Algorithmica (2015) 71:450–470
DOI 10.1007/s00453-013-9814-z

Isotonic Regression for Multiple Independent Variables

Quentin F. Stout

Received: 24 March 2012 / Accepted: 15 July 2013 / Published online: 1 August 2013

Abstract This paper gives algorithms for determining isotonic regressions for weighted data at a set of points P in multidimensional space with the standard componentwise ordering. The approach is based on an order-preserving embedding of P into a slightly larger directed acyclic graph (dag) G, where the transitive closure of the ordering on P is represented by paths of length 2 in G. Algorithms are given which, compared to previous results, improve the time by a factor of $\tilde{\Theta}(|P|)$ for the L_1, L_2, and L_∞ metrics. A yet faster algorithm is given for L_1 isotonic regression with unweighted data. L_∞ isotonic regression is not unique, and algorithms are given for finding L_∞ regressions with desirable properties such as minimizing the number of large regression errors.

Keywords Isotonic regression algorithm · Monotonic · Multidimensional ordering · Transitive closure

1 Introduction

A directed acyclic graph (dag) $G = (V, E)$ defines a partial order $\prec$ over the vertices, where for $u, v \in V$, $u \prec v$ if and only if there is a path from u to v in G. A real-valued function h on V is *isotonic* iff whenever $u \prec v$ then $h(u) \leq h(v)$, i.e., it is a weakly order-preserving map of the dag into the real numbers. By *weighted data* on G we mean a pair of real-valued functions (f, w) on G where w, the weights, is

Q.F. Stout (✉)
Computer Science and Engineering, University of Michigan, Ann Arbor, MI 48109-2121, USA
e-mail: qstout@umich.edu

nonnegative and f, the values, is arbitrary. Given weighted data (f, w) on G, an L_p *isotonic regression* is an isotonic function g on V that minimizes

$$\begin{cases} \left(\sum_{v \in V} w(v) \left| f(v) - g(v) \right|^p \right)^{1/p} & \text{if } 1 \le p < \infty \\ \max_{v \in V} w(v) \left| f(v) - g(v) \right| & p = \infty \end{cases}$$

among all isotonic functions. The *regression error* is the value of this expression, and $g(v)$ is the *regression value* of v.

Isotonic regression is an important form of non-parametric regression, and has been used in numerous settings. There is an extensive literature on statistical uses of isotonic regression going back to the 1950's [2, 3, 14, 32, 41]. It has also been applied to optimization [4, 20, 23] and classification [8, 9, 39] problems. Some recent applications involve very large data sets, including learning [19, 26, 29], analyzing data from microarrays [1] and general data mining [42]. While the earliest algorithms appeared in the statistical literature, most of the advances in exact algorithms have appeared in discrete mathematics, operations research, and computer science forums [1, 20, 23, 36, 39], using techniques such as network flows, dynamic programming, and parametric search.

We consider the case where V consists of points in d-dimensional space, $d \ge 2$, where point $p = (p_1, \ldots, p_2)$ precedes point $q = (q_1, \ldots, q_d)$ iff $p_i \le q_i$ for all $1 \le i \le d$. q is said to dominate p, and the ordering is known as domination ordering, matrix ordering, or dimensional ordering. Here it will be called *multidimensional ordering*. While "multivariate isotonic regression" might seem appropriate, this term already has a different definition [35]. Each dimension need only be a linearly ordered set. For example, the independent variables may be white blood cell count, age, and tumor severity classification I < II < III, where the dependent variable is probability of 5-year survival. Reversing the ordering on age and on tumor classification, for elderly patients the isotonic assumption is that if any two of the independent variables are held constant then an increase in the third will not decrease the survival probability. There is no assumption about what happens if some increase and some decrease.

Figure 1 is an example of isotonic regression on a set of points in arbitrary position. There are regions where it is undefined and regions on which the regression is constant. These constant regions are *level sets*, and for each its value is the L_p mean of the weighted values of its points. For L_2 this is the weighted average, for L_1 it is a weighted median, and for L_∞ is the weighted average of maximal violators, discussed in Sect. 5. Given data values f, a pair of vertices u and v is a *violating pair* iff $u \prec v$ and $f(u) > f(v)$.

Multidimensional isotonic regression has long been studied [16] but researchers have cited the difficulties of computing it as forcing them to use inferior substitutes or to restrict to modest data sets [6, 10, 12, 13, 22, 27, 30, 33, 34, 40]. Faster approximations have been developed but their developers advised: "However, when the program has to deal with four explanatory variables or more, because of the complexity in computing the estimates, the use of an additive isotonic model is proposed" [34]. The additive isotonic model uses a sum of 1-dimensional isotonic regressions, greatly restricting the ability to represent isotonic functions. For example,

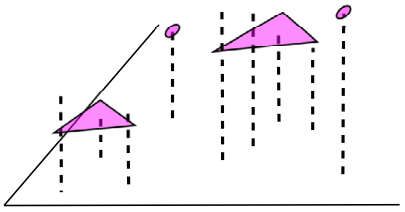


Fig. 1 Level sets for 2-dimensional isotonic regression

a sum of 1-dimensional regressions cannot adequately represent the simple function $z = x \cdot y$.

We present a systematic approach to the problem of finding isotonic regressions for data at points P at arbitrary locations in multidimensional space. P is embedded into a dag $G = (P \cup R, E)$ where $|R|, |E| = \tilde{\Theta}(|P|)$. (The "soft theta" notation, $\tilde{\Theta}$, omits poly-logarithmic factors, but the theorems give them explicitly as a function of the dimension. It is sometimes read as "approximately theta".) For any $u, v \in P$, $u \prec v$ in multidimensional ordering iff there is an $r \in R$ such that $(u, r), (r, v) \in E$. Further, r is unique. This forms a "rendezvous graph", described in Sect. 3.1, and a "condensed rendezvous graph" where one dimension has a simpler structure. In Sect. 4 we show that an isotonic regression of a set of n points in d-dimensional space can be determined in $\tilde{\Theta}(n^2)$ time for the L_1 metric, $\tilde{\Theta}(n^3)$ for the L_2 metric, and $\tilde{\Theta}(n^{1.5})$ for the L_1 metric with unweighted data. L_∞ isotonic regression is not unique, and in Sect. 5 various regression are examined, exhibiting a range of behaviors. This includes the pointwise minimum, pointwise maximum, and strict L_∞ isotonic regression, i.e., the limit, as $p \to \infty$, of L_p isotonic regression. All of these L_∞ regressions can be determined in $\tilde{\Theta}(n)$ time.

For $d \geq 3$ all of our algorithms are a factor of $\tilde{\Theta}(n)$ faster than previous results, and for L_∞ they are also faster when $d = 2$. Some of the algorithms use the rendezvous vertices to perform operations involving the transitive closure in $\tilde{\Theta}(n)$ time, instead of the $\Theta(n^2)$ that would occur if the transitive closure was used directly. For example, the algorithm for strict L_∞ regression for points in arbitrary position is $\tilde{\Theta}(n)$ faster than the best previous result for points which formed a grid, despite the fact that the latter only requires $\Theta(n)$ edges.

Section 6 has some final comments. The Appendix contains a proof of optimality of the condensed rendezvous graph for 2-dimensional points.

2 Background

The complexity of determining the optimal isotonic regression is dependent on the regression metric and the partially ordered set. For example, for a linear order it is well-known that by using "pool adjacent violators" [2] one can determine the L_2 isotonic regression in $\Theta(n)$ time, and the L_1 and L_∞ isotonic regressions in $\Theta(n \log n)$ time. For an arbitrary dag with n vertices and m edges, for L_1 Angelov, Harb, Kannan, and Wang [1] gave an algorithm taking $\Theta(nm + n^2 \log n)$ time, and for L_∞ the fastest takes $\Theta(m \log n)$ time [38]. For L_∞ with unweighted data it has long been known that it can be determined in $\Theta(m)$ time. For L_2, the algorithm of Maxwell

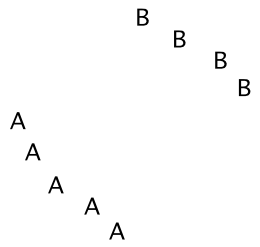


Fig. 2 Every point $a \in A$ is dominated by every point $b \in B$

and Muckstadt [23], with the modest correction by Spouge, Wan, and Wilber [36], takes $\Theta(n^4)$ time. This has recently been reduced to $\Theta(n^2m + n^3 \log n)$ [39].

Figure 2 shows that multidimensional ordering on n points can result in $\Theta(n^2)$ edges. Since the running time of the algorithms for arbitrary dags is a function of m as well as n, one way to determine an isotonic regression more quickly is to find an equivalent dag with fewer edges. A natural way to do this is via transitive reduction. The *transitive reduction* of a dag is the dag with the same vertex set but the minimum subset of edges that preserves the $\prec$ ordering. However, in Fig. 2 the transitive reduction is the same as the transitive closure, namely, all edges of the form (a, b), $a \in A$ and $b \in B$.

