Robust Nonparametric Data Approximation of Point Sets via Data Reduction*

Stephane Durocher, Alexandre Leblanc, Jason Morrison, and Matthew Skala

University of Manitoba, Winnipeg, Canada, durocher@cs.umanitoba.ca, alex_leblanc@umanitoba.ca, jason_morrison@umanitoba.ca, mskala@cs.umanitoba.ca

Abstract. In this paper we present a novel nonparametric method for simplifying piecewise linear curves and we apply this method as a statistical approximation of structure within sequential data in the plane. We consider the problem of minimizing the average length of sequences of consecutive input points that lie on any one side of the simplified curve. Specifically, given a sequence P of n points in the plane that determine a simple polygonal chain consisting of $n-1$ segments, we describe algorithms for selecting a subsequence $Q \subset P$ (including the first and last points of P) that determines a second polygonal chain to approximate P, such that the number of crossings between the two polygonal chains is maximized, and the cardinality of Q is minimized among all such maximizing subsets of P. Our algorithms have respective running times $O(n^2\sqrt{\log n})$ when P is monotonic and $O(n^2 \log^{4/3} n)$ when P is any simple polyline.

1 Introduction

Given a simple polygonal chain P (a *polyline*) defined by a sequence of points $(p_1, p_2, \ldots, p_n)$ in the plane, the *polyline approximation problem* is to produce a simplified polyline $Q = (q_1, q_2, \ldots, q_k)$, where $k < n$. The polyline Q represents an approximation of P that optimizes one or more functions of P and Q. For P to be *simple*, the points $p_1, \ldots, p_n$ must be distinct and P cannot intersect itself.

Motivation for studying polyline approximation comes from the fields of computer graphics and cartography, where approximations are used to render vector-based features such as streets, rivers, or coastlines onto a screen or a map at appropriate resolution with acceptable error [1], as well as in problems involving computer animation, pattern matching and geometric hashing (see Alt and Guibas' survey for details [3]). Our present work removes the arbitrary parameter previously required to describe acceptable error between P and Q, and provides an approximation method that is robust to some forms of noise. While previous work uses the Real RAM model, our analysis primarily assumes a Word RAM model to ensure clarity in discussions of lower bounds on time complexity. We comment on our results in both models. We further note that the Word RAM model requires that non-integer coordinates (floats and rationals) be contained in a constant number of words and be easily comparable. See the work of Han [8] and of Chan and Pătraşcu [4] for more on this model.

Typical polyline approximation algorithms require that distance between two polylines be measured using a function denoted here by $\zeta(P, Q)$. The specific measure of interest differs depending on the focus of the particular problem or article; however, three measures are popular: Chebyshev distance ζ_C, Hausdorff distance ζ_H, and Fréchet distance ζ_F. In informal terms, the Chebyshev distance is the maximum absolute difference between y-coordinates of P and Q (maximum residual); the symmetric Hausdorff distance is the distance between the most isolated point of P or Q

* Work was supported in part by the Natural Science and Engineering Research Council of Canada (NSERC).

and the other polyline; and the Fréchet distance is more complicated, being the shortest possible maximum distance between two particles each moving forward along P and Q. Alt and Guibas [3] give more formal definitions. We define a new measure of quality or similarity, to be maximized, rather than an error to be minimized. Our crossing measure is a combinatorial description of how well Q approximates P. It is invariant under a variety of geometric transformations of the polylines, and is often robust to uncertainty in the locations of individual points. Specifically, we consider the problem of minimizing the average length of sequences of consecutive input points that lie on any one side of the simplified curve. Given a sequence P of n points in the plane that determine a simple polygonal chain consisting of $n-1$ segments, we describe algorithms for selecting a subsequence $Q \subset P$ (including the first and last points of P) that determines a second polygonal chain to approximate P, such that the number of crossings between the two polygonal chains is maximized, and the cardinality of Q is minimized among all such maximizing subsets of P.

Our algorithm for minimizing $|Q|$ while optimizing our nonparametric quality measure requires $O(n^2\sqrt{\log n})$ time when P is monotonic in x, or $O(n^2 \log^{4/3} n)$ time when P is a non-monotonic simple polyline on the plane, both in $O(n)$ space. The near-quadratic times are slightly larger in their polylog exponents for the Real RAM model ($O(n^2 \log n)$ and $O(n^2 \log^2 n)$, respectively), and are remarkably similar to the optimal times achieved in the parametric version of the problem using Hausdorff distance [1, 5], suggesting the possibility that the problems may have similar complexities.

In Section 3, we define the crossing measure $\chi(Q, P)$ and relate the concepts and properties of $\chi(Q, P)$ to previous work in both polygonal curve simplification and robust approximation. In Section 4, we describe our algorithms to compute approximations of monotonic and non-monotonic simple polylines that maximize $\chi(Q, P)$.

