

# Streaming Algorithms for Line Simplification

Mohammad Ali Abam<sup>\*</sup>  
Department of Computing  
Science, TU Eindhoven,  
P.O. Box 513, 5600 MB  
Eindhoven, the Netherlands  
m.a.abam@tue.nl

Mark de Berg<sup>†</sup>  
Department of Computing  
Science, TU Eindhoven,  
P.O. Box 513, 5600 MB  
Eindhoven, the Netherlands  
mdberg@tue.nl

Peter Hachenberger<sup>‡</sup>  
Department of Computing  
Science, TU Eindhoven,  
P.O. Box 513, 5600 MB  
Eindhoven, the Netherlands  
phachenb@win.tue.nl

Alireza Zarei<sup>‡</sup>  
Computer Engineering  
Department, Sharif University  
of Technology and IPM School  
of Computer Science,  
P.O. Box 11365-9517,  
Tehran, Iran  
zareai@mehr.sharif.edu

## ABSTRACT

We study the following variant of the well-known line-simplification problem: we are getting a possibly infinite sequence of points  $p_0, p_1, p_2, \dots$  in the plane defining a polygonal path, and as we receive the points we wish to maintain a simplification of the path seen so far. We study this problem in a streaming setting, where we only have a limited amount of storage so that we cannot store all the points. We analyze the competitive ratio of our algorithms, allowing resource augmentation: we let our algorithm maintain a simplification with  $2k$  (internal) points, and compare the error of our simplification to the error of the optimal simplification with  $k$  points. We obtain the algorithms with  $O(1)$  competitive ratio for three cases: convex paths where the error is measured using the Hausdorff distance (or Fréchet distance),  $xy$ -monotone paths where the error is measured using the Hausdorff distance (or Fréchet distance), and general paths where the error is measured using the Fréchet distance. In the first case the algorithm needs  $O(k)$  additional storage, and in the latter two cases the algorithm needs  $O(k^2)$  additional storage.

<sup>\*</sup>MAA was supported by the Netherlands' Organisation for Scientific Research (NWO) under project no. 612.065.307.

<sup>†</sup>MdB and PH were supported by the Netherlands' Organisation for Scientific Research (NWO) under project no. 639.023.301.

<sup>‡</sup>AZ was supported by Sharif University and a grant from IPM school of CS (no. CS1385-2-01)

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

SCG'07, June 6–8, 2007, Gyeongju, South Korea.

Copyright 2007 ACM 978-1-59593-705-6/07/0006 ...\$5.00.

## Categories and Subject Descriptors

F.2.2 [Analysis of Algorithms and problem complexity]: Nonnumerical Algorithms and Problems—*Geometrical problems and computations*

## General Terms

Algorithms, Theory

## Keywords

line simplification, streaming algorithm, Fréchet distance, Hausdorff distance

## 1. INTRODUCTION

*Motivation.* Suppose we are tracking one, or maybe many, moving objects. Each object is equipped with a device that is continuously transmitting its position. Thus we are receiving a stream of data points that describes the path along which the object moves. The goal is to maintain this path for each object. We are interested in the scenario where we are tracking the objects over a very long period of time, as happens for instance when studying the migratory patterns of animals. In this situation it may be undesirable or even impossible to store the complete stream of data points. Instead we have to maintain an approximation of the input path. This leads us to the following problem: we are receiving a (possibly infinite) stream  $p_0, p_1, p_2, \dots$  of points in the plane, and we wish to maintain a simplification (of the part of the path seen so far) that is as close to the original path as possible, while using not more than a given (fixed) amount of available storage.

*Related work.* The problem described above is a streaming version of line simplification, one of the basic problems in GIS. In a line simplification problem one is given a polygonal path  $P := p_0, p_1, \dots, p_n$  in the plane, and the goal is

to find a path  $Q := q_0, q_1, \dots, q_k$  with fewer vertices that approximates the path  $P$  well. In fact, this problem arises whenever we want to perform data reduction on a polygonal shape in the plane, and so it plays a role not only in GIS but also in areas like image processing and computer graphics. Line simplification has been studied extensively both in these application areas as well as in computational geometry. We study line simplification in a streaming setting, where we only have a limited amount of storage so that we cannot store all the points. A similar streaming model for geometric algorithms has been used by e.g. Agarwal and Yu [3], and Zarrabi-Zadeh and Chan [17]. Also see Muthukrishnan’s survey [16] on streaming algorithms.

The line-simplification problem has many variants. For example, we can require the sequence of vertices of  $Q$  to be a subsequence of  $P$  (with  $q_0 = p_0$  and  $q_k = p_n$ )—this is sometimes called the *restricted version*—or we can allow arbitrary points as vertices. In this paper, as in most other papers, we consider the restricted version, and we limit our discussion to this version from now on; some results on the unrestricted version can be found in [8, 9, 10, 13]. In the restricted version, each link  $q_i q_{i+1}$  of the simplification corresponds to a shortcut  $p_i p_j$  (with  $j > i$ ) of the original path, and the error of the link is defined as the distance between  $p_i p_j$  and the subpath  $p_i, \dots, p_j$ . To measure the distance between  $p_i p_j$  and  $p_i, \dots, p_j$  one often uses the Hausdorff distance, but the Fréchet distance can be used as well—see below for definitions. The error of the simplification  $Q$  is now defined as the maximum error of any of its links. Once the error measure has been defined, we can consider two types of optimization problems: the min- $k$  and the min- $\delta$  problem. In the min- $k$  problem, one is given the path  $P$  and a maximum error  $\delta$ , and the goal is to find a simplification  $Q$  with as few vertices as possible whose error is at most  $\delta$ . In the min- $\delta$  problem, one is given the path  $P$  and a maximum number of vertices  $k$ , and the goal is to find a simplification with the smallest possible error that uses at most  $k$  vertices.

The oldest and most popular algorithm for line simplification under the Hausdorff distance is the Douglas-Peucker algorithm [6]. A basic implementation of this algorithm runs in  $O(n^2)$  time, but more careful implementations run in  $O(n \log n)$  time [11] or even  $O(n \log^* n)$  time [12]. However, the Douglas-Peucker algorithm is only a heuristic and it is not guaranteed to be optimal (in terms of the number of vertices used, or the error of the resulting simplification). Imai and Iri [14] showed how to solve the problem optimally in  $O(n^2 \log n)$  time by modeling it as a shortest-path problem on directed acyclic graphs. The running time of their method was improved to quadratic or near quadratic by Chin and Chan [5], and Melkman and O’Rourke [15]. Finally, Agarwal and Varadarajan [2] improved the running time to  $O(n^{4/3+\epsilon})$ , for any fixed  $\epsilon > 0$ , for the  $L_1$ -metric and the so-called uniform metric—here  $d(p, q) = |p_x - q_x|$  if  $p_x = q_x$  and  $d(p, q) = \infty$  otherwise—by implicitly representing the graph.

The line-simplification problem was first studied for the Fréchet distance by Godau [7]. Alt and Godau [4] proposed an algorithm to compute the Fréchet distance between two polygonal paths in quadratic time; combined with the approach of Imai and Iri [14] this can be used to compute an optimal solution to the min- $\delta$  or the min- $k$  problem for the Fréchet distance.

Since solving the line-simplification problem exactly is costly—the best known algorithm for the Hausdorff distance (under the  $L_2$  metric) and for the Fréchet distance take quadratic time or more—Agarwal *et al.* [1] consider approximation algorithms. In particular, they consider the min- $k$  problem for both the Hausdorff distance for  $x$ -monotone paths (in the plane) and the Fréchet distance for general paths (in  $d$ -dimensional space). They give near-linear time algorithms that compute a simplification whose error is at most  $\delta$  and whose number of vertices is at most the minimum number of vertices of a simplification of error at most  $\delta/2$ . Their algorithms are greedy and iterative. Because the algorithms are iterative they can be used in an on-line setting, where the points are given one by one and the simplification must be updated at each step. However, since they solve the min- $k$  problem, they cannot be used in a streaming setting, because the complexity of the produced simplification for an input path of  $n$  points can be  $\Theta(n)$ . (Note that an iterative greedy approach can be used in the min- $k$  problem—try to go as far as possible with each link, while staying within the error bound  $\delta$ —but that for the min- $\delta$  problem this does not work.) Moreover, their algorithm for the Hausdorff distance does not work when the normal Euclidean distance is used in the definition of Hausdorff distance, but only when the uniform distance is used. The other existing algorithms for line simplification cannot be used in a streaming setting either.

