Lipschitz Unimodal and Isotonic Regression on Paths and Trees

Pankaj K. Agarwal[1], Jeff M. Phillips[2], and Bardia Sadri[3]

[1] Duke University [2] University of Utah [3] University of Toronto

Abstract. We describe algorithms for finding the regression of t, a sequence of values, to the closest sequence s by mean squared error, so that s is always increasing (isotonicity) and so the values of two consecutive points do not increase by too much (Lipschitz). The isotonicity constraint can be replaced with a unimodular constraint, for exactly one local maximum in s. These algorithm are generalized from sequences of values to trees of values. For each we describe near-linear time algorithms.

1 Introduction

Let $\mathbb{M}$ be a triangulation of a polygonal region $P \subseteq \mathbb{R}^2$ in which each vertex is associated with a real valued height (or elevation)[1]. Linear interpolation of vertex heights in the interior of each triangle of $\mathbb{M}$ defines a piecewise-linear function $t : P \to \mathbb{R}$, called a *height function*. A height function (or its graph) is widely used to model a two-dimensional surface in numerous applications (e.g. modeling the terrain of a geographical area). With recent advances in sensing and mapping technologies, such models are being generated at an unprecedentedly large scale. These models are then used to analyze the surface and to compute various geometric and topological properties of the surface. For example, researchers in GIS are interested in extracting river networks or computing visibility or shortest-path maps on terrains modeled as height functions. These structures depend heavily on the topology of the level sets of the surface and in particular on the topological relationship between its critical points (maxima, minima, and saddle points). Because of various factors such as measurement or sampling errors or the nature of the sampled surface, there is often plenty of noise in these surface models which introduces spurious critical points. This in turn leads to misleading or undesirable artifacts in the computed structures, e.g., artificial breaks in river networks. These difficulties have motivated extensive work on topological simplification and noise removal through modification of the height function t into another one $s : \mathbb{M} \to \mathbb{R}$ that has the desired set of critical points and that is as close to t as possible [8, 20, 23]. A popular approach is to decompose the surface into pieces and modify each piece so that it has a unique minimum or maximum [23]. In some applications, it is also desirable to impose the additional constraint that the function s is Lipschitz; see below for further discussion.

[1] Supported by grants NSF: CNS-05-40347, CFF-06-35000, DEB-04-25465, 0937060-CIF32 to CRA; ARO: W911NF-04-1-0278, W911NF-07-1-0376; NIH: 1P50-GM-08183-01; DOE: OEG-P200A070505; and U.S.–Israel Binational Science Foundation.

Problem statement. Let $\mathbb{M} = (V, A)$ be a planar graph with vertex set V and arc (edge) set $A \subseteq V \times V$. We may treat $\mathbb{M}$ as undirected in which case we take the pairs (u, v) and (v, u) as both representing the same undirected edge connecting u and v. Let $\gamma \geq 0$ be a real parameter. A function $s : V \to \mathbb{R}$ is called

(L) γ-*Lipschitz* if $(u, v) \in A$ implies $s(v) - s(u) \leq \gamma$.

Note that if $\mathbb{M}$ is undirected, then Lipschitz constraint on an edge $(u, v) \in A$ implies $|s(u) - s(v)| \leq \gamma$. For an undirected planar graph $\mathbb{M} = (V, A)$, a function $s : V \to \mathbb{R}$ is called

(U) *unimodal* if s has a unique *local maximum*, i.e. only one vertex $v \in V$ such that $s(v) > s(u)$ for all $(u, v) \in A$.

For a directed planar graph $\mathbb{M} = (V, A)$, a function $s : V \to \mathbb{R}$ is called

(I) *isotonic* if $(u, v) \in A$ implies $s(u) \leq s(v)$.[2]

For an arbitrary function $t : V \to \mathbb{R}$ and a parameter γ, the γ-*Lipschitz unimodal regression (γ-LUR)* of t is a function $s : V \to \mathbb{R}$ that is γ-Lipschitz and unimodal on $\mathbb{M}$ and minimizes $\|s - t\|^2 = \sum_{v \in V}(s(v) - t(v))^2$. Similarly, if $\mathbb{M}$ is a directed planar graph, then s is the γ-*Lipschitz isotonic regression (γ-LIR)* of t if s satisfies (L) and (I) and minimizes $\|s - t\|^2$. The more commonly studied *isotonic regression (IR)* and *unimodal regression* [3, 5, 7, 22] are the special cases of LIR and LUR, respectively, in which $\gamma = \infty$, and therefore only the condition (I) or (U) is enforced.

Given a planar graph $\mathbb{M}$, a parameter γ, and $t : V \to \mathbb{R}$, the LIR (resp. LUR) problem is to compute the γ-LIR (resp. γ-LUR) of t. In this paper we propose near-linear-time algorithms for the LIR and LUR problems for two special cases: when $\mathbb{M}$ is a path or a tree. We study the special case where $\mathbb{M}$ is a path prior to the more general case where it is tree because of the difference in running time and because doing so simplifies the exposition to the more general case.

Related work. As mentioned above, there is extensive work on simplifying the topology of a height function while preserving the geometry as much as possible. Two widely used approaches in GIS are the so-called *flooding* and *carving* techniques [1, 11, 23]. The former technique raises the height of the vertices in "shallow pits" to simulate the effect of flooding, while the latter lowers the value of the height function along a path connecting two pits so that the values along the path vary monotonically. As a result, one pit drains to the other and thus one of the minima ceases to exist. Various methods based on Laplacian smoothing have been proposed in the geometric modeling community to remove unnecessary critical points; see [6, 17, 20] and references therein.

[2] A function s satisfying the isotonicity constraint (I) must assign the same value to all the vertices of a directed cycle of $\mathbb{M}$ (and indeed to all vertices in the same strongly connected component). Therefore, without loss of generality, we assume $\mathbb{M}$ to be a directed *acyclic* graph (DAG).

