社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2004－AL－96 (8)
2004／7／27

距離和最小化基準による点集合の折れ線近似

ボリス アロノフ（ポリテクニク大学）浅野 哲夫（北陸先端科学技術大学院大学）加藤 直樹（京都大学）
クルト メルホルン（マックスプランク情報学研究所）徳山 豪 (東北大学)

概要. 平面上の点集合を折れ線で近似する問題において、距離の和、もしくは２乗和を最小化する最適化基準について述べる。折れ線が１つの関節点を持つ場合の効率的なアルゴリズムと、一般のｋ関節点の場合の近似アルゴリズムの設計を行う。

Polyline Fitting of Planar Points under Min-Sum Criteria

Boris Aronov (Polytechnic University,USA) *aronov@cis.poly.edu*
Tetsuo Asano (JAIST) *t-asano@jaist.ac.jp*
Naoki Katoh (Kyoto University) *naoki@archi.kyoto-u.ac.jp*
Kurt Mehlhorn (Max-Planck-Institut für Informatik) *mehlhorn@mpi-sb.mpg.de*
Takeshi Tokuyama(Tohoku University) *tokuyama@dais.is.tohoku.ac.jp*

Abstract. Fitting a curve of a certain type to a given set of points in the plane is a basic problem in statistics and has numerous applications. We consider fitting a polyline with k joints under the min-sum criteria with respect to L_1- and L_2-metrics, which are more appropriate measures than uniform and Hausdorff metrics in statistical context. We present efficient algorithms for the 1-joint versions of the problem, and fully polynomial-time approximation schemes for the general k-joint versions.

1 Introduction

Curve fitting aims to approximate a given set of points in the plane by a curve of a certain type. This is a fundamental problem in statistics, and has numerous applications. In particular, it is a basic operation in *regression analysis*. *Linear regression* approximates a point set by a line, while *non-linear regression* approximates it by a non-linear function in a given family.

In this paper, we consider the case where the points are fitted by a polygonal curve (*polyline*) with k joints, see Figure 1. This is often referred to as *polygonal approximation* or *polygonal fitting* problem. It is used widely. For example, it is commonly employed in scientific and business analysis to represent a data set by using a polyline with a small number of joints. The best representation is the polyline minimizing the error of approximation. Error is either defined as the maximum (vertical) distance of any input point from the polyline (min-max-optimization) or the sum of vertical distances (min-sum-approximation). In either case, distance is measured in some norm. We follow common practice and restrict ourselves to norms L_1 and L_2.

Min-max-approximation by a polyline is well studied. A popular formulation is the vertical distance minimization in which we minimize the *maximum* of the vertical distance (called the *uniform metric* or *Chebyschev error function*) between the points and the curve. Hakimi and Schmeichel gave a $O(n^2 \log n)$ time algorithm for this problem [5], and the time complexity has been later improved to $O(n \log n)$ [4]. Another popular approach is to minimize the Hausdorff measure that is the maximum of the Euclidean distances between the points and the output curve. This problem can also be solved in polynomial time [12]. These problems are closely related to *curve simplification*, in which the input is a polyline with n edges rather than a set of n points; this question arises in geographic information systems (GIS, see a survey [15]), and has received much attention in computational geometry [3, 6, 9, 11].

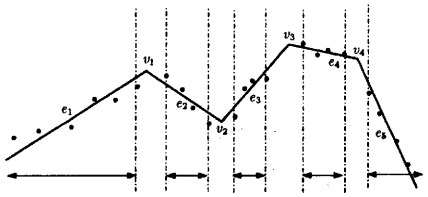

Figure 1: A 4-joint polyline fitting a set of points.

The minimize-the-maximum (*min-max*) formulation is useful in pattern recognition applications. However, in applications to statistics, its serious deficiency is fragility in presence of outliers. Even a single outlier can drastically change the output, while outliers, real or imagined, usually exist in statistical data. For this reason, minimize-the-sum-of-errors (*min-sum*) methods are considerably more popular in statistics: The most basic one is the least-squares method that minimizes the sum of the squares of vertical distances between the input points and the output curve. In this paper, we call it the *L_2-fitting* problem (the term *least-square fitting* is commonly used as well), and *regression line* in statistics usually refers to the L_2-fitting line. If the output curve is either a straight line or a low-degree algebraic curve, it is quite easy to compute the optimal L_2-fit. Another criterion is the L_1-minimization, in which we minimize the sum of vertical distances from the candidate curve to the points being fitted. L_1 fitting is more resilient to outliers than L_2 fitting; however, it is usually more expensive (or complicated) to compute the optimal solution. For linear regression, the L_1-optimal line can be computed in linear time [7] by using sophisticated computational techniques. Several other formulations have been proposed for further reducing the effect of outliers on the linear regression. Repeated median regression is a well-known example, and efficient solutions are known for several other criteria [10].

