This paper appeared in "Automata, Languages and Programming". Proceedings of the 19th International Colloquium on Automata, Languages, and Programming (ICALP 92), Wien, Austria, July 1992, ed. W. Kuich. Lecture Notes in Computer Science 623, Springer-Verlag, 1992, pp. 404–415.

A New Metric Between Polygons, and How to Compute it

(extended abstract)

Günter Rote

Technische Universität Graz, Institut für Mathematik,
Steyrergasse 30, A-8010 Graz, Austria
electronic mail: `rote@ftug.dnet.tu-graz.ac.at`

Abstract. The bounded Lipschitz distance can be used as a distance measure between polygons or between functions. It has several properties which make it attractive to use from the viewpoint of applications. There are several alternative definitions of the metric, some of which are interesting in their own right. We also give an algorithm for computing the metric, and we discuss possible implementations.

1 Introduction

1.1 Problem statement

For a given function $f\colon [0,1] \to \mathbb{R}$ we want to compute its *bounded Lipschitz norm*

$$\|f\|_{\mathrm{BL}} = \max\Big\{ \int_0^1 f(x)g(x)\,dx : |g(x)| \le M,\ |g(x)-g(y)| \le |x-y| \Big\} \qquad (1)$$

Here, M is a scaling parameter of the norm whose meaning will be explained later. In practice, f could be given by a sequence of values $(f_1,\ldots,f_n)$. In that case, the bounded Lipschitz norm looks as follows:

$$\|(f_1,\ldots,f_n)\|_{\mathrm{BL}} = \max\Big\{ \sum_{i=1}^{n} f_i g_i : |g_i| \le m,\ |g_{i+1}-g_i| \le 1 \Big\} \qquad (2)$$

For the new parameter m we have $m = M \cdot n$.

1.2 Why is it interesting to compute the distance between functions?

As defined above, the bounded Lipschitz norm is a *norm* on the vector space of all sequences or of all piecewise continuous functions. Of course, a norm gives rise to a *metric* $d_{\mathrm{BL}}(f,g)$ in the usual way:

$$d_{\mathrm{BL}}(f,h) = \|f-h\|_{\mathrm{BL}}.$$

This concept of *distance* between functions is where the applications are.

Pattern recognition: Functions of one parameter. In pattern recognition, it is often required to classify an "image" of an object by deciding which one of several given patterns looks most like it. In many cases, the image and the patterns can be specified as one-parametric functions, or, more concretely, as sequences of measured values. For example, in a vending machine that accepts money bills, an inserted bill is scanned by moving it through a beam of light, and the absorption of the light as it passes through the bill is recorded. The resulting light-and-dark pattern must be compared to each of several possible patterns.

Pattern recognition: Shapes. A shape might be given as a simple polygon or, more generally, as a simply connected region. There are several ways to represent this as a function of one variable. If the "image" polygon and the pattern polygons are *convex* bodies C, then one possibility is the support function h: $[0, 2\pi) \to \mathbb{R}$

$$h(\alpha) = \max\{\, x \cos \alpha + y \sin \alpha : (x, y) \in C \,\}.$$

Another possibility is to parameterize the boundary by arc length

$$\begin{pmatrix} x(t) \\ y(t) \end{pmatrix}, \quad \text{for } 0 \le t \le L,$$

with $\dot{x}(t)^2 + \dot{y}(t)^2 = 1$, and use the tangent direction α: $[0, L] \to [0, 2\pi)$ as the function that characterizes the curve:

$$\begin{pmatrix} \dot{x}(t) \\ \dot{y}(t) \end{pmatrix} = \begin{pmatrix} \cos \alpha(t) \\ \sin \alpha(t) \end{pmatrix},$$

This approach has for example been taken by Arkin et al. (1990), who considered the L_1 and L_2 norms of the resulting funtion $\alpha(t)$.

Other distances, including the well-known Hausdorff distance, have been investigated in this context, see Atallah (1983) and Cox et al. (1989).

Quality control. In the production of artificial fabric, one aspect of quality is that the fabric should be as smooth and uniform as possible. A real piece of fabric will always exhibit irregularities. To monitor the quality, the fabric is passed under a light beam and its thickness is continuously measured. Now it is important to assign a numerical value that "measures" the deviation of the measured thickness distribution of a piece of fabric from the ideal uniform thickness. In this application, it is important that this distance value is computed fast, because data arrive continuously at a high rate.