2 Related Work

Previous work on polyline approximation is generally divided into four categories depending on what property is being optimized and what restrictions are placed on Q [3]. Problems can be classified as requiring an approximating polyline Q having the minimum number of segments (minimizing $|Q|$) for a given acceptable error $\zeta(P, Q) \leq \epsilon$, or a Q with minimum error $\zeta(P, Q)$ for a given value of $|Q|$. These are called min-# problems and min-ϵ problems respectively. These two types of problems are each further divided into "restricted" problems, where the points of Q are required to be a subset of those in P and to include the first and last points of P ($q_1 = p_1$ and $q_k = p_n$), and "unrestricted" problems, where the points of Q may be arbitrary points on the plane. Under this classification, the polyline approximation Q we examine is a restricted min-# problem for which a subset of points of P is selected (including p_1 and p_n) where the objective measure is the number of crossings between P and Q and an optimal approximation first maximizes the crossing number (rather than minimizing it), and then has a minimum $|Q|$ given the maximum crossing number.

While the restricted min-# problems find the smallest sized approximation within a given error ϵ, an earlier approach was to find any approximation within the given error. The cartographers Douglas and Peucker [6] developed a heuristic algorithm where an initial (single segment) approximation was evaluated and the furthest point was then added to the approximation. This technique remained inefficient until Hershberger and Snoeyink [9] concluded that it could be applied in $O(n \log^* n)$ time and linear space.

The most relevant previous literature is on restricted min-# problems. Imai and Iri [10] presented an early solution to the restricted polyline approximation problem using $O(n^3)$ time and $O(n)$ space. The version they study optimizes $k = |Q|$ while maintaining that the Hausdorff metric between Q

and P is less than the parameter ϵ. Their algorithm was subsequently improved by Melkman and O'Rourke [11] to $O(n^2 \log n)$ time and then by Chan and Chin [5] to $O(n^2)$ time. Subsequently, Agarwal and Varadarajan [2] changed the approach from finding a shortest path in an explicitly constructed graph to an implicit method that runs in $O(f(\delta)n^{4/3+\delta})$ time, where δ is an abritrarily chosen constant. Agarwal and Varadarajan used the L_1 Manhattan and L_∞ Chebyshev metrics instead of the previous works' Hausdorff metric. Finally, Agarwal *et al.* [1] study a variety of metrics and give approximations of the min-# problem in $O(n)$ or $O(n \log n)$ time.

3 A Crossing Measure and its Computation

3.1 Crossing Measure

The crossing measure $\chi(Q, P)$ is defined for a sequence of n distinct points $P = (p_1, p_2, \ldots, p_n)$ and a subsequence of k distinct points $Q \subset P, Q = (q_1, q_2, \ldots, q_k)$ with the same first and last values: $q_1 = p_1$ and $q_k = p_n$. For each p_i let $(x_i, y_i) = p_i \in \mathbb{R}^2$. To understand the crossing measure, it is necessary to make use of the idea of left and right sidedness of a point relative to a directed line segment. A point p_j is on the left side of a segment $S_{i,i+1} = [p_i, p_{i+1}]$ if the signed area of the triangle formed by the points p_i, p_{i+1}, p_j is positive. Correspondingly, p_j is on the right side of the segment if the signed area is negative. The three points are collinear if the area is zero.

For any endpoint q_j of a segment in Q it is possible to determine the side of P on which q_j lies. Since Q is a polyline using a subset of the points defining P, for every segment $S_{i,i+1}$ there exists a corresponding segment of $S_{\pi(j),\pi(j+1)}$ such that $1 \leq \pi(j) \leq i < i+1 \leq \pi(j+1) \leq n$. The function $\pi : \{1, \ldots, n\} \to \{1, \ldots, k\}$ maps a point $q_j \in Q$ to its corresponding point $p_{\pi(j)} \in P$ such that $p_{\pi(j)} q_j$. The endpoints of $S_{\pi(j),\pi(j+1)}$ are given a side based on $S_{i,i+1}$ and vice versa. Two segments intersect if they share a point. Such a point is interior to both segments if and only if the segments change sides with respect to each other. The intersection is at an endpoint if at least one endpoint is collinear to the other segment [12, p. 566]. The *crossing measure* $\chi(Q, P)$ is the number of times that Q changes sides from properly left to properly right of P due to an intersection between the polylines, as shown in Figure 1. A single crossing can be generated by any of five cases (see Figure 2):

1. A segment of Q intersects P at a point distinct from any endpoints;
2. two consecutive segments of P meet and cross Q at a point interior to a segment of Q;
3. one or more consecutive segments of P are collinear to the interior of a segment of Q with the previous and following segments of P on opposite sides of that segment of Q;
4. two consecutive segments of P share their common point with two consecutive segments of Q and form a crossing; or
5. in a generalization of the previous case, instead of being a single point, the intersection comprises one or more sequential segments of P and possibly Q that are collinear or identical.

In Section 3.2, we discuss how to compute the crossings for the first three cases, which are all cases where crossings involve only one segment of Q. The remaining cases involve more than one segment of Q, because an endpoint of one segment of Q or even some entire segments of Q are coincident with one or more segments of P; those cases are discussed in Section 3.3.