*Definitions, notation, and problem statement.* To be able to state the problem we wish to solve and the results we obtain more precisely, we first introduce some terminology and definitions. Let  $p_0, p_1, \dots$  be the given stream of input points. We use  $P(n)$  to denote the path defined by the points  $p_0, p_1, \dots, p_n$ —that is, the path connecting those points in order—and for any two points  $p, q$  on the path we use  $P(p, q)$  to denote the subpath from  $p$  to  $q$ . For two vertices  $p_i, p_j$  we use  $P(i, j)$  as a shorthand for  $P(p_i, p_j)$ . A segment  $p_i p_j$  with  $i < j$  is called a *link* or sometimes a *shortcut*. Thus  $P(n)$  consists of the links  $p_{i-1} p_i$  for  $0 < i \leq n$ . We assume a function *error* is given that assigns a non-negative error to each link  $p_i p_j$ . A  $\ell$ -*simplification* of  $P(n)$  is a polygonal path  $Q := q_0, q_1, \dots, q_k, q_{k+1}$  where  $k \leq \ell$  and  $q_0 = p_0$  and  $q_{k+1} = p_n$ , and  $q_1, \dots, q_k$  is a subsequence of  $p_1, \dots, p_{n-1}$ . The error of a simplification  $Q$  for a given function *error*, denoted  $\text{error}(Q)$ , is defined as the maximum error of any of its links. We will consider two specific error functions for our simplifications, one based on the Hausdorff distance, and one based on the Fréchet distance, as defined next. For two objects  $o_1$  and  $o_2$ , we use  $d(o_1, o_2)$  to denote the Euclidean distance between  $o_1$  and  $o_2$ . (For two points  $p_i$  and  $p_j$ , we sometimes also use  $|p_i p_j|$  to denote the Euclidean distance between  $p_i$  and  $p_j$ , which is equal to the length of the segment  $p_i p_j$ .)

- In the Hausdorff error function  $\text{error}_H$ , the error of the link  $p_i p_j$  is equal to  $d_H(p_i p_j, P(i, j))$ , the Hausdorff distance of the subpath  $P(i, j)$  to the segment  $p_i p_j$ . The Hausdorff error is defined as  $d_H(p_i p_j, P(i, j)) := \max_{i \leq l \leq j} d(p_l, p_i p_j)$ .
- The Fréchet distance between two paths  $A$  and  $B$ , which we denote by  $d_F(A, B)$ , is defined as follows. Consider a man with a dog on a leash, with the man standing at the start point of  $A$  and the dog stand-

ing at the start point of  $B$ . Imagine that the man walks to the end of  $A$  and the dog walks to the end of  $B$ . During the walk they can stop every now and then, but they are not allowed to go back along their paths. Now the Fréchet distance between  $A$  and  $B$  is the minimum length of the leash needed for this walk, over all possible such walks. More formally,  $d_F(A, B)$  is defined as follows. Let  $A$  and  $B$  be specified by functions  $A : [0, 1] \rightarrow \mathbb{R}^2$  and  $B : [0, 1] \rightarrow \mathbb{R}^2$ . Any non-decreasing continuous function  $\alpha : [0, 1] \rightarrow [0, 1]$  with  $\alpha(0) = 0$  and  $\alpha(1) = 1$  defines a re-parametrization  $A_\alpha$  of  $A$  by setting  $A_\alpha(t) = A(\alpha(t))$ . Similarly, any non-decreasing continuous function  $\beta : [0, 1] \rightarrow [0, 1]$  with  $\beta(0) = 0$  and  $\beta(1) = 1$  defines a re-parametrization  $B_\beta$  of  $B$ . The Fréchet distance  $d_F(A, B)$  between two paths  $A$  and  $B$  is now defined as

$$d_F(A, B) := \inf_{\alpha, \beta} \max_{0 \leq t \leq 1} d(A_\alpha(t), B_\beta(t))$$

where the infimum is taken over all re-parametrizations  $A_\alpha$  of  $A$  and  $B_\beta$  of  $B$ . In the Fréchet error function  $\text{error}_F$ , the error of the link  $p_i p_j$  is the Fréchet distance of the subpath  $P(i, j)$  to the segment  $p_i p_j$ , that is,  $\text{error}_F(p_i p_j) := d_F(P(i, j), p_i p_j)$ .

Now consider an algorithm  $\mathcal{A} := \mathcal{A}(\ell)$  that maintains an  $\ell$ -simplification for the input stream  $p_0, p_1, \dots$ , for some given  $\ell$ . Let  $Q_{\mathcal{A}}(n)$  denote the simplification that  $\mathcal{A}$  produces for the path  $P(n)$ . Let  $\text{Opt}(\ell)$  denote an optimal off-line algorithm that produces an  $\ell$ -simplification. Thus  $\text{error}(Q_{\text{Opt}(\ell)}(n))$  is the minimum possible error of any  $\ell$ -simplification of  $P(n)$ . We define the quality of  $\mathcal{A}$  using the *competitive ratio*, as is standard for on-line algorithms. We also allow *resource augmentation*. More precisely, we allow  $\mathcal{A}$  to use a  $2k$ -simplification, but we compare the error of this simplification to  $Q_{\text{Opt}(k)}(n)$ . (This is similar to Agarwal *et al.* [1] who compare the quality of their solution to the min- $k$  problem for a given maximum error  $\delta$  to the optimal value for maximum error  $\delta/2$ .) Thus we define the competitive ratio of an algorithm  $\mathcal{A}(2k)$  as

$$\text{competitive ratio of } \mathcal{A}(2k) := \max_{n \geq 0} \frac{\text{error}(Q_{\mathcal{A}(2k)}(n))}{\text{error}(Q_{\text{Opt}(k)}(n))},$$

where  $\frac{\text{error}(Q_{\mathcal{A}(2k)}(n))}{\text{error}(Q_{\text{Opt}(k)}(n))}$  is defined as 1 if  $\text{error}(Q_{\mathcal{A}(2k)}(n)) = \text{error}(Q_{\text{Opt}(k)}(n)) = 0$ . We say that an algorithm is  $c$ -competitive if its competitive ratio is at most  $c$ .

**Our results.** We present and analyze a simple general streaming algorithm for line simplification. Our analysis shows that the algorithm has good competitive ratio under two conditions: the error function that is used is *monotone*—see Section 2 for a definition—and there is an oracle that can approximate the error of any candidate link considered by the algorithm. We then continue to show that the Hausdorff error function is monotone on convex paths and on  $xy$ -monotone paths. (It is not monotone on general paths.) The Fréchet error function is monotone on general paths. Finally, we show how to implement the error oracles for these three settings. Putting everything together leads to the following results.

- (i) For convex paths and the Hausdorff error function (or the Fréchet error function) we obtain a 3-competitive

streaming algorithm using  $O(k)$  additional storage that processes an input point in  $O(\log k)$  time.

- (ii) For  $xy$ -monotone paths and the Hausdorff error function (or the Fréchet error function) we can, for any fixed  $\varepsilon > 0$ , obtain a  $(4 + \varepsilon)$ -competitive streaming algorithm that uses  $O(k^2/\sqrt{\varepsilon})$  additional storage and processes each input point in  $O(k/\sqrt{\varepsilon})$  amortized time.
- (iii) For arbitrary paths and the Fréchet error function we can, for any fixed  $\varepsilon > 0$ , obtain a  $(4\sqrt{2} + \varepsilon)$ -competitive streaming algorithm that uses  $O(k^2/\sqrt{\varepsilon})$  additional storage and processes each input point in  $O(k/\sqrt{\varepsilon})$  amortized time.