1.3 In what respect is the proposed distance different from other metrics?

If you take any L_p norm,

$$\left(\int_0^1 |f(x)|^p \, dx \right)^{1/p},$$

for your favourite value of p, this norm will not be able to distinguish between functions that have "the same values distributed in a different way over the domain". For example,

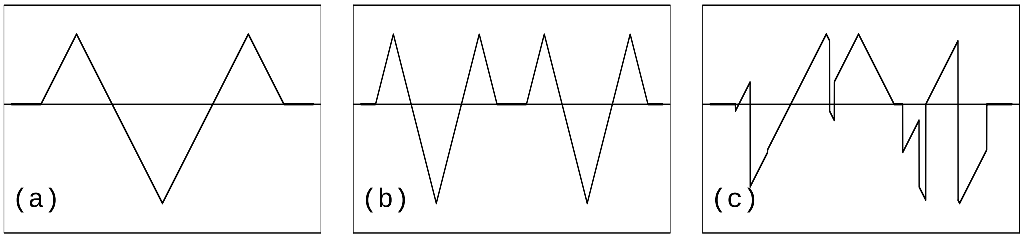

Fig. 1. Three functions which have the same L_p metrics.

in figure 1a, figure 1b, and figure 1c, the L_p norms will always be identical, although the functions look completely different. The same holds for the $L_{\max}$ or L_∞ norm,

$$\sup_{0 \le x \le 1} |f(x)|$$

Figure 1a has one big "hole", wherease figure 1b has two small holes. It is natural that small oscillations are more tolerable than big systematic irregularities, and several small holes should be better than one big hole of the same total "area". This is certainly true for the quality control of fabrics, at least.

Another distance measure is the *discrepancy*,

$$\max_{0 \le a \le b \le 1} \left| \int_a^b f(x)\, dx \right| ,$$

or the maximum sum of a contiguous subsequence of $(f_1, \ldots, f_n)$. The discrepancy does distinguish between big holes and small holes. However, contrary to the bounded Lipschitz norm, the discrepancy is influenced only by the biggest hole, and the smaller holes do not play any role at all. The situation is similar to the $L_{\max}$ norm.

The Hausdorff distance between polygons C and D,

$$\max \left\{ \max_{x \in C} \min_{y \in D} d(x, y),\ \max_{y \in D} \min_{x \in C} d(x, y) \right\} ,$$

suffers from the same limitation.

To my knowledge, the bounded Lipschitz norm is the only distance measure between functions that is both sensitive to locality on the x-axis and maintains a global view of the function on the whole interval. In other words, it distinguishes between big holes and small holes, and it still takes care of every hole, big and small.

1.4 Examples

We will illustrate the bounded Lipschitz norm of a function and not of a sequence, because this exhibits the essential properties of the bounded Lipschitz distance more clearly and more intuitively.

Figure 2a shows a function f and the corresponding function g which optimizes the integral in (1). In general, g will try to follow the trend of f, being as large as possible