A less obvious approach to minimizing edges is via embedding. Given dags $G = (V, E)$ and $G' = (V', E')$, an *order-embedding of G into G'* is an injective map $\pi : V \to V'$ such that, for all $u, v \in V$, $u \prec v$ in G iff $\pi(u) \prec \pi(v)$ in G'. In Fig. 2 a single vertex c could have been added in the middle and then the ordering could be represented via edges of the form (a, c) and (c, b), i.e., only n edges.

Given weighted data (f, w) on G and an order-embedding of G into G', an *extension of the data to G'* is a weighted function (f', w') on G' where

$$f'(v), w'(v) = \begin{cases} f(u), w(u) & \text{if } v = \pi(u) \\ y, 0, \ y \text{ arbitrary} & \text{otherwise} \end{cases}$$

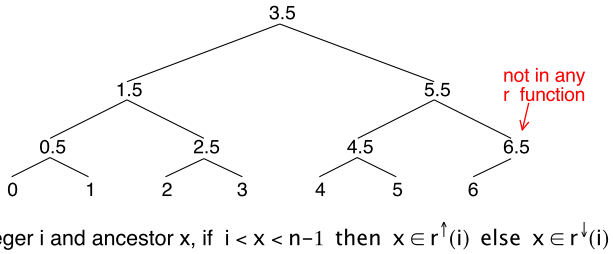
For any L_p metric, given an order-embedding of G into G' and an extension of the data from G to G', an optimal isotonic regression $\hat{f}'$ on G' gives an optimal isotonic regression $\hat{f}$ on G, where $\hat{f}(v) = \hat{f}'(\pi(v))$. For any embedding it will always be that $V \subset V'$ and $\pi(v) = v$ for all $v \in V$, so we omit π from the notation.

3 Multidimensional Ordering

Other than linear orders, the most common orderings for which isotonic regressions are determined are multidimensional orderings. Algorithms for isotonic regressions on multidimensional orderings, without imposing restrictions such as additive models, have concentrated on 2 dimensions [5, 11, 12, 14, 25, 30, 31, 36]. For the L_2 metric and 2-dimensional points, the fastest known algorithms take $\Theta(n^2)$ time if the points are in a grid [36] and $\Theta(n^2 \log n)$ time for points in general position [39], while for L_1 the times are $\Theta(n \log n)$ and $\Theta(n \log^2 n)$, respectively [39]. These algorithms, which are based on dynamic programming, are significantly faster than merely utilizing the best algorithm for arbitrary dags.

There do not appear to be any algorithms giving exact results for dimensions >2 which don't also apply to arbitrary dags. Applying algorithms for arbitrary dags

Fig. 3 Rendezvous tree, $n = 7$



to points in general position results in $\Theta(n^3)$, $\Theta(n^4)$, and $\Theta(n^2)$ time for L_1, L_2, and L_∞, respectively. Here algorithms will be given which reduce the time by a factor of $\tilde{\Theta}(n)$, based on an order embedding into a sparse graph described below.

3.1 Rendezvous Dags

Ordering on a set V of points will be represented by an order-embedding into dag $G = (V \cup R, E)$ where $|R|, |E| = \tilde{\Theta}(|V|)$. G is bipartite, with all edges having one endpoint in V and one in R. For $u, v \in V$, $u \prec v$ in multidimensional ordering iff there is an $x \in R$ such that $(u, x), (x, v) \in E$. Further, x is unique.

Since multidimensional ordering and isotonic regression only depend on relative coordinates, for each dimension i, $1 \le i \le d$, if the points have n_i different coordinates in that dimension then we assume they are $0, \ldots, n_i - 1$. We call these *normalized coordinates*. Points can be converted to normalized coordinates in $\Theta(dn \log n)$ time, and in the same time can be sorted into lexical order. Lexical ordering respects multidimensional ordering. From now on we assume that the points are in normalized coordinates and in order.

Given an integer $n \ge 1$, let $\ell = \lceil \log_2 n \rceil$ and let $j_\ell \cdots j_2 j_1$ be the binary representation of $j \in \{0 \ldots n-1\}$. Define sets of real values $r^\uparrow(j)$ and $r^\downarrow(j)$ in $\{\frac{1}{2}, \frac{3}{2} \ldots, n - \frac{3}{2}\}$ as follows:

$$r^\uparrow(j) = \left\{ j_\ell \cdots j_{k+1} 0111 + \frac{1}{2} : 1 \le k \le \ell,\ j_k = 0,\ j_\ell \cdots j_{k+1} 0111 + \frac{1}{2} < n - 1 \right\}$$

$$r^\downarrow(j) = \left\{ j_\ell \cdots j_{k+1} 0111 + \frac{1}{2} : 1 \le k \le \ell,\ j_k = 1 \right\}$$

E.g., if $j = 11010$ then $r^\uparrow(j) = \{11010.1, 11011.1\}$ and $r^\downarrow(j) = \{11001.1, 10111.1, 01111.1\}$. Figure 3 shows that $r^\uparrow$ and $r^\downarrow$ have a simple interpretation: for integers $a < b$ their least common ancestor s is the unique point in $r^\uparrow(a) \cap r^\downarrow(b)$. We call s the *rendezvous* point of a and b. Technically $r^\uparrow$ and $r^\downarrow$ should include n as a parameter, but since it will always be clear from the context it is omitted.

Rendezvous points provide an order embedding of the linear ordering on $V = \{0, \ldots, k-1\}$ into the bipartite dag $G = (V \cup R, E)$ where

$$R = \left\{ \frac{1}{2}, \ldots, k - \frac{3}{2} \right\}$$

$$E = \bigcup_{i=0}^{k-1} \left(\left\{ (i, x) : x \in r^\uparrow(i) \right\} \cup \left\{ (y, i) : y \in r^\downarrow(i) \right\} \right)$$

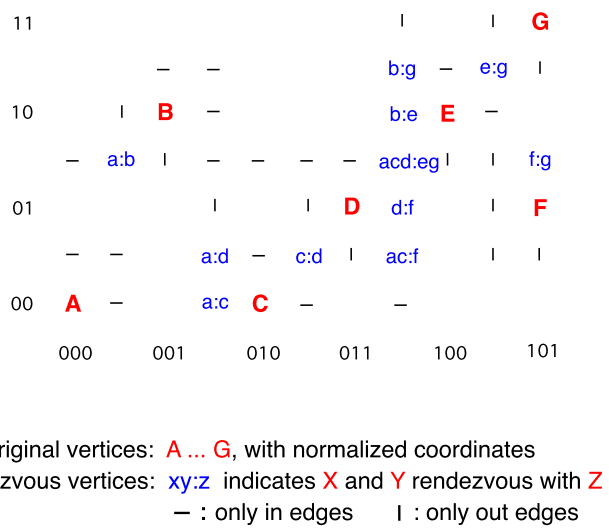


Fig. 4 Rendezvous vertices in 2 dimensions

For $i, j \in V$, $i \prec j$ iff there is an $s \in R$ such that $(i, s), (s, j) \in E$, and if $i \not\prec j$ then there is no path from i to j in G. Thus it is an order embedding. The embedding is not very efficient, in that $|E| = \Theta(k \log k)$, but it naturally extends to higher dimensions.

Let V be a set of n points in d-dimensional space. For a point $x = (x_1, \ldots, x_d) \in V$ let $R^{\uparrow}(x)$, $R^{\downarrow}(x)$ be the sets of d-dimensional points:

$$R^{\uparrow}(x) = \left\{(y_1, \ldots, y_d) : y_i \in r^{\uparrow}(x_i) \cup \{x_i\}, i = 1..d\right\} \setminus \{x\}$$
$$R^{\downarrow}(x) = \left\{(y_1, \ldots, y_d) : y_i \in r^{\downarrow}(x_i) \cup \{x_i\}, i = 1..d\right\} \setminus \{x\}$$

Since different points in $d \geq 2$ dimensional space may have the same coordinates in some dimensions, x_i is added to $r^{\uparrow}(x_i)$ and $r^{\downarrow}(x_i)$. However, using this at all dimensions would give x, so it is removed from $R^{\uparrow}(x)$ and $R^{\downarrow}(x)$. Just as for the linear order this is an order embedding, and for $u, v \in V$, $u \prec v$ in multidimensional ordering iff there is a unique $s \in R^{\uparrow}(u) \cap R^{\downarrow}(v)$. Figure 4 illustrates rendezvous points, many of which could be pruned if desired. E.g., the point (010.1, 01.1) has only an incoming edge from C, while (000.1, 10) has only an outgoing edge to B.