In the case where the x-coordinates of P are monotonic, P describes a function Y of x and Q is an approximation $\hat{Y}$ of that function. The signs of the residuals $r = (r_1, r_2, \ldots, r_n) = Y_P - \hat{Y}$ are computed at the x-coordinates of P and are equivalent to the sidedness described above. The

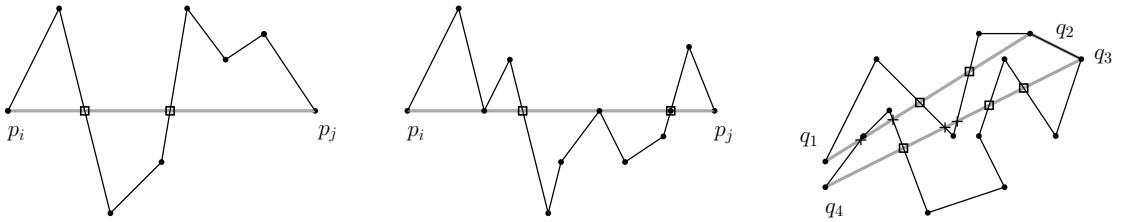


Fig. 1: Crossings are indicated with a square and false crossings are marked with a +. Crossings are only counted when the simplifying segment intersects the chain that begins and ends with its own endpoints.

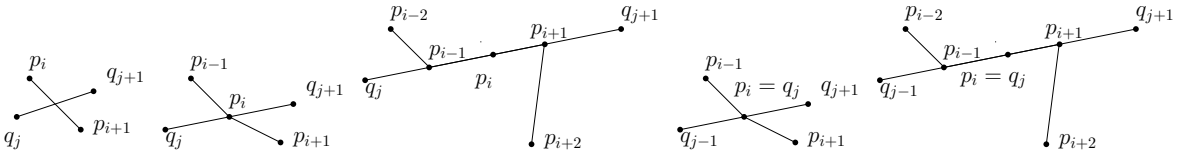


Fig. 2: Examples of the five cases generating a single crossing.

crossing number is the number of proper sign changes in the sequence of residuals. The resulting approximation maximizes the likelihood that adjacent residuals would have different signs, while minimizing the number of original data points retained conditional on that number of sign changes. Note that if r was independently and identically selected at random from a distribution with median zero, then any adjacent residuals in the sequence r would have different signs with probability 1/2.

3.2 Counting Crossings with a Segment

To compute an approximation Q with optimal crossing number for a given P, we consider the optimal number of crossings for segments of P and combine them in a dynamic programming algorithm. Starting from a point p_i we compute optimal crossing numbers for each of the $n - i$ segments that start at point p_i and end at some p_j with $i < j \leq n$. Computing all $n - i$ optimal crossing numbers for a given p_i simultaneously in a single pass is more efficient than computing them for each (p_i, p_j) pair separately. These batched computations are performed for each p_i and the results used to find Q.

To compute a single batch, we will consider the angular order of points in $P_{i+1,n} = \{p_{i+1}, \ldots, p_n\}$ with respect to p_i. Let $\rho_i(j)$ be a function on the indices representing the angular order of segments (p_i, p_j) within this set with respect to vertical, such that $\rho_i(j) = 1$ for all p_j that are closest to the vertical line passing through p_i, and $\rho_i(j) \leq \rho_i(k)$ if and only if the clockwise angle for p_j from the vertical is less than or equal to the corresponding angle for p_k. Note that within axis-aligned quadrants that are centered on p_i, a larger angle corresponds to a smaller slope. This confirms that angular order comparisons are $O(1)$ time computable using basic arithmetic in the Word RAM Model. See Figure 3(a). Using this angular ordering we partition $P_{i+1,n}$ into chains and process the batch of crossing number problems as discussed below.

We define a *chain* with respect to p_i to be a consecutive sequence $P_{\ell,\ell'} \subset P_{i+1,n}$ with non-decreasing angular order. That is, either $\rho_i(\ell') \geq \rho_i(j+1) \geq \cdots \geq \rho_i(\ell)$ or $\rho_i(\ell) \leq \rho_i(j+1) \leq \cdots \leq \rho_i(\ell')$, with the added constraint that chains cannot cross the vertical ray above p_i. Each segment

that does cross is split into two pieces using two artificial points on the ray per crossing segment. The points on the low segment portions have rank $\rho_i = 1$ and the identically placed other points have rank $\rho_i = n+1$. These points do not increase the complexities by more than a constant factor and are not mentioned further. Processing $P_{i+1,n}$ into its chains is done by first computing the slope and quadrant for each point and storing that information with the points. Then the points are sorted by angular order around p_i and $\rho_i(j)$ is computed as the rank of p_j in the sorted list. This step requires $O(n \log\log n)$ time and linear space in the Word RAM model [7] with linear extra space to store the slopes, quadrants and ranks. Creating a list of chains is then feasable in $O(n)$ time and space by storing the indices of the beginning and end of each chain encountered while checking points p_j in increasing order from $j = i+1$ to $j = n$. Identifying all chains involves two steps. First, all segments are checked to determine whether they intersect the vertical ray, each in $O(1)$ time. Such an intersection implies that the previous chain should end and the segment that crosses the ray should be a new chain (note that an artificial index of $i + \frac{1}{2}$ can denote the point that crosses the vertical). The second check is to determine whether the most recent pair of points has a different angular rank ordering from the previous pair. If so, the previous chain ended with the previous point and the new chain begins with the current segment. Each chain is oriented from lowest angular index to highest angular index.

