Isotonic Regression via Partitioning

Quentin F. Stout

Abstract Algorithms are given for determining weighted isotonic regressions satisfying order constraints specified via a directed acyclic graph (DAG). For the L_1 metric a partitioning approach is used which exploits the fact that L_1 regression values can always be chosen to be data values. Extending this approach, algorithms for binary-valued L_1 isotonic regression are used to find L_p isotonic regressions for $1 < p < \infty$. Algorithms are given for trees, 2-dimensional and multidimensional orderings, and arbitrary DAGs. Algorithms are also given for L_p isotonic regression with constrained data and weight values, L_1 regression with unweighted data, and L_1 regression for DAGs where there are multiple data values at the vertices.

Keywords Isotonic regression · Median regression · Monotonic · Nonparametric · Tree · DAG · Multidimensional

1 Introduction

A directed acyclic graph (DAG) G with n vertices $V = \{v_1, v_2, \ldots, v_n\}$ and m edges E defines a partial order over the vertices, where $v_i \prec v_j$ if and only if there is a path from v_i to v_j. It is assumed that the DAG is connected, and hence $m \geq n - 1$. If it isn't connected then the regression of each component is independent of the others. For a real-valued function $\mathbf{f}$ over V, f_i denotes $f(v_i)$. By weighted *data* on G we mean a pair of real-valued functions $(\mathbf{y}, \mathbf{w})$ on V where $\mathbf{y}$ is the data values and $\mathbf{w}$ is the non-negative weights. A function $\mathbf{f}$ on G is *isotonic* iff whenever $v_i \prec v_j$, then

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

Q.F. Stout
King Abdul-Aziz University, Jeddah, Saudi Arabia

Table 1 Times of fastest previously known isotonic regression algorithms	Ordered set	Metric		
		L_1	L_2	L_∞
	Linear	$\Theta(n \log n)$ [1, 22]	$\Theta(n)$ PAV	$\Theta(n \log n)$ *
	Tree	$\Theta(n^2)$ [5]	$\Theta(n \log n)$ [15]	$\Theta(n \log n)$ *
	2-dim grid	$\Theta(n^2 \log n)$ *	$\Theta(n^2)$ [21]	$\Theta(n \log n)$ *
	2-dim arbitrary	$\Theta(n^3)$ *	$\Theta(n^3)$ [21]	$\Theta(n \log^2 n)$ [24]
	$d \geq 3$ dim grid	$\Theta(n^2 \log n)$ *	$\Theta(n^4)$ *	$\Theta(n \log n)$ *
	$d \geq 3$ dim arbitrary	$\Theta(n^3)$ *	$\Theta(n^4)$ *	$\Theta(n \log^d n)$ [24]
* indicates that it is implied by the result for arbitrary ordered sets	Arbitrary	$\Theta(nm + n^2 \log n)$ [2]	$\Theta(n^4)$ [14]	$\Theta(m \log n)$ [24]

$f_i \leq f_j$. An L_p *isotonic regression* of the data is a real-valued isotonic function $\mathbf{z}$ over V that minimizes

$$\left(\sum_{i=1}^{n} w_i |y_i - z_i|^p \right)^{1/p} \qquad 1 \leq p < \infty$$

$$\max_{i=1}^{n} w_i |y_i - z_i| \qquad\qquad p = \infty$$

among all isotonic functions. The *regression error* is the value of this expression. For a vertex $v_i \in V$ and isotonic regression $\mathbf{z}$, the *regression value* of v_i is $\mathbf{z}_i$. Isotonic regressions partition the vertices into *level sets* which are connected regions where the regression value is a constant. The regression value of a level set is the L_p weighted mean of the data values.

Isotonic regression is an important alternative for standard parametric regression when there is no confidence that the relationship between independent variables and the dependent one is parametric, or when an independent variable is discrete. For example, one may believe that average weight is an increasing function of height and of S < M < L < XL shirt size. That is, if the shirt size is held constant the average weight increases with height, and similarly the weight increases if the height is fixed and the shirt size increases. However, there are no assumptions of what happens if height increases but shirt size decreases. Barlow et al. [3] and Robertson et al. [20] review work on isotonic regression dating back to the 1940's. This includes use of the L_1, L_2, and L_∞ metrics and orderings including linear, tree, multidimensional, and arbitrary DAGs. These books also give examples showing uses of isotonic regression for optimization problems such as nonmetric multidimensional scaling.

Table 1 shows the fastest known isotonic regression algorithms for numerous combinations of metric and ordered set. "Dim" ordered sets have vertices that are points in d-dimensional space with the natural ordering, i.e., $(a_1, \ldots, a_d) \prec (b_1, \ldots, b_d)$ iff $a_i \leq b_i$ for $1 \leq i \leq d$. For $d = 2$ this is also known as "matrix", "row and column", or

"bimonotonic" ordering, and for general d is sometimes called "domination". Here it will be called *multidimensional ordering*. "Grid" sets are arranged in a d-dimensional grid of extent n_i in the ith dimension, where $n = \prod_{i=1}^{d} n_i$, and "arbitrary" sets have points in arbitrary locations.

For unweighted data the L_∞ isotonic regression can be found in $\Theta(m)$ time using the well-known fact that the regression value at $v \in V$ can be chosen to be $(\min_{v \preceq v_j} y_j + \max_{v_i \preceq v} y_i)/2$. For $1 < p < \infty$, restricting to unweighted data does not appear to help, but Sect. 5.4 shows that restricting both data and weight values can reduce the time required. For L_1, restricting to unweighted data can be useful for dense DAGs, reducing the time to $\Theta(n^{2.5} \log n)$. This is discussed in Sect. 7.

For linear orders, *pair adjacent violators*, PAV, has been repeatedly rediscovered. Starting with the initial data as n level sets, whenever two consecutive level sets violate the isotonic condition they are merged into a single level set. No matter what order the sets are merged in, for any p the result is optimal. Simple left-right parsing yields the L_2 isotonic regression in $\Theta(n)$ time, L_1 in $\Theta(n \log n)$ time [1, 22], and L_∞ in $\Theta(n \log n)$ time [24]. Thompson's extension of PAV to trees [27] is discussed in Sect. 3.

For more general orderings PAV can give incorrect results. For arbitrary DAGs, Maxwell and Muckstadt [14], with a small correction by Spouge et al. [21], gave a $\Theta(n^4)$ time algorithm for L_2 isotonic regression; Angelov et al. [2] gave a $\Theta(nm + n^2 \log n)$ time algorithm for L_1; and Stout [24] gave a $\Theta(m \log n)$ time algorithm for L_∞, based on a small modification of an algorithm of Kaufman and Tamir [12].

For $1 < p < \infty$ the L_p mean of a set is unique (see the Appendix), as is the isotonic regression, but this is not always true for L_1. The L_1 mean of a set is a weighted median, and L_1 regression is often called median regression. A weighted median of the set of data $(\mathbf{y}, \mathbf{w})$, with total weight $W = \sum w_i$, is any number z such that $\sum_{y_i \leq z} w_i \geq W/2$ and $\sum_{y_j \geq z} w_j \geq W/2$. Since the weighted median of a set can always be chosen to be one of the values in the set, there is always an L_1 isotonic regression where all regression values are original data values. In Sect. 2 a partitioning approach is introduced, one which repeatedly halves the number of possible regression values at each vertex. Since there are at most n possible regression values being considered, only a logarithmic number of iterations are needed. This approach is applied to tree (Sect. 3) and 2-dimensional (Sect. 4) orderings.

Partitioning is extended to general L_p isotonic regression in Sect. 5, where each stage involves a binary-valued L_1 regression. Algorithms are developed for "semi-exact" regressions where the answers are as accurate as the ability to compute the L_p mean, for approximations to within a specified accuracy, and for exact regressions when the data values and weights are constrained.

Section 6 considers L_1 isotonic regression when there are multiple values at each vertex, and Sect. 7 contains final remarks, including an observation concerning L_1 isotonic regression on unweighted data.

For data $(\mathbf{y}, \mathbf{w})$, let $\mathrm{err}(v_i, z) = w_i |y_i - z|$, i.e., it is the error of using z as the regression value at v_i.

2 Partitioning

For a closed set S of real numbers, an *S-valued isotonic regression* is an isotonic regression where the regression values are elements of S, having minimal regression error among all isotonic S-valued functions. *Rounding x to $\{a, b\}$*, where $a < b$ and $x \notin (a, b)$, results in a if $x \leq a$ and b if $x \geq b$.