## 2. A GENERAL ALGORITHM

In this section we describe a general strategy for maintaining an  $\ell$ -simplification of an input stream  $p_0, p_1, \dots$  of points in the plane, and we will show that it has a good competitive ratio under two conditions: the error function is *monotone* (as defined below), and we have an *error oracle* at our disposal that computes or approximates the error of a link. We denote the error computed by the oracle for a link  $p_i p_j$  by  $\text{error}^*(p_i p_j)$ . In later sections we will prove that the Hausdorff error metric is monotone on convex or  $xy$ -monotone paths and that the Fréchet error function is monotone on general paths, and we will show how to implement the oracles for these settings.

Our algorithm is quite simple. Suppose we have already handled the points  $p_0, \dots, p_n$ . (We assume  $n > \ell + 1$ ; until that moment we can simply use all points and have zero error.) Let  $Q := q_0, q_1, \dots, q_\ell, q_{\ell+1}$  be the current simplification. Our algorithm will maintain a priority queue  $\mathcal{Q}$  that stores the points  $q_i$  with  $1 \leq i \leq \ell$ , where the priority of a point is the error (as computed by the oracle) of the link  $q_{i-1} q_{i+1}$ . In other words, the priority of  $q_i$  is (an approximation of) the error that is incurred when  $q_i$  is removed from the simplification. Now the next point  $p_{n+1}$  is handled as follows:

1. Set  $q_{\ell+2} := p_{n+1}$ , thus obtaining an  $(\ell+1)$ -simplification of  $P(n+1)$ .
2. Compute  $\text{error}^*(q_\ell q_{\ell+2})$  and insert  $q_{\ell+1}$  into  $\mathcal{Q}$  with this error as priority.
3. Extract the point  $q_s$  with minimum priority from  $\mathcal{Q}$ ; remove  $q_s$  from the simplification.
4. Update the priorities of  $q_{s-1}$  and  $q_{s+1}$  in  $\mathcal{Q}$ .

Next we analyze the competitive ratio of our algorithm.

We say that a link  $p_i p_j$  *encloses* a link  $p_l p_m$  if  $i \leq l \leq m \leq j$ , and we say that  $\text{error}$  is a  $c$ -*monotone error function* for a path  $P(n)$  if for any two links  $p_i p_j$  and  $p_l p_m$  such that  $p_i p_j$  encloses  $p_l p_m$  we have

$$\text{error}(p_l p_m) \leq c \cdot \text{error}(p_i p_j).$$

In other words, an error function is  $c$ -monotone if the error of a link cannot be worse than  $c$  times the error of any link that encloses it.

Furthermore, we denote an error oracle as an  $e$ -*approximate error oracle* if

$$\text{error}(p_i p_j) \leq \text{error}^*(p_i p_j) \leq e \cdot \text{error}(p_i p_j)$$

for any link  $p_i p_j$  for which the oracle is called by the algorithm above.

**THEOREM 2.1.** *Suppose that we use a  $c$ -monotone error function and that we have an  $e$ -approximate error oracle at our disposal. Then the algorithm described above with  $\ell = 2k$  is  $ce$ -competitive with respect to  $Opt(k)$ . The time the algorithm needs to update the simplification  $Q$  upon the arrival of a new point is  $O(\log k)$  plus the time spent by the error oracle. Besides the storage needed for the simplification  $Q$ , the algorithm uses  $O(k)$  storage plus the storage needed by the error oracle.*

**PROOF.** Consider an arbitrary  $n \geq 0$ , and let  $Q(n)$  denote the  $2k$ -simplification produced by our algorithm. Since the error of  $Q(n)$  is the maximum error of any of its links, we just need to show that  $error(\sigma) \leq c \cdot e \cdot error(Q_{Opt(k)}(n))$  for any link  $\sigma$  in  $Q(n)$ . Let  $m \leq n$  be such that  $\sigma$  appears in the simplification when we receive point  $p_m$ . If  $m \leq 2k + 2$ , then  $error(\sigma) = 0$  and we are done. Otherwise, let  $Q(m-1) := q_0, \dots, q_{2k+1}$  be the  $2k$ -simplification of  $P(m-1)$ . Upon the arrival of  $p_m = q_{2k+2}$  we insert  $q_{2k+1} = p_{m-1}$  into  $Q$ . A simple counting argument shows that at least one of the shortcuts  $q_{t-1}q_{t+1}$  for  $1 \leq t \leq 2k+1$ , let's call it  $\sigma'$ , must be enclosed by one of the at most  $k+1$  links in  $Q_{Opt(k)}(n)$ . Since  $\sigma$  is the link with the smallest priority among all links in  $Q$  at that time, its approximated error is smaller than that of  $\sigma'$ . Therefore,

$$\begin{aligned} error(Q_{Opt(k)}(n)) &\geq \frac{1}{c} error(\sigma') &\geq \frac{1}{c \cdot e} error^*(\sigma') \\ &\geq \frac{1}{c \cdot e} error^*(\sigma) &\geq \frac{1}{c \cdot e} error(\sigma). \end{aligned}$$

We conclude that our algorithm is  $ce$ -competitive with respect to  $Opt(k)$ .

Besides the time and storage needed by the error oracle, the algorithm only needs  $O(k)$  space to store the priority queue and  $O(\log k)$  for each update of the priority queue.  $\square$

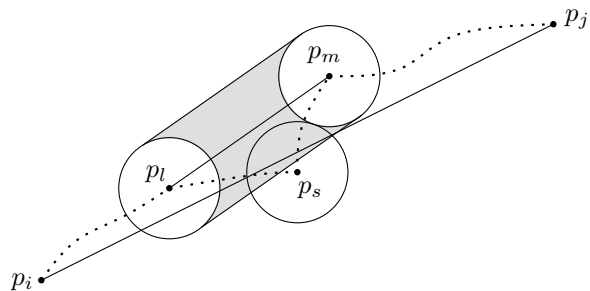
### 3. THE HAUSDORFF ERROR FUNCTION

The algorithm presented above has good competitive ratio if the error function being used is monotone and can be approximated well. In this section we show that these properties hold for the Hausdorff error function on convex and  $xy$ -monotone paths. (A path is convex if by connecting the last point to the first point on the path we obtain a convex polygon. A path is  $xy$ -monotone if any horizontal or vertical line intersects it in at most one point.) Note that for these two cases the Hausdorff distance between a link  $p_i p_j$  and the subpath  $P(i, j)$  is identical to the Fréchet distance between them. Thus the results from this section hold for the Fréchet distance as well. They improve on the result that will be given in the next section for the Fréchet distance on general curves. During this section all results stated for the Hausdorff distance also hold for the Fréchet distance.

The following lemma gives results on the monotonicity of various types of paths under the Hausdorff error function.

**LEMMA 3.1.** *The Hausdorff error function is 1-monotone on convex paths and 2-monotone on  $xy$ -monotone paths. Moreover, there is no constant  $c$  such that the Hausdorff error function is  $c$ -monotone on  $y$ -monotone paths.*

**PROOF.** It is easy to see that the Hausdorff error function is 1-monotone on convex paths. It is also not difficult, given any constant  $c$ , to give an example of an  $y$ -monotone path such that the Hausdorff error function is not  $c$ -monotone—a zigzag with four vertices such that the first and third are



**Figure 1:** The Hausdorff error function is 2-monotone on any  $xy$ -monotone path.

very close together and the second and fourth are very close together will do.