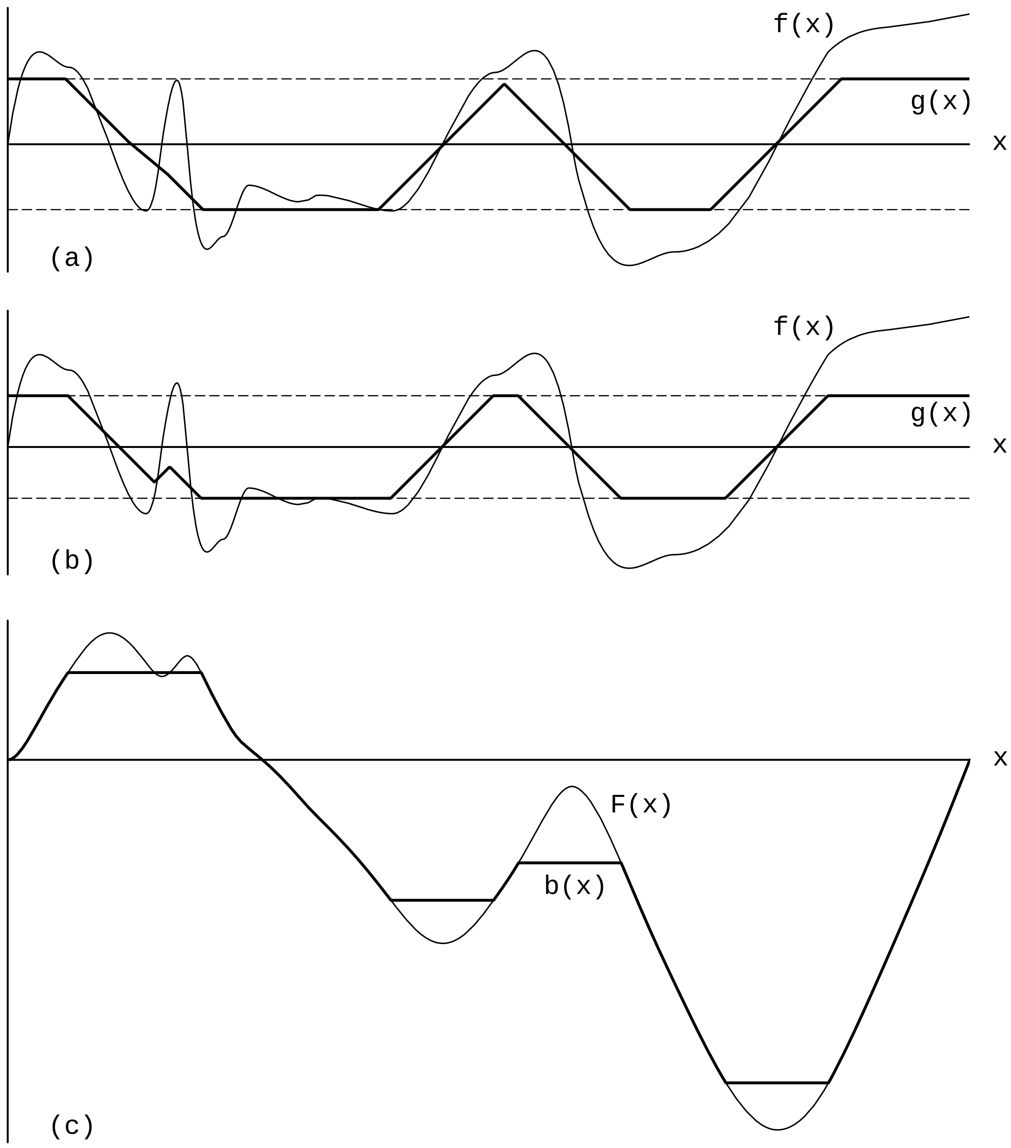

Fig. 2. An example. Part (c) shows the dual problem (section 2.2).

when f is positive and as small as possible when f is negative. We see that g cannot follow f too closely in the left part because f wiggles too fast, but the big wave in the right part is fully taken into account. Figure 2b shows what happens when the parameter M in the constraint $|g(x)| \leq M$ is decreased. Previously, the wave in the middle was too short for $g(x)$ to follow it completely, but now there is no longer a difference between the big right wave and the middle wave. Thus M is something like the "precision" of the norm or the intended "granularity" of the function f. If M goes to 0, the effect of the Lipschitz condition $|g(x) - g(y)| \leq |x - y|$ becomes more and more negligible, and $g(x)$ will be equal to $+M$ or $-M$ most of the time. Thus, we essentially approach the L_1 norm.

1.5 Relation to other work

As mentioned above, the idea of measuring the distance between polygonal shapes by computing a metric between functions has been used in Arkin et mult. al. (1990).

Neunzert and Wetton (1987) and Hackh (1990) proposed the bounded Lipschitz norm for applications in quality control. They discussed its advantages, but rejected it for practical purposes because they did not know how to compute it fast.

We can omit the boundedness condition $|g(x)| \leq 1$ in definition (1) and require only the Lipschitz condition $|g(x) - g(y)| \leq |x - y|$. This corresponds to making the parameter M very big (bigger than 1/2). (Of course we have to insist that $\int_0^1 f(x)\,dx = 0$, because otherwise the solution of (1) is unbounded.) The resulting norm becomes the Monge-Kantorovich mass transfer problem, whose history goes back to 18th century work of G. Monge (1781) and to L. Kantorovitch (1942). The bounded Lipschitz norm is a special case of the Kantorovich-Rubinstein norm which is known in functional analysis and probability theory, see Kantorovich and Rubinshteĭn (1958) or Rachev (1984, 1991). This relation is described in more detail in section 2.3.