Lemma 2.1 *For a DAG G and data $(\mathbf{y}, \mathbf{w})$, let $S = \{a, b\}$, $a < b$, be such that there is an L_1 isotonic regression with no values in (a, b). Let $\mathbf{z^s}$ be an S-valued L_1 isotonic regression. Then there is an unrestricted L_1 isotonic regression $\mathbf{z}$ such that $\mathbf{z}$ has no values in (a, b) and $\mathbf{z^s}$ is $\mathbf{z}$ rounded to S.*

Proof Let $\tilde{\mathbf{z}}$ be an optimal regression with no values in (a, b). If there is no vertex v_i where $\tilde{z}_i \leq a$ and $z_i^s = b$, nor any vertex v_j where $\tilde{z}_j \geq b$ and $z_j^s = a$, then $\mathbf{z^s}$ is $\tilde{\mathbf{z}}$ rounded to S. Otherwise the two cases are similar, so assume the former holds. Let y be the largest value such that there is a vertex v_i where $\tilde{z}_i = y \leq a$ and $z_i^s = b$, and let U be the subset of V where $z_i^s = b$ and $\tilde{z}_i = y$.

Let $[c, d]$ be the range of weighted medians of the data on U. If $c > a$ then the function which equals $\tilde{\mathbf{z}}$ on $V \setminus U$ and $\min\{c, b\}$ on U is isotonic and has L_1 error strictly less than that of $\tilde{\mathbf{z}}$, contradicting the assumption that it is optimal. If $c \leq a$ and $d < b$, then the function which is $\mathbf{z^s}$ on $V \setminus U$ and a on U is an isotonic S-valued function with error strictly less than that of $\mathbf{z^s}$, contracting the assumption that is optimal. Therefore $[a, b] \subseteq [c, d]$, so the isotonic function which is $\tilde{\mathbf{z}}$ on $V \setminus U$ and b on U has the same error as $\tilde{\mathbf{z}}$, i.e., is optimal, and has more vertices where it rounds to $\mathbf{z^s}$ than $\tilde{\mathbf{z}}$ has. In a finite number of iterations one obtains an unrestricted regression which rounds to $\mathbf{z^s}$ everywhere. $\square$

The L_1 algorithms herein have an initial sort of the data values. Given data $(\mathbf{y}, \mathbf{w})$, let $\acute{y}_1 < \cdots < \acute{y}_{n'}$ be the distinct data values in sorted order, where n' is the number of distinct values, and let $\acute{y}[k, \ell]$ denote $\{\acute{y}_k, \ldots, \acute{y}_\ell\}$. L_1 partitioning proceeds by identifying, for each vertex, subintervals of $\acute{y}[1, n']$ in which the final regression value lies. Let $V[a, b]$ denote the vertices with interval $\acute{y}[a, b]$ and let $\mathsf{Int}(v_i)$ denote the interval that v_i is currently assigned to. Initially $\mathsf{Int}(v_i) = \acute{y}[1, n']$ for all $v_i \in V$.

Lemma 2.1 shows that by finding an optimal $\{\acute{y}_\ell, \acute{y}_{\ell+1}\}$-valued isotonic regression $\mathbf{z^s}$, the vertices where $\mathbf{z^s} = \acute{y}_\ell$ can be assigned to $V[1, \ell]$, and those where $\mathbf{z^s} = \acute{y}_{\ell+1}$ can be assigned to $V[\ell + 1, n']$. The level sets of $V[1, \ell]$ are independent of those of $V[\ell + 1, n']$, and hence an optimal regression can be found by recursively finding an optimal regression on $V[1, \ell]$ using only regression values in $\acute{y}[1, \ell]$, and an optimal regression on $V[\ell + 1, n']$ using only regression values in $\acute{y}[\ell + 1, n']$. Keeping track of a vertex's interval merely requires keeping track of k and ℓ (in fact, ℓ does not need to be stored with the vertex since at each stage, vertices with the same lower bound have the same upper bound). When determining the new interval for a vertex one only uses vertices with the same interval. At each stage the assignment is isotonic in that if $v_i \prec v_j$ then either $\mathsf{Int}(v_i) = \mathsf{Int}(v_j)$ or the intervals have no overlap and the upper limit of $\mathsf{Int}(v_i)$ is less than the lower limit of $\mathsf{Int}(v_j)$. Each stage halves the size of each interval, so eventually they are a single value, i.e., the
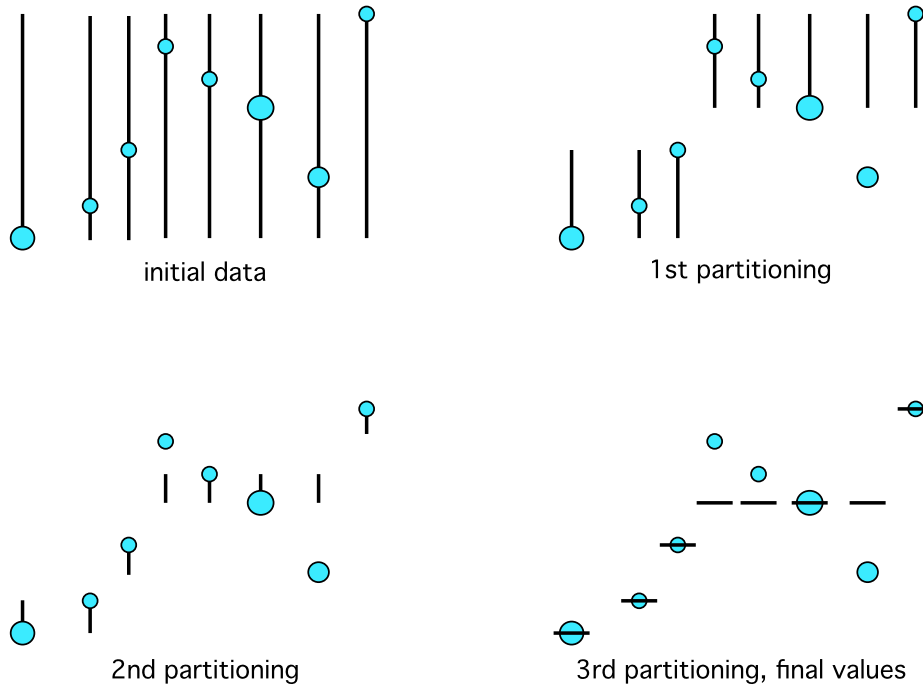
Fig. 1 Partitioning in Theorem 3.1: size is weight, bars are the interval regression value will be in

regression value has been determined. Thus there are $\lceil \lg n' \rceil$ stages. Figure 1 shows the partitioning intervals that are produced by the algorithm in Theorem 3.1.

An aspect that one needs to be careful about is that after partitioning has identified the vertices with regression values in a given interval, then the next partitioning step on that subgraph does not necessarily use the median of the data values in the subgraph. For example, suppose the values and weights on a linear ordering are $(4, 1)$, $(3, 10)$, $(1, 1)$, $(2, 1)$. The first partitioning involves values 2 and 3, and results in all vertices being assigned to the interval $[3, 4]$. Since the subgraph is the original graph, using the median of the data values in the subgraph results in an infinite loop. Instead, the next stage uses partitioning values 3 and 4, quartiles from the original set of values. One can use the median of the subgraph if when the subgraph corresponds to the range of values $[a, b]$ then only data values in $[a, b]$ are used to determine the median. Also note that it is the median of the values, not their weighted median.

While having some similarities, Ahuja and Orlin's scaling for L_1 regression [1] is quite different from partitioning. In their approach, all vertices are initially in their own level set, and whenever two vertices are placed in the same level set then they remain in the same level set. In partitioning, all vertices are initially in the same level set, and whenever two are placed in different level sets they stay in different level sets. An important partitioning approach is "minimum lower sets", first used by Brunk in the 1950's [4].

3 Trees

Throughout, all trees are rooted, with each vertex except the root pointing to its parent. Star ordering, where all nodes except the root have the root as their parent, is often called tree ordering in statistical literature concerning tests of hypotheses, where several hypotheses are compared to the null hypothesis.

Thompson [27] showed that PAV can be extended to tree orderings by using a bottom-up process. A level set S is merged with a violating level set T containing a child of a vertex in S iff T's mean is the largest among all level sets containing a child of a vertex in S. At each step, every level set is a subtree. Using this, for an arbitrary tree the regression can be found in $\Theta(n \log n)$ time for the L_2 [15] and L_∞ [24] metrics. These times can be reduced to $\Theta(n)$ when the tree is a star [15, 24].