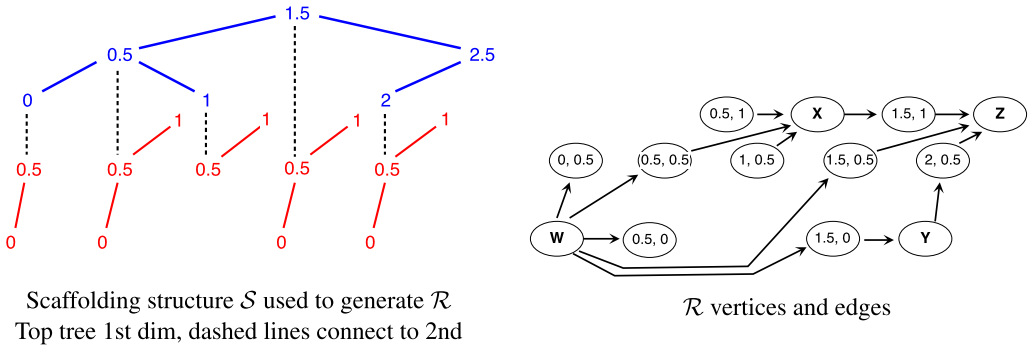
The *rendezvous dag of* V, $\mathcal{R}(V)$, is $(V \cup R, E)$, where

$$R = \bigcup_{v \in V} \left(R^{\uparrow}(v) \cup R^{\downarrow}(v)\right)$$
$$E = \bigcup_{v \in V} \left(\left\{(v, x) : x \in R^{\uparrow}(v)\right\} \cup \left\{(x, v) : x \in R^{\downarrow}(v)\right\}\right)$$

Since $|R^{\uparrow}(v)|, |R^{\downarrow}(v)| = \Theta(\log^d n)$, $|R|, |E| = \Theta(n \log^d n)$. However, there are $\Theta(n^d)$ potential rendezvous points, so to efficiently construct $\mathcal{R}(V)$ one needs to generate only the relevant ones.

Proposition 1 *Given a set V of n points in d-dimensional space, $d \geq 1$, $\mathcal{R}(V)$ can be constructed in $\Theta(n \log^d n)$ time, where the implied constants depend upon d.*

Proof To construct $\mathcal{R}(V)$, the fundamental difficulty is that to add an edge (v, r) for some $v \in V$ and rendezvous vertex r there is no canonical list of the rendezvous ver-



Scaffolding structure $\mathcal{S}$ used to generate $\mathcal{R}$
Top tree 1st dim, dashed lines connect to 2nd

$\mathcal{R}$ vertices and edges

Fig. 5 2-d Rendezvous graph for $W = (0, 0)$, $X = (1, 1)$, $Y = (2, 0)$, $Z = (2, 1)$

tices. One could use a d-dimensional array of all possibilities, but that could be of size $\Theta(n^d)$. This could be improved through hashing, but that would only give an algorithm with expected time guarantees. Instead, a simple multidimensional structure, the *scaffolding*, $\mathcal{S}$, is used to help create $\mathcal{R}(V)$. The top level of $\mathcal{S}$ is the rendezvous tree corresponding to dimension 1, where the nodes point to rendezvous trees corresponding to dimension 2, etc., until dimension d is reached. Each node of the rendezvous trees in the d^{th} dimension points to a vertex of $V \cup R$, where each vertex has linked lists for incoming and outgoing edges. At each level the trees are as in Fig. 3. If all trees at all levels were complete then the base would correspond to the d-dimensional array mentioned above. The scaffolding is the part of this tree needed to reach the potential rendezvous vertices that are actually used. The scaffolding is used to index vertices but is not part of $\mathcal{R}(V)$ and can be deleted once $\mathcal{R}(V)$ has been created.

Initially $\mathcal{S}$ is empty. Whenever vertices are added to a tree at some dimension, the entire path to that vertex is created (an initial segment, and perhaps the entire path, may already exist). I.e., we are not building a binary search tree by the usual insertion process for maintaining a balanced binary search tree, but rather building the rendezvous tree, with predetermined shape, for that dimension. For $v \in V$, first v is inserted, then $R^{\uparrow}(v)$ and $R^{\downarrow}(v)$. The vertices in $R^{\uparrow}(v)$, and similarly $R^{\downarrow}(v)$, are added in a depth-first order, where the first dimension recursively calls the second dimension, etc. At the d^{th} dimension all of v's rendezvous points lie along a single path, and hence it takes $O(\log n)$ time to create them (if they haven't yet been created by some other vertex) and add the edges. Since $O(\log^{d-1} n)$ trees are reached in the d^{th} dimension, the total time is as claimed. Note that if each of v's edges to rendezvous points had been inserted one at a time by starting at the top then the time would have increased by a logarithmic factor. □

Figure 5 is an example of the scaffolding that would be constructed for the points $W = (0, 0)$, $X = (1, 1)$, $Y = (2, 0)$, $Z = (2, 1)$ in $[0..2] \times [0..1]$. The top tree corresponds to the first dimension, and at each of its vertices the dotted vertical lines indicate the tree corresponding the second dimension. Each vertex of the lower tree creates a vertex of $\mathcal{R}$. The vertices of $\mathcal{R}$ are a subset of $[0, 0.5, 1, 1.5, 2, 2.5] \times [0, 0.5, 1]$.

One can reduce the size of the dag and the time to construct it by observing that the construction is inefficient for 1-dimensional space. E.g., the rendezvous dag

for $\{0, 1, \ldots, 7\}$ has edges from $\{0, 1, 2, 3\}$ to rendezvous vertex 3.5, from 3.5 to $\{4, 5, 6, 7\}$, from $\{0, 1\}$ to 1.5, etc., for a total of 24 edges. A more efficient representation would just have edges from 0 to 1, 1 to 2, etc. We use this linearization of one dimension in a condensed version of the rendezvous graph.

For a point $x \in V$, where $x = (x_1, \ldots, x_d)$, let $R_c^{\uparrow}(x)$ and $R_c^{\downarrow}(x)$ be sets of d-dimensional points where

$$R_c^{\uparrow}(x) = \left\{(x_1, y_2, \ldots, y_d) : y_i \in r^{\uparrow}(x_i) \cup \{x_i\},\ i = 2..d\right\} \setminus \{x\}$$
$$R_c^{\downarrow}(x) = \left\{(x_1, y_2, \ldots, y_d) : y_i \in r^{\downarrow}(x_i) \cup \{x_i\},\ i = 2..d\right\} \setminus \{x\}$$

The *condensed rendezvous dag of* V, $\mathcal{R}_c(V)$, is $(V \cup R_c, E_c)$, where

$$R_c = \bigcup_{v \in V} \left(R_c^{\uparrow}(v) \cup R_c^{\downarrow}(v)\right)$$
$$E_c = E' \cup \bigcup_{v \in V} \left(\left\{(v, x) : x \in R_c^{\uparrow}(v)\right\} \cup \left\{(x, v) : x \in R_c^{\downarrow}(v)\right\}\right)$$

and E' is edges of the form (p, q) where $p = (p_1, \ldots, p_d)$, $q = (q_1, \ldots, q_d)$, $p_i = q_i$ for $2 \le i \le d$, $p_1 < q_1$, and there is no $r = (r_1, p_2, \ldots, p_d)$ in $V \cup R_c$ such that $p_1 < r_1 < q_1$. That is, for all points agreeing on their last $d - 1$ coordinates, E' puts them in linear order by their first coordinate. For two such points u and v, with $u \prec v$, it is no longer true that there is a rendezvous vertex x such that (u, x) and (x, v) are edges in $\mathcal{R}_c(V)$.

Clearly V is order embedded in $\mathcal{R}_c(V)$ and $|E_c|, |R_c| = \Theta(n \log^{d-1} n)$. Figure 6 shows the condensed version of the dag in Fig. 4.

Proposition 2 *Given a set V of n points in d-dimensional space with multidimensional ordering, $d \ge 2$, in $\Theta(n \log^{d-1} n)$ time one can construct $\mathcal{R}_c(V)$, where the implied constants depend upon d.*

Proof A $(d - 1)$-dimensional scaffolding structure $\mathcal{S}_c$ will be constructed, similar to the scaffolding used in $\mathcal{R}(V)$. $\mathcal{S}_c$ corresponds to dimensions $2 \ldots d$. Instead of pointing to a vertex, a node in the d^{th} dimension points to a linked list of all vertices with the same final $d - 1$ coordinates. Points in V are processed in lexical order. When $x = (x_1, \ldots, x_d) \in V$ is processed, as before it is inserted and then $R_c^{\uparrow}(x)$ and $R_c^{\downarrow}(x)$ are inserted using depth-first recursion. For each $y \in R_c^{\uparrow}(x)$ a vertex will be added to the tail of the list at the node of $\mathcal{S}_c$ representing y, with an edge from x to this new vertex, and for $y \in R_c^{\downarrow}(x)$ the new tail vertex has an edge to x. □

Even in $\mathcal{R}_c$ many of the rendezvous vertices may be unnecessary, raising the question of whether the results in Proposition 2 are optimal in terms of the number of edges required. In the Appendix it is shown that they are optimal for points in 2 dimensions, but the optimality for $d \ge 3$ is unknown.