In this paper, we focus on the L_1- and L_2-fitting problems when the desired curve is a k-joint polyline; in other words, it is a continuous piecewise-linear x-monotone curve with $k+1$ linear components. We assume that a coordinate system is fixed, and the input points are sorted with respect to their x-coordinate values. To the authors' knowledge, the computational complexity of the optimal k-joint problem under either of these minimization criteria has not been previously investigated. More specifically, it seems that an efficient solution of the L_1-fitting problem extending the result of Imai *et al.* [7] is theoretically challenging even for the 1-joint problem.

In this paper, we begin by considering the 1-joint problem. We give algorithms of complexity $O(n)$ and $\tilde{O}(n^{1.4})$ time for the L_2 and L_1 criteria, respectively.[1] The L_2-fitting algorithm is simple and practical, whereas the L_1-fitting algorithm depends on using a semi-dynamic range search data structure and parametric search. For general k, we describe approximation schemes. Let z_{opt} be the minimum fitting error for a k-joint polyline and let ϵ be a positive constant. We give a polynomial-time approximation scheme (PTAS) to compute a $\lfloor(1+\epsilon)k\rfloor$-joint fitting whose error is at most z_{opt} and a fully polynomial-time approximation scheme (FPTAS) to compute a k-joint polyline with $(1+\epsilon)z_{\mathrm{opt}}$ fitting error, and consequently show that the problems cannot be strongly NP-hard, although their NP-hardness remains open.

Intuitively, why are the problems we consider in this paper more difficult than some related questions? We have mentioned that the uniform metric fitting problem can be solved efficiently. The key point is that its corresponding decision problem to determine whether there exists a k-joint polyline with a uniform error less than a given value w is geometrically a *stabbing* problem. The k-joint path must go through n vertical line segments of length $2w$ centered at the input points, and we can continuously move a feasible polyline such that each link becomes extremal in geometric sense, that is, goes through a pair of endpoints of vertical segments. Hence, we can design a polynomial-time algorithm. The L_1- and L_2-fitting problems seem to be more subtle: We do not have reformulation of the corresponding decision problems in terms of stabbing.

We remark that the curve simplification prob-

[1] We write $f(n) = \tilde{O}(g(n))$ if there exists an absolute constant $c \geq 0$ such that $f(n) = O(g(n)\log^c n)$.

lem under the L_1-measure is to minimize the area between input and output polylines. In a restricted case where the vertex set or the set of lines supporting edges of the output polyline is a subset of that of the input polyline, the problem is reduced to the k-link shortest path problem in a graph. In particular, if the input polyline is convex, this problem is related to matrix searching (see [1]). However, for the general case the authors are not aware of an efficient algorithm.

2 Preliminaries

A k-joint polyline is an alternating sequence $P = (e_1, \mathbf{v}_1, e_2, \mathbf{v}_2, \ldots, e_k, \mathbf{v}_k, e_{k+1})$ of line segments (*links*) and joint vertices (or simply *joints*), where e_s and e_{s+1} have $\mathbf{v}_s$ as a shared endpoint, for $s = 1, 2, \ldots, k$, and e_1 and e_{k+1} are infinite rays. We denote the link e_s on line $y = a_s x - b_s$ by (a_s, b_s) if the interval of the values of x corresponding to the link is understood. A joint $\mathbf{v}_s$ is represented by the pair (u_s, v_s) of its coordinate values. Thus, the connectivity and monotonicity of the polyline can be guaranteed by requiring that $v_s = a_s u_s - b_s = a_{s+1} u_s - b_{s+1}$, for $s = 1, 2, \ldots, k+1$, and $u_1 < \ldots < u_k$.