For L_1, Chakravarti [5] used linear programming in an algorithm taking $\Theta(n^2)$ time, while Qian [18] used PAV but no details were given. While PAV can be used to determine L_1 isotonic regression in $\Theta(n \log n)$ time, a partitioning approach is simpler and is used in the extensions to L_p isotonic regression in Sect. 5. Figure 1 illustrates the stages of the following theorem for a linear ordering.

Theorem 3.1 *For a tree of n vertices and data* $(\mathbf{y}, \mathbf{w})$, *an L_1 isotonic regression can be determined in* $\Theta(n \log n)$ *time.*

Proof Initially each vertex has interval $\acute{y}[1, n']$. At each partitioning stage, suppose vertex v_i has interval $\acute{y}[k, \ell]$. If $k = \ell$ then the regression value at v_i is $\acute{y}_k$. Otherwise, let $j = \lfloor (k + \ell)/2 \rfloor$. Define $Z(v_i, 0)$ to be the minimal possible error in the subtree rooted at v_i with vertices in $V[k, \ell]$, given that the regression at v_i is $\acute{y}_j$, and similarly for $Z(v_i, 1)$ and $\acute{y}_{j+1}$. Then Z satisfies the recursive equations:
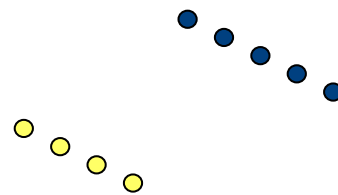
$$Z(v_i, 0) = \text{err}(v_i, \acute{y}_j) + \sum \{Z(u, 0) : u \text{ child of } v_i \text{ and } \text{Int}(v_i) = \text{Int}(u)\}$$

$$Z(v_i, 1) = \text{err}(v_i, \acute{y}_{j+1})$$
$$+ \sum \{\min\{Z(u, 0), Z(u, 1)\} : u \text{ child of } v_i \text{ and } \text{Int}(v_i) = \text{Int}(u)\}$$

which can be computed using a bottom-up postorder traversal.

To recover the optimal solution for this stage and to determine the subintervals for each vertex, a top-down preorder traversal is used. For the root r, its new interval is $\acute{y}[k, j]$ or $\acute{y}[j + 1, \ell]$ depending on which of $Z(r, 0)$ and $Z(r, 1)$ is smallest, respectively. Ties can be broken arbitrarily. For a node p, if its parent initially had the same interval and the parent's new interval is $\acute{y}[k, j]$, then that is p's new interval as well. If the parent's new interval is $\acute{y}[j + 1, \ell]$, or had an initial interval different than p's, then p's new interval is determined using the same procedure as was used for the root. The traversals for computing Z and updating the intervals can all be done in $\Theta(n)$ time per stage. $\square$

A faster algorithm, using neither partitioning nor sorting, is possible when the tree is star and, more generally, when the graph is a complete directed bipartite DAG, i.e., the vertices are partitioned into V_1 and V_2, there is a directed edge from every vertex in V_1 to every vertex in V_2, and no additional edges exist. A star corresponds to $|V_2| = 1$.

Fig. 2 The complete directed
bipartite DAG $K_{4,5}$ given via
2-dimensional ordering

Proposition 3.2 *For a complete directed bipartite DAG of n vertices, given data $(\mathbf{y}, \mathbf{w})$, an L_1 isotonic regression can be determined in $\Theta(n)$ time.*

Proof Let V_1 and V_2 be the bipartite partition of the vertices. Note that there is an optimal regression $\mathbf{z}$ of the form $z_i = \min\{y_i, c\}$ for $v_i \in V_1$ and $z_j = \max\{y_j, c\}$ for $v_j \in V_2$, for some data value c. Let a be the largest data value such that

$$\sum_{v_i \in V_1, \, y_i \geq a} w_i \geq \sum_{v_j \in V_2, \, y_j \leq a} w_j$$

and let b be the smallest data value $> a$. Then there is an optimal L_1 isotonic regression where either $c = a$ or $c = b$. In linear time one can determine a, b, and c. □

Punera and Ghosh [17] consider regression on a tree with the monotonicity constraint that $\hat{z}_i = \max\{\hat{z}_j : v_j$ a child of $v_i\}$ whenever v_i is not a leaf node. This can easily be computed by modifying the definition of $Z(v_i, 1)$ to require that for at least one child $Z(u, 1)$ is used (even if it is not smaller than $Z(u, 0)$), and recording this child so that the proper intervals can be determined. The time remains $\Theta(n \log n)$, which improves upon their $\Theta(n^2 \log n)$ time algorithm.

4 Multidimensional Orderings

Isotonic regression involving more than one independent variable has often been studied (see the numerous references in [3, 20]). A problem with d independent ordered variables naturally maps to a DAG where the vertices are points in a d-dimensional space and the edges are implied by domination ordering, i.e., $(a_1, \ldots, a_d) \prec (b_1, \ldots, b_d)$ iff $a_i \leq b_i$ for $1 \leq i \leq d$. In statistics, when the observations are over a grid it is sometimes called a complete layout, and when they are arbitrarily distributed it is an incomplete layout.

A d-dimensional grid has $\Theta(dn)$ grid edges, and thus the general algorithm in [2] shows that the L_1 isotonic regression can be found in $\Theta(n^2 \log n)$ time. However for vertices in arbitrary positions $\Theta(n^2)$ edges may be needed, as is shown in Fig. 2, which would result in $\Theta(n^3)$ time. The following theorem shows that for points in general position the time can be reduced to $\tilde{\Theta}(n)$ for $d = 2$ and $\tilde{\Theta}(n^2)$ for $d \geq 3$.

Theorem 4.1 *For a set of n vertices in d-dimensional space, given data $(\mathbf{y}, \mathbf{w})$, an L_1 isotonic regression can be determined in*

(a) $\Theta(n \log n)$ *time if the vertices form a grid, $d = 2$,*
(b) $\Theta(n \log^2 n)$ *time if the vertices are in arbitrary locations, $d = 2$,*

(c) $\Theta(n^2 \log n)$ *time if the vertices form a grid*, $d \geq 3$,

(d) $\Theta(n^2 \log^d n)$ *time if the vertices are in arbitrary locations*, $d \geq 3$,

where the implied constants depend on d.

The proof will be broken into parts, with part (a) in Sect. 4.1 and part (b) in Sect. 4.2. Part (c) was discussed above.

For (d), let V be the set of vertices. In [25] it is shown that in $\Theta(n \log^{d-1} n)$ time one can construct a DAG $G' = (V', E')$ where $V \subseteq V'$, $|V'| = |E'| = \Theta(n \log^{d-1} n)$, and for all $v_i, v_j \in V$, $v_i \prec v_j$ in domination ordering iff $v_i \prec v_j$ in G'. One obtains (d) by placing arbitrary values with weight 0 on $V' \setminus V$ and applying the general algorithm in [2], noting that the number of stages is linear in the number of vertices.

4.1 2-Dimensional Grids

Our algorithm for Theorem 4.1(a) is similar to that of Spouge, Wan and Wilbur [21] for L_2 regression.

Suppose the vertices are a 2-dimensional grid with rows $1, \ldots, r$ and columns $1, \ldots, c$, where $n = r \cdot c$. Let $\mathrm{err}((g, h), z)$ denote the error of using z as the regression value at grid location (g, h). For the initial round of partitioning, let $\ell = \lceil n'/2 \rceil$ and $S = \{\acute{y}_\ell, \acute{y}_{\ell+1}\}$. To determine an optimal S-valued regression, let $Z(g, h)$ denote the optimal S-valued regression error on the subgrid $[1, \ldots, g] \times [1, \ldots, c]$ subject to the constraint that the regression value at (g, h) is $\acute{y}_{\ell+1}$ (and hence the regression values in row g, columns $[h+1, \ldots, c]$, are also $\acute{y}_{\ell+1}$), and that this is the leftmost $\acute{y}_{\ell+1}$ value in this row (hence all regression values in $[1, \ldots, g] \times [1, \ldots, h-1]$ are $\acute{y}_\ell$). Define $Z(g, c+1)$ to be the optimal regression error when all of the values on row g are $\acute{y}_\ell$.