A prominent line of research on topological simplification was initiated by Edelsbrunner et. al. [13] who introduced the notion of *persistence*; see also [12, 28]. Roughly speaking, each homology class of the contours in sublevel sets of a height function is characterized by two critical points at one of whom the class is born and at the other it is destroyed. The persistence of this class is then the height difference between these two critical points. Efficient algorithms [13, 14, 4] simplify topology based on persistence, optimally eliminating all critical points of persistence below a threshold measured as $\|s - t\|_\infty = \max_{v \in V} |s(v) - t(v)|$. No efficient algorithm is known to minimize $\|s - t\|_2$.

The isotonic-regression (IR) problem has been studied in statistics [5, 7, 22] since the 1950s. It has many applications ranging from statistics [25] to bioinformatics [7], and from operations research [19] to differential optimization [16]. The *pool adjacent violator algorithm* (PAVA) [5] solves the IR problem on paths in $O(n)$ time, by merging consecutive level sets of vertex values that are out of order. Brunk [10] and Thompson [27] initiated the study of the IR problem on general DAGs and trees, respectively. Jewel [18] introduced the problem of Lipschitz isotonic regression on DAGs and showed connections between this problem and network flow with quadratic cost functions. Stout [26] solves the UR problem on paths in $O(n)$ time. Pardalos and Xue [21] give an $O(n \log n)$ algorithm for the IR problem on trees. For the special case when the tree is a star they give an $O(n)$ algorithm. Spouge et. al.[24] give an $O(n^4)$ time algorithm for the IR problem on DAGs. The problems can be solved under the L_1 and L_∞ norms on paths [26] and DAGs [3] as well, with an additional $\log n$ factor for L_1. To our knowledge there is no prior work on efficient algorithms for Lipschitz isotonic/unimodal regressions in the literature.

Our results. We present efficient exact algorithms for LIR and LUR problems on two special cases of planar graphs: paths and trees. In particular, we present an $O(n \log n)$ algorithm for computing the LIR on a path of length n (Section 4), and an $O(n \log n)$ algorithm on a tree with n nodes (Section 6). We present an $O(n \log^2 n)$ algorithm for computing the LUR problem on a path of length n (Section 5). Our algorithm can be extended to solve the LUR problem on an unrooted tree in $O(n \log^3 n)$ time (Section 7). The LUR algorithm for a tree is particularly interesting because of its application in the aforementioned carving technique [11, 23]. The carving technique modifies the height function along a number of trees embedded on the terrain where the heights of the vertices of each tree are to be changed to vary monotonically towards a chosen "root" for that tree. In other words, to perform the carving, we need to solve the IR problem on each tree. The downside of doing so is that the optimal IR solution happens to be a step function along each path toward the root of the tree with potentially large jumps. Enforcing the Lipschitz condition prevents sharp jumps in function value and thus provides a more natural solution to the problem.

Section 3 presents a data structure, called *affine composition tree* (ACT), for maintaining a xy-monotone polygonal chain, which can be regarded as the graph of a monotone piecewise-linear function $F : \mathbb{R} \to \mathbb{R}$. Besides being crucial for our algorithms, ACT is interesting in its own right. A special kind of binary search

tree, an ACT supports a number of operations to query and update the chain, each taking $O(\log n)$ time. Besides the classical insertion, deletion, and query (computing $F(x)$ or $F^{-1}(x)$ for a given $x \in \mathbb{R}$), one can apply an INTERVAL operation that modifies a contiguous subchain provided that the chain remains x-monotone after the transformation, i.e., it remains the graph of a monotone function. For space, several proofs are missing; see the full version [2].

2 Energy Functions

On a discrete set U, a real valued function $s : U \to \mathbb{R}$ can be viewed as a point in the $|U|$-dimensional Euclidean space in which coordinates are indexed by the elements of U and the component s_u of s associated to an element $u \in U$ is $s(u)$. We use the notation $\mathbb{R}^U$ to represent the set of all real-valued functions defined on U.

Let $\mathbb{M} = (V, A)$ be a directed acyclic graph on which we wish to compute γ-Lipschitz isotonic regression of an *input function* $t \in \mathbb{R}^V$. For any set of vertices $U \subseteq V$, let $\mathbb{M}[U]$ denote the subgraph of $\mathbb{M}$ induced by U, i.e. the graph $(U, A[U])$, where $A[U] = \{(u, v) \in A : u, v \in U\}$. The set of γ-Lipschitz isotonic functions on the subgraph $\mathbb{M}[U]$ of $\mathbb{M}$ constitutes a convex subset of $\mathbb{R}^U$, denoted by $\Gamma(\mathbb{M}, U)$. It is the common intersection of all half-spaces determined by the isotonicity and Lipschitz constraints associated with the edges in $A[U]$, i.e., $s_u \leq s_v$ and $s_v \leq s_u + \gamma$ for all $(u, v) \in A[U]$.

For $U \subseteq V$ we define $E_U : \mathbb{R}^V \to \mathbb{R}$ as $E_U(s) = \sum_{v \in U}(s(v) - t(v))^2$. The γ-Lipschitz isotonic regression of the input function t is $\sigma = \arg\min_{s \in \Gamma(\mathbb{M}, V)} E_V(s)$. For a subset $U \subseteq V$ and $v \in U$ define the function $E^v_{\mathbb{M}[U]} : \mathbb{R} \to \mathbb{R}$ as

$$E^v_{\mathbb{M}[U]}(x) = \min_{s \in \Gamma(\mathbb{M}, U); s(v) = x} E_U(s).$$

Lemma 1. *For any $U \subseteq V$ and $v \in U$, the function $E^v_{\mathbb{M}[U]}$ is continuous and strictly convex.*