1.6 Overview of the paper

In section 2 we give several alternate formulations of the problem; among them is a function approximation or function smoothing problem, and two network flow models. Section 3 develops an algorithm and discusses the possible options for data structures. Is section 4 we mention some open problems.

The main contribution of this paper is not only in the algorithmic part; here we use more or less standard techniques. It lies rather in the modeling aspect: We unify several seemingly different problems by showing their equivalence. This opens the way for deeper understanding of the problems, and algorithms for one of the problems can be translated into solutions for the others.

2 Different representations

Let us first mention the easily proved fact that we really have a norm:

Theorem 1. *The bounded Lipschitz norm is in fact a norm, and hence the bounded Lipschitz distance* d_{BL} *satisfies the triangle inequality.*

2.1 A network flow model

It is possible to translate the restrictions in (2) into a minimum cost network flow model as shown in figure 3. The flow on the arc from A_{i-1} to A_i is the variable g_i and it is restricted between the capacities $-m \leq g_i \leq +m$. The cost on this arc is f_i. On the arcs between A_i and B the flow can be at most 1 in either direction, except for A_0 and A_n, where the flow is unrestricted. This models the Lipschitz condition. The cost on these arcs is 0.

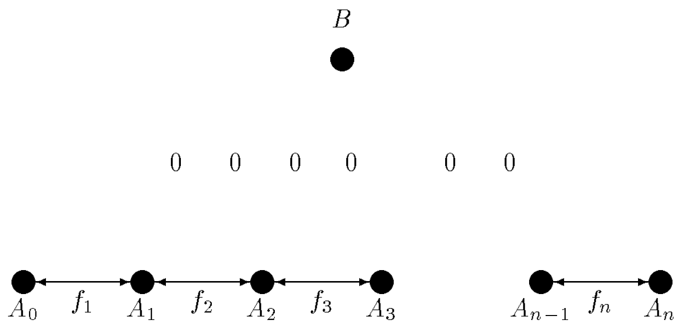

Fig. 3. The network flow model.

2.2 The dual problem: smoothing a function

The linear programming dual of problem (2), after some transformation, can be written as follows: Let F_i denote the partial sums of the sequence f_i:

$$F_k := f_1 + f_2 + \cdots + f_k \tag{3}$$

Then we have

$$\|(f_1, \ldots, f_n)\|_{\mathrm{BL}} = \min\Big\{\, m \cdot \sum_{i=1}^{n} |b_i - b_{i-1}| + \sum_{i=0}^{n} |b_i - F_i| : b_0 = F_0,\ b_n = F_n \,\Big\}. \tag{4}$$

This problem is of interest in itself because it can be interpreted as a problem of smoothing a function which is given by a sequence $(F_0, F_1, \ldots, F_n)$. We are looking for a "smooth" sequence $(b_0, b_1, \ldots, b_n)$ that approximates the given one. The right sum in the above expression (4) is the approximation error, and the left sum is a penalty term that keeps the sequence smooth. By varying the parameter m one can control the desired degree of smoothness. Figure 2c shows the dual problem to the example of figure 2b. (This is the continuous version of the problem again.) The graph of $b(x)$ is shown as a thick curve overlaid over the graph of $F(x) := \int_0^x f(t)\,dt$. The reader may notice a correspondence between the ranges where $b(x)$ follows $F(x)$ or where it remains constant, and the ranges in figure 2a where $g(x)$ is at its upper or lower bound or where it has slope ± 1, respectively. This is nothing but a manifestation of complementary slackness in linear programming duality.

A similar curve fitting problem has been considered by Ohmura et al. (1990) and by Imai (1991):

$$\min\Big\{\, \sum_{i=1}^{n} |b_i - F_i| : b_1 \le b_2 \le \cdots \le b_n \,\Big\}. \tag{5}$$

Here a given sequence must be approximated by a monotone sequence.

Theorem 2. *The monotone approximation problem* (5) *is a special case of* (4).