We now formulate the problem of fitting of k-joint polyline to an n-point set. Given a set of points $S = \{p_1 = (x_1, y_1), p_2 = (x_2, y_2), \ldots, p_n = (x_n, y_n)\}$ with $x_1 < x_2 < \cdots < x_n$ and an integer k, and setting $u_0 = -\infty$ and $u_{k+1} = \infty$ for convenience, find a polyline $P = ((a_1, b_1), (u_1, v_1), (a_2, b_2), (u_2, v_2), \ldots, (u_k, v_k), (a_{k+1}, b_{k+1}))$ minimizing one of the following three quantities for L_1-, L_2-, and uniform (i.e. L_∞) metric fitting, respectively:
$L_1 : \sum_{s=1}^{k+1} \sum_{u_{s-1} < x_i \le u_s} |a_s x_i - b_s - y_i|$
$L_2 : \sum_{s=1}^{k+1} \sum_{u_{s-1} < x_i \le u_s} (a_s x_i - b_s - y_i)^2$, and
$L_\infty : \max_{1 \le s \le k+1} \{\max_{u_{s-1} \le x_i \le u_s} |a_s x_i - b_s - y_i|\}$.

For $k = 0$, the problems are linear regression problems. The L_2-linear regression is well known as the *Gaussian least-squares method*. Once we compute $\mathrm{A}_n = \sum_{i=1}^n x_i$, $\mathrm{B}_n = \sum_{i=1}^n y_i$, $\mathrm{C}_n = \sum_{i=1}^n x_i^2$, $\mathrm{D}_n = \sum_{i=1}^n x_i^2$, and $\mathrm{E}_n = \sum_{i=1}^n x_i y_i$ in linear time, we can construct an optimal fitting line $y = ax - b$ by considering the partial derivatives of the objective function and solving a 2×2 system of linear equations. The linear regression problem with respect to the uniform error is to find a pair of parallel lines at the minimum vertical distance that contain all the given points between them. This can be done by applying the *rotating caliper method* that computes antipodal pairs of points on the convex hull of the point set. For an x-sorted point set this can be done in $O(n)$ time [13]. The L_1-linear regression problem is more involved; however, a linear-time algorithm has been devised by Imai *et al.* [7] based on Megiddo's prune-and-search paradigm.

3 Fitting a 1-joint polyline

We consider the problem of fitting a 1-joint polyline to a set of points. We proceed in two steps. We first assume that the joint vertex lies in a fixed interval $[x_q, x_{q+1}]$ and later eliminate this assumption. Let $S_1(q) = \{p_1, p_2, \ldots, p_q\}$ and $S_2(q) = \{p_{q+1}, \ldots, p_n\}$. Our objective polyline consists of two links lying on lines $\ell_1 : y = a_1 x - b_1$ and $\ell_2 : y = a_2 x - b_2$, respectively. We call a tuple (a_1, b_1, a_2, b_2) *feasible* if the two lines $y = a_1 x - b_1$ and $y = a_2 x - b_2$ meet at a point whose x-coordinate $u = \frac{b_1 - b_2}{a_1 - a_2}$ lies in the interval $[x_q, x_{q+1}]$. Our goal here is to find a feasible tuple (a_1, b_1, a_2, b_2) representing a 1-joint polyline minimizing (for a given q)
$\sum_{i=1}^q |a_1 x_i - b_1 - y_i| + \sum_{i=q+1}^n |a_2 x_i - b_2 - y_i|$ and
$\sum_{i=1}^q (a_1 x_i - b_1 - y_i)^2 + \sum_{i=q+1}^n (a_2 x_i - b_2 - y_i)^2$, for L_1- and L_2-fitting, respectively.

Lemma 3.1. *For either L_1- or L_2-fitting criterion, the 1-joint problem for a fixed q reduces to solving two convex programming problems.*

Proof. Disregarding the feasibility constraint, the problem is clearly a quadratic programming problem for the L_2 case and a linear programming problem for the L_1 case. The feasibility constraint requiring that the lines ℓ_1 and ℓ_2 meet in the strip between x_q and x_{q+1} can be expressed by different linear constraints depending on whether $a_1 \le a_2$. Thus, we can decompose the (L_1 or L_2) problem into two subproblems. If $a_1 \le a_2$, the lines meet in the strip if and only if ℓ_1 is not below ℓ_2 at x_q and is not above at x_{q+1}. Thus, the

additional constraint becomes

$$x_q(a_2 - a_1) \leq b_2 - b_1 \leq x_{q+1}(a_2 - a_1). \quad (1)$$

In the opposite case, the additional constraint is

$$x_{q+1}(a_2 - a_1) \leq b_2 - b_1 \leq x_q(a_2 - a_1). \quad (2)$$

Clearly, each subproblem is a convex programming problem, as claimed. □

From the above lemma, it is clear that the optimal 1-joint polyline can be computed by using linear/quadratic programming. However, we can design combinatorial algorithms. Indeed, we can classify the solution into two cases: (1) An inequality in (1), (2) holds with equality. (2) All of the inequalities in (1), (2) are strict. We call the solution *fixed* in the former case and *free* otherwise. From the form of the expressions in (1), (2) we deduce the following simple observation.