So now consider an  $xy$ -monotone path  $p_0, \dots, p_n$ . Let  $p_i p_j$  and  $p_l p_m$  be two links such that  $p_i p_j$  encloses  $p_l p_m$ , and let  $p_s$  be a point on the subpath  $P(l, m)$  such that  $d(p_s, p_l p_m) = error_H(p_l p_m)$ . Consider the circles  $C_l$ ,  $C_m$  and  $C_s$  of radius  $error_H(p_i p_j)$  centered at points  $p_l$ ,  $p_m$  and  $p_s$ —see Figure 1. Since the distance of the link  $p_i p_j$  to the points  $p_l$ ,  $p_s$ , and  $p_m$  is at most  $error_H(p_i p_j)$ , it must intersect these circles. Let  $p'_s$ ,  $p'_l$ , and  $p'_m$  be the orthogonal projections of  $p_s$ ,  $p_l$ , and  $p_m$  onto the link  $p_i p_j$ . Clearly,  $p'_s$ ,  $p'_l$  and  $p'_m$  are inside  $C_s$ ,  $C_l$  and  $C_m$ , respectively. Since  $P(i, j)$  is  $xy$ -monotone,  $p'_s$  lies between  $p'_l$  and  $p'_m$ , which implies

$$d(p'_s, p_l p_m) \leq \max(d(p'_l, p_l), d(p'_m, p_m)) \leq error_H(p_i p_j).$$

Therefore,

$$\begin{aligned} error_H(p_l p_m) &= error_H(p_s, p_l p_m) \\ &\leq d(p_s, p'_s) + d(p'_s, p_l p_m) \\ &\leq 2 error_H(p_i p_j). \end{aligned}$$

Note that the link  $p_i p_j$  can be tangent to  $C_s$ ,  $C_l$ , and  $C_m$ , which shows that the monotonicity factor 2 is tight.  $\square$

The next step is to implement the error oracles for convex paths and for  $xy$ -monotone paths. We start with the case of convex paths.

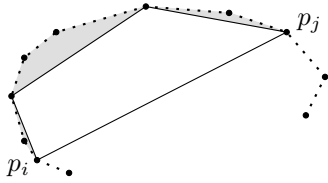
#### 3.1 The error oracle for convex paths

The idea of the error oracle is to maintain an approximation of the area enclosed by  $p_i p_j$  and the path  $P(i, j)$  for each link  $p_i p_j$ . Let  $area(i, j)$  denote this area. If the two angles  $\angle p_{i+1} p_i p_j$  and  $\angle p_{j-1} p_j p_i$  are at most 90 degrees, we can deduce an approximation of  $error_H(p_i p_j)$  from  $area(i, j)$  and  $|p_i p_j|$ . Indeed, if  $d_H(p_i p_j, P(i, j)) = d$ , then the maximum area enclosed by  $p_i p_j$  and  $P(i, j)$  is achieved by a rectangle with base  $p_i p_j$  and height  $d$ , and the minimum area is achieved by a triangle with base  $p_i p_j$  and height  $d$ . Hence,

$$\begin{aligned} d_H(p_i p_j, P(i, j)) &\leq 2 \cdot area(i, j) / |p_i p_j| \\ &\leq 2 \cdot d_H(p_i p_j, P(i, j)), \end{aligned}$$

and so  $2 \cdot area(i, j) / |p_i p_j|$  can be used as a 2-approximate error oracle. Unfortunately this approach does not work if  $\angle p_{i+1} p_i p_j$  and/or  $\angle p_{j-1} p_j p_i$  are bigger than 90 degrees. We therefore proceed as follows.

For each shortcut  $p_i p_j$  used in the current approximation, partition the path  $P(i, j)$  into at most five pieces by splitting it at each vertex that is extreme in  $x$ - or  $y$ -direction. (If, say, there is more than one leftmost vertex on the path, we



**Figure 2: The areas maintained by the error oracle for convex paths.**

cut at the first such vertex.) The information we maintain for  $p_i p_j$  is the set of cut points as well as area enclosed by each such piece  $P(l, m)$  and the corresponding shortcut  $p_l p_m$ —see Figure 2. Notice that if  $P(i, j)$  does not contain an extreme point we simply maintain  $area(i, j)$ , as before.

As  $d_H(p_i p_j, P(i, j))$  is the maximum of  $d_H(p_i p_j, P(l, m))$  over all pieces  $P(l, m)$  into which  $P(i, j)$  is cut, it is sufficient to approximate  $d_H(p_i p_j, P(l, m))$  for each piece. Note that  $\angle p_{l+1} p_l p_m$  and  $\angle p_{m-1} p_m p_l$  are at most 90 degrees. We approximate  $d_H(p_i p_j, P(l, m))$  by

$$(2 \cdot area(l, m) / |p_l p_m|) + d_H(p_i p_j, p_l p_m).$$

We claim this gives us a 3-approximation. We have

$$\begin{aligned} d_H(p_i p_j, P(l, m)) &\leq d_H(p_l p_m, P(l, m)) + d_H(p_i p_j, p_l p_m) \\ &\leq \frac{2 \cdot area(l, m)}{|p_l p_m|} + d_H(p_i p_j, p_l p_m). \end{aligned}$$

On the other hand,

$$\begin{aligned} 3 \cdot d_H(p_i p_j, P(l, m)) &\geq 3 \cdot \max(d_H(p_l p_m, P(l, m)), \\ &\quad d_H(p_i p_j, p_l p_m)) \\ &\geq \frac{2 \cdot area(l, m)}{|p_l p_m|} + d_H(p_i p_j, p_l p_m), \end{aligned}$$

so  $(2 \cdot area(l, m) / |p_l p_m|) + d_H(p_i p_j, p_l p_m)$  is indeed a 3-approximation of  $d_H(p_i p_j, P(l, m))$ .

What remains is to show that we can maintain this information as more points are received and the simplification changes. First consider step 2 of the algorithm, where we need to compute  $error^*(q_\ell q_{\ell+2})$ . Since we have the information described above available for  $q_\ell q_{\ell+1}$ , and  $q_{\ell+1}$  and  $q_{\ell+2}$  are consecutive points of the original path  $P$ , we can compute the necessary information for  $q_\ell q_{\ell+2}$  in  $O(1)$  time. Similarly, in step 4 we can update the information in  $O(1)$  time. We omit the easy details.

**LEMMA 3.2.** *There is a 3-approximate error oracle for the Hausdorff error function on convex paths that uses  $O(k)$  storage and can be updated in  $O(1)$  time.*

**THEOREM 3.3.** *There is a streaming algorithm that maintains a  $2k$ -simplification for convex planar paths under the Hausdorff error function (or the Fréchet error function) and that is 3-competitive with respect to  $Opt(k)$ . The algorithm uses  $O(k)$  additional storage and each point is processed in  $O(\log k)$  time.*

### 3.2 The error oracle for $xy$ -monotone paths

We use the notion of width for approximating  $error_H$  of an  $xy$ -monotone path. The *width* of a set of points with respect to a given direction  $\vec{d}$  is the minimum distance of two lines being parallel to  $\vec{d}$  that enclose the point set. Let  $w(i, j)$  be the width of the points in subpath  $P(i, j)$  with respect to the direction  $\vec{p_i p_j}$ . Since  $P(i, j)$  is  $xy$ -monotone,

it is contained inside the axis-parallel rectangle defined by  $p_i$  and  $p_j$ . Therefore,  $w(i, j)/2 \leq error_H(p_i p_j) \leq w(i, j)$  and  $w(i, j)$  can be used as a 2-approximate error oracle for  $error_H(p_i p_j)$ .

Agarwal and Yu [3] have described a streaming algorithm for maintaining a core-set that can be used to approximate the width of a set in any direction. More precisely, given a data stream  $p_0, p_1, \dots$ , they maintain an  $\varepsilon$ -core-set of size  $O(1/\sqrt{\varepsilon})$  in  $O(1/\sqrt{\varepsilon})$  amortized time per point; with this core-set one can get a  $(1 + \varepsilon)$ -approximation of the width in any direction, which can be used to get a  $(2 + \varepsilon)$ -approximate error oracle.