This can be seen as follows. We add a value $F_0 := \min\{ F_i : 1 \le i \le n \}$ at the beginning and a value $F_{n+1} := \max\{ F_i : 1 \le i \le n \}$ at the end of the sequemce, and we set $n := n+1$ and $m := n+2$. By the constraints of (4), the sequence (b_i) must rise from $b_0 = F_0$ to $b_n = F_n$, and thus the penalty term is at least $m(F_n - F_0)$. Since the multiplier m is so large it would never pay off to go down again:

Lemma 3. *With the data as defined above, we will always have $b_i \le b_{i+1}$ in an optimal solution of* (4).

Thus the penalty term is constant and may be neglected.

Imai gave an $O(n \log n)$ algorithm which uses mergeable heaps. Our algorithm which we give in the next section is very simple and has the same complexity, and, when appropriately specialized, it uses just plain heap operations. We also give a few other algorithms whose complexity depends on the size of the F_i values.

2.3 Another network flow model — the Monge-Kantorovich mass transfer problem

A different reformulation leads to the following dual definition:

$$\|(f_1, \ldots, f_n)\|_{\mathrm{BL}} = \min\Big\{ \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} : \sum_{j=1}^{n} x_{ij} - \sum_{j=1}^{n} x_{ji} = f_i, \text{ for } i = 1, 2, \ldots, n;\;\; x_{ij} \ge 0 \Big\}$$

where $c_{ij} := \min\{|i-j|, 2m\}$. This is a mininum-cost network flow problem with nodes $C_1, C_2, \ldots, C_n$. The original problem (2) is the dual problem to this. The restrictions are the flow conservation equations. The supply/demand at node C_i is f_i. We assume $\sum_{i=1}^{n} f_i = 0$. By routing the flow x_{ij} on arc (i,j) over the vertices $C_{i+1}, C_{i+2}, \ldots, C_{j-1}$ if $|i-j| \le 2m$, and via an auxiliary vertex D otherwise, we can correctly model the costs c_{ij} with the arcs shown in the network of figure 4. The costs on the arcs between C_{i-1} and C_i are 1, in both directions, and the costs on the arcs between C_i and D are m, in both directions. The flow must be nonnegative, but there are no upper capacities.

This model has a nice direct interpretation: f_i is the excess or the lack of material at location i. We have to transport the material to balance the excesses and the demands in the cheapest possible way. Without the additional vertex D, this is the original mass transfer problem that goes back to G. Monge (1781) and to L. V. Kantorovitch (1942). In our case, the price that we have to pay is proportional to the distance, except that we only have to pay a flat rate of $2m$ for very long distances.

The cost coefficients c_{ij} can be generalized to be an arbitrary metric on the point set $\{1, 2, \ldots, n\}$ (or on the interval $[0,1]$ in the continuous formulation), and unter suitable conditions the equivalence between the different dual formulations still holds. The resulting general metric between functions is known as the Kantorovich-Rubinstein distance or the Wasserstein distance, depending on the formulation, see Rachev (1984, 1991).

Note that the network in both network flow models of figure 3 and figure 4 is series-parallel. Booth and Tarjan (1993) have given an $O(E \log E)$ algorithm for a mininum-cost

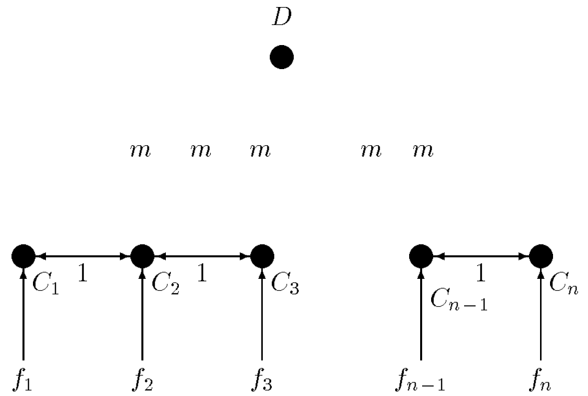

Fig. 4. Another network flow model.

network flow in a series-parallel network with E arcs. When applied to our networks, the algorithm coincides for the two networks; the algorithm in section 3 can actually be derived as a special case and simplified version of Booth and Tarjan's algorithm.