Lemma 3.2. *If the solution is fixed, the joint is located on either of the two vertical lines $x = x_q$, $x = x_{q+1}$.*

If the joint is on the line $x = x_{q+1}$, we can regard it as a solution for the partition into $S_1(q+1) = S_1(q) \cup \{p_{q+1}\}$ and $S_2(q+1) = S_2(q) \setminus \{p_{q+1}\}$. Thus, for each partition, we essentially need to solve two subproblems: (1) the free problem and (2) the fixed problem where the joint is on the vertical line $x = x_q$. This leads to the following generic algorithm: For each partition of S into two intervals S_1 and S_2, we first consider the free problem ignoring the constraints, and check whether the resulting solution is feasible or not, i.e., we verify that (1) or (2), as appropriate, is satisfied. If it is feasible, it is the best solution for the partition. Otherwise, we consider the fixed solution adding the constraint that the joint lie on $x = x_q$, and report the solution for the partition. After processing all $n-1$ possible partitions, we report the solution with the smallest error.

If it takes $O(f(n))$ time to process a subproblem for each partition, the total time complexity is $O(nf(n))$. For efficiency, we design a dynamic algorithm to process each partition so that $f(n)$ is reduced in the amortized sense.

3.1 The L_2 1-joint problem

We show how to construct an optimal L_2-fitting 1-joint polyline in linear time. We process the partitions $(S_1(q), S_2(q))$ starting from $q = 1$ to $q = n - 1$, in order. We maintain the sums, variances, and covariances $\mathrm{A}_q = \sum_{i=1}^{q} x_i$, $\mathrm{B}_q = \sum_{i=1}^{q} y_i$, $\mathrm{C}_q = \sum_{i=1}^{q} x_i^2$, $\mathrm{D}_q = \sum_{i=1}^{q} y_i^2$, and $\mathrm{E}_q = \sum_{i=1}^{q} x_i y_i$ incrementally, at constant amortized cost. They also provide us the corresponding values for $S_2(q)$ if we precompute those values for S, i.e., $\sum_{i=q+1}^{n} x_i = \mathrm{A}_n - \mathrm{A}_q$ etc.

For the free case, the objective function is separable, in the sense that the optimal solution can be identified by finding (a_1, b_1) minimizing $\sum_{i=1}^{q}(a_1 x_i - b_1 - y_i)^2$ and (a_2, b_2) minimizing $\sum_{j=q+1}^{n}(a_2 x_j - b_2 - y_j)^2$ independently. This can be computed in $O(1)$ time from the values of $\mathrm{A}_q, \ldots, \mathrm{E}_q$ as explained in section 2. The feasibility check of the solution is done in $O(1)$ time by computing the intersection point of the corresponding pair of lines. It remains to solve the subproblems with the additional constraint that the joint is at $x = x_q$. This can be solved by using a standard method such as the Lagrange's method of indeterminate coefficients.

Theorem 3.3. *L_2-optimal 1-joint fitting can be computed in linear time.*

3.2 The L_1 1-joint problem

3.2.1 Semi-dynamic L_1 linear regression

We start with the problem of computing the optimal linear L_1-fitting (i.e., linear regression) of the input point set, i.e., we seek the line $\ell_{\mathrm{opt}}: y = ax - b$ minimizing $\sum_{i=1}^{n} |ax_i - b - y_i|$.

The difficulty with the L_1-fitting problem is that, written in linear programming terms, it has $n + 2$ variables, in contrast to the least-squares case where the problem is directly solved as a bivariate problem. Nonetheless, the problem can solved by brute-force in $O(n^3)$ time, since there are $O(n^2)$ possible linear dissections of the point set which can be enumerated in $\Theta(n^2)$ worst-case time by constructing the dual arrangement, and we can compute the optimal line in linear time once the dissection by the line is given (this algo-

rithm can be easily sped up to constant or near-constant amortized time per dissection). Moreover, it can be easily seen that the optimal line bisects S into two equal-size subsets; in other words, the line is a halving line. Using this fact, Imai *et al.* [7] devised an optimal linear-time algorithm for computing ℓ_{opt} based on the multidimensional prune-and-search paradigm.