$\mathcal{R}$ and $\mathcal{R}_c$ have similarities with the "Manhattan networks" in Gudmundsson et al. [15]. These undirected graphs have their edges parallel to the axes, as does $\mathcal{R}_c$ for 2-dimensional points. For 2-dimensional points they give networks with a diameter vs. size tradeoff, with the smallest diameter being 6 using $\Theta(n \log^2 n)$ nodes

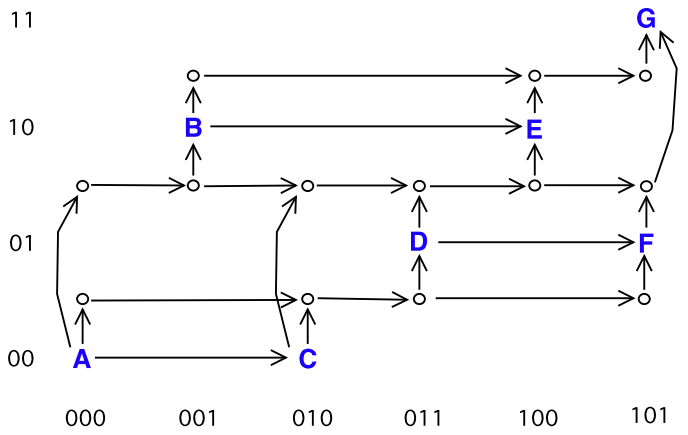


Fig. 6 Condensed rendezvous dag corresponding to Fig. 4

and edges, i.e., the same size (in O-notation) as $\mathcal{R}$. (The diameter is the maximum length, over all pairs of points, of the shortest path between them.) For d-dimensional points the diameter is $\Omega(d)$ since each edge traverses a single dimension. Multidimensional divide-and-conquer is used to create the networks. In contrast, $\mathcal{R}$ treats all dimensions symmetrically (or all but the first, for $\mathcal{R}_c$) and, independent of dimension, when $u \prec v$ there is a path of length 2 from u to v.

4 L_1 and L_2 Isotonic Regression

Recall that for $d = 2$ there are dynamic programming algorithms which take $\Theta(n \log^2 n)$ and $\Theta(n^2 \log n)$ time for L_1 and L_2, respectively [39], and $\Theta(n \log n)$ [39] and $\Theta(n^2)$ [36] time if they form a grid. When $d \geq 3$, for a set of n d-dimensional points, applying the algorithms for arbitrary dags results in $\Theta(n^3)$ time for L_1 [1], $\Theta(n^4)$ [23] for L_2 and $\Theta(n^{2.5} \log n)$ for L_1 on unweighted data [39]. If the points are in a grid the times are $\Theta(n^2 \log n)$, $\Theta(n^3 \log n)$, and $\Theta(n^{2.5} \log n)$, respectively. By using rendezvous graphs one can significantly reduce the times for points in general position, and for L_1 with unweighted data can also reduce the time for points in a grid. Theorem 3 was also noted in [39] (citing this paper). $\mathcal{R}_c$ is used in a straightforward manner in Theorem 3, while the structure of $\mathcal{R}$ plays an important role in Theorem 4.

Theorem 3 *Given a set V of n points in d-dimensional space with multidimensional ordering, $d \geq 3$, and given weighted data (f, w) on V, an isotonic regression of (f, w) can be found in*

- $\Theta(n^2 \log^d n)$ *time for the L_1 metric, and*
- $\Theta(n^3 \log^d n)$ *time for the L_2 metric,*

where the implied constants depend upon d.

Proof If N and M represent the number of vertices and edges in $\mathcal{R}_c(V)$, then the result for L_1 comes from using the algorithm of Angelov et al. [1]. It has a number of stages, each taking $\Theta(M + N \log N)$ time. The number of stages is linear in n, not

N, since the flow problem being solved at each stage is only needed for vertices with nonzero weights. This yields the time claimed for L_1.

For L_2, the partitioning approach in [39] solves a binary L_1 isotonic regression problem at each stage, using the algorithm just described. The number of stages of the L_2 algorithm is linear in the number of vertices, and here too only the vertices of nonzero weight are relevant. □

When the data is unweighted a different approach can be used for L_1. The algorithm uses stages of isotonic regressions on binary functions. Let h be a $\{0, 1\}$-valued function on V. Since the regression values for L_1 isotonic regression can always be chosen to be data values, an L_1 regression of h can itself be $\{0, 1\}$-valued. The *violation graph* is (V, E'), where there is an edge $(u, v) \in E'$ iff $u \prec v$ and $h(u) > h(v)$, i.e., iff they are a violating pair. The violation graph can be constructed by pairwise comparisons in $\Theta(n^2)$ time, and might have $\Theta(n^2)$ edges. Using this the approach below would take $\Theta(n^2 \log n)$ time. However, there are fewer rendezvous vertices, and they allow one to perform operations involving violating pairs more quickly than can be done on the violation graph itself.

Theorem 4 *Given a set V of n points in d-dimensional space with multidimensional ordering, $d \geq 3$, and given unweighted data f on V, an L_1 isotonic regression can be found in $\Theta(n^{1.5} \log^{d+1} n)$ time, where the implied constants depend upon d.*

Proof A sequence of partitioning stages will be used. For data values $a < b$, where no data values are in (a, b), let f' be the function which is a at $v \in V$ if $f(v) \leq a$, and b otherwise. Let g be an $\{a, b\}$-valued L_1 isotonic regression of f' and let $V_a = \{v \in V : g(v) = a\}$ and $V_b = \{v \in V : g(v) = b\}$. In [39] it is proven that an isotonic regression of f on V can be formed by an isotonic regression of f on V_a where all regression values are data values $\leq a$, and an isotonic regression of f on V_b where all regression values are data values $\geq b$. By choosing a, b so that at least one is a median data value, after $\Theta(\log n)$ partitioning stages an isotonic regression has been determined. Lemma 6 shows that the binary isotonic regression at each stage can be completed in $\Theta(n^{1.5} \log^d n)$ time. □

The fact that an L_1 isotonic regression of a $\{a, b\}$-valued function can be chosen to be $\{a, b\}$-valued greatly simplifies its structure when the data is unweighted.

Lemma 5 *Given a dag $G = (V, E)$ and an unweighted $\{0, 1\}$-valued function h on V, let C be a minimum cardinality vertex cover of the violation graph. Then the function $\hat{h}$ given by*

$$\hat{h}(v) = \begin{cases} h(v) & \text{if } v \in V \setminus C \\ 1 - h(v) & \text{otherwise} \end{cases}$$

is an L_1 isotonic regression of h.

Proof For any isotonic function g on V, the set of vertices where $g \neq h$ must be a vertex cover of the violation graph. Let C be a minimal cardinality vertex cover. The

size of C sets a lower bound on the L_1 regression error, and the L_1 regression error of $\hat{h}$ is $|C|$.

To show that $\hat{h}$ is isotonic we need only show that no new violating pairs were introduced. Let $V_i = \{v \in V : h(v) = i\}$ for $i \in \{0, 1\}$. There are two cases, $u, v \in V_0, u \prec v, u \in C$ but $v \notin C$, or $u, v \in V_1, u \prec v, v \in C$ but $u \notin C$. Their proofs are similar, so assume the former holds. Let $x \prec u$ with $x \in V_1$, i.e., they are a violating pair. Because $u \prec v$, then x, v is also a violating pair. Since we assumed that $v \notin C$, it must be that $x \in C$. However, since this is true for all x for which x, u is violating, then $C \setminus \{u\}$ is also a vertex cover, contradicting the minimality of C. □

Lemma 6 *Given a set V of n points in d-dimensional space with multidimensional ordering, $d \geq 2$, and given an unweighted $\{0, 1\}$-valued function f on V, an L_1 isotonic regression of f, with all regression values in $\{0, 1\}$, can be found in $\Theta(n^{1.5} \log^d n)$ time, where the implied constants depend on d.*

Proof A concise representation of the violation graph can be easily constructed from $\mathcal{R}(V)$. Let $V_i = \{v \in V : f(v) = i\}$, $i \in \{0, 1\}$. Let $\mathcal{R}'$ be $\mathcal{R}(V)$ minus the edges in $R^{\uparrow}(v)$ if $v \in V_0$, and minus the edges in $R^{\downarrow}(v)$ if $v \in V_1$. For $u \prec v \in V$, there is a path of length 2 from u to v in $\mathcal{R}'$ iff they are a violating pair.

Lemma 5 shows that to find an isotonic regression it suffices to find a minimum cardinality vertex cover of the violation graph. It is well known that such a cover can be obtained in linear time from a maximum matching of the violation graph. Since the violation graph is bipartite and unweighted, the Hopcroft-Karp algorithm can be used to find a maximum matching in $\Theta(|E|\sqrt{n})$ time, where E is the set of edges. We use $\mathcal{R}'$ to reduce the number of edges required. Without it, the time would be $\Theta(n^{2.5})$.