Lemma 1. *Consider any chain $P_{\ell,\ell'}$ (w.l.o.g. assume $\ell < \ell'$). With respect to p_i the segment $S_{i,j} : (i < j \leq n)$ can have at most one crossing strictly interior to $P_{\ell,\ell'}$.*

Proof. Case 1. Suppose $\rho_i(\ell) = \rho_i(j)$ or $\rho_i(j) = \rho_i(\ell')$. If $\ell = n$ then no crossing can exist because at least one end (or all) of $P_{k,\ell}$ is collinear with $S_{i,j}$ and no proper change in sidedness can occur in this chain to generate a crossing.

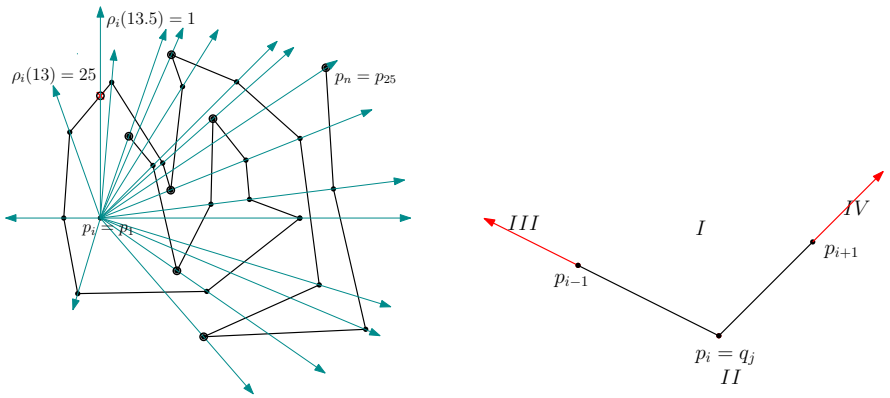
Case 2. Suppose $\rho_i(j) \notin [\rho_i(\ell), \rho_i(\ell')]$. These cases have no crossings with the chain because $P_{k,\ell}$ is entirely on one side of $S_{i,j}$. A ray exists between either $\rho_i(j) < \rho_i(\ell)$ or $\rho_i(m) < \rho_i(j)$ that separates $P_{k,l}$ from $S_{i,j}$ and thus no crossings can occur between the segment and the chain.

Case 3. Suppose $\rho_i(j) \in (\rho_i(\ell), \rho_i(\ell'))$. Assume that the chain causes at least two crossings. Pick the lowest index segment for each of the two crossings that are the fewest segments away from p_i. By definition there are no crossings of segments between these two segments. Label the point with lowest index of these two segments p_λ and the point with greatest index $p_{\lambda'}$. Define a possibly degenerate cone Φ with a base p_i and rays through p_λ and $p_{\lambda'}$. This cone, by definition, separates the segments from $p_{\lambda+1}$ to $p_{\lambda'-1}$ from the remainder of the chain. Since this sub-chain cannot circle p_i entirely there must exist one or more points that have a maximum (or minimum) angular order, which contradicts the definition of the chain. Hence there must be at most one crossing. □

The algorithm for computing the crossing measure on a batch of segments depends on the nature of P. If P is x-monotone, then the chains can be ordered by increasing x-coordinates or equivalently by the greatest index among the points that define them. In this case, a segment $S_{i,j}$ intersects any chain $P_{\ell,\ell'}$ exactly once if its x-coordinates are less than p_j and $\rho_i(j) \in (\rho_i(\ell), \rho_i(\ell'))$ (i.e., Case 3 of Lemma 1).

The algorithm represents each of the $O(n)$ segments $S_{i,j}$ as a blue point $(j, \rho_i(j))$ and each chain $P_{\ell,\ell'}$ as two red points: the start $(\max(\ell,\ell'), \min(\rho_i(\ell), \rho_i(\ell')))$ and end $(\max(\ell,\ell'), \max(\rho_i(\ell), \rho_i(\ell')))$. For every starting red point that is dominated (strictly greater than in both coordinates) by a single blue point a crossing is generated only if the corresponding red point is not dominated. The count of red start points dominated by each blue point is an offline counting query solvable in $O(n\sqrt{\log n})$ time and linear space using the Word RAM model [4, Theorem 2]. The count of red end points

dominated by each blue point must then be subtracted from the start domination counts using the same method. Correctness of the result follows from x-monotonicity and the proof of Lemma 1.