**LEMMA 3.4.** *There is a  $(2 + \varepsilon)$ -approximate error oracle for the Hausdorff error function on  $xy$ -monotone paths that uses  $O(k^2/\sqrt{\varepsilon})$  storage and has  $O(k/\sqrt{\varepsilon})$  amortized update time.*

**PROOF.** Although our algorithm only needs the approximate errors of the links  $q_{i-1} q_{i+1}$  to decide which point  $q_s$  is erased next, we must maintain a core-set for each link that might be needed at some later time in our simplification. These are the links  $q_i q_j$ , with  $0 \leq i < j - 1 < 2k + 1$ . So we need to maintain a core-set for each of these  $O(k^2)$  links. Considering a new point  $q_{2k+2} = p_{n+1}$ , we must create  $O(k)$  new core-sets, one for each of the links  $q_i p_{n+1}$ , with  $0 \leq i \leq 2k$ . We create such core-sets for the links  $q_i p_{n+1}$ , by copying the core-sets  $q_i q_{2k+1}$  and ‘inserting’ point  $p_{n+1}$  to them using the algorithm by Agarwal and Yu. When some point  $q_s$  is removed from the simplification in Step 3 of our algorithm and the link  $q_{s-1} q_{s+1}$  is added, the core-sets for all links that start or end at  $q_s$  have become meaningless and are therefore deleted. The bounds follow.  $\square$

**THEOREM 3.5.** *There is a streaming algorithm that maintains a  $2k$ -simplification for  $xy$ -monotone planar paths under the Hausdorff error function (or the Fréchet error function) and that is  $(4 + \varepsilon)$ -competitive with respect to  $Opt(k)$ . The algorithm uses  $O(k^2/\sqrt{\varepsilon})$  additional storage and each point is processed in  $O(k/\sqrt{\varepsilon})$  amortized time.*

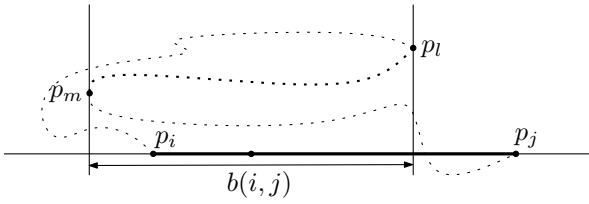
## 4. THE FRÉCHET ERROR FUNCTION

We now turn our attention to the Fréchet error function. We will show that we can obtain an  $O(1)$ -competitive algorithm for arbitrary paths. The first property we need is that the Fréchet error function is monotone. This has in fact already been proven by Agarwal *et al.* [1].

**LEMMA 4.1.** [1] *The Fréchet error function is 2-monotone on arbitrary paths.*

### 4.1 The error oracle

Next we turn our attention to the implementation of the error oracle for the Fréchet error function. We use two parameters to approximate  $error_F(p_i p_j)$ . The first one is  $w(i, j)$ , the width of the points of  $P(i, j)$  in the direction of  $\vec{p_i p_j}$ , which we also used to approximate the Hausdorff error in the case of  $xy$ -monotone paths. The other parameter is the length of the largest back-path in the direction of  $\vec{p_i p_j}$ , which is defined as follows. Assume without loss of generality that  $\vec{p_i p_j}$  is horizontal with  $p_j$  to the right of  $p_i$ . For two points  $p_l, p_m$  on the path  $P(i, j)$  with  $l < m$  we define  $P(l, m)$  to be a *back-path* on  $P(i, j)$  if  $(p_m)_x < (p_l)_x$ . In other words  $P(l, m)$  is a back-path if, relative to the direction  $\vec{p_i p_j}$ , we go back when we move from  $p_l$  to  $p_m$ —see



**Figure 3: Relation between Fréchet distance and back-paths.**

Figure 3. The *length* of a back-path  $P(l, m)$  on  $P(i, j)$  is defined to be the length of the projection of  $p_l p_m$  onto a line parallel to  $p_i p_j$ , which is equal to  $(p_l)_x - (p_m)_x$  since we assumed  $p_i p_j$  is horizontal. We define  $b(i, j)$  to be the maximum length of any back-path on  $P(i, j)$ .

LEMMA 4.2. *The Fréchet error of a shortcut  $p_i p_j$  satisfies  $\max(\frac{w(i, j)}{2}, \frac{b(i, j)}{2}) \leq \text{error}_F(p_i p_j) \leq 2\sqrt{2} \max(\frac{w(i, j)}{2}, \frac{b(i, j)}{2})$*

PROOF. As above we will without loss of generality assume that  $p_i p_j$  is horizontal with  $p_j$  to the right of  $p_i$ .

We observe that  $\text{error}_F(p_i p_j) \geq \text{error}_H(p_i p_j) \geq w(i, j)/2$ . Next we will show that  $b(i, j)/2 \leq \text{error}_F(p_i p_j)$ . Consider a back-path  $P(l, m)$  on  $P(i, j)$  determining  $b(i, j)$ , as shown in Figure 3. Let  $r$  be the point on the line through  $p_i p_j$  midway between  $p_l$  and  $p_m$ , that is, the point on the line through  $p_i p_j$  with  $x$ -coordinate  $((p_l)_x + (p_m)_x)/2$ . Note that  $r$  does not necessarily lie on  $p_i p_j$ . The Fréchet distance between  $p_i p_j$  and  $P(i, j)$  is determined by some optimal pair of parametrizations of  $p_i p_j$  and  $P(i, j)$  that identifies each point  $p$  of  $P(i, j)$  with a point  $\bar{p}$  on  $p_i p_j$  in such a way that if  $p$  comes before  $q$  along  $P(i, j)$  then  $\bar{p}$  does not come later than  $\bar{q}$  along  $p_i p_j$ . Now consider the images  $\bar{p}_l$  and  $\bar{p}_m$ . If  $\bar{p}_l$  lies to the left of  $r$  then  $|p_l \bar{p}_l| > b(i, j)/2$ . If, on the other hand,  $\bar{p}_l$  lies on or to the right of  $r$  then  $\bar{p}_m$  lies on or to the right of  $r$  as well, and we have  $|p_m \bar{p}_m| > b(i, j)/2$ . We conclude that  $\max(w(i, j)/2, b(i, j)/2) \leq \text{error}_F(p_i p_j)$ , which proves the first part of the lemma.

For the second part of the lemma we need to show that  $\text{error}_F(p_i p_j) \leq \sqrt{2} \max(w(i, j), b(i, j))$ . It is convenient to think about the Fréchet distance in terms of the man-dog metaphor. In these terms, we have to find a walking schedule where the man walks along  $p_i p_j$  and the dog walks along  $P(i, j)$  such that they never go back along their paths and their distance is never more than  $\sqrt{2} \max(w(i, j), b(i, j))$ . We can find such a walk as follows. Denote the position of the man by  $p_{\text{man}}$  and the position of the dog by  $p_{\text{dog}}$ . Initially  $p_{\text{man}} = p_{\text{dog}} = p_i$ . Let  $\ell$  be the vertical line through  $p_i$ . Of all the intersection points of  $\ell$  with  $P(i, j)$ , let  $p$  be one furthest along  $P(i, j)$ . (If  $\ell$  does not intersect  $P(i, j)$  except at  $p_i$ , then  $p = p_i$ .) We let the dog walk along  $P(p_i p)$ , while the man waits at  $p_i$ . Let  $q$  be an arbitrary point on  $P(p_i p)$ . Then there must be points  $p_l, p_m$  with  $l < m$  such that  $(p_l)_x \geq (p_i)_x$  and  $(p_m)_x \leq (q)_x$ . Hence, we have  $|(q)_x - (p_i)_x| \leq (p_l)_x - (p_m)_x \leq b(i, j)$ . Furthermore,  $|(q)_y - (p_i)_y| \leq w(i, j)$ . Hence, during this first phase we have  $|p_{\text{man}} p_{\text{dog}}| \leq \sqrt{2} \max(w(i, j), b(i, j))$ .