3 Algorithms

3.1 Dynamic programming

We can solve the linear program in (2) by considering all possible values $j = -m, \ldots, m$ for g_k, successively for $k = 1, \ldots n$. Define

$$G_k(j) := \max\Big\{ \sum_{i=1}^{k} f_i g_i : |g_i| \leq m,\ |g_{i+1} - g_i| \leq 1, g_k = j \Big\}.$$

Then we can set up the recursion

$$G_{k+1}(j) = f_{k+1} \cdot j + \max\{G_k(j-1), G_k(j), G_k(j+1)\}, \text{ for } -m \leq j \leq m, \tag{6}$$

where we take $G_k(-m-1) = G_k(m+1) = -\infty$ for the "out-of-boundary" values.

Lemma 4. *$G_k(j)$ is a concave function of j.*

Thus, we can represent the function $G_k(j)$ by the "start value" $S := G_k(-m)$ and the decreasing sequence of increments $E_j := G_k(j+1) - G_k(j)$. Let us see how G_{k+1} is derived from G_k. The maximum of the three expressions in (6) can be obtained by taking three copies of the graph of the function G_k: one which is shifted one unit to the left, one which is shifted one unit to the right, and an unshifted one. Then we take the pointwise maximum, see figure 5. The result will have the same sequence of increments E_j as the original G_k, except that two horizontal pieces $E_j = 0$ are inserted in the middle, and the first and the last entry is deleted. Adding the linear function $f_{k+1} \cdot j$ means that we have to add the value f_{k+1} to each E_j. We also have to compute the new start value $S = G_{k+1}(-m)$ from $G_k(-m)$. The following algorithm summarizes this procedure.

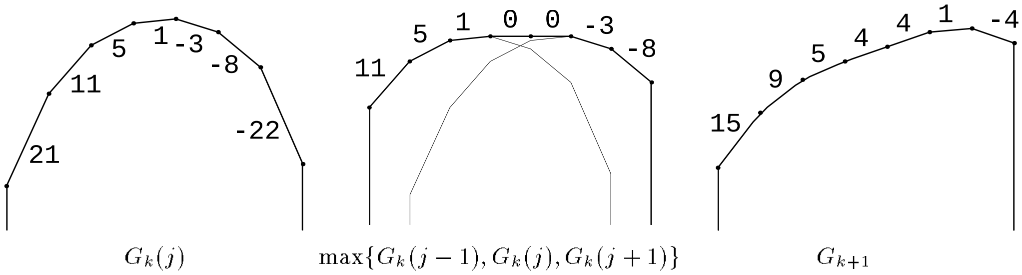

Fig. 5. Deriving G_{k+1} from G_k.

(∗ Start with $G_0(j) \equiv 0$: ∗)
Let $E = (0, 0, \ldots, 0)$; ($2m$ zeros)
$S := 0$;
for $k := 1$ **to** n **do**
 (∗ compute G_k: ∗)
 insert 0,0 into the decreasing sequence E (at the correct position);
 $S := S +$ the largest (i. e., first) element of E;
 remove the largest and the smallest element from E;
 add f_k to all elements of E;
 $S := S + (-m) \cdot f_k$;
end for;
add all positive elements of E to S; this is the result.

Instead of adding a constant to all elements of E, which costs too much time, we can also add it to a separate value F which we maintain; we represent the "true value" of E_j by $E'_j + F$. The resulting algorithm is as follows:

Let $E' = (0, 0, \ldots, 0)$; ($2m$ zeros)
$S := 0$; $F := 0$;
for $k := 1$ **to** n **do**
 (∗ compute G_k: ∗)
 insert $-F$,$-F$ into E';
 $S := S + F +$ the largest element of E';
 remove the largest and the smallest element from E';
 $F := F + f_k$;
 $S := S + (-m) \cdot f_k$;
end for;
for all elements e' of E' with $e' + F > 0$, add $e' + F$ to S; this is the result.

3.2 Data structures

Clearly, there is really no need to represent E' as a sorted list. The amount of work is constant per iteration of the loop, except for the operations of the set E'. Let us discuss the necessary operations on E':

1. insert
2. find and remove the smallest and largest element.

We have $2n$ operations of each type. E' has always $2m$ or $2m+2$ elements. The elements that are inserted are the partial sums F_i of (3).