(a) An example of the rays' angular order of vertices in $P_{i+1,n}$ and the resulting chains

(b) Regions around p_j that determine a crossing at p_j

Fig. 3: Angular ordering with chains around p_i and regions local to p_i

The problem becomes more difficult if we assume that P is simple but not necessarily monotonic in x. While chains describe angular order nicely, a non-monotonic P does not imply a consistent ordering of chain boundaries. Thus, queries will be of a specific nature: for a given point p_j, we must determine how many chains are closer to i, have a lower maximum index than j, and are within the angular order (as with the monotonic example). We use the same strategy of two sets of dominance queries as in the monotonic case. The difference is that, instead of using the maximum index on a chain both for the chain's location within the polyline and its distance from p_i, we precompute a distance ranking of chains from p_i as well as the maximum index. Then, the start and end of each chain is represented as a three-dimensional red point.

Chains do not cross, and can only intersect at their endpoints, due to the non-overlapping definition of chains and the simplicity of P. Therefore, to compute the closeness of chains, we sweep a ray from p_i, initially vertical, in increasing ρ_i order (increasing angle). This defines a partial order on chains with respect to their distance from p_i. Using a topological sweep [12, p. 481] it is possible to determine a unique order that preserves this partial ordering of chains. Since there are $O(n)$ chains and changes in neighbours defining the partial order occur only at chain endpoints, there are $O(n)$ edges in the partial order. This further implies that $O(n \log n)$ time is required to determine the events in a sweep and $O(n)$ time to compute the topological ordering. Without loss of generality assume that the chains closest to p_i have a lower topological index. This is a traditional Real RAM approach, and while a more efficient Word RAM approach could be found, it would be unnecessary given that this is a preprocessing step. Computing each batch of domination queries requires $O(n \log^{4/3} n)$ time and linear space using a result of Chan and Pătraşcu [4, Corollary 3.2].

3.3 Counting Crossings Due to Neighbouring Approximation Segments

We now address the two cases of a crossing generated by more than one segment of Q. Suppose that $p_i = q_j$. Then there is an intersection between P and Q at this point, and we must detect whether

a change in sidedness accompanies this intersection. Assume initially that P does not contain any consecutively collinear segments; we will consider the other case later.

Categorization	Conditions for End of $S_{\pi(j-1),i}$	Conditions for Beginning of $S_{i,\pi(j+1)}$
collinear (1)	$q_{j-1} = p_{i-1}$ $q_{j+1} = p_{i+1}$	$q_{j-1} = p_{i-1}$ $q_{j+1} = p_{i+1}$
left (2)	$q_{j-1} \in I$ $(q_{j-1} \in III) \wedge (p_{i-2} \in II)$ $(q_{j-1} \in IV) \wedge (p_{i+2} \in II)$	$q_{j+1} \in I$ $(q_{j+1} \in III) \wedge (p_{i-2} \in II)$ $(q_{j+1} \in IV) \wedge (p_{i+2} \in II)$
right (3)	$q_{j-1} \in II$ $(q_{j-1} \in III) \wedge (p_{i-2} \in I)$ $(q_{j-1} \in IV) \wedge (p_{i+2} \in I)$	$q_{j+1} \in II$ $(q_{j+1} \in III) \wedge (p_{i-2} \in I)$ $(q_{j+1} \in IV) \wedge (p_{i+2} \in I)$

Table 1: Left, right, and collinear labels applied to beginning or end of a segment at p_j

We begin with the non-degenerate case where $(p_{i-1}, p_{i+1}, q_{i-1}, q_{i+1})$ are all distinct points (i.e., case 4 in Figure 2). Each of the points q_{j-1} and q_{j+1} can be in one of four locations: in the cone left of (p_{i-1}, p_i, p_{i+1}); in the cone right of (p_{i-1}, p_i, p_{i+1}); on the ray defined by $S_{i,i-1}$; or on the ray defined by $S_{i,i+1}$. These are labelled in Figure 3(b) as regions I through IV, respectively. In Cases III and IV it may also be necessary to consider the location of q_{i-1} or q_{j+1} with respect to $S_{i-2,i-1}$ or $S_{i+1,i+2}$.

For the degenerate case where the points may not be unique, if $p_i = q_j$ and $p_{i+1} \neq q_{j+1}$, then any change in sidedness is handled at p_i and can be detected by verifying the previous side from the polyline. If, however, $p_{i+1} = q_{j+1}$, then any change in sidedness will be counted further along in the approximation.

By examining these points it is possible to assign a sidedness to the end of $S_{\pi(j-1),\pi(j)}$ and the beginning of $S_{\pi(j),\pi(j+1)}$. Note that the sidedness of a point q_{j-1} with respect to $S_{i-2,i-1}$ can be inferred from the sidedness of p_{i-2} with respect to $S_{\pi(j-1),i-1}$, and that property is used in the case of regions III and IV. The assumed lack of consecutive collinear segments requires that $\{p_{i-2}, p_{i+2}\} \in I \cup II$ and thus Table 1 is a complete list of the possible cases when $|P| \geq 5$. Cases involving III or IV where $i \notin [3, n-2]$ are labelled collinear. We discuss the consequences of this choice later.