We continue the walk as follows. Sweep  $\ell$  to the right. Initially  $\ell$  will intersect  $P(p_i p_j)$  in only one point. As long as this is the case, we set  $p_{\text{man}} = \ell \cap p_i p_j$  and we set  $p_{\text{dog}} = \ell \cap P(p_i p_j)$ . During this part we clearly have  $|p_{\text{man}} p_{\text{dog}}| \leq w(i, j)$ . At some point  $\ell$  may intersect  $P(p_{\text{dog}}, p_j)$  in one (or

more) point(s) other than  $p_{\text{dog}}$ . When this happens we take the intersection point  $p$  that is furthest along  $P(p_{\text{dog}}, p_j)$ , and let the dog proceed to  $p$  while the man waits at his current position. By the previous argument,  $|p_{\text{man}} p_{\text{dog}}| \leq \sqrt{2} \max(w(i, j), b(i, j))$  during this phase. Then we continue to sweep  $\ell$  to the right again, letting  $p_{\text{man}} = \ell \cap p_i p_j$  and  $p_{\text{dog}} = \ell \cap P(p_i p_j)$ . The process ends when the sweep line reaches  $p_j$ . We have thus found a walking schedule with  $|p_{\text{man}} p_{\text{dog}}| \leq \sqrt{2} \max(w(i, j), b(i, j))$  at all times, finishing the proof of the lemma.  $\square$

According to the above lemma, in order to approximate  $\text{error}_F(p_i p_j)$  it suffices to approximate  $\max(w(i, j), b(i, j))$ . In the previous section we already described how to approximate  $w(i, j)$ , when we were studying the Hausdorff error function for  $xy$ -monotone paths. Next we describe a method for approximating  $b(i, j)$ , and show how to combine these two methods to build the oracle for  $\text{error}_F(p_i p_j)$ . (Note that if there are no back-paths, then the Fréchet error is equal to the Hausdorff error, so the case of  $xy$ -monotone paths for Hausdorff error is a special case of our current setting.)

In the algorithm as presented in Section 2 we need to maintain (an approximation of) the error of each shortcut  $q_l q_{l+2}$  in the current simplification. For this we need to know the maximum length of a back-path on the path from  $q_l$  to  $q_{l+2}$ . The operations we must do are to add a point  $q_{\ell+2} = p_{n+1}$  at the end of the simplification, and to remove a point  $q_s$  from the simplification. To this end we maintain the following information. For the moment let's assume that all we need is the maximum length of the back-path with respect to the positive  $x$ -direction. Then we maintain for each link  $p_i p_j$  of the simplification the following values:

- (i)  $b(i, j)$ , the maximum length of a back-path (w.r.t. the positive  $x$ -direction) on  $P(i, j)$ ;
- (ii)  $x_{\max}(i, j)$ , the maximum  $x$ -coordinate of any point on  $P(i, j)$ ;
- (iii)  $x_{\min}(i, j)$ , the minimum  $x$ -coordinate of any point on  $P(i, j)$ .

Now consider a shortcut  $q_l q_{l+2}$ . Let  $q_l = p_i$ ,  $q_{l+1} = p_t$  and  $q_{l+2} = p_j$ . Then  $b(i, j)$ , the maximum length of a back-path on  $P(q_l, q_{l+2}) = P(i, j)$ , is given by

$$\max(b(i, t), b(t, j), x_{\max}(i, t) - x_{\min}(t, j)).$$

Adding a point  $q_{\ell+2}$  is easy, because we only have to compute the above three values for  $q_{\ell+1} q_{\ell+2}$ , which is trivial since  $q_{\ell+1}$  and  $q_{\ell+2}$  are consecutive points on the original path. Removing a point  $q_s$  can also be done in  $O(1)$  time (let  $q_{s-1} = p_i$  and  $q_{s+1} = p_j$ ): above we have shown how to compute  $b(i, j)$  from the available information for  $q_{s-1} q_s$  and  $q_s q_{s+1}$ , and computing  $x_{\max}(i, j)$  and  $x_{\min}(i, j)$  is even easier.

Thus we can maintain the maximum length of a back-path. There is one catch, however: the procedure given above maintains the maximum length of a back-path *with respect to a fixed direction* (the positive  $x$ -direction). But in fact we need to know for each  $q_i q_{i+2}$  the maximum length of a back-path with respect to the direction  $\vec{q_i q_{i+2}}$ . These directions are different for each of the links and, moreover, we do not know them in advance. To overcome this problem we define  $2\pi/\alpha$  equally spaced canonical directions, for a suitable  $\alpha > 0$ , and we maintain, for every link  $p_i p_j$ , the

information described above for each direction. Now suppose we need to know the maximum length of a back-path for  $p_i p_j$  with respect to the direction  $\overrightarrow{p_i p_j}$ . Then we will use  $b_{\vec{d}}(p_i p_j)$ , the maximum length of a back-path with respect to  $\vec{d}$  instead, where  $\vec{d}$  is the canonical direction closest to  $\overrightarrow{p_i p_j}$  in clockwise order. In general, using  $\vec{d}$  may not give a good approximation of the maximum length of a back-path in direction  $\overrightarrow{p_i p_j}$ , even when  $\alpha$  is small. However, the approximation is only bad when  $w(i, j)$  is relatively large, which means that the Fréchet distance can still be approximated well. This is made precise in the following lemmas.

**LEMMA 4.3.** *Let  $w$  be the width of  $P(i, j)$  in direction  $\overrightarrow{p_i p_j}$ , let  $b$  be the maximum length of a back-path on  $P(i, j)$  in direction  $\overrightarrow{p_i p_j}$ , and let  $b^*$  be the maximum length of a back-path on  $P(i, j)$  in direction  $\vec{d}$ , where  $\vec{d}$  is the canonical direction closest to  $\overrightarrow{p_i p_j}$  in clockwise order. Then we have:  $b^* - \tan(\alpha) \cdot w \leq b \leq b^* + \tan(\alpha) \cdot (b^* + w)$ .*

**PROOF.** We first show that  $b \leq b^* + \tan(\alpha) \cdot (b^* + w)$ . Let the sub-path  $P(l, m)$  have the maximum back-path length in the direction  $\overrightarrow{p_i p_j}$ . Consider two half-lines originating from  $p_m$  and being parallel to  $\overrightarrow{p_i p_j}$  and  $\vec{d}$ . Let  $\beta$  denote the angle between these two half-lines. Because  $\vec{d}$  is the canonical direction closest to  $\overrightarrow{p_i p_j}$  in clockwise order, clearly  $\beta \leq \alpha$ . Let  $p$  and  $q$  be the orthogonal projections of  $p_l$  onto the lines through  $p_m$  in direction  $\overrightarrow{p_i p_j}$  and  $\vec{d}$ , respectively. We distinguish four cases, depending on whether  $p_l$  is above or below the line through  $p_m$  in direction  $\overrightarrow{p_i p_j}$  and whether  $p$  is right or left to  $p_m$  in direction  $\vec{d}$ . Note that  $q$  always is right to  $p_m$  in direction  $\overrightarrow{p_i p_j}$ . All possible cases are illustrated in Figure 4. The corresponding proof to each case is as follows.