The Hopcroft-Karp algorithm has $O(\sqrt{n})$ stages, where at each stage a maximal set of disjoint augmenting paths of minimal length is found. Each stage involves a breadth-first search followed by edge-disjoint depth-first searches. The time of each stage is linear in the number of edges, assuming all isolated vertices have first been removed. It is straightforward to implement each stage using $\mathcal{R}'$ where a path of length ℓ used in the violation graph is represented by a path of length 2ℓ in $\mathcal{R}'$. E.g., there might be vertices $A = \{a_1, \ldots, a_j\} \in V_1$ and $B = \{b_i, \ldots, b_k\} \in V_0$ where all A, B pairs are violators and share the same rendezvous vertex r. However, in the Hopcroft-Karp algorithm one never uses more than $\min\{j, k\}$ edges from A to B, and any subset of edges of this size can be represented by paths through r in $\mathcal{R}'$. It remains true that the time is linear in the number of edges, which is $\Theta(n \log^d n)$. Thus the total time is as claimed. □

5 L_∞ Isotonic Regression

Isotonic regression using the L_∞ metric is not unique. For example, with unweighted data 5, −5, 4 on $\{1, 2, 3\}$, an L_∞ isotonic regression must be 0 at 1 and 2, with a regression error of 5. The regression value at 3 can be anything in the range $[0, 9]$ and maintain the isotonic property without increasing the error. This flexibility can be both a blessing and a curse. In general the algorithms are much faster than those for L_1 and L_2, but the regressions produced may have some undesirable properties.

We slightly extend the definition of violating pairs to *weakly violating pairs*, i.e., pairs of vertices u, v where $u \preceq v$ and $f(u) \geq f(v)$. Note that a single vertex is a weakly violating pair. For vertices u, v let $\mathsf{wmean}(f, w : u, v)$ be the weighted average $(w(u)f(u) + w(v)f(v))/(w(u) + w(v))$. Using this as their regression value has equal error at both vertices, denoted $\mathsf{mean\_err}(f, w : u, v)$. Using $\mathsf{wmean}(f, w : u, v)$ as the regression value at u and v minimizes their L_∞ regression error if they are violating since if the regression value at u has smaller error then it must be greater than $\mathsf{wmean}(f, w : u, v)$. The isotonic constraint forces the regression value at v to be at least as large as that at u, which increases the error at v. Similarly, the error at v cannot be reduced. More generally, for a level set $L \subset V$, its L_∞ mean is $\mathsf{wmean}(f, w : u', v')$, where u', v' maximize $\mathsf{mean\_err}(f, w : u, v)$ among all $u, v \in L$.

For dag G and data (f, w), let Max denote the pointwise maximum of all L_∞ isotonic regressions. It is straightforward to show that Max is an L_∞ isotonic regression, as is Min, which is the pointwise minimum. The fastest algorithm known for L_∞ isotonic regression on arbitrary dags produces Max or Min in $\Theta(m \log n)$ time [38]. It is a slight modification of the approach used in [20], based on a feasibility test to determine if there is a regression with error ϵ, using parametric search to narrow down the range of ϵ. Parametric search is impractical, and fortunately is not needed to obtain the results given here. If any L_∞ isotonic regression can be found in time $\Omega(m)$, then Max and Min can be found in the same time by using the regression error of the first in a feasibility test. The feasibility test is essentially based on producing Max (or Min, at the user's discretion).

Since the 1970's the most commonly studied L_∞ isotonic regression is Basic [41]. For data (f, w) on dag G, $\mathsf{Basic}(f, w)$ at vertex x is $\mathsf{wmean}(f, w : u', v')$, where u', v' maximize $\mathsf{mean\_err}(f, w : u, v)$ among all weakly violating pairs u, v such that $u \preceq x \preceq v$. A related L_∞ isotonic regression algorithm is Prefix [38]. Let

$$\mathsf{pre}(v) = \max\{\mathsf{wmean}(f, w : u, v) : u \preceq v\}$$
$$\mathsf{Prefix}(f, w)(u) = \min\{\mathsf{pre}(v) : u \preceq v\}$$

For unweighted data, Basic at v simplifies to $\big(\max\{f(u) : u \preceq v\} + \min\{f(u) : u \succeq v\}\big)/2$, which can be computed in $\Theta(m)$ time via topological sort. Prefix can similarly be computed in the same time. For weighted data, for arbitrary dags the fastest known approach for computing either is to first determine the transitive closure. Given the transitive closure, Prefix can easily be determined in $\Theta(n^2)$ time since $\mathsf{pre}(v)$ only involves predecessors of v. The combination of predecessors and successors used in Basic makes it more complicated, though it too can be determined in $\Theta(n^2)$ time by using an approach described in [24].

The *strict* L_∞ *isotonic regression*, Strict, is the limit, as $p \to \infty$, of L_p isotonic regression. It has been called the "best best" L_∞ isotonic regression [21], and the limit process is known as the "Polya algorithm" [28]. For arbitrary dags, given the transitive closure Strict can be determined in $\Theta(n^2 \log n)$ time [37].

We call the above L_∞ *isotonic regression mappings* since they map a dag G and weighted data (f, w) to an isotonic function on G. These mappings differ significantly in their mathematical properties. For example, a mapping M is *monotonic* iff for every dag G and weight function w, if pointwise $f_1 \leq f_2$, then pointwise

$\mathsf{M}(f_1, w) \leq \mathsf{M}(f_2, w)$. Prefix and Basic are monotonic, as can be easily seen from their construction, and Strict inherits monotonicity since all L_p isotonic regressions are monotonic for $1 < p < \infty$. However, Min is not monotonic as can be seen by considering the unweighted functions $(0, 0, 0)$ and $(0, 2, 0)$, for which Min is $(0, 0, 0)$ and $(-1, 1, 1)$, respectively. Similarly, Max is not monotonic. Also, Prefix, Basic and Strict always have regression values that are within the range of data values, while the above example shows that this need not be true for Min, and similarly for Max. In some applications this would not be acceptable.

Another aspect concerns large regression errors. Strict minimizes the number of large errors [37], in that, for any dag G and data (f, w), if $g \neq \mathsf{Strict}(f, w)$ is an isotonic function on G, then there is a $C > 0$ such that g has more vertices with regression error $\geq C$ than does $\mathsf{Strict}(f, w)$, and for any $D > C$, g and $\mathsf{Strict}(f, w)$ have the same number of vertices with regression error $\geq D$ (there might be a $d < C$ where g has fewer vertices with regression error $\geq d$ than does Strict, but the emphasis is on large errors). For example, for the function with values 3, 1, 2.5 and weights 2, 2, 1, Strict is 2, 2, 2.5, as are all L_p isotonic regressions for $1 < p < \infty$, but all of the other L_∞ regression mappings considered here have a nonzero error for the third value. Prefix and Basic have a weaker property that is straightforward to prove [38]: let C be the regression error of L_∞ isotonic regression. If Prefix has regression error C at vertex v then all L_∞ regressions have the same value at v. The same is true of Basic. Min and Max have far different behavior, in that at any v, one or both of them has the largest regression error at v among all L_∞ isotonic regressions.

The best previous results are [20, 37, 38]:

- $d = 1$: $\Theta(n \log n)$ for all of the regressions mentioned above, and $\Theta(n)$ for Prefix and Basic if the data is unweighted;
- $d \geq 2$, points form a grid: Min, Max: $\Theta(n \log n)$; Prefix, Basic: $\Theta(n^2)$; Strict: $\Theta(n^2 \log n)$; and $\Theta(n)$ for Prefix and Basic if the data is unweighted;
- $d \geq 2$, points in general position: Prefix, Basic: $\Theta(n^2)$; Stict, Min, Max: $\Theta(n^2 \log n)$;

where the implied constants depend upon d. The results for $d \geq 2$ are based on using the best algorithm for arbitrary dags, though, as noted above, the times for Min and Max can be reduced to that of Prefix. Note that for weighted data and $d \geq 2$, for Prefix, Basic and Strict there is no difference between points in a grid and points in arbitrary position, and for points in general position restricting to unweighted data does not speed up Prefix nor Basic. Also, unlike L_1 and L_2, the previous results for $d = 2$ are no better than those for $d \geq 3$.

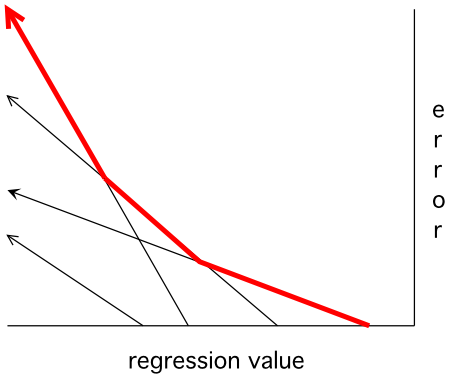
For points in general position, using rendezvous graphs we will show:

Theorem 7 *Given a set V of n points in d-dimensional space with multidimensional ordering, $d \geq 2$, and given weighted data (f, w) on V, an L_∞ isotonic regression can be found in*

(a) $\Theta(n \log^{d-1} n)$ *time for* Basic *and* Prefix (*and hence* Min *and* Max) *if the data is unweighted*,
(b) $\Theta(n \log^{d} n)$ *time for* Prefix (*and hence* Max *and* Min),
(c) $\Theta(n \log^{d+1} n)$ *time for* Strict,

where the implied constants depend upon d.