A single crossing occurs if and only if the end of $S_{\pi(j-1),i}$ is on the left or right of P, while the beginning of $S_{i,\pi(j+1)}$ is on the opposite side. Furthermore, the end of any approximation $Q_{1,j}$ of $P_{1,i}$ that ends in $S_{\pi(j-1),i}$ inherits the same labelling as the end of $S_{\pi(j-1),i}$. This labelling is consistent with the statement that the approximation last approached the polyline P from the side indicated by the labelling. To maintain this invariant in the labelling of the end of polylines, if $S_{\pi(j-1),i}$ is labelled as collinear then the approximation $Q_{1,j}$ needs to have the same labelling as $Q_{1,j-1}$. As a basis case, the approximations of $P_{1,2}$ and $P_{1,1}$ are the result of the identity operation so they must be collinear. Note that an approximation labelled collinear has no crossings.

The constant number of cases in Table 1, and the constant complexity of the sidedness test, imply that we can compute the number of crossings between a segment and a chain, and therefore the labelling for the segment, in constant time. Let $\eta(Q_{1,j}, S_{\pi(j-1),i})$ represent the number of extra crossings (necessarily 0 or 1) introduced at p_j by joining $Q_{1,j}$ and $S_{\pi(j-1),i}$. We have $\chi(Q_{1,j}, P_{1,i}) = \chi(Q_{1,j-1}, P_{1,\pi(j-1)}) + \chi(S_{\pi(j-1),i}, P_{\pi(j-1),i}) + \eta(Q_{1,j}, S_{\pi(j-1),i})$, which lends itself to computing the optimal approximation incrementally using dynamic programming.

It remains to consider the case of sequential collinear segments (i.e., case 5 in Figure 2). The polyline P' can be simplified into P by merging sequential collinear segments, effectively removing points of P' without changing its shape. When joining two segments where $p'_i = q_j$, p'_{i-1} and p'_{i+1}

define the regions as before but there is no longer a guarantee regarding non-collinearity of p'_{i-2} or p'_{i+2} with respect to the other points. The points q_{j-1} and q_{j-2} are now collinear if and only if either of them are entirely collinear to the relevant segments of P. Our check for equality is changed to a check for equality or collinearity. We examine the previous and next points of P' that are not collinear to the two segments $[p'_{i-1}, p'_i]$ and $[p'_i, p'_{i+1}]$. We find such points for every p'_i in a preprocessing step requiring linear time and space, by scanning the polyline for turns and keeping two queues of previous and current collinear points.

4 Finding a Polyline that Maximizes the Crossing Measure

This section describes our dynamic programming approach to computing a polyline Q that is a subset of P that maximizes the crossing measure $\chi(Q, P)$. Our algorithm returns a subset of minimum cardinality k among all such maximizing subsets. We compute $\chi(S_{i,j}, P_{i,j})$ in batches, as described in the previous section. Our algorithm maintains the best known approximations of $P_{1,i}$ for all $i \in [1, n]$ and each of the three possible labellings of the ends. We refer to these paths as $\mathcal{Q}_{\sigma,i}$ where σ describes the labelling at i: $\sigma = 1$ for collinear, $\sigma = 2$ for left, or $\sigma = 3$ for right.

To reduce the space complexity we do not explicitly maintain the (potentially exponential-size) set of all approximations $\mathcal{Q}_{\sigma,i}$. Instead, for each approximation corresponding to (σ, i) we maintain: $\chi(\mathcal{Q}_{\sigma,i}, P_{1,i})$ (initially zero); the size of the approximation found, $|\mathcal{Q}_{\sigma,i}|$ (initially $n+1$); the starting index of the last segment added, $\beta_{\sigma,i}$ (initially zero); and the end labelling of the best approximation to which the last segment was connected, $\tau_{\sigma,i}$ (initially zero). The initial values described represent the fact that no approximation is yet known. The algorithm begins by setting the values for the optimal identity approximation for $P_{1,1}$ to the following values (note $\sigma = 1$):

$$\chi(\mathcal{Q}_{1,1}, P_{1,1}) = 0, \qquad |\mathcal{Q}_{1,1}| = 1, \qquad \beta_{1,1} = 1, \qquad \tau_{1,1} = 1.$$