For $g \in [1, \ldots, r]$ and $h \in [1, \ldots, c+1]$ let

$$R(g, h) = \sum_{k=1}^{h-1} \mathrm{err}\big((g, k), \acute{y}_\ell\big) + \sum_{k=h}^{c} \mathrm{err}\big((g, k), \acute{y}_{\ell+1}\big)$$

Z satisfies a simple recurrence which can be computed in increasing row order:

$$Z(g, h) = \begin{cases} R(g, h) + \min\{Z(g-1, k) : k \in [h, \ldots, c+1]\} & g > 1 \\ R(g, h) & g = 1 \end{cases} \quad (1)$$

i.e., if the leftmost $\acute{y}_{\ell+1}$ value in row g is at column h, then the optimal S-valued regression on $[1, \ldots, g] \times [1, \ldots, c]$ given this constraint will contain the optimal regression on $[1, \ldots, g-1] \times [1, \ldots, c]$ that on row $g-1$ has its leftmost $\acute{y}_{\ell+1}$ value no further left than h. For any row, the R and Z values can be easily computed in $\Theta(c)$ time, and thus for all entries they can be computed in $\Theta(n)$ time. The minimal regression error is the minimum of $Z(r, h)$ for $h \in [1, \ldots, c+1]$. As usual, by storing the value of k that minimizes (1), the optimal regression values and new intervals can be recovered in linear time.

For subsequent partitions the calculation of $R(g, h)$ is modified to include only values corresponding to grid points in the same level set, i.e., those with the same

interval as (g, h). To determine $Z(g, h)$, if $\mathsf{Int}((g, h)) = \mathsf{Int}((g-1, h))$ then the minimum is taken over values at grid points with the same interval (these grid points are consecutive), while if $\mathsf{Int}((g, h)) \neq \mathsf{Int}((g-1, h))$ then $Z(g, h) = R(g, h)$. This completes the proof of Theorem 4.1(a).

4.2 2-Dimension Data, Arbitrary Position

To prove Theorem 4.1(b), since the coordinates in each dimension are only used for ordering, their specific values are not important. Assume the first coordinates have values $1, \ldots, r$, and second coordinates have values $1, \ldots, c$. One can view the vertices as being in an $r \times c$ grid where some of the grid points have no data. It is possible that $r \cdot c = \Theta(n^2)$ and hence a naive use of Theorem 4.1(a) would require $\Theta(n^2 \log n)$ time. However, one can exploit the fact that most of the grid points have no data.

Suppose that the points have been sorted in increasing row order, breaking ties by sorting in increasing column order. Thus if $u \succ v$ then u occurs after v in the ordering. Relabel the vertices so that v_i is the ith point in this ordering, with coordinates (r_i, c_i). For $g \in [0, \ldots, n]$ and $h \in [1, \ldots, c+1]$, when solving the $S = \{\acute{y}_\ell, \acute{y}_{\ell+1}\}$ partitioning problem let $Z(g, h)$ be the minimal regression error for an S-valued regression on the first g points such that the leftmost $\acute{y}_{\ell+1}$ value is in column h. (If none of $\{z_1, \ldots, z_g\}$ are in column h then $Z(g, h) = Z(g, h+1)$.) Define $Z(0, h)$ to be 0 for $h \in [1, \ldots, c+1]$.

For $g \in [1, \ldots, n]$ and $h \in [1, \ldots, c+1]$, one has

$$
Z(g, h) = \begin{cases}
\mathrm{err}(v_g, \acute{y}_\ell) + Z(g-1, h) & \text{if } h > c_g \\
\mathrm{err}(v_g, \acute{y}_{\ell+1}) + \min\{Z(g-1, k) : k \in [h, \ldots, c+1]\} & \text{if } h = c_g \\
\mathrm{err}(v_g, \acute{y}_{\ell+1}) + Z(g-1, h) & \text{if } h < c_g
\end{cases}
$$

This can be efficiently computed in a bottom-up manner by means of operations on a balanced binary search tree with nodes $1, \ldots, c+1$, where at the gth step the information about the gth point is added.

Ignoring the case of equality, the other cases add $\mathrm{err}(v_g, \acute{y}_\ell)$ to the interval of columns $c_g + 1, \ldots, c+1$ and $\mathrm{err}(v_g, \acute{y}_{\ell+1})$ to the interval of columns $1, \ldots, c_g - 1$, where the value of $Z(g, h)$ is the sum of the values added to column h by vertices v_1 through v_g. It is well-known how to create a balanced tree where the addition of a value to an interval of keys, and determining the sum of the intervals covering an index, can be computed in $\Theta(\log n)$ time. Further, it can be augmented to also perform the minimum calculation needed for the equality case in the same time bound. The final modification needed is in the equality case to insure that for points above the gth one, when the intervals covering c_g are summed they give the correct value. To do this, to the degenerate index interval $[c_g, c_g]$ add the value which is the correct value of $V(g, c_g)$ minus $V(g-1, c_g)$. This makes the sum of the intervals covering column c_g equal to the correct value.

The optimal S-valued regression error is $\min\{Z(n, h) : h \in [1, \ldots, c+1]\}$, and the S-valued recurrence values can be determined in reverse order, as for trees. For subsequent rounds one can partition the vertices into two subsets containing the two new level sets. Each round of partitioning can be completed in $\Theta(n \log n)$ time, completing the proof of Theorem 4.1(b). Thus Theorem 4.1 is proven.

5 L_p Isotonic Regression, $1 < p < \infty$

In this section we consider L_p isotonic regression for $1 < p < \infty$. While it has been mentioned in the literature, few concrete algorithms have appeared for $p \neq 2$. The only ones we are aware of involve linear orders [1, 7, 26], with the fastest being Ahuja and Orlin's algorithm [1] which takes $\Theta(n \log U)$ time, where U is the maximum difference of any two data values and the results are restricted to integers.

The approach used here builds on the results for L_1. Section 5.1 introduces "ϵ-partitioning", which plays the same role as partitioning did for L_1. In Sect. 5.2 we find "semi-exact" isotonic regressions which are as exact as the capability to compute the L_p mean. E.g., they are exact for L_2. In Sect. 5.3 we find approximate solutions to within a given error, and in Sect. 5.4 it is shown that these can be made to be exact, not just semi-exact, in restricted circumstances.

Runtime of the L_p Algorithms L_p isotonic regression will be determined via partitioning using L_1 binary-valued isotonic regression at each stage. The running time of an L_p algorithm will be the number of stages multiplied by the time to determine the regression values used for the binary regression and then solve it. For a given DAG the time to solve the binary regression at each stage is:

- tree: $\Theta(n)$, from Sect. 3,
- 2-dimensional grid: $\Theta(n)$, from Sect. 4.1,
- 2-dimensional arbitrary: $\Theta(n \log n)$, from Sect. 4.2,
- d-dimensional grid, $d \geq 3$: $\Theta(n^2 \log n)$, from Theorem 4.1(c),
- d-dimensional arbitrary, $d \geq 3$: $\Theta(n^2 \log^d n)$, from Theorem 4.1(d).
- arbitrary: $\Theta(nm + n^2 \log n)$, from Angelov et al. [2].

The number of stages will not necessarily be $\Theta(\log n)$ since the regression values might not be data values, and determining the values used for the binary regression may take a nonnegligible amount of time.

5.1 ϵ-Partitioning

We introduce a partitioning method which generalizes the partitioning step used for L_1 regression.

Minimizing the weighted L_1 error in (2) below is the same as minimizing the weighted L_p error for $\{C, D\}$-partitioning since the new weights reflect the increase in the L_p error between rounding to the closer of C, D and rounding to the further. The proof directly follows that of Lemma 2.1, with the added simplification that the L_p mean of a set is unique when $p > 1$ (i.e., in the proof there, $c = d$). That proof shows that g is unique as well, from which one infers that it has no values in $(0, 1)$.

Lemma 5.1 *Given $1 < p < \infty$, a DAG G, and data $(\mathbf{y}, \mathbf{w})$, let f be the L_p isotonic regression. Let $C < D$ be such that no data value nor level set value of f is in (C, D), and let g be an L_1 isotonic regression on G with data $(\mathbf{y}', \mathbf{w}')$, where*

$$(y_i', w_i') = \begin{cases} (0, \ w_i \cdot [(D - y_i)^p - (C - y_i)^p]) & \textit{if } y_i \leq C \\ (1, \ w_i \cdot [(y_i - C)^p - (y_i - D)^p]) & \textit{otherwise} \end{cases} \tag{2}$$

Then g is unique, {0, 1}-valued, and $g_i = 0$ iff $f_i \leq C$.