In order to design an efficient algorithm for the 1-joint fitting problem, we consider a semi-dynamic version of the L_1 linear regression for a point set P with low amortized time complexity, where we dynamically maintain P with insertions and deletions under an assumption that P is always a subset of a fixed universe S of size n that is given from the outset. (In fact, for our application, it is sufficient to be able to start with $P = \emptyset$ and handle only insertions, and to start with $P = S$ and handle only deletions. Moreover, the order of insertions and deletions is known in advance. The data structure we describe below is more general.)

Consider the dual space, with $p_i = (x_i, y_i)$ transformed to the dual line $Y = f_i(X)$ where $f_i(X) = x_i X - y_i$. The line $y = ax - b$ is transformed to the point (a, b) in the dual space. The kth *level* of the arrangement $\mathcal{A} = \mathcal{A}(S^*)$ of the set S^* of dual lines is the trajectory of the kth largest value among $f_i(X)$. We call the $\lceil n/2 \rceil$th level the *median level.*

Lemma 3.4 (Imai *et al.* [7]). *If the optimal L_1-fitting line is given by $y = a_{\text{opt}}x - b_{\text{opt}}$, its dual point $(a_{\text{opt}}, b_{\text{opt}})$ is on the median level if n is odd, and between the median level and the $\lfloor (n+1)/2 \rfloor$th level if n is even.*

Now, given X-value t, consider the point $(t, f_i(t))$ for each $i = 1, 2, \ldots, n$, and let $F(t)$ to be the sum of the $\lfloor n/2 \rfloor$ largest values in $\{f_i(t) : i = 1, 2, \ldots, n\}$ and let $G(t)$ be the sum of the $\lfloor n/2 \rfloor$ smallest values in the same set. Put $H(t) = F(t) - G(t)$ and let a_{opt} be the value of t which minimizes H. The lemma below follows directly from the fact that, in any line arrangement, the portions of the lines lying on or below any fixed level k can be decomposed into k non-overlapping concave chains (see, for example, [2]).

Lemma 3.5. *$F(t)$ is a convex function, while $G(t)$ is a concave function. As a consequence, $H(t)$ is also convex. $H(t)$ has either slope 0 at $t = a_{\text{opt}}$ or its slope changes from negative to positive at $t = a_{\text{opt}}$.*

Suppose a fixed universe S^* of lines is given. We need a data structure that at any given moment maintains a subset $P^* \subseteq S^*$ and supports the following operations:

Slope-sum query For a query value t, return the sum of the slopes of the $\lfloor n/2 \rfloor$ highest lines at $X = t$.

Height-sum query For a query value t, return the sum of the Y-coordinates of the $\lfloor n/2 \rfloor$ highest lines at $X = t$. The height-sum query reduces to a slope-sum-query plus a constant-term-sum-query.

Update A line in S^* is added to or removed from P^*.

Similar query problems are given in [8] prviously. Suppose a data structure supporting such queries on a set $P^* \subseteq S^*$ of lines in $O(\tau(n))$ time is available, where $n = |S^*|$. Then we can query the slopes of F and G at t, and hence compute the slope of H at t in $O(\tau(n))$ time. Because of convexity of H, we have the following:

Lemma 3.6. *Given t, we can decide whether $t < a_{\text{opt}}$, $t > a_{\text{opt}}$, or $t = a_{\text{opt}}$ in $O(\tau(n))$ time.*

Thus, we can perform binary search to find a_{opt}. We will show below how to make the search for a_{opt} strongly polynomial. Once we know a_{opt}, we determine b_{opt} by determining the median level at $t = a_{\text{opt}}$.

Because of space limitation, we omit construction of the query data structure, and just declare that $\tau(n) = \tilde{O}(n^{0.4})$ is possible.

Lemma 3.7. *A linear-space data structure attaining $\tau(n) = \tilde{O}(n^{0.5})$ can be constructed in $O(n \log n)$ time. If we can use $O(n^{1.2})$ space, a data structure with $\tilde{O}(n^{0.4})$ query time is constructed in $\tilde{O}(n^{1.4})$ preprocessing time.*

3.2.2 Algorithm for L_1 1-joint fitting

Finally, we describe the algorithm to find the L_1-optimal 1-joint polyline fitting a set S of n points in the plane. Recall that there are two different types of solutions:

Type 1 There is an index q such that the 1-joint polyline consists of the optimal L_1-fitting line of $S_1(q) = \{p_1, p_2, \ldots, p_q\}$ and that of $S_2(q) = \{p_{q+1}, p_{q+2}, \ldots, p_n\}$.