A total of $n-1$ iterations are performed, one for each $i \in [1, n-1]$, where for each of a batch of segments $S_{i,j} : i < j \leq n$ the algorithm considers a possible approximation ending in that segment. Each iteration begins with the set of approximations $\{\forall \sigma, \mathcal{Q}_{\sigma,\ell} : \ell \leq i\}$ being optimal, with maximal values of $\chi(\mathcal{Q}_{\sigma,\ell}, P_{1,\ell})$ and minimum size $|\mathcal{Q}_{\sigma,\ell}|$ for each of the specified σ and ℓ combinations. The iteration proceeds to calculate the crossing numbers of all segments starting at i and ending at a later index, $\{\chi(S_{i,j}, P_{i,j}) | j \in (i, n]\}$, using the method from Section 3. For each of the segments $S_{i,j}$ we compute the sidedness of both the end at j (σ'_j) and the start at i (υ'_j). Using υ'_j and all values of $\{\sigma : \beta_{\sigma,i} \geq 0\}$ it is possible to compute $\eta(\mathcal{Q}_{\sigma,i}, S_{i,j})$ using just the labellings of the two inputs (see Table 2). It is also possible to determine the labelling of the end of the concatenated polyline $\psi(\sigma, \sigma'_j)$ using the labelling of the end of the previous polyline σ and the end of the additional segment σ'_j (also shown in Table 2).

$\eta(\beta_{\sigma,i}, \upsilon'_j)$		$\beta_{\sigma,i}$ 1	2	3
	1	0	0	0
υ'_j	2	0	0	1
	3	0	1	0

$\psi(\sigma, \sigma'_j)$		σ 1	2	3
	1	1	2	3
σ'_j	2	2	2	2
	3	3	3	3

Table 2: Crossings $\eta(\beta_{\sigma,i}, \upsilon'_j)$ due to concatenation, and the end labelling $\psi(\sigma, \sigma'_j)$ of the polyline

With these values computed, the current value of $\chi(\mathcal{Q}_{\psi(\sigma,\sigma'_j),j}, P_{1,j})$ is compared to $\chi(\mathcal{Q}_{\sigma,i}, P_{1,j}) + \chi(S_{i,j}, P_{i,j}) + \eta(\beta_{\sigma,i}, \upsilon'_j)$ and if the new approximation has a greater or equal number of crossings, then we compute:

$$\chi(\mathcal{Q}_{\psi(\sigma,\sigma'_j),j}, P_{1,j}) = \chi(\mathcal{Q}_{\sigma,i}) + \chi(S_{i,j}, P_{i,j}) + \eta(\beta_{\sigma,i}, \upsilon'_j),$$

$$|\mathcal{Q}_{\psi(\sigma,\sigma'_j),j}| = |\mathcal{Q}_{\sigma,i}| + 1, \qquad \beta_{\psi(\sigma,\sigma'_j),j} = i, \qquad \tau_{\psi(\sigma,\sigma'_j),j} = \sigma.$$

Correctness of this algorithm follows from the fact that each possible segment ending at $i+1$ is considered before the $(i+1)$-st iteration. For each segment and each labelling, at least one optimal polyline with that labelling and leading to the beginning of that segment must have been considered, by the inductive assumption. Since the number of crossings in a polyline depends only on the crossings within the segments and the labelings where the segments meet, the inductive hypothesis is maintained through the $(i+1)$-st iteration. It is also trivially true in the basis case $i = 1$. With the exception of computing the crossing number for all of the segments, the algorithm requires $O(n)$ time and space to update the remaining information in each iteration. The final post-processing step is to determine $\sigma_{max} = \arg\max_\sigma \chi(\mathcal{Q}_{\sigma,n})$, finding the approximation that has the best crossing number. We use the β and τ information to reconstruct $\mathcal{Q}_{\sigma_{max},n}$ in $O(k)$ time.

The algorithm requires $O(n)$ space in each iteration and $O(n \log^{4/3} n)$ time per iteration to compute crossings of each batch of segments dominates the remaining time per iteration. Thus for simple polylines, $\mathcal{Q}_{\sigma_{max},n}$ is computable in $O(n^2 \log^{4/3} n)$ time and $O(n)$ space and for monotonic polylines it is computable in $O(n^2\sqrt{\log n})$ time and $O(n)$ space.

5 Results

Here we present results of applying our method to approximate x-monotonic data with and without noise in a parameter-free fasion.

Our first point set is given by $p = (x, x^2 + 10 \cdot \sin x)$ for 101 equally spaced points $x \in [-10, 10]$. The maximal-crossing approximation for this point set has 5 points and 7 crossings. We generated a second point set by adding standard normal noise generated in MATLAB with *randn* to the first point set. The maximal-crossing approximation of the data with standard normal noise has 19 points and a crossing number of 54. We generated a third point set from the first by adding heavy-tailed noise consisting of standard normal noise for 91 data points and standard normal noise multiplied by ten for ten points selected uniformly at random without replacement. The maximal-crossing approximation of the signal contaminated by heavy-tailed noise has 20 points and a crossing number of 50, which is quite comparable to the results obtained with the uncontaminated normal noise. These results are shown in Figure 4.

