

$$v_U(t) = \begin{cases} v_0(t) & \text{if } t \in T(v_0) \setminus U \\ * & \text{o.w.} \end{cases}$$

We claim the following algorithm APPROX-OUTLIER outputs a voltage assignment v with $\left\|\text{grad}_G[v]\right\|_\infty \leq \alpha^*$ and $\left\|v(T) - v_0(T)\right\|_0 \leq 2k$.

> Algorithm APPROX-OUTLIER:
>
> 0. Initialize $U = \emptyset$.
> 1. Using the algorithm STEEPESTPATH (Algorithm 7), find a steepest terminal path in G w.r.t. v_U. Denote this path P and let s and t be its terminal endpoints. If there is no terminal path with positive gradient, skip to step 4.
> 2. Add s and t to the set U.
> 3. If $|U| \leq 2k - 2$ then repeat from step 1.
> 4. Using the algorithm COMPINFMIN (Algorithm 5), compute voltages v that extend v_U and output v.

From the stopping conditions, it is clear that $|U| \leq 2k$. If in step 1 we ever find that no terminal paths have positive gradient then our v that extends v_U will have $\left\|\text{grad}_G[v]\right\|_\infty = 0 \leq \alpha^*$, by Lemma 6. Similarly if we find a steepest path with gradient less than α^* w.r.t. v_U, then for this U there exists v that extends v_U and has $\left\|\text{grad}_G[v]\right\|_\infty \leq \alpha^*$. This will continue to hold when if we add vertices to U. Therefore, for the final U, there will exist an v that extends v_U and has $\left\|\text{grad}_G[v]\right\|_\infty \leq \alpha^*$.

If we never find a steepest terminal path P with $\nabla P(v_0) \leq \alpha^*$, then each steepest path we find corresponds to an edge in G_{α^*} that is not yet covered by U and our algorithm in fact implements the greedy approximation algorithm for vertex cover described in Theorem 38. This implies that the final U is a vertex cover of G_{α^*} of size at most $2k$. By Lemma 36, this implies that there exists a voltage assignment u extending v_U that has $\left\|\text{grad}_G[u]\right\|_\infty \leq \alpha^*$. This implies by Theorem 16 that the v we output has $\left\|\text{grad}_G[v]\right\|_\infty \leq \alpha^*$.

In all cases, the v we output extends v_U, so $\left\|v(T) - v_0(T)\right\|_0 \leq |U| \leq 2k$. ∎

Appendix E. Proof of Hardness of l_0 regularization for l_2

We will prove Theorem 21, by a reduction from minimum bisection. To this end, let $G = (V, E)$ be any graph. We will reduce the minimum bisection problem on G to our regularization problem. Let $n = |V|$.

The graph on which we will perform regularization will have vertex set

$$V \cup \widehat{V},$$

where $\widehat{V}$ is a set of n vertices that are in 1-to-1 correspondence with V. We assume that every edge in G has weight 1. We now connect every vertex in $\widehat{V}$ to the corresponding vertex in V by an edge of weight B, for some large B to be determined later. We also connect all of the vertices in $\widehat{V}$ to each other by edges of weight B^3. So, we have a complete graph of weight B^3 edges on $\widehat{V}$, a matching of weight B edges connecting $\widehat{V}$ to V, and the original graph G on V. The input potential function will be

$$v(a) = \begin{cases} 0 & \text{for } a \in \widehat{V}, \text{ and} \\ 1 & \text{for } a \in V. \end{cases}$$

Now set $k = n/2$. We claim that we will be able to determine the value of the minimum bisection from the solution to the regularization problem.

If S is the set of vertices on which v and w differ, then we know that the w is harmonic on S: for every $a \in S$, $w(a)$ is the weighted average of the values at its neighbors. In the following, we exploit the fact that $|S| \leq n/2$.

Claim 39 *For every $a \in S \cap \widehat{V}$, $w(a) \leq 2/nB^2$.*

Proof Let a be the vertex in $S \cap \widehat{V}$ that maximizes $w(a)$. So, a is connected to at least $n/2$ neighbors in $\widehat{V}$ with w-value equal to 0 by edges of weight B^3. On the other hand, a has only one neighbor that is not in $\widehat{V}$, that vertex has w-value at most 1, and it is connected to that vertex by an edge of weight B. Call that vertex c. We have

$$((n-1)B^3 + B)w(a) = Bw(c) + \sum_{b \in \widehat{V}, b \neq a} B^3 w(b)$$

$$= Bw(c) + \sum_{b \in \widehat{V} \cap S, b \neq a} B^3 w(b) + \sum_{b \in \widehat{V} - S} B^3 w(b)$$

$$\leq B + \sum_{b \in \widehat{V} \cap S, b \neq a} B^3 w(a)$$

$$\leq B + (n/2 - 1)B^3 w(a).$$

Subtracting $(n/2 - 1)B^3 w(a)$ from both sides gives

$$((n/2)B^3 + B)w(a) \leq B,$$

which implies the claim. ∎

Claim 40 *For $a \in S \cap V$, $w(a) \leq n/B$.*

Proof Vertex a has exactly one neighbor in $\widehat{V}$. Let's call that neighbor c. We know that $w(c) \leq 2/B^2 n$. On the other hand, vertex a has fewer than $n - 1$ neighbors in V, and each of these have w-value at most 1. Let d_a denote the degree of a in G. Then,

$$(B + d_a)w(a) \leq d_a + B \frac{2}{B^2 n}.$$

31

So,

$$w(a) \leq \frac{d_a + 2/Bn}{d_a + B}$$
$$\leq \frac{n + (2/Bn)}{B + n}$$
$$\leq n/B.$$