Depending on the size of the universe $\hat{F} := \max_k |F_k|$, there are several choices of data structures.

Ordinary priority queues. This leads to $O(n \log m)$ time and $O(m)$ storage if we simply use heaps. (Actually, we have to use max-min heaps, see Atkinson et al. 1986.) Since the list E' is initialized with zeros, we can in fact achieve $O(n \log \min\{m, n\})$ time and $O(\min\{m, n\})$ space. By using a balanced search tree augmented with appropriate information fields, we can solve the problem on-line with $O(\log m)$ time per data point: Immediately after we have read the next value f_i, we can output the bounded Lipschitz norm of the sequence so far.

Subsets of a small universe. The remaining possibilities are more "data-dependent", because they are only useful when the data are integers or $\hat{F}$ is small. In applications, $\hat{F}$ will not be too large, because it does not make sense to measure the image with a very high precision. If $\hat{F}$ is large or the data are "real numbers", this approach can still be used if one is satisfied with an approximate value, because the data can be scaled and rounded to small integers. Using the appropriate data structures (see Mehlhorn and Näher 1990) we can achieve $O(m)$ storage and $O(n \log \log \hat{F})$ time. If the f_i are very small it might even be preferable to use a much simpler sequential search, which would lead to a time of $O(\sum_i |f_i|)$ with $O(\hat{F})$ storage.

The off-line min approach. If we can afford to sort the values F_i by some method of distribution sort or radix sort, using $O(n + \hat{F})$ time or $O(n \log_n \hat{F})$ time, respectively, we can then use the algorithm for the off-line-min problem (cf. the three Americans' book, Aho et al. 1974).

This leads to a very simple algorithm with a complexity of $O(n\alpha(m, m))$, where $\alpha(m, n)$ is the extremely slowly growing inverse function of the Ackermann function, cf. Tarjan and van Leeuwen (1984). Using the improvement of Gabow and Tarjan (1985), which makes the algorithm slightly more complicated, we can even achieve $O(n)$ time. (All these times are in addition to sorting.)

However, there is a slight catch: Usually, in the off-line min problem, we are given a sequence of INSERT and DELETE-MIN operations. We are given the whole sequence in advance. We start by looking at the smallest element that was inserted and find out at what time it was deleted: this is the nearest DELETE-MIN that follows the respective INSERT. We match off this INSERT and this DELETE-MIN, look at the second-smallest element, and so on.

In our case, we also have DELETE-MAX operations, and these might interfere with the DELETE-MIN's: We cannot be sure whether an element for which we are looking for the corresponding DELETE-MIN operation should not really have been removed by a DELETE-MAX. There is no easy way to decide this, except by carrying out the operations in sequence, which is just what we want to avoid.

We solve this dilemma by cutting the sequence of operations into *blocks* of length $4m$, i. e., we have $2m$ INSERT's, m DELETE-MIN's and m DELETE-MAX's.

Lemma 5. *Let x be the median of the $2m$ elements that are initially in the list before the a block starts. Then all elements which are removed by DELETE-MIN's in the block are smaller than x, and all elements which are removed by DELETE-MAX's in the block are larger than x.*

It follows that we can process the DELETE-MIN's and DELETE-MAX's in a block separately without caring about interference.

4 Open questions and further research

The algorithm can be extended compute the norm of a piecewise linear functions. For a function with n linear pieces, it takes $O(n \log n)$ time. In addition, $O(n)$ equations involving at most $O(n)$ square roots must be solved. More precisely, if the function f has s sign changes, the algorithm needs $O(n \log n + s^2)$ time. The details will be given in a later version of this paper.

A question for further research is the minimization of the bounded Lipschitz distance under various transformations of the functions, like scalings of coordinates or cyclic rotations of the definition interval. This is of interest when comparing two polygonal shapes, as described in the introduction. The arbitrary choice of the starting point for the parameterization of the boundary or a rotation of the polygons should not have an influence on their distance measure.

The definition of the bounded Lipschitz norm in section 2.3 can be extended naturally to two- and higher-dimensional arrays (functions of more than one variable). Already in two dimensions, it is an open question whether the bounded Lipschitz norm can be computed more efficiently than by solving a network flow problem. Even with $m = \infty$, i. e., with the distance coefficients $c((x_i, y_i), (x_j, y_j)) = |x_i - x_j| + |y_i - y_j|$, there is no simple way to compute the metric. (Setting $m = \infty$ corresponds to deleting the supervertex D in figure 4.)