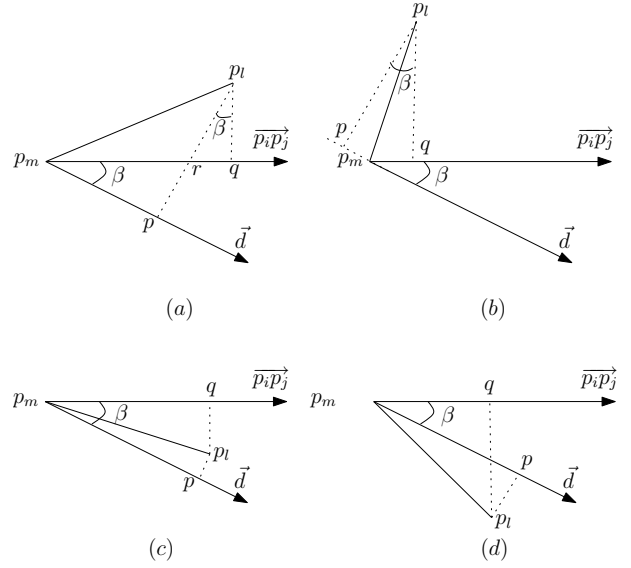
$$\begin{aligned}
\text{(a)} \quad b = |p_m q| &\leq |p_m p| + |pr| + |rq| \\
&\leq |p_m p| + |p_m p| \tan(\beta) + |p_l q| \tan(\beta) \\
&\leq b^* + \tan(\alpha) \cdot (b^* + w) \\
\text{(b)} \quad b = |p_m q| &\leq |p_l q| \tan(\beta) \\
&\leq w \tan(\alpha) \\
&\leq b^* + \tan(\alpha) \cdot (b^* + w) \\
\text{(c)} \quad b = |p_m q| &\leq |p_m p_l| \\
&\leq |p_m p| + |pp_l| \\
&\leq |p_m p| + |p_m p| \tan(\beta) \\
&\leq b^* + \tan(\alpha) \cdot (b^* + w) \\
\text{(d)} \quad b = |p_m q| &\leq |p_m p| \\
&\leq b^* \\
&\leq b^* + \tan(\alpha) \cdot (b^* + w)
\end{aligned}$$

The same elementary arguments can be used to show that  $b^* - \tan(\alpha) \cdot w \leq b$ .  $\square$

The final oracle is now defined as follows. Let  $w^*$  be the approximation of the width of  $P(i, j)$  in direction  $\overrightarrow{p_i p_j}$  as given by Agarwal and Yu's  $\varepsilon$ -core-set method, and let  $b^*$  be the maximum length of a back-path on  $P(i, j)$  in direction  $\vec{d}$ , where  $\vec{d}$  is the canonical direction closest to  $\overrightarrow{p_i p_j}$  in clockwise order. Then we set

$$error_{\mathbb{F}}^*(p_i p_j) := \sqrt{2} \cdot \max(w^*, b^* + \tan(\alpha) \cdot (b^* + w^*)).$$

Combing Lemma 4.2 with the observations above, we can prove the following lemma.



**Figure 4: Illustration for the proof of Lemma 4.3.**

**LEMMA 4.4.**  $error_{\mathbb{F}}(p_i p_j) \leq error_{\mathbb{F}}^*(p_i p_j) \leq 2\sqrt{2}(1 + \varepsilon)(1 + 4 \tan(\alpha)) \cdot error_{\mathbb{F}}(p_i p_j)$

**PROOF.** Let  $w$  be the width of  $P(i, j)$  in direction  $\overrightarrow{p_i p_j}$ , let  $b$  be the maximum length of a back-path on  $P(i, j)$  in direction  $\overrightarrow{p_i p_j}$ . Because  $w^*$  is the width of an  $\varepsilon$ -core-set, we have  $w \leq w^* \leq (1 + \varepsilon)w$ . Using Lemma 4.2 we get

$$\begin{aligned}
error_{\mathbb{F}}(p_i p_j) &\leq 2\sqrt{2} \cdot \max\left(\frac{w}{2}, \frac{b}{2}\right) \\
&\leq \sqrt{2} \cdot \max(w^*, b^* + \tan(\alpha) \cdot (b^* + w)) \\
&\leq \sqrt{2} \cdot \max(w^*, b^* + \tan(\alpha) \cdot (b^* + w^*)) \\
&= error_{\mathbb{F}}^*(p_i p_j).
\end{aligned}$$

On the other hand

$$\begin{aligned}
error_{\mathbb{F}}^*(p_i p_j) &= \sqrt{2} \cdot \max(w^*, b^* + \tan(\alpha) \cdot (b^* + w^*)) \\
&\leq \sqrt{2} \cdot \max((1 + \varepsilon)w, b + \tan(\alpha)w \\
&\quad + \tan(\alpha) \cdot (b + \tan(\alpha)w + (1 + \varepsilon)w)) \\
&\leq \sqrt{2}(1 + \varepsilon) \cdot \max(w, b + b \tan(\alpha) \\
&\quad + 3w \tan(\alpha)) \\
&\leq \sqrt{2}(1 + \varepsilon)(1 + 4 \tan(\alpha)) \cdot \max(w, b) \\
&\leq 2\sqrt{2}(1 + \varepsilon)(1 + 4 \tan(\alpha)) \cdot \max\left(\frac{w}{2}, \frac{b}{2}\right) \\
&\leq 2\sqrt{2}(1 + \varepsilon)(1 + 4 \tan(\alpha)) \cdot error_{\mathbb{F}}(p_i p_j)
\end{aligned}$$

$\square$

Taking  $\varepsilon$  and  $\alpha$  sufficiently small, we get our final result.

**THEOREM 4.5.** *There is a streaming algorithm that maintains a  $2k$ -simplication for arbitrary planar paths under the Fréchet error function and that is  $(4\sqrt{2} + \varepsilon)$ -competitive with respect to  $Opt(k)$ . The algorithm uses  $O(k^2/\sqrt{\varepsilon})$  additional storage and each point is processed in  $O(k/\sqrt{\varepsilon})$  amortized time.*

## 5. CONCLUDING REMARKS

We presented the first line-simplification algorithms in the streaming model. We obtained algorithms with  $O(1)$  competitive ratio for convex planar paths and  $xy$ -monotone planar paths under the Hausdorff error function (or the Fréchet error function), and for general planar paths under the Fréchet distance. Our results imply linear-time approximation when  $k$  is a constant (where the approximation

factor is with respect to the optimal solution using half the number of links).

Our algorithms all use resource augmentation: they maintain a  $2k$ -simplification but we compare the error of our simplification to the error of an optimal  $k$ -simplification. One obvious question is whether we can do with less, or maybe no, resource augmentation. In the appendix we show that this is not the case for general planar paths under the Hausdorff error function, but note that we have not been able to give any  $O(1)$ -competitive algorithm for this case, not even with resource augmentation. Thus there is a significant gap between our positive and our negative results.

Another aspect where improvement may be possible is the implementation of the error oracles, which need  $O(k^2)$  storage for  $xy$ -monotone paths under the Hausdorff error function and for general paths under the Fréchet distance. For instance, if we can maintain core-sets in a streaming setting such that one can also merge two core-sets, then this will reduce the dependency on  $k$  in the storage from quadratic to linear. (Note that we need to be able to do an unbounded number of merges.)

Our general approach extends to higher dimensions (but the approximation factors and running times will change).

## 6. REFERENCES

- [1] P.K. Agarwal, S. Har-Peled, N.H. Mustafa and Y. Wang. Near-linear time approximation algorithms for curve simplification. *Algorithmica* 42:203–219 (2005).
- [2] P.K. Agarwal and K. R. Varadarajan. Efficient algorithms for approximating polygonal chains. *Discr. Comput. Geom.* 23:273–291 (2000).
- [3] P.K. Agarwal, H. Yu. A Space-Optimal Data-Stream Algorithm for Coresets in the Plane. In: *Proc. 23th ACM Symp. Comput. Geom.*, to appear.
- [4] H. Alt and M. Godau. Computing the Fréchet distance between two polygonal curves. *Int. J. Comput. Geom. Appl.* 5:75–91 (1995).
- [5] W.S. Chan and F. Chin. Approximation of polygonal curves with minimum number of line segments. In *Proc. 3rd Annu. Internat. Sympos. Algorithms Comput.*, LNCS 650, pages 378–387, 1992.
- [6] D.H. Douglas and T.K. Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Canad. Cartog.* 10:112–122, 1973.
- [7] M. Godau. A natural metric for curves: Computing the distance for polygonal chains and approximation algorithms. In *Proc. 8th Annu. Sympos. Theoret. Asp. Comput. Sci. (STACS)*, pages 127–136, 1991.
- [8] M.T. Goodrich. Efficient piecewise-linear function approximation using the uniform metric. *Discr. Comput. Geom.* 14:445–462 (1995).
- [9] L.J. Guibas, J.E. Hershberger, J.S.B. Mitchell, and J.S. Snoeyink. Approximating polygons and subdivisions with minimum link paths. *Int. J. Comput. Geom. Appl.* 3:383–415 (1993).
- [10] S.L. Hakimi and E.F. Schmeichel. Fitting polygonal functions to a set of points in the plane. *CVGIP: Graph. Models Image Process.* 53:132–136, 1991.
- [11] J. Hershberger and J. Snoeyink. An  $O(n \log n)$  implementation of the Douglas-Peucker algorithm for

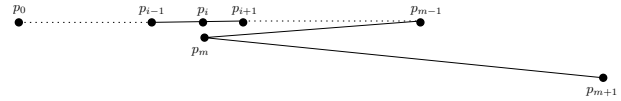


Figure 5: Base path component for Theorem A.1.

line simplification. In *Proc. 10th ACM Symp. Comput. Geom.*, pages 383–384, 1994.