Let C and $\epsilon > 0$ be arbitrary. To apply the above lemma with $D = C + \epsilon$, since there are only finitely many regression and data values there is an ϵ sufficiently small so that none are in $(C, C + \epsilon)$. However, since we don't know the regression, we don't *a priori* know how small ϵ needs to be, so we treat it as an infinitesimal. For $y_i \leq C$ the weight for the binary regression is the amount the weighted L_p distance will increase from C to $C + \epsilon$. Since ϵ is an infinitesimal, this is merely ϵ times the derivative $w_i p (C - y_i)^{p-1}$. Similarly, for $y_i > C$ the weight is $\epsilon \cdot w_i p (y_i - C)^{p-1}$. Since all weights have factors of ϵ and p they are removed, i.e., the binary L_1 regression problem being solved is on data of the form

$$(y_i', w_i') = \begin{cases} (0, \ w_i(C - y_i)^{p-1}) & \text{if } y_i \leq C \\ (1, \ w_i(y_i - C)^{p-1}) & \text{otherwise} \end{cases} \tag{3}$$

We call this ϵ-*partitioning*.

Let f be the L_p isotonic regression of the original data. For a level set with mean $< C$ the sum of the regression errors is a strictly increasing function of its distance from C, as it is if the mean is $> C$. Thus, if g is an isotonic regression for the ϵ-partitioning, to minimize L_1 error it must be 0 on all level sets of f with mean $< C$ and 1 on all level sets with mean $> C$. For level sets of f with mean C, g might be any value in $[0, 1]$. We partition G by putting all vertices with $g \in [0, 1)$ into one subgraph, and those with $g = 1$ into the other. Regression values in the former will be in $(-\infty, C]$, and those in the latter will be in $[C, \infty)$.

5.2 Semi-exact Isotonic Regression

We call the below *semi-exact* since it is as exact as one wishes to compute the L_p mean of a set.

The values of C used for ϵ-partitioning are based on the "minimum lower sets" approach [4]. At the first step C is the L_p mean of the entire DAG. If there is more than one level set in the isotonic regression there is at least one with mean $< C$ and at least one with mean $> C$ and hence ϵ-partitioning produces a non-trivial partitioning. Thus if it produces a trivial partition the L_p regression is the constant function C and the algorithm is finished, while otherwise the two pieces of the partition are further refined. There is the possibility that the L_p regression is constant but ϵ-partitioning produces a non-trivial partitioning, where each piece again has L_p mean C. For analyzing worst-case time complexity this is no worse than the partitioning when the isotonic regression is not constant. An alternative approach is to check if one piece has L_p mean C, in which case both do and the regression value is C.

L_p isotonic regression may require $\Theta(n)$ stages. For example, if partitioning is used for a linear order, the L_2 metric, and unweighted data $1, 4, 27, \ldots, n^n$, then at each stage only the largest remaining value is determined to be a level set. Another

complication is that one cannot, in general, determine the L_p mean exactly, and generating a sufficiently accurate approximation may entail significant time. For a DAG of n vertices with data $(\mathbf{y}, \mathbf{w})$, the time to determine all of the means utilized throughout the algorithm will involve a part that is independent of the data, taking $\Theta(n^2)$ time, and a data-dependent part, $\mathcal{T}_p(n, \mathbf{y}, \mathbf{w})$, that depends upon the accuracy desired. This assumes the bracketing data values have been determined, where, if the mean is C, then the *bracketing data values* are $\acute{y}_i$, $\acute{y}_{i+1}$ such that $\acute{y}_i \leq C \leq \acute{y}_{i+1}$. The significance of the bracketing values is discussed in the Appendix. They can be determined by first using $O(\log n)$ stages of ϵ-partitioning on the data values and then continuing with ϵ-partitioning based on the means.

Combining the time to determine the L_p means with the fact that there may be $\Theta(n)$ stages, each using an ϵ-partitioning taking time given at the beginning of Sect. 5, results in the following:

Theorem 5.2 *Let* $1 < p < \infty$. *Given a DAG G and data* $(\mathbf{y}, \mathbf{w})$, *one can determine the semi-exact* L_p *isotonic regression in the following time*:

- *tree*: $\Theta(n^2 + \mathcal{T}_p(n, \mathbf{y}, \mathbf{w}))$,
- *2-dim grid*: $\Theta(n^2 + \mathcal{T}_p(n, \mathbf{y}, \mathbf{w}))$,
- *2-dim arb*: $\Theta(n^2 \log n + \mathcal{T}_p(n, \mathbf{y}, \mathbf{w}))$,
- *d-dim grid*, $d \geq 3$: $\Theta(n^3 \log n + \mathcal{T}_p(n, \mathbf{y}, \mathbf{w}))$,
- *d-dim arb*, $d \geq 3$: $\Theta(n^3 \log^d n + \mathcal{T}_p(n, \mathbf{y}, \mathbf{w}))$,
- *arbitrary*: $\Theta(n^2 m + n^3 \log n + \mathcal{T}_p(n, \mathbf{y}, \mathbf{w}))$,

where the implied constants for d-dimensional DAGs are functions of d. Further, for $p = 2, 3, 4$, *and* 5 *the regressions are exact and* $\mathcal{T}_p(n, \mathbf{y}, \mathbf{w}) = 0$.

Proof Beyond the timing analysis discussed above, the only additional proof needed is the exactness claim. That follows from the fact that polynomials of degree ≤ 4 can be solved exactly (see the Appendix). $\qquad\square$

The results for L_2 improve upon Spouge, Wan, and Wilbur's $\Theta(n^3)$ time algorithm [21] for arbitrary points in 2-dimensional space; for $m = o(n^2)$ improve upon the $\Theta(n^4)$ algorithm of Maxwell and Muckstadt [14] for arbitrary DAGs; and improve upon using their algorithm for d-dimensional data, both grids and arbitrary, for $d \geq 3$.

By using PAV one can improve the result for trees when p is an integer:

Proposition 5.3 *Let* p *be an integer* ≥ 2. *Given a tree G and data* $(\mathbf{y}, \mathbf{w})$, *one can determine the semi-exact* L_p *isotonic regression in the following time*:

- *G linear*: $\Theta(n + \mathcal{T}_p(n, \mathbf{y}, \mathbf{w}))$ *if* p *is even and* $\Theta(n \log n + \mathcal{T}_p(n, \mathbf{y}, \mathbf{w}))$ *if* p *is odd*,
- *G arbitrary tree*: $\Theta(n \log n + \mathcal{T}_p(n, \mathbf{y}, \mathbf{w}))$.

Further, for $p = 2, 3, 4$, *and* 5 *the regressions are exact and* $\mathcal{T}_p(n, \mathbf{y}, \mathbf{w}) = 0$.

Proof For PAV algorithms there are two components to time: the time to keep track of which level sets to merge, and the time to merge the sets and compute their mean. For

a linear order the time to keep track of the level sets is $\Theta(n)$, while for a tree order it can be done in $\Theta(n \log n)$ time [15]. In the Appendix it is shown that for even positive integers the time to merge the sets and compute their means is $\Theta(n + \mathcal{T}_p(n, \mathbf{y}, \mathbf{w}))$, and for $p = 2$ and 4 each root can be found exactly in constant time. For odd positive integers, merging the sets and finding their means takes $\Theta(n \log n + \mathcal{T}_p(n, \mathbf{y}, \mathbf{w}))$ time, and for $p = 3$ and 5 each root can be found exactly in constant time. □

5.3 Approximation to Within δ

To find an approximation where the value at each vertex is at most δ from the optimal regression value one need only use regression values of the form $k\delta + \min_{i=1}^{n} y_i$, for $0 \leq k \leq \lfloor (\max_{i=1}^{n} y_i - \min_{i=1}^{n} y_i)/\delta \rfloor$. Just as for L_1, one can do binary search on the possible regression values. Using the remarks at the beginning of Sect. 5 concerning running time gives:

Theorem 5.4 *Let $1 \leq p < \infty$. Given a DAG G, data $(\mathbf{y}, \mathbf{w})$, and $\delta > 0$, the time to determine the L_p isotonic regression to within δ is*:

- *tree*: $\Theta(n \log K)$,
- *2-dim*: $\Theta(n \log K)$ *for a grid, and* $\Theta(n \log n \log K)$ *for arbitrary points,*
- *d-dim, $d \geq 3$*: $\Theta(n^2 \log n \log K)$ *for a grid, and* $\Theta(n^2 \log^d n \log K)$ *for arbitrary points,*
- *arbitrary*: $\Theta((nm + n^2 \log n) \log K)$,

where $K = (\max_{i=1}^{n} y_i - \min_{i=1}^{n} y_i)/\delta$ and the implied constants for d-dimensional DAGs depend on d.

There has been extensive work on approximations, e.g., [3, 9, 10, 20]. A problem related to the one solved here is to be given a constant $C > 0$ and find an approximation with regression error at most C times that of the optimal error. However, to date there is none with a running time which depends on C but is independent of the data.