Type 2 There is an index q such that the joint lies on the vertical line $x = x_q$.

If the optimal solution is of type 1, we compute an optimal L_1-fitting line for $S_1(q)$ and $S_2(q)$ separately, for every $q = 1, 2, \ldots, n$, by using the semi-dynamic algorithm with S as the universe. If we use quasi-linear space $\tilde{O}(n)$, the time complexity is $\tilde{O}(n^{1.5})$, and if we use $O(n^{1.2})$ space, the time complexity is $\tilde{O}(n^{1.4})$.

Otherwise, the optimal solution is of type 2. For each q, we guess the y-coordinate value η of the joint vertex (x_q, η). Then, we can compute the best line, in the sense of L_1 fitting, approximating $S_1(q)$ going through the (for now, fixed) joint by using almost the same strategy as in section 3.2.1. Indeed, it suffices to determine the slope of this line. In the dual space, we just need to compute a point $p = (a(p), b(p))$ on the line $Y = x_q X + \eta$ such that $\sum_{i=1}^{q} |a(p)x_i - b(p)|$ is minimized. We observe that the above function is convex if it is regarded as a function of a, and hence $\theta(p) = \theta^+(p) - \theta^-(p)$ is monotone and changes the sign at p, where $\theta^+(p)$ $(\theta^-(p))$ is the sum of slopes of lines above p (resp. below p). Thus, we can apply binary search by using slope-sum query, and this binary search can be performed in $O(\log n)$ steps by using the filtration as described above.

Moreover, because of the convexity of the objective function, once we know the optimal solution for a given η_0, we can determine whether the optimal value η is greater than η_0 or not by using the height-sum query. Indeed, when we infinitesimally slide η_0, the gain (or loss) of the objective function can computed from the slope sums and height sums associated with each of the sets of points lying above, below, and on the current polyline (for each of $S_1(q)$ and $S_2(q)$).

Thus, we can apply binary search for computing the optimal value of η. In order to construct a strongly polynomial algorithm, we apply parametric search. Note that given η, our algorithm has a natural parallel structure inherited from the range-searching algorithms, and runs in polylogarithmic time using $\tilde{O}(\tau(n))$ processors. Thus, popular framework for the parametric searching [14] works. Therefore, for a fixed q, the second case of the problem is computed in $\tilde{O}(\tau(n))$ time. Thus, we have the following:

Theorem 3.8. *The optimal L_1-fitting 1-joint polyline is computed in $\tilde{O}(n^{1.5})$ randomized time using linear space, and $\tilde{O}(n^{1.4})$ randomized time using $O(n^{1.2})$ space.*

4 Fitting a k-joint polyline

The k-joint fitting problem is polynomial-time solvable if k is a fixed constant. We describe the algorithm in a non-deterministic fashion. We guess the partition of $x_1, \ldots, x_n$ into k intervals each of which corresponds to a line segment in the polyline. Also, we guess whether each joint is free or fixed. We decompose the problem at the free joints and have a set of subproblems. In each subproblem, we give the linear constraints corresponding to the fixed condition (i.e., each joint is located on a guessed vertical line). Thus, each subproblem is a convex programming problem: a linear program for L_1, and a quadratic program for L_2. We solve each subproblem separately to obtain the solution of the whole problem. Note that this strategy works because of the convexity of each subproblem. There are $O((3n)^k)$ different choices of the guesses, thus we can be replace guessing by a brute-force search to have a polynomial-time deterministic algorithm if k is a constant.

For a general k, we do not know whether the problem is in the class P or not. Thus, we consider approximation algorithms. One possible approach is to relax the requirement that the number of joints is exactly k. We can design a PTAS for it.