3 Affine Composition Tree

In this section we introduce a data structure, called *affine composition tree* (ACT), for representing an xy-monotone polygonal chain in $\mathbb{R}^2$, which is being deformed dynamically. Such a chain can be regarded as the graph of a piecewise-linear monotone function $F : \mathbb{R} \to \mathbb{R}$, and thus is bijective. A *breakpoint* of F is the x-coordinate of a *vertex* of the graph of F (a vertex of F for short), i.e., a $b \in \mathbb{R}$ at which the left and right derivatives of F disagree. The number of breakpoints of F will be denoted by $|F|$. A continuous piecewise-linear function F with breakpoints $b_1 < \cdots < b_n$ can be characterized by its vertices $q_i = (b_i, F(b_i)), i = 1, \ldots, n$ together with the *slopes* μ_- and μ_+ of its *left* and *right unbounded pieces*, respectively, extending to $-\infty$ and $+\infty$. An *affine transformation* of $\mathbb{R}^2$ is a map $\phi : \mathbb{R}^2 \to \mathbb{R}^2$, $q \mapsto M \cdot q + c$ where M is a nonsingular

2×2 matrix, (a *linear transformation*) and $c \in \mathbb{R}^2$ is a *translation vector* — in our notation we treat $q \in \mathbb{R}^2$ as a column vector.

An ACT supports the following operations on a monotone piecewise-linear function F with vertices $q_i = (b_i, F(b_i)), i = 1, \ldots, n$:

1. EVALUATE(a) and EVALUATE^{-1}(a): Given any $a \in \mathbb{R}$, return $F(a)$ or $F^{-1}(a)$.
2. INSERT(q): Given a point $q = (x, y) \in \mathbb{R}^2$ insert q as a new vertex of F. If $x \in (b_i, b_{i+1})$, this operation removes the segment $q_i q_{i+1}$ from the graph of F and replaces it with two segments $q_i q$ and $q q_{i+1}$, thus making x a new breakpoint of F with $F(x) = y$. If $x < b_1$ or $x > b_k$, then the affected unbounded piece of F is replaced with one parallel to it but ending at q and a segment connecting q and the appropriate vertex of F (μ_+ and μ_- remain intact). We assume that $F(b_i) \le y \le F(b_{i+1})$.
3. DELETE(b): Given a breakpoint b of F, removes the vertex $(b, F(b))$ from F; a delete operation modifies F in a manner similar to insert.
4. AFFINE(ψ): Given an affine transformation $\psi : \mathbb{R}^2 \to \mathbb{R}^2$, modify the function F to one whose graph is the result of application of ψ to the graph of F. See Figure 1(Left).
5. INTERVAL$_p$(ψ, τ^-, τ^+): Given an affine transformation $\psi : \mathbb{R}^2 \to \mathbb{R}^2$ and $\tau^-, \tau^+ \in \mathbb{R}$ and $p \in \{x, y\}$, this operation applies ψ to all vertices v of F whose p-coordinate is in the range $[\tau^-, \tau^+]$. Note that AFFINE(ψ) is equivalent to INTERVAL$_p$($\psi, -\infty, +\infty$) for $p = \{x, y\}$. See Figure 1(Right).

AFFINE and INTERVAL are applied with appropriate choice of transformation parameters so that the resulting chain remains xy-monotone.

An ACT $\mathcal{T} = \mathcal{T}(F)$ is a red-black tree that stores the vertices of F in the sorted order, i.e., the ith node of $\mathcal{T}$ is associated with the ith vertex of F. However, instead of storing the actual coordinates of a vertex, each node z of $\mathcal{T}$ stores an affine transformation $\phi_z : q \mapsto M_z \cdot q + c_z$. If $z_0, \ldots, z_k = z$ is the path in $\mathcal{T}$ from the root z_0 to z, then let $\Phi_z(q) = \phi_{z_0}(\phi_{z_1}(\ldots \phi_{z_k}(q) \ldots))$. Notice that Φ_z is also an affine transformation. The actual coordinates of the vertex associated with z are $(q_x, q_y) = \Phi_z(\overline{0})$ where $\overline{0} = (0, 0)$.

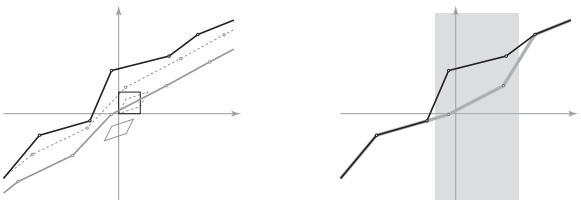

Fig. 1: Left: the graph of a monotone piecewise linear function F (in solid black). The dashed curve is the result of applying the linear transform $\psi_0(q) = Mq$ where $M = \left(\begin{smallmatrix} 1 & 1/3 \\ 1/3 & 2/3 \end{smallmatrix}\right)$. The gray curve, a translation of the dashed curve under vector $c = \left(\begin{smallmatrix} -1 \\ -2 \end{smallmatrix}\right)$, is the result of applying AFFINE(ψ) to F where $\psi(q) = Mq + c$. Right: INTERVAL(ψ, τ^-, τ^+), only the vertices of curve F whose x-coordinates are in the marked interval $[\tau^-, \tau^+]$ are transformed. The resulting curve is the thick gray curve.

Given a value $b \in \mathbb{R}$ and $p \in \{x, y\}$, let $\textsc{Pred}_p(b)$ (resp. $\textsc{Succ}_p(b)$) denote the rightmost (resp. leftmost) vertex q of F such that the p-coordinate of q is at most (resp. least) b. Using ACT $\mathcal{T}$, $\textsc{Pred}_p(b)$ and $\textsc{Succ}_p(b)$ can be computed in $O(\log n)$ time by following a path in $\mathcal{T}$, composing the affine transformations along the path, evaluating the result at $\overline{0}$, and comparing its p-coordinate with b. $\textsc{Evaluate}(a)$ determines the vertices $q^- = \textsc{Pred}_x(a)$ and $q^+ = \textsc{Succ}_x(a)$ of F immediately preceding and succeeding a and interpolates F linearly between q^- and q^+; if $a < b_1$ (resp. $a > b_k$), then $F(a)$ is calculated using $F(b_1)$ and μ_- (resp. $F(b_k)$ and μ_+). Since b^- and b^+ can be computed in $O(\log n)$ time and the interpolation takes constant time, $\textsc{Evaluate}(a)$ is answered in time $O(\log n)$. Similarly, $\textsc{Evaluate}^{-1}(a)$ is answered using $\textsc{Pred}_y(a)$ and $\textsc{Succ}_y(a)$.