5.4 Constrained Data Values

In certain situations one can use approximate solutions to achieve exact solutions. We first give a bound on how close different level set values can be for unweighted binary data.

Lemma 5.5 *For $1 < p < \infty$ there are constants $\alpha(p), \beta(p) > 0$ such that if S_1 and S_2 are sets of no more than n binary values then either their L_p means are the same or they differ by at least $\alpha(p)/n^{\beta(p)}$.*

Proof Let $M(i, j)$ denote the L_p mean of i 0's and j 1's. Simple calculus shows that

$$M(i, j) = \frac{j^q}{i^q + j^q} \tag{4}$$

where $q = 1/(p - 1)$. The difference between means, $M(i_1, j_1) - M(i_2, j_2)$, is

$$\frac{J_1 I_2 - J_2 I_1}{(I_1 + J_1) \cdot (I_2 + J_2)} \tag{5}$$

where $I_1 = i_1^q$, $J_1 = j_1^q$, $I_2 = i_2^q$, and $J_2 = j_2^q$. We will obtain a lower bound for this.

When $i_1 + j_1 \leq n$ and $i_2 + j_2 \leq n$, the denominator is no more than $4n^{2q}$. Letting $A = j_1 i_2$ and $B = j_2 i_1$, the numerator is $A^q - B^q$, where A and B are integers with $0 \leq A, B \leq n^2$. Either $A = B$, in which case the level sets have the same value, or else they differ by at least 1. If $q \leq 1$, by concavity and evaluating the derivative of x^q at n^2, $|A^q - B^q| \geq qn^{2q-2}$. Thus the absolute value of (5) is either 0 or at least $2qn^{2q-2}/4n^{2q} = \alpha(p)/n^{\beta(p)}$ for constants $\alpha(p)$ and $\beta(p)$. A similar results holds when $q > 1$, using the fact that x^q is convex and the numerator is at least $1^q - 0^q = 1$. $\square$

To convert an approximate isotonic regression into an exact one, suppose level set values of the exact isotonic regression f are in the range a to b and differ by at least δ. Let $S = \{s_1 < s_2 < \cdots < s_k\}$ be such that $s_1 \leq a$, $s_k \geq b$, and $s_{i+1} \leq s_i + \delta/2$ for $1 \leq i < k$. Let h be an S-valued isotonic regression of the data. Since consecutive values of S are sufficiently close, no two level sets of f of different mean will round to the same value of S, even if one rounds up and the other rounds down. Therefore for each vertex v of G, $f(v)$ is the L_p-mean of the level set of h containing v.

Theorem 5.6 *For $1 < p < \infty$, if data values are in $\{0, 1\}$ and weights are in $\{1, \ldots, W\}$, then for a DAG G of n vertices the exact L_p isotonic regression can be found in the time given in Theorem* 5.4, *where $\log K$ is replaced by $\log nW$. The implied constants depend on p.*

Further, if data values are in $\{0, \ldots, D\}$ and weights are in $\{1, \ldots, W\}$, then the exact L_2 isotonic regression can be found in the same time, where $\log K$ is replaced by $\log nDW$.

Proof Sets with $\leq n$ binary values with weights in $\{1, \ldots, W\}$ have an L_p mean contained in the L_p means of sets with $\leq nW$ binary values. Therefore means of such sets differ by at least $\alpha(p)/(nW)^{\beta(p)}$. If S is multiples of half of this, from 0 to 1, an optimal S-valued isotonic regression will be a sufficiently close approximation so that the true L_p mean of its level sets is the desired regression. The number of elements in S is $\Theta((nW)^{\beta(p)})$, so only $\Theta(\log nW)$ rounds of ϵ-scaling are needed to identify the level sets of the exact regression. Replacing their approximate value with the exact mean (4) finishes the calculation.

Extending the range of data values for L_p is complicated by the non-analytic form of the L_p mean. However, for L_2, if data values are in $\{0, \ldots, D\}$ and weights are in $\{1, \ldots, W\}$, then, by the same analysis as above, the means of unequal level sets differ by at least $1/(nW)^2$, independent of D, and their means are in the interval $[0, D]$. Thus at most $\Theta(\log nDW)$ iterations are required. $\square$

6 Multiple Values per Vertex

Several authors have considered the extension where there are multiple values at each vertex, i.e., at vertex v_i there is a set of weighted values $\{(y_{i,j}, w_{i,j}) : 1 \leq j \leq n_i\}$ for some $n_i \geq 1$, and the L_p isotonic regression problem, $1 \leq p < \infty$, is to find the
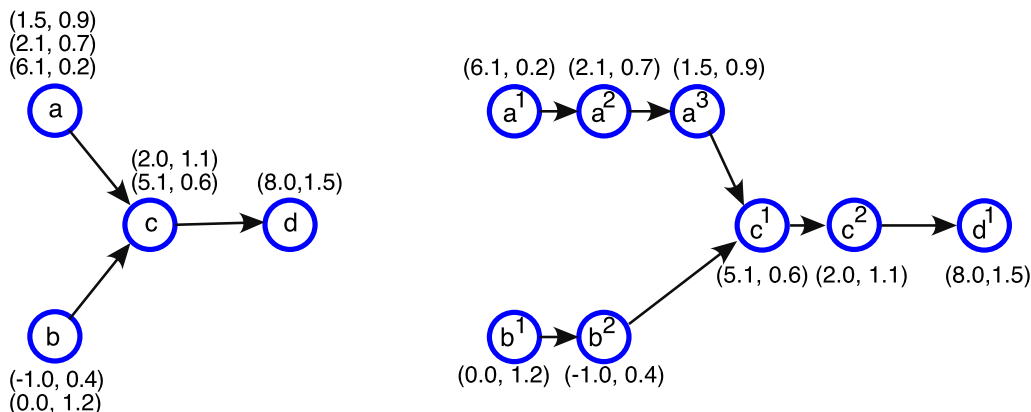
Fig. 3 Converting a DAG with multiple values per vertex into one with a single value per vertex

isotonic function **z** that minimizes

$$\left(\sum_{i=1}^{n} \sum_{j=1}^{n_i} w_{i,j} |y_{i,j} - z_i|^p \right)^{1/p} \tag{6}$$

among all isotonic functions. For L_2 this simplifies to single-valued isotonic regression where at each vertex the value is the weighted mean of the values and the weight is the sum of the weights, but this is no longer true when $p \neq 2$. Let $N = \sum_{i=1}^{n} n_i$.

Robertson and Wright [19] studied the statistical properties of L_1 isotonic regression with multiple values and two independent variables, but no algorithm for finding it was given. Chakravarti [5] gave a $\Theta(Nn)$ time algorithm for L_1 isotonic regression for a tree, and for a linear order Pardalos, Xue and Yong [16] gave one taking $\Theta(N \log^2 N)$ time, and asked if $\Theta(N \log N)$ was possible.

A simple approach to this problem is to transform it into one where there is a single value per vertex. Given a DAG G with multiple values per vertex, create a new DAG G' with one value per vertex as follows: if vertex $v_i \in G$ has k weighted values $(y_{i,j}, w_{i,j})$, $1 \leq j \leq k$, then in G' represent v_i with vertices $v_{i,j}$, $1 \leq j \leq k$; represent edge (v_i, v_ℓ) in G with edge $(v_{i,k}, v_{\ell,1})$ in G'; and add edges $(v_{i,j}, v_{i,j+1})$ to G' for $1 \leq j < k$. Vertex $v_{i,j}$ has the weighted value $(y'_{i,j}, w'_{i,j})$, where $y'_{i,j}$ is the jth data value at v_i in decreasing order and $w'_{i,j}$ is its corresponding weight. See Fig. 3. G' has N vertices and $m + N - n$ edges. An optimal isotonic regression on G' will have the same value for $v_{i,1}, \ldots, v_{i,k}$, and this is the value for $v_i \in G$ in an optimal solution to (6). The initial sorting at the vertices takes $\Theta(N \log N)$ time.

An advantage of this approach is that one can invoke whatever algorithm is desired on the resulting DAG. When the initial DAG was a tree the new DAG is as well, and thus the L_1 regression can be determined in $\Theta(N \log N)$ time. This answers the question of Pardalos et al. and improves upon Chakravarti's result.

For L_1 partitioning algorithms another approach is to merely use all N values for the partitioning, resulting in running times that are the same as in Theorem 5.4 where $\log K$ is replaced with $\log N$ and an $N \log N$ term is added, representing the initial sorting of the values and the evaluation of regression error throughout the algorithm. For large N, to reduce the time yet further this approach will be refined.