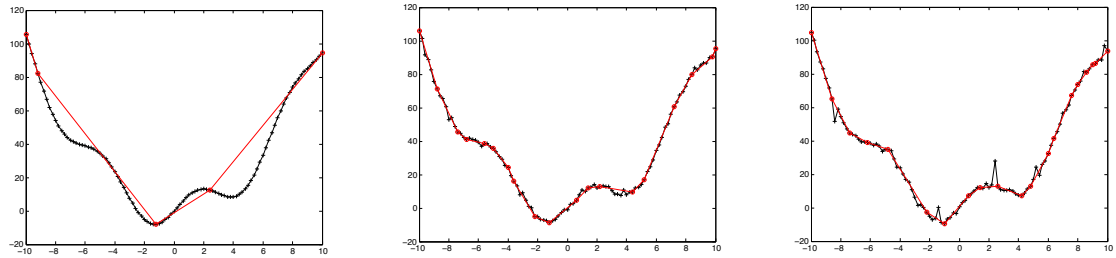


Fig. 4: The optimal crossing path for $p = (x, x^2 + 10 \cdot \sin x)$, without noise, with standard normal noise and with heavy-tailed (mixed gaussian) noise.

As can be seen in Figure 4, the crossing-maximization procedure gives a much closer approximation to the signal when there is some nonzero amount of noise present to provide opportunities for crossings. It is reasonable to conclude that, in the case of a clean signal, we could obtain a more useful approximation by artificially adding some noise centered at zero before computing our maximal-crossing polyline. However, to do so requires choosing an appropriate distribution for the added noise, and we wish to keep our procedure free of any parameters.

6 Discussion and Conclusions

The maximal-crossing approximation is robust to small changes of x- or y-coordinates of any p_i when the points are in general position. This robustness can be seen by considering that the crossing number of every approximation depends on the arrangement of lines induced by the line segments, and any point in general position can be moved some distance ϵ without affecting the combinatorial structure of the arrangement. The approximation is also invariant under affine transformations because these too do not modify the combinatorial structure of the arrangement. For x-monotonic polylines, the approximation possesses another useful property: the more a point is an outlier, the less likely it is to be included in the approximation. To see this, consider increasing the y-coordinate of any point p_i to infinity while x-monotonicity remains unchanged. In the limit, this will remove p_i from the approximation. That is, if p_i is initially in the approximation, then once p_i moves sufficiently upward, the two segments of the approximation adjacent to p_i cease to cross any segments of P.

An additional improvement in speed is achievable by bounding sequence lengths. If a parameter m is chosen in advance such that we require that the longest segment considered can span at most $m-2$ vertices, with the appropriate changes, the algorithm can then find the minimum sized approximation conditional on maximum crossing number and having a longest segment of length at most m in $O(nm\log^{4/3} m)$ time for simple polylines or $O(nm\sqrt{\log m})$ time for monotonic polylines, both with linear space. As it seems natural for long segments to be rare in good approximations, setting m to a relatively small value should still lead to good approximations while significantly improving speed.

Acknowledgements

The authors are grateful to Timothy Chan for discussions on the complexity of our solution in the Word RAM model that helped reduce the running times.

References

1. P. K. Agarwal, S. Har-Peled, M. H. Mustafa, and Y. Wang. Near-linear time approximation algorithms for curve simplification. In *ESA*, volume 2461 of *LNCS*, pages 195–202. Springer Verlag, 2002.
2. P. K. Agarwal and K. R. Varadarajan. Efficient algorithms for approximating polygonal chains. *Discrete and Computational Geometry*, 23:273–291, 2000.
3. H. Alt and L.J. Guibas. Discrete geometric shapes: Matching, interpolation, and approximation. In J.-R. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*, pages 121–153. Elsevier, 2000.
4. T. M. Chan and M. Pătraşcu. Counting inversions, offline orthogonal range counting and related problems. In *SODA*, pages 161–173, 2010.
5. W.S. Chan and F. Chin. Approximation of polygonal curves with minimum number of line segments. In *ISAAC*, volume 650 of *LNCS*, pages 378–387. Springer, 1992.
6. D. Douglas and T. Peucker. Algorithms for the reduction of points required to represent a digitised line or its caricature. *The Canadian Cartographer*, 10:112–122, 1973.
7. Y. Han. Deterministic sorting in $o(n\log\log n)$ time and linear space. *J. of Algorithms*, 50:96–105, 2004.
8. Y. Han and M. Thorup. Integer sorting in $o(n\sqrt{\log\log n})$ expected time and linear space. In *FOCS*, pages 135–144, 2002.
9. J. Hershberger and J. Snoeyink. Cartographic line simplification and polygon csg formulae in $o(n\log^* n)$ time. *Computational Geometry: Theory and Applications*, 11(3-4):175–185, 1998.
10. H. Imai and M. Iri. Polygonal approximation of curve-formulations and algorithms. In G. T. Toussaint, editor, *Computational Morphology*, pages 71–86. North-Holland, 1988.
11. A. Melkman and J. O'Rourke. On polygonal chain approximation. In G. T. Toussaint, editor, *Computational Morphology*, pages 87–95. North-Holland, 1988.
12. S. S. Skiena. *The Algorithm Design Manual.* Springer, 2^{nd} edition, 2008.