A key observation of ACT is that a standard rotation on any edge of $\mathcal{T}$ can be performed in $O(1)$ time by modifying the stored affine transformations in a constant number of nodes (see Figure 2(left)) based on the fact that an affine transformation $\phi : q \mapsto M \cdot q + c$ has an inverse affine transformation $\phi^{-1} : q \mapsto M^{-1} \cdot (q - c)$; provided that the matrix M is invertible. A point $q \in \mathbb{R}^2$ is inserted into $\mathcal{T}$ by first computing the affine transformation Φ_u for the node u that will be the parent of the leaf z storing q. To determine ϕ_z we solve, in constant time, the system of (two) linear equations $\Phi_u(\phi_z(\overline{0})) = q$. The result is the translation vector c_z. The linear transformation M_z can be chosen to be an arbitrary invertible linear transformation, but for simplicity, we set M_z to the identity matrix. Deletion of a node is handled similarly.

To perform an $\textsc{Interval}_p(\psi, \tau^-, \tau^+)$ query, we first find the nodes z^- and z^+ storing the vertices $\textsc{Pred}_p(\tau^-)$ and $\textsc{Succ}_p(\tau^+)$, respectively. We then successively rotate z^- with its parent until it becomes the root of the tree. Next, we do the same with z^+ but stop when it becomes the right child of z^-. At this stage, the subtree rooted at the left child of z^+ contains exactly all the vertices q for which $q_p \in [\tau^-, \tau^+]$. Thus we compose ψ with the affine transformation at that node and issue the performed rotations in the reverse order to put the tree back in its original (balanced) position. Since z^- and z^+ were both within $O(\log n)$ steps from the root of the tree, and since performing each rotation on the tree can only increase the depth of a node by one, z^- is taken to the root in

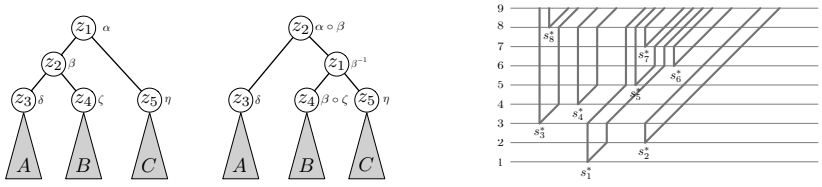

Fig. 2: Left: Rotation in affine composition trees, with each node's affine function in greek. When rotating (z_1, z_2), changing the functions as shown the leaves remain unchanged. Right: The breakpoints of the function $F_i = d\tilde{E}_i/dx$. For each i, s^*_{i-1} and $s^*_{i-1} + \gamma$ are the "new" breakpoints of F_i. All other breakpoints of F_i come from F_{i-1} where those smaller than s^*_i remain unchanged and those larger are increased by γ.

$O(\log n)$ steps and this increases the depth of z^+ by at most $O(\log n)$. Thus the whole operation takes $O(\log n)$ time.

We can augment $\mathcal{T}(F)$ with additional information so that for any $a \in \mathbb{R}$ the function $E(a) = E(b_1) + \int_{b_1}^{a} F(x)\,dx$, where b_1 is the leftmost breakpoint and $E(b_1)$ is value associated with b_1, can be computed in $O(\log n)$ time; we refer to this operation as INTEGRATE(a). We provide the details in the full version [2].

Theorem 1. *A continuous piecewise-linear monotonically increasing function F with n breakpoints can be maintained using a data structure $\mathcal{T}(F)$ such that*

1. EVALUATE *and* EVALUATE^{-1} *queries can be answered in $O(\log n)$ time,*
2. *an* INSERT *or a* DELETE *operation can be performed in $O(\log n)$ time,*
3. AFFINE *and* INTERVAL *operations can be performed in $O(1)$ and $O(\log n)$ time, respectively.*
4. INTEGRATE *operation can be performed in $O(\log n)$ time.*

One can use the above operations to compute the sum of two increasing continuous piecewise-linear functions F and G as follows: we first compute $F(b_i)$ for every breakpoint b_i of G and insert the pair $(b_i, F(b_i))$ into $\mathcal{T}(F)$. At this point the tree still represents $\mathcal{T}(F)$ but includes all the breakpoints of G as *degenerate* breakpoints (at which the left and right derivates of F are the same). Finally, for every consecutive pair of breakpoints b_i and b_{i+1} of G we apply an INTERVAL$_x(\psi_i, b_i, b_{i+1})$ operation on $\mathcal{T}(F)$ where ψ_i is the affine transformation $q \mapsto Mq + c$ where $M = \left(\begin{smallmatrix} 1 & 0 \\ \alpha & 1 \end{smallmatrix}\right)$ and $c = \left(\begin{smallmatrix} 0 \\ \beta \end{smallmatrix}\right)$, in which $G_i(x) = \alpha x + \beta$ is the linear function that interpolates between $G(b_i)$ at b_i and $G(b_{i+1})$ at b_{i+1} (similar operation using μ_- and μ_+ of G for the unbounded pieces of G must can be applied in constant time). It is easy to verify that after performing this series of INTERVAL's, $\mathcal{T}(F)$ turns into $\mathcal{T}(F + G)$. The total running time of this operation is $O(|G| \log |F|)$. Note that this runtime can be reduced to $O(|G|(1 + \log |F| / \log |G|))$, for $|G| < |F|$, by using an algorithm of Brown and Tarjan [9] to insert all breakpoints and then applying all INTERVAL operations in a bottom up manner. Furthermore, this process can be reversed (i.e. creating $\mathcal{T}(F - G)$ without the breakpoints of G, given $\mathcal{T}(F)$ and $\mathcal{T}(G)$) in the same runtime. We therefore have shown:

Lemma 2. *Given $\mathcal{T}(F)$ and $\mathcal{T}(G)$ for a piecewise-linear isotonic functions F and G where $|G| < |F|$, $\mathcal{T}(F + G)$ or $\mathcal{T}(F - G)$ can computed in $O(|G|(1 + \log |F| / \log |G|))$ time.*

Tree sets. We will be repeatedly computing the sum of two functions F and G. It will be too expensive to compute $\mathcal{T}(F + G)$ explicitly using Lemma 2, therefore we represent it implicitly. More precisely, we use a *tree set* $\mathcal{S}(F) = \{\mathcal{T}(F_1), \ldots, \mathcal{T}(F_k)\}$ consisting of affine composition trees of monotone piecewise-linear functions $F_1, \ldots, F_k$ to represent the function $F = \sum_{j=1}^{k} F_j$. We perform several operations on F or $\mathcal{S}$ similar to those of a single affine composition tree.

EVALUATE(x) on F takes $O(k \log n)$ time, by evaluating $\sum_{j=1}^{k} F_j(x)$. And EVALUATE$^{-1}(y)$ on F takes $O(k \log^2 n)$ time using Frederickson and Johnson [15].

Given the ACT $\mathcal{T}(F_0)$, we can convert $\mathcal{S}(F)$ to $\mathcal{S}(F + F_0)$ in two ways: an INCLUDE$(\mathcal{S}, F_0)$ operation sets $\mathcal{S} = \{\mathcal{T}(F_1), \ldots, \mathcal{T}(F_k), \mathcal{T}(F_0)\}$ in $O(1)$ time. A MERGE$(\mathcal{S}, F_0)$ operations sets $\mathcal{S} = \{\mathcal{T}(F_1 + F_0), \mathcal{T}(F_2), \ldots, \mathcal{T}(F_k)\}$ in time $O(|F_0| \log |F_1|)$. We can also perform UNINCLUDE$(\mathcal{S}, F_0)$ and UNMERGE$(\mathcal{S}, F_0)$, operations that reverse the respective above operations in the same runtimes.

We can perform an AFFINE$(\mathcal{S}, \psi)$ where ψ describes a linear transform M and a translation vector c. To update F by ψ we update F_1 by ψ and for $j \in [2, k]$ update F_j by just M. This takes $O(k)$ time. It follows that we can perform INTERVAL$(\mathcal{S}, \psi, \tau^-, \tau^+)$ in $O(k \log n)$ time, where $n = |F_1| + \ldots + |F_k|$. Here we assume that the transformation ψ is such that each F_i remains monotone after the transformation.

4 LIR on Paths

In this section we describe an algorithm for solving the LIR problem on a path, represented as a directed graph $P = (V, A)$ where $V = \{v_1, \ldots, v_n\}$ and $A = \{(v_i, v_{i+1}) : 1 \leq i < n\}$. A function $s : V \to \mathbb{R}$ is isotonic (on P) if $s(v_i) \leq s(v_{i+1})$, and γ-Lipschitz for some real constant γ if $s(v_{i+1}) \leq s(v_i) + \gamma$ for each $i = 1, \ldots, n-1$. For the rest of this section let $t : V \to \mathbb{R}$ be an input function on V. For each $i = 1, \ldots, n$, let $V_i = \{v_1, \ldots, v_i\}$, let P_i be the subpath $v_1, \ldots, v_i$, and let E_i and $\tilde{E}_i$, respectively, be shorthands for E_{V_i} and $E^{v_i}_{P_i}$. By definition, if we let $\tilde{E}_0 = 0$, then for each $i \geq 1$:

$$\tilde{E}_i(x) = (x - t(v_i))^2 + \min_{x-\gamma \leq y \leq x} \tilde{E}_{i-1}(y). \tag{1}$$

By Lemma 1, $\tilde{E}_i$ is convex and continuous and thus has a unique minimizer s^*_i.

Lemma 3. *For $i \geq 1$, the function $\tilde{E}_i$ is given by the recurrence relation:*

$$\tilde{E}_i(x) = (x - t(v_i))^2 + \begin{cases} \tilde{E}_{i-1}(x - \gamma) & x > s^*_{i-1} + \gamma \\ \tilde{E}_{i-1}(s^*_{i-1}) & x \in [s^*_{i-1}, s^*_{i-1} + \gamma] \\ \tilde{E}_{i-1}(x) & x < s^*_{i-1}. \end{cases} \tag{2}$$

Thus by Lemmas 1 and 3, $\tilde{E}_i$ is strictly convex and piecewise quadratic. We call a value x that determines the boundary of two neighboring quadratic pieces of $\tilde{E}_i$ a *breakpoint* of $\tilde{E}_i$. Since $\tilde{E}_1$ is a simple (one-piece) quadratic function, it has no breakpoints. For $i > 1$, the breakpoints of the function $\tilde{E}_i$ consist of s^*_{i-1} and $s^*_{i-1} + \gamma$, as determined by recurrence (2), together with breakpoints that arise from recursive applications of $\tilde{E}_{i-1}$. Examining equation (2) reveals that all breakpoints of $\tilde{E}_{i-1}$ that are smaller than s^*_{i-1} remain breakpoints in $\tilde{E}_i$ and all those larger than s^*_{i-1} are increased by γ and these form all of the breakpoints of $\tilde{E}_i$ (see Figure 2(right)). Thus $\tilde{E}_i$ has precisely $2i - 2$ breakpoints. To compute the point s^*_i at which $\tilde{E}_i(x)$ is minimized, it is enough to scan over these $O(i)$ quadratic pieces and find the unique piece whose minimum lies between its two ending breakpoints.