Fig. 7 Error envelope



The proof will be in parts. Part (a) comes from creating $\mathcal{R}_c(V)$ and then using the fastest algorithm for arbitrary dags, (b) is proven in Sect. 5.1, and (c) is proven in Sect. 5.2. Basic can also be determined in $\Theta(n \log^d n)$ time, but the algorithm is more complicated than that for Prefix and will be omitted.

5.1 Prefix L_∞ Isotonic Regression

There is a geometric approach to computing the pre function used in Prefix. On a graph of regression error as a function of regression value, the ray with endpoint $(f(v), 0)$ and slope $w(v)$ gives the regression error at v of using a regression value $\geq f(v)$. For any $u \prec v$, the ray with endpoint $(f(u), 0)$ and slope $-w(u)$ gives the regression error at u of using a regression value $\leq f(u)$. If $u \preceq v$ are a weakly violating pair then the intersection of these rays gives their weighted mean and corresponding error. The set of rays corresponding to all of v's predecessors is as in Fig. 7. The line segments corresponding to the maximum error at each regression value is an upper envelope, which will be called a decreasing *error envelope*. The intersection of the increasing ray corresponding to v with the decreasing error envelope corresponding to its predecessors gives the value of $\mathsf{pre}(v)$ and the regression error at v of using this value. By storing the error envelope in a balanced tree ordered by slope, it is easy to insert a ray, and determine the intersection of a ray with the envelope, in $O(\log t)$ time per operation, where t is the number of segments in the envelope.

Proof of Theorem 7(b) The time is dominated by the time to compute pre since once it has been computed a reverse topological sort of $\mathcal{R}_c(V)$ yields Prefix. The motivation behind $\mathcal{R}_c(V)$ will be used, but an order embedding will not be constructed. Instead, a sweep is made through the first dimension, keeping a $(d-1)$-dimensional scaffolding T corresponding to the union of all rendezvous vertices encountered so far, merging vertices with the same final $d-1$ coordinates. In $\mathcal{R}_c(V)$ a leaf of T pointed to a linked list of all vertices with the corresponding final $d-1$ coordinates, but now it has an error envelope corresponding to all points which have an incoming edge to the rendezvous vertices with these coordinates. When the sweep encounters a new vertex $v \in V$, first $\mathsf{pre}(v)$ is computed by computing error envelope intersections at vertices corresponding to $R_c^{\downarrow}(v)$. There are only $\Theta(\log^{d-1} n)$ of these and thus the time to compute $\mathsf{pre}(v)$ is $\Theta(\log^d n)$. Once $\mathsf{pre}(v)$ has been computed, then the ray

corresponding to v is inserted into the error envelopes corresponding to $R_c^{\uparrow}(v)$. This too can be performed in $\Theta(\log^d n)$ total time.

The vertices are processed in topological order, and thus if u is a predecessor of v then its information has been added to the error envelopes before $\mathsf{pre}(v)$ is computed. Further, any envelope encountered in the calculation of $\mathsf{pre}(v)$ will only contain rays corresponding to predecessors of v. □

5.2 Strict L_∞ Isotonic Regression

Algorithm 1 computes Strict. A proof of correctness, and the fact that there are only n iterations of the loop, appears in [37]. The algorithm in [37], applicable to all dags, first uses the transitive closure to find all violating pairs. Here rendezvous vertices are used to find important violating pairs incrementally. $L(v)$ and $U(v)$ are lower and upper bounds, respectively, on the regression value at v. Violating pairs are examined in decreasing order of mean_err, which is how Strict minimizes the number of vertices with large errors.

As an example, on vertices $v_1 < v_2 < v_3 < v_4 < v_5 < v_6$ suppose the f values are $(4, 3, 2, -3, 3, 0)$ with weights $(1, 4, 1, 4, 1, 1)$. The first violating pair examined is v_2, v_4, with $\mathsf{mean\_err} = 12$ and $\mathsf{wmean} = 0$. At the end of the loop, L is $(-\infty, 0, 0, 0, 0, 0)$, U is $(0, 0, 0, 0, \infty, \infty)$, and S is $(-, 0, -, 0, -, -)$. v_2, v_6 were a violating pair with $\mathsf{mean\_err} = 2.4$, but v_2 was removed from consideration once its S value was determined, and hence the next violating pair is v_1, v_6, with $\mathsf{mean\_err} = 2$ and $\mathsf{wmean} = 2$. At line 5, $U(v_1) < y$ because the constraint lowering $U(v_1)$ imposed by the earlier violating pair forces greater error at v_1 than v_1 and v_6 impose on each other. $S(v_1)$ is thus determined (and assigned in line 11) because any future constraint on it would have error at most that imposed by v_6. Since $S(v_1)$ is closer to $f(v_6)$ than $f(v_1)$ was, the old constraint is not relevant at v_6 and lines 6–9 are skipped. At the end of the loop S is $(0, 0, -, 0, -, -)$. The next violating pair is v_5, v_6, with $\mathsf{mean\_err} = 1.5$ and $\mathsf{wmean} = 1.5$, resulting in $S = (0, 0, -, 0, 1.5, 1.5)$. The final violating pair is v_3, v_3, with $\mathsf{mean\_err} = 0$, $\mathsf{wmean} = 2$, $U(v_3) = L(v_3) = 0$. $S(v_3)$ is set to 0 at line 6.

Given the transitive closure, Algorithm 1 is quite simple to implement. The violating pairs can be determined and sorted in decreasing mean_err value, so line 3 is merely stepping through this order. Each edge of the transitive closure is used at most once in terms of updating L or U values, and hence the total time of the while loop is linear in the number of edges. Thus the total time is dominated by the initial sorting.

Applying this directly would take $\Theta(n^2 \log n)$ time, but this can be reduced via rendezvous graphs.

Proof of Theorem 7 (c) There are only two important aspects: keeping track of the L and U values, and determining the violating pair maximizing mean_err. The full rendezvous dag, $\mathcal{R}(V)$, will be used.

We keep track of L and U values by storing them at rendezvous vertices. Whenever $L(v)$ is needed (e.g., line 10), the vertices in $R^{\downarrow}(v)$ are examined and the largest L value is used. Whenever L values are updated (e.g., line 8), they are updated at the vertices in $R^{\uparrow}(v)$. Similar calculations are done for $U(v)$.

Algorithm 1: Computing $S = \mathsf{Strict}(f, w)$ on dag $G = (V, E)$

```
1   for all v ∈ V, L(v) = −∞, U(v) = ∞
2   while there is a violating pair
3       let u ≺ v be a violating pair maximizing mean_err(f, w : u, v)
4       y = wmean(f, w : u, v)
5       if y ≤ U(u) then
6           S(v) = max{y, L(v)}
7           for all u' ≺ v, U(u') = min{U(u'), S(v)}
8           for all v' ≻ v, L(v') = max{L(v'), S(v)}
9           remove v from violating pairs
10      if y ≥ L(v) then
11          S(u) = min{y, U(u)}
12          for all u' ≺ u, U(u') = min{U(u'), S(u)}
13          for all v' ≻ u, L(v') = max{L(v'), S(u)}
14          remove u from violating pairs
15  end while
```

We keep track of the globally worst violating pairs by using a priority queue based on the rendezvous points. For each rendezvous point its priority is the worst error among all pairs rendezvousing at that point. Thus line 2 is implemented by removing the worst pair at rendezvous point at the head of the queue. For the next iteration of the loop, to efficiently find new maximal violating pairs at line 2 note that only rendezvous points involving u or v might have a new worst violating pair. Whenever a rendezvous point's priority changes its position in the queue may change, moving it towards the back. There are at most $\Theta(\log^d n)$ rendezvous points involving u or v, and the time to change their positions in the priority queue is $O(\log n)$ per point, so the total time for maintaining the priority queue over all iterations of the while-loop is $\Theta(n \log^{d+1} n)$.