Theorem 4.1. *Let z_{opt} be the optimal L_1 (or L_2) error of a k-joint fitting. Then, for any constant $\epsilon > 0$, we can compute a $\lfloor(1+\epsilon)k\rfloor$-joint fitting whose error is at most z_{opt} in polynomial time.*

Proof. We ignore continuity and approximate the points by using a piecewise-linear (not necessarily continuous) function with k linear pieces. This can be done by preparing the optimal linear regression for each subinterval of consecutive points of S, and then applying dynamic programming. We can restore the continuity by inserting at most k steep (nearly vertical) line segments. The resulting polyline has at most $2k$ joints and error at most z_{opt}. We can improve $2k$ to $\lfloor\frac{3k}{2}\rfloor$ by applying the 1-joint algorithm instead of linear regression algorithm, and further improve to $\lfloor(1+\epsilon)k\rfloor$ by using the r-joint algorithm mentioned above for $r = \lceil\epsilon^{-1}\rceil$. □

Another approach is to keep the number of joints at k and approximate the fitting error. We give a FPTAS for it. We only discuss the L_1 case, since the L_2 case is analogous. Let z_{opt} be the optimal L_1-error, and we aim to find a k-joint polyline whose error is at most $(1+\epsilon)z_{\mathrm{opt}}$. We remark that if $z_{\mathrm{opt}} = 0$, our solution is exactly the same as the solution for the uniform metric fitting problem, and thus we may assume $z_{\mathrm{opt}} > 0$. Recall that the uniform metric fitting problem can be solved in $O(n \log n)$ time [4]. The following is a trivial but crucial observation:

Lemma 4.2. *Let z_∞ be the optimal error for the uniform metric k-joint fitting problem. Then, $z_\infty \leq z_{\mathrm{opt}} \leq n z_\infty$.*

Our strategy is as follows: We call the n vertical lines through our input points the *column lines*. We give a set of *portal points* on each column line, and call a k-joint polyline a *tame polyline* if each of its links satisfies the condition that the line containing the link goes through a pair of portal points.

On each column line, the distance between its data point and the intersection point with the optimal polyline is at most z_{opt}, thus at most nz_∞. Thus, on the ith column line, we place the portals in the vertical range $[y_i - nz_\infty, y_i + nz_\infty]$. The portal points are placed symmetrically above and below y_i. The j-th portal above y_i is located at the y-value $y_i + (1+\frac{\epsilon}{2})^{j-1}\delta$, where $\delta = \frac{z_\infty \epsilon}{2n}$ and $j = 1, 2, \ldots, M$. We choose M maximal with $(1+\frac{\epsilon}{2})^M \delta \leq nz_\infty$, and hence $M = O(\epsilon^{-1}\log(n + \epsilon^{-1}))$. We also put portals at heights y_i and $y_i \pm nz_\infty$. In this way the number of portals in any column is at most $2M+3$. We call a closed interval between adjacent portals in a column a *prime interval*.

Lemma 4.3. *There exists a tame polyline whose L_1 error is at most $(1+\epsilon)z_{\mathrm{opt}}$.*

Proof. We start from the optimal polyline ℓ_{opt}, and deform it to obtain a tame polyline. We proceed sequentially, left to right. Consider the line containing the leftmost link of ℓ_{opt}. We continuously move the line to a tame line without crossing any portal point during the movement; if the line started off passing through a portal point, we rotate it around it; if the line started off passing through two, it is already tame. The right joint of the current link is accordingly moved to the intersection of the new line and the line containing the right neighbor link. It may happen that during this transformation a joint crosses a column line. However, the intersection points of the original and the deformed polylines with a column line are located in a common prime interval. We repeat this operation, proceeding from left to right, to obtain a tame polyline.

Now, consider the change of vertical distances between a point p_i and the two polylines. The polylines go through the same prime interval of the column line through p_i. An index i is called a near-index if the polylines goes through a prime interval containing y_i as its endpoint; otherwise it is called a far-index. For the near-indices, the summation of the errors caused by the new polyline is bounded by $n\delta = \frac{z_\infty \epsilon}{2}$. For each far-index, the errors caused by the new polyline at the column is bounded by $(1+\frac{\epsilon}{2})$ times the one caused by the old (i.e. optimal) polyline. Thus, the total error of the new polyline for all the far indices is at most $(1+\frac{\epsilon}{2})z_{\mathrm{opt}}$. In total, the error of the new polyline is bounded by $(1+\frac{\epsilon}{2})z_{\mathrm{opt}} + \frac{z_\infty \epsilon}{2} \leq (1+\epsilon)z_{\mathrm{opt}}$. □