Lemma 4. *Given the sequence $s_1^*, \ldots, s_n^*$, one can compute the γ-LIR s of input function t in $O(n)$ time.*

One can compute the values of $s_1^*, \ldots, s_n^*$ in n iterations. The ith iteration computes the value s_i^* at which $\tilde{E}_i$ is minimized and then uses it to compute $\tilde{E}_{i+1}$ via (2) in $O(i)$ time. Hence, the γ-LIR of $t \in \mathbb{R}^V$ can be computed in linear time. However, this gives an $O(n^2)$ algorithm for computing the γ-LIR of t. We now show how this running time can be reduced to $O(n \log n)$.

For simplicity we assume each s_i^* is none of the breakpoints of $\tilde{E}_i$. Hence s_i^* belongs to the interior of some interval on which $\tilde{E}_i$ is quadratic, and its derivative is zero at s_i^*. If we know to which quadratic piece of $\tilde{E}_i$ the point s_i^* belongs, we can determine s_i^* by setting the derivative of that piece to zero.

Lemma 5. *The derivative of $\tilde{E}_i$ is continuous isotonic and piecewise-linear.*

Let F_i denote the derivative of $\tilde{E}_i$ with the recurrence (via (2)):

$$F_{i+1} = 2(x - t(v_{i+1})) + \hat{F}_i(x); \tag{3}$$

$$\hat{F}_i(x) = \begin{cases} F_i(x-\gamma) & x > s_i^* + \gamma, \\ 0 & x \in [s_i^*, s_i^* + \gamma], \\ F_i(x) & x < s_i^*. \end{cases} \tag{4}$$

if we set $F_0 = 0$. As mentioned above, s_i^* is simply the solution of $F_i(x) = 0$, which, by Lemma 5 always exists and is unique. Intuitively, $\hat{F}_i$ is obtained from F_i by splitting it at s_i^*, shifting the right part by γ, and connecting the two pieces by a horizontal edge from s_i^* to $s_i^* + \gamma$ (lying on the x-axis).

In order to find s_i^* efficiently, we use an ACT $\mathcal{T}(F_i)$ to represent F_i. It takes $O(\log |F_i|) = O(\log i)$ time to compute $s_i^* = \textsc{Evaluate}^{-1}(0)$ on $\mathcal{T}(F_i)$. Once s_i^* is computed, we store it in a separate array for back-solving through Lemma 4. We turn $\mathcal{T}(F_i)$ into $\mathcal{T}(\hat{F}_i)$ by performing a sequence of $\textsc{Insert}((s_i^*, 0))$, $\textsc{Interval}_x(\psi, s_i^*, \infty)$, and $\textsc{Insert}((s_i^*, 0))$ operations on $\mathcal{T}(F_i)$ where $\psi(q) = q + \binom{\gamma}{0}$; the two insert operations add the breakpoints at s_i^* and $s_i^* + \gamma$ and the interval operation shifts the portion of F_i to the right of s_i^* by γ. We then turn $\mathcal{T}(\hat{F}_i)$ into $\mathcal{T}(F_{i+1})$ by performing $\textsc{Affine}(\phi_{i+1})$ operation on $\mathcal{T}(\hat{F}_i)$ where $\phi_{i+1}(q) = Mq + c$ where $M = \left(\begin{smallmatrix} 1 & 0 \\ 2 & 1 \end{smallmatrix}\right)$ and $c = \binom{0}{-2t(v_{i+1})}$.

Given ACT $\mathcal{T}(F_i)$, s_i^* and $\mathcal{T}(F_{i+1})$ can be computed in $O(\log i)$ time. Hence, we can compute $s_1^*, \ldots, s_n^*$ in $O(n \log n)$ time. By Lemma 4, we can conclude:

Theorem 2. *Given a path $P = (V, A)$, a function $t \in \mathbb{R}^V$, and a constant γ, the γ-Lipschitz isotonic regression of t on P can be found in $O(n \log n)$ time.*

Update operation. We define a procedure $\textsc{Update}(\mathcal{T}(\hat{F}_i), t(v_{i+1}), \gamma)$ that encapsulates the process of turning $\mathcal{T}(\hat{F}_i)$ into $\mathcal{T}(\hat{F}_{i+1})$ and returning s_{i+1}^*. Specifically, it performs $\textsc{Affine}(\phi_{i+1})$ of $\mathcal{T}(\hat{F}_i)$ to produce $\mathcal{T}(F_{i+1})$, then it outputs $s_{i+1}^* = \textsc{Evaluate}^{-1}(0)$ on $\mathcal{T}(F_{i+1})$, and finally a sequence of $\textsc{Insert}((s_{i+1}^*, 0))$, $\textsc{Interval}(\psi, s_{i+1}^*, \infty)$, and $\textsc{Insert}((s_{i+1}^*, 0))$ operations on $\mathcal{T}(F_{i+1})$ for $\mathcal{T}(\hat{F}_{i+1})$.

Performed on $\mathcal{T}(F)$ where F has n breakpoints, an UPDATE takes $O(\log n)$ time. An UNUPDATE$(\mathcal{T}(\hat{F}_{i+1}), t(v_{i+1}), \gamma)$ reverts affects of UPDATE$(\mathcal{T}(\hat{F}_i), t(v_{i+1}), \gamma)$. This requires that s^*_{i+1} is stored for the reverted version. Similarly, we can perform UPDATE$(\mathcal{S}, t(v_i), \gamma)$ and UNUPDATE$(\mathcal{S}, t(v_i), \gamma)$ on a tree set $\mathcal{S}$, in $O(k \log^2 n)$ time, the bottleneck coming from EVALUATE$^{-1}(0)$.