Another open question is whether the heap operations in section 3.2 are really necessary if nothing is assumed about the values F_i. Since we are not interested in the *sequence* in which the elements are removed from the data structure but only in the *set* of elements which are removed by DELETE-MAX and DELETE-MIN operations (actually, only in their *sum*), it is not clear whether $\Omega(n \log \min\{m, n\})$ is a lower bound for the problem.

References

A. V. Aho, J. E. Hopcroft, and J. D. Ullman: *The Design and Analysis of Computer Algorithms*, Addison-Wesley, 1974.

E. Arkin, L. P. Chew, D. P. Huttenlocher, K. Kedem, and J. S. B. Mitchell: An efficiently computable metric for comparing polygonal shapes, in: *SODA'90, Proc. 1st ACM-SIAM Symp. Discrete Algorithms, San Francisco, January 1990*, Society for Industrial and Applied Mathematics, Philadelphia 1990, pp. 129–137.

M. J. Atallah: A linear time algorithm for the Hausdorff distance between convex polygons, *Inform. Process. Lett.* **17** (1983), 207–209.

M. D. Atkinson, J.-R. Sack, N. Santoro, T. Strothotte: Min-max heaps and generalized priority queues, *Commun. ACM* **29** (1986), 996–1000.

H. Booth and R. E. Tarjan: Finding the minimum-cost maximum flow in a series-parallel Network, *J. Algorithms* **15** (1993), 416–446.

P. Cox, H. Maitre, M. Minoux, C. Ribeiro: Optimal matching of convex polygons, *Pattern Recognition Letters* **9** (1989), 327–334.

H. N. Gabow and R. E. Tarjan: A linear-time algorithm for a special case of disjoint set union, *J. Comput. Syst. Sci.* **30** (1985), 209–221.

P. Hackh: Quality control of artificial fabrics, Forschungsbericht, Arbeitsgruppe Technomathematik, Universität Kaiserslautern, 1990.

H. Imai: A geometric fitting problem of two corresponding sets of points on a line, *IEICE Trans. Fund. Electron. Commun. Comput. Sci.* **E 74** (1991), 665–667 plus Hiroshi Imai's picture on p. 668.

L. Kantorovitch: On the translocation of masses, *Comptes Rendues (Doklady) de l'Académie des Sciences de l'URSS* **37** (1942), 199–201.

L. V. Kantorovich and G. Sh. Rubinshteĭn: On a space of completely additive functions (in Russian), *Vestnik Leningrad. Univ.* **13**, 7 (1958), 52–59.

K. Mehlhorn and S. Näher: Bounded ordered dictionaries in $O(\log\log N)$ time and $O(n)$ space. *Inform. Proc. Lett.* **35** (1990), 183–189.

G. Monge: Mémoire sur la théorie des déblais et des remblais, *Mémoires de l'Académie Royale des Sciences*, Paris, 1781, 666–704 and pl. XVIII–XIX; see also: Sur les déblais et les remblais, *Histoire de l'Académie Royale des Sciences*, Paris, 1781, 34–38.

H. Neunzert and B. Wetton: Pattern recognition using measure space metrics, Forschungsbericht Nr. 28, Arbeitsgruppe Technomathematik, Universität Kaiserslautern, 1987.

M. Ohmura, S. Wakabayashi, J. Miyao, and N. Yoshida: Improvement of one-dimensional placement in VLSI design (in Japanese), *Trans. Inst. Electron. Commun. Eng. Japan* **J73-A**, 11 (1990), 1858-1866.

S. T. Rachev: The Monge-Kantorovich mass transfer problem and its stochastic applications (in Russian), *Teor. Veroyatn. Primen.* **29** (1984), 625–653; English translation: *Theory Prob. Appl.* **29** (1984), 647–676.

S. T. Rachev: *Probability Metrics and the Stability of Stochastic Models*, Wiley, Chichester 1991.

R. E. Tarjan and J. van Leeuwen: Worst-case analysis of set union algorithms, *J. Assoc. Comput. Mach.* **31** (1984), 245–281.

This article was processed using the LaTeX macro package with LMAMULT style