Thus, it suffices to compute the optimal tame polyline. There are Mn portals, and thus $N = O(M^2n^2)$ lines going through a pair of portals. Let $\mathcal{L}$ be the set of these lines. We design a dynamic programming algorithm. For the i-th column, for each line $\ell \in \mathcal{L}$ and each $m \leq k$, we record the approximation error of the best m-joint tame polyline up to the current column whose (rightmost) link covering p_i is on ℓ. When we proceed to the $(i+1)$-th column, each approximation error is updated. If there is an intersection between lines ℓ and ℓ' in the interval $(x_i, x_{i+1}]$, we consider the polylines that have the intersection as a possible joint. This can be done by copying the data for ℓ to ℓ' and vice versa incrementing the join number by one, and then keeping the smaller of the current and the new (copied) error for each of the pairs (ℓ, m) and (ℓ', m) for $m = 1, 2, \ldots, k$. Then, we add the distance from p_{i+1} to each polyline. Finally, we select the minimum error at the n-th column, and retrieve the polyline by backtracking.

There are $O(N^2)$ intersections of lines, and it takes $O(k)$ time for each intersection for copying and updating. This requires $O(N^2k)$ work and dominates the running time. Since $N = O(n^2M^2) = O(n^2\epsilon^{-2}\log^2(n+\epsilon^{-1}))$, we have the following:

Theorem 4.4. *An $(1+\epsilon)$-approximation, i.e., a k-joint polyline with error $(1+\epsilon)$ times the optimal, for each of the L_1 and L_2 k-joint problems can be computed in $O(kn^4\epsilon^{-4}\log^4(n+\epsilon^{-1}))$ time.*

Acknowledgment: The authors would like to thank Jirí Matoušek for a stimulating discussion on convexity.

References

[1] A. Aggarwal, B. Schieber, and T. Tokuyama, "Finding a minimum-weight k-link path in graphs with the concave Monge property and applications," *Discrete Comput. Geom.*, **12** (1994) 263–280.

[2] P. Agarwal, B. Aronov, T. Chan, M. Sharir, "On Levels in Arrangements of Lines, Segments, Planes, and Triangles," *Discrete Comput. Geom.*, **19** (1998) 315–331.

[3] P. K. Agarwal and K. R. Varadarajan, "Efficient algorithms for approximating polygonal chains," *Discrete Comput. Geom.*, **23** (2000) 273–291.

[4] M. Goodrich, "Efficient piecewise-linear function approximation using the uniform metric," *Discrete Comput. Geom.*, **14** (1995) 445–462.

[5] S. Hakimi and E. Schmeichel, "Fitting polygonal functions to a set of points in the plane," *Graphical Models and Image Processing*, **53** (1991) 132–136.

[6] H. Imai and M. Iri: "Polygonal approximations of a curve - Formulations and algorithms," *Computational Morphology*, Elsevier Science Publishers B.V. (North Holland), 1988, 71–86.

[7] H. Imai, K. Kato, and P. Yamamoto: "A linear-time algorithm for linear L_1 approximation of points," *Algorithmica*, **4** (1989) 77–96.

[8] N. Katoh and T. Tokuyama, "Notes on computing peaks in k-levels and parametric spanning trees," Proc. 17th ACM Symp. on Computational Geometry, 2001, pp. 241–248.

[9] Y. Kurozumi and W.A. Davis: "Polygonal approximation by the minimax method," *Computer Graphics and Image Processing*, **19** (1982) 248–264.

[10] S. Langerman and W. Steiger, "Optimization in arrangements." *Proc. Symp. Theor. Aspects Computer Science (STACS2003)*, LNCS 2607, 2003, pp. 50–61.

[11] A. Melkman and J. O'Rourke: "On polygonal chain approximation," *Computational Morphology*, Elsevier Science Publishers B.V. (North Holland), 1988, 87–95.

[12] J. O'Rourke and G. Toussaint, "Pattern Recognition," Chapter 43 of *Handbook of Discrete and Computational Geometry* (eds. J. Goodman and J. O'Rourke), CRC Press, 1997.

[13] F. P. Preparata and M. I. Shamos, *Computational Geometry, an Introduction*, Springer-Verlag, New York, 1985.

[14] J. Salowe, "Parametric Search," Chapter 37 of *Handbook of Discrete and Computational Geometry* (eds. J. Goodman and J. O'Rourke), CRC Press, 1997.

[15] Robert Weibel, "Generalization of Spatial Data: Principles and Selected Algorithms," *Algorithmic Foundations of Geographic Information Systems*, LNCS 1340, 1997, pp. 99–152.