- [12] J. Hershberger and J. Snoeyink. Cartographic line simplification and polygon CSG formulae in  $O(n \log^* n)$  time. In *Proc. 5th Int. Workshop Algorithms and Data Structures (WADS)*, LNCS 1272, pages 93–103, 1997.
- [13] H. Imai and M. Iri. An optimal algorithm for approximating a piecewise linear function. *Inf. Proc. Lett.* 9:159–162 (1986).
- [14] H. Imai and M. Iri. Polygonal approximations of a curve-formulations and algorithms. In: G.T. Toussaint (ed.), *Computational Morphology*, North-Holland, pages 71–86, 1988.
- [15] A. Melkman and J. ORourke. On polygonal chain approximation. In: G.T. Toussaint (ed.), *Computational Morphology*, North-Holland, pages 87–95, 1988.
- [16] S.M. Muthukrishnan. Data Streams: Algorithms and Applications. <http://athos.rutgers.edu/~muthu/stream-1-1.ps>, 2003.
- [17] H. Zarrabi-Zadeh and T. Chan. A Simple Streaming Algorithm for Minimum Enclosing Balls. In: *Proc. 18th Can. Conf. Comput. Geom.*, pages 139–142, 2006.

## APPENDIX

### A. THE HAUSDORFF ERROR FUNCTION FOR GENERAL PATHS

In this appendix we show that for the Hausdorff error function it is not possible to have a streaming algorithm that maintains a path with less than  $2k$  points whose competitive ratio (with respect to  $Opt(k)$ ) is bounded, unless the algorithm uses  $\Omega(n/k)$  additional storage. In fact, this even holds when the input path is known to be  $y$ -monotone.

**THEOREM A.1.** *Let  $\mathcal{A}$  be a streaming algorithm that maintains an  $(2k - 1)$ -simplification for an arbitrary planar path  $P(n)$ , and that is able to store at most  $m - 1$  of the input points, where  $2k + 2 \leq m \leq n/k$ . For any  $c > 0$  and  $n \geq km + 1$ , there is an  $y$ -monotone path  $p_0, p_1, \dots, p_n$  such that  $error_H(Q_{\mathcal{A}(2k-1)}(n)) > c \cdot error_H(Q_{Opt(k)}(n))$ .*

**PROOF.** Figure 5 shows the basic component of the path which has the following properties.

- (i) Points  $p_0, \dots, p_{m-1}$  are collinear,
- (ii)  $|p_i p_{i+1}| > c \cdot d_H(p_{m-1}, p_i p_{m+1})$  for all  $0 \leq i \leq m - 2$ ,
- (iii)  $d_H(p_{m-1}, p_{i-1} p_{m+1}) > c \cdot d_H(p_{m-1}, p_i p_{m+1})$  for all  $1 \leq i \leq m - 1$

To obtain a configuration with the above properties, we take a horizontal line  $\ell$  and a point  $p_{m-1}$  on  $\ell$ . We put  $p_{m+1}$  below  $\ell$  and arbitrarily far from  $p_{m-1}$  to the right of  $p_{m-1}$  such that its distance to  $\ell$  is greater than  $(c+\epsilon)^{m-1}$  where  $\epsilon > 0$  is an arbitrarily small number. We put  $p_i$  ( $i = 0, \dots, m - 2$ ) on  $\ell$  to the left of  $p_{m-1}$  such that  $p_i p_{m+1}$  is tangent to the

circle whose radius is  $(c+\epsilon)^{m-i-1}$  and whose center is  $p_{m-1}$ . The point  $p_i$  always exists, because  $d_H(p_{m+1}, \ell) > (c+\epsilon)^i$ .

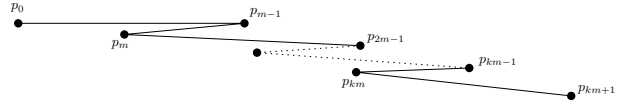
Let  $\mathcal{A}$  be a simplification algorithm being able to store at most  $m-1$  points. Upon the arrival of  $p_{m-1}$ , the algorithm  $\mathcal{A}$  is required to delete one of the past points, because it cannot store  $m$  points. Let  $p_i$ , with  $1 \leq i \leq m-2$ , be the deleted point, and let the next point,  $p_m$ , be slightly below  $p_i$  (i.e.  $|p_i p_m| \cong 0$ ). Up to here, by choosing  $p_m$  in  $Q_{\mathcal{A}(1)}(m)$ , we have  $error_H(Q_{\mathcal{A}(1)}(m)) = error_H(Q_{Opt(1)}(m)) = 0$ . Now consider the next point  $p_{m+1}$ , which lies on its position according to our construction. Obviously,  $Q_{Opt(1)}(m+1)$  is  $p_0, p_i, p_{m+1}$ , and its Hausdorff error is  $d_H(p_{m-1}, p_i p_{m+1})$ . Since  $\mathcal{A}$  has missed the point  $p_i$ ,  $Q_{\mathcal{A}(1)}(m+1)$  is  $p_0, p_j, p_{m+1}$  for some  $j \neq i$ . There are three possibilities for  $j$ :

1.  $0 \leq j < i$ : using the property (iii) we have:

$$\begin{aligned} error_H(Q_{\mathcal{A}(1)}(m+1)) &= d_H(p_{m-1}, p_j p_{m+1}) \\ &> c \cdot d_H(p_{m-1}, p_i p_{m+1}) \\ &= c \cdot error_H(Q_{Opt(1)}(m+1)) \end{aligned}$$

2.  $i < j \leq m-1$ : using the property (ii) we have:

$$\begin{aligned} error_H(Q_{\mathcal{A}(1)}(m+1)) &\geq d_H(p_m, p_j p_{m+1}) \cong |p_i p_j| \\ &> c \cdot d_H(p_{m-1}, p_i p_{m+1}) \\ &= c \cdot error_H(Q_{Opt(1)}(m+1)) \end{aligned}$$



**Figure 6: A path that cannot be simplified within a bounded competitive ratio.**

3.  $j = m$ : using the property (ii) we have:

$$\begin{aligned} error_H(Q_{\mathcal{A}(1)}(m+1)) &= d_H(p_{m-1}, p_0 p_m) \cong |p_i p_{m-1}| \\ &> c \cdot d_H(p_{m-1}, p_i p_{m+1}) \\ &= c \cdot error_H(Q_{Opt(1)}(m+1)) \end{aligned}$$

Therefore, in order to be within a bounded competitive ratio,  $\mathcal{A}$  must store at least the two points  $p_{m-1}$  and  $p_m$ , which leads to a 2-simplification.

We concatenate  $k$  of these components in such a way that for any two consecutive components, the first two points of the latter lie on the last two points of the former as illustrated in Figure 6. Other than the first and the last points, it is straightforward to show that  $\mathcal{A}$  has to store 2 points of each component to be within a bounded competitive ratio. This implies  $error_H(Q_{\mathcal{A}(2k-1)}) > c \cdot error_H(Q_{Opt(k)}(n))$ .  $\square$