Theorem 6.1 *Given a DAG G with multiple weighted values per vertex, with N total values, an L_1 isotonic regression can be found in the same running time as in Theorem* 5.4, *where* $\log K$ *is replaced by* $\log N$ *and an* $(N + n \log N) \log \log N$ *term is added.*

Further, if G is fixed and only N varies then the regression can be found in $\Theta(N)$ *time.*

Proof It will be shown that $O(\log N)$ stages are needed and that the total time to initialize data structures, find the values used for the binary regressions, and evaluate the regression error, is $O((N + n \log N) \log \log N)$. This, plus the time to solve the binary regressions on the DAG, yields the results claimed.

For $V[a, b]$, a and b now refer to regression values a and b, rather than their ranks as in Sect. 2, since their global rank will not be known. The values in (a, b) at the vertices in $V[a, b]$ will be called *active values*. Rather than finding a median z of all values in (a, b), it suffices to choose an approximate median among the active values. This can be achieved by determining at each vertex in $V[a, b]$ the unweighted median of its active values and weighting this by the number of active values it has. Let z be the weighted median of these medians, breaking ties by choosing the smaller value, and let z' be the next largest value at vertices in $V[a, b]$ (if z is b then set z' to be b and z the next smallest value). No more than 3/4 of the active values in $V[a, b]$ can be in (a, z) and no more than 3/4 are in (z', b). Hence, no matter how the vertices are partitioned into $V[a, z]$ and $V[z', b]$, in each of them the number of active values is no more than 3/4 of those in $V[a, b]$, and thus there are at most $\Theta(\log N)$ iterations until there are no active values. A final partitioning step determines which vertices should have regression value a and which should be b.

Initially the values at vertex v_i are partially sorted into $\lg N$ "bags" of size $n_i / \lg N$, where the values in each bag are smaller than the values in the next. For each bag the smallest value, number of values, and sum of weights in the bag are determined. The bags are organized into a binary tree using their lowest value as the key. At each node of the tree is kept the sum of the weights and the number of values in the bags below. This can all be done in $\Theta(n_i \log \log N)$ time at v_i, and $\Theta(N \log \log N)$ time overall.

Finding the median of the vertex's active values, the next largest value above z, and the relative error of using z vs. z', can each be determined by a traversal through the tree and a linear time operation on at most two bags (e.g., to find the relative error of using z vs. z' one needs to determine it for values inside the bag where z would go, and then use the tree to determine it for values outside the bag). Thus the time at v_i at each stage is $\Theta(n_i / \log N + \log \log N)$, and $\Theta(N / \log N + n \log \log N)$ over all vertices. Since the number of stages is $\Theta(\log N)$, this proves the statement at the beginning of the proof.

To prove the claim concerning a fixed DAG and varying N no bags are used. Each stage involves multiple iterations, n per stage, where for each vertex the unweighted median of its active values is used for one of the iterations. After each stage the number of active values at a node is no more than half of what it was to begin with, and thus stage k takes $\Theta(n(T(G) + N/2^k))$ time, where $T(G)$ is the time to solve a binary-valued L_1 isotonic regression on G. Thus the total time is $\Theta(n(T(G) \log N + N))$, which is $\Theta(N)$ since G and n are fixed.   □

Table 2 Times of fastest known isotonic regression algorithms "semi-exact" is exact for $p = 2, 3, 4, 5$, and $\mathcal{T}_2 = \mathcal{T}_3 = \mathcal{T}_4 = \mathcal{T}_5 = 0$

Ordered set	Metric			
	L_1	L_p, $1 < p < \infty$, semi-exact	L_p, $1 < p < \infty$, δ approx	L_∞
Linear	$\Theta(n \log n)$ [1, 22]	$\Theta(n + \mathcal{T}_p)$, p even $\Theta(n \log n + \mathcal{T}_p)$, p odd $\Theta(n^2 + \mathcal{T}_p)$, p arb PAV, +	$\Theta(n \log K)$ [1]	$\Theta(n \log n)$ *
Tree	$\Theta(n \log n)$ +	$\Theta(n \log n + \mathcal{T}_p)$, p integer $\Theta(n^2 + \mathcal{T}_p)$, p arb [15] ($p = 2$), +	$\Theta(n \log K)$ +	$\Theta(n \log n)$ *
2-dim grid	$\Theta(n \log n)$ +	$\Theta(n^2 + \mathcal{T}_p)$ [21] ($p = 2$), +	$\Theta(n \log K)$ +	$\Theta(n \log n)$ *
2-dim arbitrary	$\Theta(n \log^2 n)$ +	$\Theta(n^2 \log n + \mathcal{T}_p)$ +	$\Theta(n \log n \log K)$ +	$\Theta(n \log^2 n)$ [24]
$d \geq 3$ dim grid	$\Theta(n^2 \log n)$ *	$\Theta(n^3 \log n + \mathcal{T}_p)$ *	$\Theta(n^2 \log n \log K)$ *	$\Theta(n \log n)$ *
$d \geq 3$ dim arbitrary	$\Theta(n^2 \log^d n)$ +	$\Theta(n^3 \log^d n + \mathcal{T}_p)$ +	$\Theta(n^2 \log^d n \log K)$ +	$\Theta(n \log^d n)$ [24]
Arbitrary	$\Theta(nm + n^2 \log n)$ [2]	$\Theta(n^2 m + n^3 \log n + \mathcal{T}_p)$ +	$\Theta((nm + n^2 \log n) \log K)$ +	$\Theta(m \log n)$ [24]

+ Shown here

* Implied by the result for arbitrary ordered sets

$K = (\max_{i=1}^n y_i - \min_{i=1}^n y_i)/\delta$

$\mathcal{T}_p$ Total data-dependent time to determine L_p means

7 Final Comments

Table 2 is an update of Table 1, incorporating the results in this paper. Results for L_p were not included in Table 1 since polynomial time algorithms had only appeared for linear orderings [1, 7, 26]. The semi-exact algorithms, and the exact L_2 algorithms in [14, 21], have a worst-case behavior where each partitioning stage results in one piece having only a single vertex. Partitioning using the mean probably works well in practice, but a natural question is whether the worst-case time can be reduced by a factor of $\tilde{\Theta}(n)$.

Partitioning can also improve the time for L_1 isotonic regression for unweighted data on an arbitrary DAG $G = (V, E)$. Partitioning on values $a < b$ is equivalent to finding a minimum vertex cover on the directed bipartite graph $H = (V, E')$ with vertex partition V_1, V_2, where V_1 is those vertices with data $\leq a$ and V_2 is the vertices with data $\geq b$. There is an edge $(v_1, v_2) \in E'$, with $v_1 \in V_1$ and $v_2 \in V_2$, iff $v_2 \prec v_1$ in G, i.e., iff they violate the isotonic constraint. Vertices in the minimum cover correspond to those where the value of the $\{a, b\}$-valued regression is the further of a or b, instead of the closer. A simple analysis using the fact that the cover is minimal shows that no new violating pairs are introduced. For unweighted bipartite graphs the maximum matching algorithm of Hopcroft and Karp takes $\Theta(E\sqrt{V})$ time,

from which a minimum vertex cover can be constructed in linear time. H can be constructed in linear time from the transitive closure of G, and since the transitive closure can be found in $O(n^{2.376})$ time (albeit with infeasible constants), the total time over all $\Theta(\log n)$ partitioning stages is $\Theta(n^{2.5} \log n)$. For multidimensional DAGs this can be reduced to $\tilde{\Theta}(n^{1.5})$ [25]. However, this approach does not help for L_p regression since the binary L_1 regression problems generated are weighted even when the data is unweighted.

The L_1 median, and hence L_1 isotonic regression, is not always unique, and a natural question is whether there is a "best" one. A similar situation occurs for L_∞ isotonic regression. Algorithms have been developed for strict L_∞ isotonic regression [23, 25], which is the limit, as $p \to \infty$, of L_p isotonic regression. This is also known as "best best" L_∞ regression [13]. Analogously one can define strict L_1 isotonic regression to be the limit, as $p \to 1^+$, of L_p isotonic regression. It can be shown to be well defined, and it is straightforward to derive the appropriate median for a level set [11]. However, like L_p means, this median cannot always be computed exactly. Apparently no algorithms have yet been developed for strict L_1 isotonic regression, though PAV is still applicable for linear orders and trees.