The only remaining step is to determine maximal violators at rendezvous vertices. Initially each rendezvous vertex r has an decreasing error envelope of rays corresponding to vertices $v \in V$ for which $r \in R^{\uparrow}(v)$ and an increasing error envelope corresponding to vertices $u \in V$ for which $r \in R^{\downarrow}(u)$. The intersection of these envelopes corresponds to the maximal violating pair which rendezvous at r. When a vertex v is being removed from all violating pairs (e.g., line 9), its ray must be removed from the envelopes at $R^{\uparrow}(v)$ and $R^{\downarrow}(v)$, and at each the (potentially) new maximal violators are determined. Deleting rays from an envelope is more complex than inserting them since rays that did not have a segment on the envelope may suddenly be uncovered and have one. However, the "repeated violators" lemma in [38] shows that at a rendezvous vertex with a total of k incoming and outgoing edges, the total time to do all operations is $\Theta(k \log k)$. Since there are at most $\Theta(n \log^d n)$ edges, this completes the proof of the theorem. □

The repeated violators lemma mentioned above is based on the semi-dynamic algorithm of Hershberger and Suri [17], which shows that for a collection of k rays, after a $\Theta(k \log k)$ setup phase each deletion takes $O(\log k)$ time. "Semi-dynamic"

refers to the fact that only deletions occur after the setup. There is as of yet no algorithm guaranteeing $O(\log k)$ time per operation for both insertions and deletions.

6 Final Comments

Isotonic regression is becoming increasingly important as researchers analyze large, complex data sets and minimize the assumptions they make [1, 29, 42]. However, there has been dissatisfaction with the slowness of algorithms for multidimensional isotonic regression, and researchers have been forced to use approximations and inferior models [7, 10, 12, 30, 34]. As recently as 2011, Saarela and Arjas [33] said "... the question remains whether one can find computationally feasible generalizations to multiple dimensions". The algorithms developed here are exact and reduce the time by a factor of $\tilde{\Theta}(n)$.

Some of the algorithms merely use the fact that $\mathcal{R}_c$ is not very large, while for others the structure of $\mathcal{R}$ plays a central role. The L_∞ Prefix and Strict algorithms for arbitrary dags utilize the transitive closure [37, 38], and the rendezvous vertices in $\mathcal{R}$ were used to perform operations on the transitive closure more efficiently than can be done on the closure itself. For weighted data and $d \geq 2$ these algorithms are faster for points in general position then the fastest previous results for points in a grid. The same is true for L_1 isotonic regression of unweighted data when $d \geq 3$. For L_1 and L_2 the algorithms for arbitrary dags do not rely on the transitive closure [1, 23, 36, 39] and there had been a factor of $\tilde{\Theta}(n)$ difference in time between grids and points in arbitrary position for $d \geq 3$. Condensed rendezvous graphs narrow the difference to a poly-log factor by reducing the number of edges.

The results are given for points in general position, but when they form a complete grid the constants for $\mathcal{R}$ (and similarly $\mathcal{R}_c$) are significantly better. For dimension $d \geq 1$, one can show that the number of vertices is $< 2^d n$, not $\Theta(n \log^d n)$, and the number of edges is $< n(1 + \log_2 \lceil n^{1/d} \rceil)^d$, which is $\approx d^{-d}$ times the worst case of $n(1 + \lceil \log_2 n \rceil)^d$. Of course, this is only relevant for algorithms which utilize the structure of the rendezvous graph, not merely the fact that it is small, since the grid has $< dn$ edges.

For L_2, one practical aspect is that the worst-case time, $\tilde{\Theta}(n^3)$, requires rather extreme data. Let g be the isotonic regression. At each stage the weighted average C of the data values is determined and the vertices are partitioned into those where $g < C$ and those where $g \geq C$ (the values of g aren't known, but it can be determined where they will be above or below C) [23, 36, 39]. The algorithm is then recursively applied to each piece. It is possible that at each stage the largest element ends up in a partition by itself, resulting in $n - 1$ stages, but in practice the behavior will likely be more like $\tilde{\Theta}(n^2)$. Luss et al. [22] found such a reduction for the fastest previous algorithm, with an observed behavior of $\Theta(n^3)$ time vs. the worst case of $\Theta(n^4)$. Whether there is an algorithm for multidimensional orderings which is $\tilde{\Theta}(n^2)$ in the worst case, and whether there is one which is $\tilde{\Theta}(n^3)$ for arbitrary dags, are open questions.

Isotonic regression is in general not unique for L_1 nor L_∞, so natural questions arise as to the range of the regressions and the properties of the various choices. For

L_∞, Min and Max are the pointwise minimum and maximum, respectively, of all isotonic regressions, and can be determined in $\tilde{\Theta}(n)$ time (Theorem 7). An important aspect of achieving this time is that it is based on using Prefix to determine the optimal regression error and then a simple scan using a topological sort to determine Min an Max (see [38]). For arbitrary dags the fastest algorithm [38] (an improvement of one appearing in [20]) uses the decidedly impractical parametric search to determine the error. For L_1, algorithms to generate the pointwise minimum and maximum seem to be more complex than the ones presented here. For L_∞ the Strict isotonic regression has been called the "best best" regression, and Theorem 7 shows that it can be computed with only $\Theta(\log n)$ slowdown compared to the time for the fastest, Prefix. Strict is the limit, as $p \to \infty$, of L_p isotonic regression, so the natural "strict L_1" isotonic regression is the limit, as $p \to 1$, of L_p isotonic regression. The author knows of no algorithms for computing this, though the question of what the level set values should be was settled by Jackson long ago [18].

Finally, the rendezvous graph itself may be of interest in other settings where a parsimonious representation of multidimensional ordering is useful. The fact that the transitive closure was represented by unique paths of length 2 proved quite useful, so a natural question is to determine which other classes of dags have rendezvous graphs of size not much more than that of the original vertices.

Acknowledgements Research partially supported by NSF grant CDI-1027192, DOE grant DE-FC52-08NA28616 and King Abdulaziz University. The author thanks Günter Rote for pointing out the paper by Gudmundsson et al. [15], and the reviewer for helpful comments.

Appendix: Optimal Size of Order-Embeddings

For points in 2-dimensional space the condensed rendezvous dag has $\Theta(n \log n)$ edges, and here we show that there are sets of points in 2-space for which $\Omega(n \log n)$ edges are needed. One example is the bit-reversal permutation: given integer $b > 0$, let V be the set of 2-dimensional points $\{(i_b i_{b-1} \ldots i_1, i_1 i_2 \ldots i_b) : i_j \in \{0, 1\}\}$, i.e., a point is in V if its b-bit 2nd coordinate, in binary, is the reversal of its 1st coordinate.

Proposition 8 *Let V be the b-bit bit-reversal permutation, ordered by 2-dimensional ordering. Then for any dag G, if there is an order-embedding of V into G then G has at least $\frac{1}{2}|V| \log_2 |V|$ edges.*

Proof Note that G is not restricted to 2-dimensional ordering.

To prove this, let $G = (V', E')$ be a dag and π be an order-embedding of V into G. Let k^r denote the bit reversal of k. Then $V = \{p(i) : 0 \le i \le n - 1\}$, where $p(i)$ is the 2-dimensional point (i^r, i) and $n = 2^b$. Note that $p(0), p(1), \ldots, p(n-1)$ have been ordered by their second coordinate. Let $V_0 = \{p(2i) : 0 \le i < n/2\}$ and $V_1 = \{p(2i+1) : 0 \le i < n/2\}$, i.e., V_x is those points where the highest bit of their first coordinate is x. Elements of V_0 can precede elements of V_1, but not vice versa. In particular, for any i, j, where $0 \le i, j \le n - 1$, then $p(i) \prec p(j)$ if and only if $i < j$ and either $p(j) \notin V_0$ or $p(i) \notin V_1$.

Vertex $p(n-2) \in V_0$ is dominated by $p(n-1) \in V_1$, and thus there is a path in G from $\pi(p(n-2))$ to $\pi(p(n-1))$. Let $e(1)$ denote the first edge on this path. Since no other point dominates $p(n-2)$, removing $e(1)$ from E' does not alter the fact that π preserves the 2-dimensional ordering, with the possible exception that there may be elements $p(j) \in V_0$ for which $p(j) \prec p(n-1)$ but $\pi(p(j))$ no longer precedes $\pi(p(n-1))$.

Let $U_i = \{p(n-2j+1) : 1 \le j \le i\} \subset V_1$. By induction, for $1 \le i < n/2$, suppose one has found i edges $E_i = \{e(j) : 1 \le j \le i\}$ such that, for any $p, q \in V$, if $p \notin V_0$ or $q \notin U_i$, then $p \prec q$ in V if and only if $\pi(p) \prec \pi(q)$ in $(V', E' - E_i)$. To find edge $e(i+1)$, since $p(n-2i-2) \in V_0$ is dominated by $p(n-2i-1) \in V_1 - U_i$, there is a path τ in $(V', E' - E_i)$ from $\pi(p(n-2i-2))$ to $\pi(p(n-2i-1))$. Let $v \in V'$ be the first vertex on τ such that there is no path from v to any vertex in $\pi(V_0)$. Since the endpoint $\pi(p(n-2i-1))$ has this property v must exist, and it cannot be $\pi(p(n-2i-2))$ since there must be a path from it to $\pi(p(n-2i)) \in \pi(V_0)$. Let $e(i+1)$ denote the incoming edge at v in τ. This edge cannot be on any path leading to $\pi(V_1 - U_{i+1})$ since otherwise $\pi(p(n-2i-2))$ would be dominated by such a point, violating the order-embedding property because $p(n-2i-2)$ is not dominated by any point in $V_1 - U_{i+1}$. Similarly, it cannot be on any path that starts in $\pi(V_1)$ since otherwise such a point would have a path to the predecessor of v in τ, and hence would have a path to the point $\pi(p(n-2i)) \in \pi(V_0)$, which is impossible. Therefore $e(i+1)$ is only on paths from $\pi(V_0)$ to $\pi(U_{i+1})$. Thus $E_{i+1} = E_i + e(i+1)$ has the requisite properties for the induction.