$\blacksquare$

We now estimate the value of the regularized objective function. To this end, we assume that

$$|S| = k = n/2.$$

Let

$$T = S \cap V,$$

and

$$t = |T|.$$

We will prove that $S \subset V$ and so $S = T$ and $t = n/2$.

Let δ denote the number of edges on the boundary of T in V. Once we know that $t = n/2$, δ is the size of a bisection.

Claim 41 *The contribution of the edges between V and $\widehat{V}$ to the objective function is at least*

$$(n - t)B - 4/B$$

and at most

$$(n - t)B + tn^2/B.$$

Proof For the lower bound, we just count the edges between vertices in $V \setminus T$ and $\widehat{V}$. There are $n - t$ of these edges, and each of them has weight B. The endpoint in $V \setminus T$ has w-value 1, and the endpoint in $\widehat{V}$ has w-value at most $2/nB^2$. So, the contribution of these edges is at least

$$(n - t)B(1 - 2/nB^2)^2 \geq (n - t)B(1 - 4/nB^2) \geq (n - t)B - 4/B.$$

For the upper bound, we observe that the difference in w-values across each of these $n - t$ edges is at most 1, so their total contribution is at most

$$(n - t)B.$$

Since for every vertex $a \in T$, $w(a) \leq n/B$, and also every vertex $b \in \widehat{V}$, $w(b) \leq 2/nB^2$, the contribution due to edges between T and $\widehat{V}$ is at most

$$t(n/B)^2 B = tn^2/B.$$

$\blacksquare$

We will see that this is the dominant term in the objective function. The next-most important term comes from the edges in G.

Claim 42 *The contribution of the edges in G to the objective function is at least*

$$\delta(1 - 2n/B)$$

and at most

$$\delta + (t^2/2)(n/B)^2$$

Proof Let $(a, b) \in E$. If neither a nor b is in T, then $w(a) = w(b) = 1$, and so this edge has no contribution. If $a \in T$ but $b \notin T$, then the difference in w-values on them is between $(1 - n/B)$ and 1. So, the contribution of such edges to the objective function is between

$$\delta(1 - 2n/B) \text{ and } \delta.$$

Finally, if a and b are in T, then the difference in w-values on them is at most n/B, and so the contribution of all such edges to the objective function is at most

$$(t^2/2)(n/B)^2.$$

$\blacksquare$

Claim 43 *The edges between pairs of vertices in $\widehat{V}$ contribute at most $2/B$ to the objective function.*

Proof As $0 \leq w(a) \leq 2/B^2 n$ for every $a \in \widehat{V}$, every edge between two vertices in $\widehat{V}$ can contribute at most

$$B^3 (2/B^2 n)^2 = 4/Bn^2.$$

As there are fewer than $n^2/2$ such edges, their total contribution to the objective function is at most

$$(n^2/2)(4/Bn^2) = 2/B.$$

$\blacksquare$

Lemma 44 *If $n \geq 4$ and $B = 2n^3$, the value of the objective function is at least*

$$(n - t)B + \delta - 1/2$$

and at most

$$(n - t)B + \delta + 1/3.$$

Proof Summing the contributions in the preceding three claims, we see that the value of the objective function is at least

$$
\begin{aligned}
(n - t)B - 4/B + \delta(1 - 2n/B) &\geq (n - t)B + \delta - 4/B - 2n\delta/B \\
&\geq (n - t)B + \delta - n^3/B \\
&\geq (n - t)B + \delta - 1/2,
\end{aligned}
$$

as $\delta \leq (n/2)^2$.

Similarly, the objective function is at most

$$(n-t)B + tn^2/B + \delta + (t^2/2)(n/B)^2 + 2/B \le (n-t)B + n^3/2B + \delta + n^4/8B^2 + 2/B$$
$$\le (n-t)B + n^3/2B + \delta + 1/32n^2 + 1/n^3$$
$$\le (n-t)B + \delta + 1/3.$$

■

Claim 45 *If $n \ge 2$ and $B = 2n^3$, then $S \subset V$.*

Proof The objective function is minimized by making t as large as possible, so $t = n/2$ and $S \subset V$. ■

Theorem 46 *The value of the objective function reveals the value of the minimum bisection in G.*

Proof The value of the objective function will be between

$$(n/2)B + \delta - 1/2$$

and

$$(n/2)B + \delta + 1/3.$$

So, the objective function will be smallest when δ is as small as possible. ■

Theorem 46 immediately implies Theorem 21.