One can view Proposition 3.2, concerning complete bipartite graphs, and Sect. 6, concerning multiple values per vertex, as being related. The complete bipartite graph partitions the vertices into sets P_1 and P_2. Viewing P_1 and P_2 as two nodes in a DAG with an edge from P_1 to P_2, then the isotonic requirement in Proposition 3.2 is that no element in P_1 has a regression value greater than the regression value of any element of P_2, but there are no constraints among elements of P_1 nor among elements of P_2. In contrast, the model in Sect. 6 corresponds to requiring that every element in P_1 has the same regression value, and similarly for P_2. The interpretation of multiple values per vertex as partitioning elements, with no constraints among elements in the same partition, can be extended to arbitrary DAGs, resulting in each node of the DAG having an interval of regression values, rather than a single one. A technique similar to that in Sect. 6 can be used to convert this problem into a standard isotonic regression with one value per vertex, but the special structure can likely be exploited to produce faster algorithms.

Finally, the classification problem called *isotonic separation* [6] or isotonic *minimal reassignment* [8] is an isotonic regression problem. In the simplest case, suppose there are two types of objects, red and blue, and at each vertex there is an observation of a red or blue object. There is a penalty for misclassifying a red object as blue, and a perhaps different penalty for the opposite error. The goal is to minimize the sum of the misclassification penalties given that red $<$ blue and the classification must be isotonic. This is a straightforward binary-valued L_1 isotonic regression problem. In the general case with k ordered categories there is a penalty for misclassifying an object of type i as being of type j. Unfortunately, penalties as simple as 0–1 correct/incorrect do not satisfy the partitioning property in Lemma 2.1. However, for trees, the dynamic programming approach used in Sect. 3 can be modified to find an optimal isotonic separation in a single pass taking $\Theta(kn)$ time. The time required for arbitrary DAGs is unknown.

Appendix: Computing the L_p Mean

The computation of the L_p-mean of a level set for general p is more complicated than for $p = 1, 2$, or ∞. We are interested in identifying those aspects which might be more than linear in the time of the remainder of the algorithm. We assume that the data values have been presorted.

Simple calculus [11] shows that the L_p mean of the data $(\mathbf{y}, \mathbf{w})$ is the unique C such that

$$\sum_{y_i > C} w_i (y_i - C)^{p-1} = \sum_{y_j < C} w_j (C - y_j)^{p-1} \tag{7}$$

which does not, in general, have an analytic solution. An important aspect is to determine the data values that bracket the L_p mean, i.e., to determine which are in the RHS and which are on the LHS of (7).

Given the bracketing values, the time to find the mean to desired accuracy may depend upon the data and we make no assumptions concerning it. Given p, n, and data $(\mathbf{y}, \mathbf{w})$, the total time to find all L_p means used in the algorithm is decomposed into a part independent of the data and the remaining portion, denoted $\mathcal{T}_p(n, \mathbf{y}, \mathbf{w})$, that is data dependent. For general p the data independent portion requires evaluating (7) for each set at least once per partitioning stage, and thus $\Theta(n)$ time per stage. Since there can be $\Theta(n)$ stages in each of the algorithms in Sect. 5, the data independent time for finding means is $\Theta(n^2)$ once the bracketing values are known. By first partitioning on data values, all of the partitioning algorithms can determine the bracketing values in time no more than the remaining time of the algorithm.

If p is an even integer, (7) reduces to finding the unique solution of $\sum_{i=1,n} w_i (y_i - C)^{p-1} = 0$, i.e., no bracketing values nor presorting are required. This is a sum of $(p-1)$st degree polynomials, and hence can be determined by summing up, over the data values, the coefficients of each term and then evaluating the result. For p an odd integer once again the terms are polynomials so, given the bracketing values, one can use sums of coefficients to evaluate the summations in (7). Since roots of polynomials of degree ≤ 4 can be found exactly in constant time, $\mathcal{T}_2 = \mathcal{T}_3 = \mathcal{T}_4 = \mathcal{T}_5 = 0$.

For PAV sets are merged rather than partitioned. This does not change the data independent time for general p, but for p an even integer each set is reduced to $p + 1$ coefficients and merging sets merely requires adding coefficients. Hence the data independent time is $\Theta(n)$. For p odd the situation is a bit more complex since the bracketing values are needed. One can construct a balanced search tree where data values are keys and at each node, for each term there is the sum of the coefficients beneath it. The only change from being on the LHS vs. the RHS of (7) is whether the term has a plus or minus sign, and hence, given this tree, one can descend through the tree and determine the bracketing data values in logarithmic time. Merging sets corresponds to merging their trees, including maintaining the information about sums of coefficients in subtrees. With careful merging this can be accomplished in $\Theta(n \log n)$ total time.

References

1. Ahuja, R.K., Orlin, J.B.: A fast scaling algorithm for minimizing separable convex functions subject to chain constraints. Oper. Res. **49**, 784–789 (2001)
2. 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)
3. Barlow, R.E., Bartholomew, D.J., Bremner, J.M., Brunk, H.D.: Statistical Inference Under Order Restrictions. Wiley, New York (1972)
4. Brunk, H.D.: Maximum likelihood estimates of monotone parameters. Ann. Math. Stat. **26**, 607–616 (1955)
5. Chakravarti, N.: Isotonic median regression for orders represented by rooted trees. Nav. Res. Logist. **39**, 599–611 (1992)
6. Chandrasekaran, R., Rhy, Y.U., Jacob, V.S., Hong, S.: Isotonic separation. INFORMS J. Comput. **17**, 462–474 (2005)
7. Chepoi, V., Cogneau, D., Fichet, B.: Polynomial algorithms for isotonic regression. In: L_1 Statistical Procedures and Related Topics. Lecture Notes—Monograph Series, vol. 31, pp. 147–160 (1967). Institute of Mathematical Statistics
8. 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 of Knowledge Discovery in Databases. Springer Lecture Notes in Computer Science, vol. 4702, pp. 164–175 (2007)
9. Dykstra, R.L., Robertson, T.: An algorithm for isotonic regression of two or more independent variables. Ann. Stat. **10**, 708–716 (1982)
10. Gebhardt, F.: An algorithm for monotone regression with one or more independent variables. Biometrika **57**, 263–271 (1970)
11. Jackson, D.: Note on the median of a set of numbers. Bull. Am. Math. Soc. **27**, 160–164 (1921)
12. Kaufman, Y., Tamir, A.: Locating service centers with precedence constraints. Discrete Appl. Math. **47**, 251–261 (1993)
13. Legg, D., Townsend, D.: Best monotone approximation in $L_\infty[0, 1]$. J. Approx. Theory **42**, 30–35 (1984)
14. Maxwell, W.L., Muckstadt, J.A.: Establishing consistent and realistic reorder intervals in production-distribution systems. Oper. Res. **33**, 1316–1341 (1985)
15. Pardalos, P.M., Xue, G.: Algorithms for a class of isotonic regression problems. Algorithmica **23**, 211–222 (1999)
16. Pardalos, P.M., Xue, G.-L., Yong, L.: Efficient computation of an isotonic median regression. Appl. Math. Lett. **8**, 67–70 (1995)
17. Punera, K., Ghosh, J.: Enhanced hierarchical classification via isotonic smoothing. In: Proceedings of the International Conference on the World Wide Web 2008, pp. 151–160 (2008)
18. Qian, S.: An algorithm for tree-ordered isotonic median regression. Stat. Probab. Lett. **27**, 195–199 (1996)
19. Robertson, T., Wright, F.T.: Multiple isotonic median regression. Ann. Stat. **1**, 422–432 (1973)
20. Robertson, T., Wright, F.T., Dykstra, R.L.: Order Restricted Statistical Inference. Wiley, New York (1988)
21. Spouge, J., Wan, H., Wilbur, W.J.: Least squares isotonic regression in two dimensions. J. Optim. Theory Appl. **117**, 585–605 (2003)
22. Stout, Q.F.: Unimodal regression via prefix isotonic regression. Comput. Stat. Data Anal. **53**, 289–297 (2008). A preliminary version appeared in "Optimal algorithms for unimodal regression". Comput. Stat. **32** (2000)
23. Stout, Q.F.: Strict L_∞ isotonic regression. J. Optim. Theory Appl. **152**, 121–135 (2012)
24. Stout, Q.F.: Weighted L_∞ isotonic regression (2012, submitted)
25. Stout, Q.F.: Isotonic regression for multiple independent variables (2012, submitted)
26. Strömberg, U.: An algorithm for isotonic regression with arbitrary convex distance function. Comput. Stat. Data Anal. **11**, 205–219 (1991)
27. Thompson, W.A. Jr.: The problem of negative estimates of variance components. Ann. Math. Stat. **33**, 273–289 (1962)