Once $E_{n/2}$ has been determined, let $E'' = E' - E_{n/2}$, i.e., $n/2$ edges have been removed. The dag $G'' = (V', E'')$ has the property that if $p \prec q$ and $p, q \in V_0$ or $p, q \in V_1$, then $\pi(p) \prec \pi(q)$ in G''. If we further remove all edges in E'' which are not on any path from $\pi(p)$ to $\pi(q)$ where $p, q \in V_0$ or $p, q \in V_1$, then the resulting dag has the same property. Further, it has connected components, one of which contains $\pi(V_0)$ and another of which contains $\pi(V_0)$. V_0 and V_1 are both isomorphic to the $b-1$ bit-reversal permutation, and thus we can recursively remove edges from their components. The process can be repeated b times, removing $n/2$ edges at each stage, so there must be at least $bn/2 = \frac{1}{2}|V| \log_2 |V|$ edges. □

A somewhat similar proof appears in [15] when there is no partial ordering on the points and G is required to be a Manhattan network, i.e., the endpoints of edges differ in exactly one coordinate.

References

1. Angelov, S., Harb, B., Kannan, S., Wang, L.-S.: Weighted isotonic regression under the L_1 norm. In: Symposium on Discrete Algorithms (SODA), pp. 783–791 (2006)
2. Ayer, M., Brunk, H.D., Ewing, G.M., Reid, W.T., Silverman, E.: An empirical distribution function for sampling with incomplete information. Ann. Math. Stat. **5**, 641–647 (1955)
3. Barlow, R.E., Bartholomew, D.J., Bremner, J.M., Brunk, H.D.: Statistical Inference Under Order Restrictions: the Theory and Application of Isotonic Regression. Wiley, New York (1972)
4. Barlow, R.E., Brunk, H.D.: The isotonic regression problem and its dual. J. Am. Stat. Soc. **67**, 140–147 (1972)
5. Beran, R., Dümbgen, L.: Least squares and shrinkage estimation under bimonotonicity constraints. Stat. Comput. **20**, 177–189 (2010)

6. Bril, G., Dykstra, R., Pillars, C., Robertson, T.: Algorithm AS 206: isotonic regression in two independent variables. J. R. Stat. Soc., Ser. C, Appl. Stat. **33**, 352–357 (1984)
7. Burdakov, O., Grimvall, A., Hussian, M.: A generalized PAV algorithm for monotonic regression in several variables. In: Proceedings of COMPSTAT'2004 (2004)
8. Chandrasekaran, R., Rhy, Y.U., Jacob, V.S., Hong, S.: Isotonic separation. INFORMS J. Comput. **17**, 462–474 (2005)
9. Dembczynski, K., Greco, S., Kotlowski, W., Slowinski, R.: Statistical model for rough set approach to multicriteria classification. In: PKDD 2007: 11th European Conference on Principles and Practice Knowledge Discovery in Databases. Springer Lecture Notes in Computer Science, vol. 4702, pp. 164–175 (2007)
10. Dette, H., Scheder, R.: Strictly monotone and smooth nonparametric regression for two or more variables. Can. J. Stat. **34**, 535–561 (2006)
11. Dykstra, R., Hewett, J., Robertson, T.: Nonparametric, isotonic discriminant procedures. Biometrika **86**, 429–438 (1999)
12. Dykstra, R.L., Robertson, T.: An algorithm for isotonic regression of two or more independent variables. Ann. Stat. **10**, 708–716 (1982)
13. Gamarnik, D.: Efficient learning of monotone concepts via quadratic optimization. In: Proceedings of Computational Learning Theory (COLT), pp. 134–143 (1998)
14. Gebhardt, F.: An algorithm for monotone regression with one or more independent variables. Biometrika **57**, 263–271 (1970)
15. Gudmundsson, J., Klein, O., Knauer, C., Smid, M.: Small Manhattan networks and algorithmic applications for the earth mover's distance. In: Proceedings 23rd European Workshop on Computational Geometry, pp. 174–177 (2007)
16. Hanson, D.L., Pledger, G., Wright, F.T.: On consistency in monotonic regression. Ann. Stat. **1**, 401–421 (1973)
17. Hershberger, J., Suri, S.: Applications of a semi-dynamic convex hull algorithm. BIT Numer. Math. **32**, 249–267 (1992)
18. Jackson, D.: Note on the median of a set of numbers. Bull. Am. Math. Soc., pp. 160–164 (1921)
19. Kalai, A.T., Sastry, R.: The isotron algorithm: high-dimensional isotonic regression. In: Proceedings of Computational Learning Theory (COLT) (2009)
20. Kaufman, Y., Tamir, A.: Locating service centers with precedence constraints. Discrete Appl. Math. **47**, 251–261 (1993)
21. Legg, D., Townsend, D.: Best monotone approximation in $L_\infty[0, 1]$. J. Approx. Theory **42**, 30–35 (1984)
22. Luss, R., Rosset, S., Shahar, M.: Decomposing isotonic regression for efficiently solving large problems, In: Proceedings of Neural Information Processing Systems (2010)
23. Maxwell, W.L., Muckstadt, J.A.: Establishing consistent and realistic reorder intervals in production-distribution systems. Oper. Res. **33**, 1316–1341 (1985)
24. Megido, N.: Linear-time algorithms for linear programming in $\Re^3$ and related problems. SIAM J. Comput. **12**, 759–776 (1983)
25. Meyer, M.: Inference for multiple isotonic regression (2010). www.stat.colostate.edu/research/TechnicalReports/2010/2010_2.pdf
26. Moon, T., Smola, A., Chang, Y., Zheng, Z.: IntervalRank—isotonic regression with listwise and pairwise constraints. In: Proceedings Web Search and Data Mining, pp. 151–160 (2010)
27. Mukarjee, H., Stern, S.: Feasible nonparametric estimation of multiargument monotone functions. J. Am. Stat. Assoc. **89**, 77–80 (1994)
28. Pólya, G.: Sur une algorithme toujours convergent pour obtenir les polynomes de meillure approximation de Tchebycheff pour un fonction continue quelconque. Comptes Rendus **157**, 840–843 (1913)
29. Punera, K., Ghosh, J.: Enhanced hierarchical classification via isotonic smoothing. In: Proceedings International Conference on the World Wide Web, pp. 151–160 (2008)
30. Qian, S., Eddy, W.F.: An algorithm for isotonic regression on ordered rectangular grids. J. Comput. Graph. Stat. **5**, 225–235 (1996)
31. Robertson, T., Wright, F.T.: Multiple isotonic median regression. Ann. Stat. **1**, 422–432 (1973)
32. Robertson, T., Wright, F.T., Dykstra, R.L.: Order Restricted Statistical Inference. Wiley, New York (1988)
33. Saarela, O., Arjas, E.: A method for Bayesian monotonic multiple regression. Scand. J. Stat. **38**, 499–513 (2011)
34. Salanti, G., Ulm, K.: Multidimensional isotonic regression and estimation of the threshold value. Discussion paper 234. Institute für Statistik, Ludwig-Maximilians Universität, Munchen (2001)

35. Sasabuchi, S., Inutsuka, M., Kulatungaz, D.D.S.: A multivariate version of isotonic regression. Biometrika **70**, 465–472 (1983)
36. Spouge, J., Wan, H., Wilber, W.J.: Least squares isotonic regression in two dimensions. J. Optim. Theory Appl. **117**, 585–605 (2003)
37. Stout, Q.F.: Strict L_∞ isotonic regression. J. Optim. Theory Appl. **152**, 121–135 (2012)
38. Stout, Q.F.: Weighted L_∞ isotonic regression, submitted (2012). Available at. www.eecs.umich.edu/~qstout/pap/LinfinityIso.pdf
39. Stout, Q.F.: Isotonic regression via partitioning. Algorithmica **66**, 93–112 (2013)
40. Sysoev, O., Burdakov, O., Grimvall, A.: A segmentation-based algorithm for large-scale partially ordered monotonic regression. Comput. Stat. Data Anal. **55**, 2463–2476 (2011)
41. Ubhaya, V.A.: Isotone optimization, I, II. J. Approx. Theory **12**, 146–159, 315–331 (1974)
42. Velikova, M., Daniels, H.: Monotone Prediction Models in Data Mining. VDM Verlag (2008)