Lemma 6. *For $\mathcal{T}(\hat{F}_i)$,* UPDATE$(\mathcal{T}(\hat{F}_i), t(v_i), \gamma)$ *and* UNUPDATE$(\mathcal{T}(\hat{F}_i), t(v_i), \gamma)$ *take $O(\log n)$ time.*

5 LUR on Paths

Let $P = (V, A)$ be an *undirected* path, $V = \{v_1, \ldots, v_n\}$, $A = \{\{v_i, v_{i+1}\}, 1 \leq i < n\}$, and $t \in \mathbb{R}^V$. For $v_i \in V$ let $P_i = (V, A_i)$ be a directed graph in which all edges are directed towards v_i; that is, for $j < i$, $(v_j, v_{j+1}) \in A_i$ and for $j > i$ $(v_j, v_{j-1}) \in A_i$. For each $i = 1, \ldots, n$, let $\mathit{\Gamma}_i = \mathit{\Gamma}(P_i, V) \subseteq \mathbb{R}^V$ and let $\sigma_i = \arg\min_{s \in \mathit{\Gamma}_i} E_V(s)$. If $\kappa = \arg\min_i E_V(\sigma_i)$, then σ_κ is the γ-LUR of t on P.

We find σ_κ in $O(n \log^2 n)$ time by solving the LIR problem, then traversing the path while maintaining the optimal solution using UPDATE and UNUPDATE. Specifically, for $i = 1, \ldots, n$, let $V_i^- = \{v_1, \ldots, v_{i-1}\}$ and $V_i^+ = \{v_{i+1}, \ldots, v_n\}$. For P_i, let $\hat{F}^-_{i-1}$, $\hat{F}^+_{i+1}$ be the functions on directed paths $P[V_i^-]$ and $P[V_i^+]$, respectively, as defined in (3). Set $\bar{F}_i(x) = 2(x - t(v_i))$. Then the function $F_i(x) = dE_{v_i}(x)/dx$ can be written as $F_i(x) = \hat{F}^-_{i-1}(x) + \hat{F}^+_{i+1}(x) + \bar{F}_i(x)$. We store F_i as the tree set $S_i = \{\mathcal{T}(\hat{F}^-_{i-1}), \mathcal{T}(\hat{F}^+_{i+1}), \mathcal{T}(\bar{F}_i)\}$. By performing EVALUATE$^{-1}(0)$ we can compute s^*_i in $O(\log^2 n)$ time (the rate limiting step), and then we can compute $E_V(s^*_i)$ in $O(\log n)$ time using INTEGRATE(s^*_i). Assuming we have $\mathcal{T}(\hat{F}^-_{i-1})$ and $\mathcal{T}(\hat{F}^+_{i+1})$, we can construct $\mathcal{T}(\hat{F}^-_i)$ and $\mathcal{T}(\hat{F}^+_{i+2})$ in $O(\log n)$ time be performing UPDATE$(\mathcal{T}(F^-_{i-1}), t(v_i), \gamma)$ and UNUPDATE$(\mathcal{T}(F^+_{i+1}), t(v_{i+1}), \gamma)$. Since $\mathcal{S}_1 = \{\mathcal{T}(F^-_0), \mathcal{T}(F^+_2), \mathcal{T}(\bar{F}_1)\}$ is constructed in $O(n \log n)$ time, finding κ by searching all n tree sets takes $O(n \log^2 n)$ time.

Theorem 3. *Given an undirected path $P = (V, A)$ and a $t \in \mathbb{R}^V$ together with a real $\gamma \geq 0$, the γ-LUR of t on P can be found in $O(n \log^2 n)$ time.*

6 LIR on Rooted Trees

Let $T = (V, A)$ be a rooted tree with root r and let for each vertex v, $T_v = (V_v, A_v)$ denote the subtree of T rooted at v. Similar to the case of path LIR, for each vertex $v \in V$ we use the shorthands $E_v = E_{V_v}$ and $\tilde{E}_v = E^v_{T_v}$. Since the subtrees rooted at distinct children of a node v are disjoint, one can write an equation corresponding to (1) in the case of paths, for any vertex v of T:

$$\tilde{E}_v(x) = (x - t(v))^2 + \sum_{u \in \delta^-(v)} \min_{x - \gamma \leq y \leq x} \tilde{E}_u(y), \tag{5}$$

where $\delta^-(v) = \{u \in V \mid (u,v) \in A\}$. An argument similar to that of Lemma 5 together with Lemma 1 implies that for every $v \in V$, the function $\tilde{E}_v$ is convex and piecewise quadratic, and its derivative F_v is continuous, monotonically increasing, and piecewise linear. We can prove that F_v satisfies the following recurrence where $\hat{F}_u$ is defined analogously to $\hat{F}_i$ in (4):

$$F_v(x) = 2(x - t(v)) + \sum_{u \in \delta^-(v)} \hat{F}_u(x). \tag{6}$$

Thus to solve the LIR problem on a tree, we post-order traverse the tree (from the leaves toward the root) and when processing a node v, we compute and sum the linear functions $\hat{F}_u$ for all children u of v and use (6) to compute the function F_v. We then solve $F_v(x) = 0$ to find s_v^*. As in the case of path LIR, $\hat{F}_v$ can be represented by an ACT $\mathcal{T}(\hat{F}_v)$. For simplicity, we assume each non-leaf vertex v has two children $h(v)$ and $l(v)$, where $|\hat{F}_{h(v)}| \geq |\hat{F}_{l(v)}|$. Given $\mathcal{T}(\hat{F}_{h(v)})$ and $\mathcal{T}(\hat{F}_{l(v)})$, we can compute $\mathcal{T}(\hat{F}_v)$ and s_v^* with the operation $\textsc{Update}(\mathcal{T}(\hat{F}_{h(v)} + \hat{F}_{l(v)}), t(v), \gamma)$ in $O(|\hat{F}_{l(v)}|(1 + \log|\hat{F}_{h(v)}|/\log|\hat{F}_{l(v)}|))$ time. The merging of two functions dominates the time for the update. By careful, global analysis of all merge costs we can achieve the following result:

Theorem 4. *Given rooted tree $T = (V, A)$, function $t \in \mathbb{R}^V$, and Lipschitz constant γ, we can find in $O(n \log n)$ time, the γ-LIR of t on T.*

7 LUR on Trees

We extend this framework to solve the LUR on trees problem. Given an undirected input tree, we direct the tree towards an arbitrary vertex chosen as the root and invoke Theorem 4. Similar to LUR on paths, we traverse the tree, letting each vertex be the root, and maintaining enough information to recompute the LIR solution for the new root in $O(\log^3 n)$ time. We return the solution for the root which minimizes the error. The details are left for the full version [2].

Theorem 5. *Given unrooted tree $T = (V, A)$, function $t \in \mathbb{R}^V$, and Lipschitz constraint γ, we can find in $O(n \log^3 n)$ time the γ-LUR of t on T.*

References

1. P. K. Agarwal, L. Arge, and K. Yi. I/O-efficient batched union-find and its applications to terrain analysis. In *Proc. 22 ACM Symp. on Comp. Geometry*, 2006.
2. P. K. Agarwal, J. M. Phillips, and B. Sadri. Lipschitz unimodal and isotonic regression on paths and trees. Technical report, arXiv:0912.3624, 2009.
3. S. Angelov, B. Harb, S. Kannan, and L.-S. Wang. Weighted isotonic regression under the l_1 norm. In *Proc. 17 ACM-SIAM Symp. on Discrete Algorithms*, 2006.
4. D. Attali, M. Glisse, S. Hornus, F. Lazarus, and D. Morozov. Persistence-sensitive simplification of functions on surfaces in linear time. Manuscript, INRIA, 2008.

5. M. Ayer, H. D. Brunk, G. M. Ewing, W. T. Reid, and E. Silverman. An empirical distribution function for sampling with incomplete information. *Annals of Mathematical Statistics*, 26:641–647, 1955.
6. C. Bajaj, V. Pascucci, and D. Schikore. Visualization of scalar topology for structural enhancement. In *IEEE Visualizationw*, pages 51–58, 1998.
7. R. E. Barlow, D. J. Bartholomew, J. M. Bremmer, and H. D. Brunk. *Statistical Inference Under Order Restrictions: The Theory and Application of Isotonic Regression*. John Wiley and Sons, 1972.
8. P. Bremer, H. Edelsbrunner, B. Hamann, and V. Pascucci. A multi-resolution data structure for two-dimensional morse functions. In *Proceedings 14th IEEE Visualization Conference*, pages 139–146, 2003.
9. M. R. Brown and R. E. Tarjan. A fast merging algorithm. *J. Alg*, 15:416–46, 1979.
10. H. D. Brunk. Maximum likelihood estimates of monotone parameters. *Annals of Mathematical Statistics*, 26:607–616, 1955.
11. A. Danner, T. Mølhave, K. Yi, P. K. Agarwal, L. Arge, and H. Mitasova. Terrastream: from elevation data to watershed hierarchies. In *15th ACM International Symposium on Advances in Geographic Information Systems*, 2007.
12. H. Edelsbrunner and J. Harer. *Persistent Homology: A Survey*. Number 453 in Contemporary Mathematics. American Mathematical Society, 2008.
13. H. Edelsbrunner, D. Letscher, and A. Zomorodian. Topological persistence and simplification. In *Proc. 41 Symp. on Foundatons of Computer Science*, 2000.
14. H. Edelsbrunner, D. Morozov, and V. Pascucci. Persistence-sensitive simplification functions on 2-manifolds. In *Proc. 22 ACM Symp. on Comp. Geometry*, 2006.
15. G. N. Frederickson and D. B. Johnson. The complexity of selection and ranking in $x + y$ and matrices with sorted columns. *J. Comput. Syst. Sci.*, 24:192–208, 1982.
16. S. J. Grotzinger and C. Witzgall. Projection onto order simplexes. *Applications of Mathematics and Optimization*, 12:247–270, 1984.
17. I. Guskov and Z. J. Wood. Topological noise removal. In *Graphics Interface*, 2001.
18. W. S. Jewel. Isotonic optimization in tariff construction. *ASTIN*, 8:175–203, 1975.
19. W. L. Maxwell and J. A. Muckstadt. Establishing consistent and realistic reorder intervals in production-distribution systems. *Operations Res.*, 33:1316–1341, 1985.
20. X. Ni, M. Garland, and J. C. Hart. Fair Morse functions for extracting the topological structure of a surface mesh. *ACM Transact. Graphics*, 23:613–622, 2004.
21. P. M. Pardalos and G. Xue. Algorithms for a class of isotonic regression problems. *Algorithmica*, 23:211–222, 1999.
22. F. P. Preparata and M. I. Shamos. *Computational geometry: an introduction*. Springer-Verlag, New York, NY, USA, 1985.
23. P. Soille, J. Vogt, and R. Cololmbo. Carbing and adaptive drainage enforcement of grid digital elevation models. *Water Resources Research*, 39(12):1366–1375, 2003.
24. J. Spouge, H. Wan, and W. J. Wilbur. Least squares isotonic regression in two dimensions. *Journal of Optimization Theory and Applications*, 117:585–605, 2003.
25. Q. F. Stout. Optimal algorithms for unimodal regression. *Computing Science and Statistics*, 32:348–355, 2000.
26. Q. F. Stout. Unimodal regression via prefix isotonic regression. *Computational Statistics and Data Analysis*, 53:289–297, 2008.
27. W. A. Thompson, Jr. The problem of negative estimates of variance components. *Annals of Mathematical Statistics*, 33:273–289, 1962.
28. A. Zomorodian. Computational topology. In Atallah and Blanton, editors, *Algorithms and Theory of Computation Handbook*. Chapman & Hall/CRC Press, 2009.