A Streaming Algorithm for the Convex Hull

Raimi A. Rufai*† Dana S. Richards‡

Abstract

Consider a base station in a wireless sensor network that receives incoming input points and must maintain a running convex hull within a memory constraint. We give a new streaming algorithm that processes each point in time $\mathcal{O}(\log k)$ where k is the memory constraint, while maintaining an optimal area error of $\mathcal{O}\left(1/k^2\right)$.

1 Introduction

A streaming algorithm is an on-line approximation algorithm constrained to work within a memory budget. When more memory than the allowed budget is demanded, we must make decisions on what is worth keeping and what must be discarded. A streaming algorithm has three parts: an initialization procedure, a processing algorithm for each successive input, and a facility for answering queries using the restricted memory.

2 Related Work

Preparata gave an exact online algorithm [5] but with no memory constraints. The streaming algorithm proposed by Hershberger and Suri [1, 3, 2] maintains extreme points in k uniformly spaced directions and another k extreme points in adaptively sampled directions. Their algorithm has a distance error of $\mathcal{O}\left(1/k^2\right)$; no area measure was reported.

Lopez and Reizner [4] proposed an algorithm for approximating an n-gon by a k-gon, $k < n$. Their algorithm builds an inscribed k-gon by repeatedly removing an ear of minimum area until only k vertices remain. (An *ear* of a convex polygon is any triangle formed by three consecutive vertices.) However their algorithm, unlike ours, is not on-line, as all the vertices of the n-gon are known ahead of time.

3 Streaming Algorithm

Let $C = (p_1, p_2, ..., p_n)$ be a sequence of vertices of a convex polygon in counter-clockwise order. Each contiguous triple (p, q, r) in C defines a measure $\Delta_q = \textsc{goodness}(p, q, r)$, which is associated with the vertex q. We will think of Δ_q as measuring the *goodness* of q. Note that Δ_q is a local measure and depends only on q and its two immediate neighbors in C. When a direct neighbor is inserted or deleted, the goodness must be recomputed. The function GOODNESS can be defined in various ways: as the area of the triangle $\triangle pqr$, as its perimeter, as the length of the segment pr, as the height of the triangle pqr relative to base pr, or even as the angle $\angle q$ in $\triangle pqr$. This yields different variants of the same algorithm. In this section, we shall mainly address the area variant.

3.1 Initialize

The procedure INITIALIZE in Algorithm 1 initializes a balanced binary search tree T and a priority queue H to store the NODE references using two different keys. While points in T are ordered by their polar angles relative to a centroid, the points in H are keyed on their goodness.

Algorithm 1: INITIALIZE(P)

Input : P: The first 3 input points in a data stream S.
Output: T: balanced BST with vertices of conv(P) sorted by angles about centroid c;
H: min-heap of vertices of $conv(P)$ using GOODNESS as priority.

1 $L \leftarrow \text{conv}(P)$
2 $c \leftarrow \textsc{centroid}(L)$
3 $(N, W, S, E) \leftarrow \textsc{directionalExtrema}(L)$
4 **foreach** $p \in L$ **do**
5 　$\Theta \leftarrow \textsc{polar}(p, c)$
6 　**if** $p \in (N, W, S, E)$ **then**
7 　　$\Delta \leftarrow \infty$
8 　**else**
9 　　$\Delta \leftarrow \textsc{goodness}(L.\textsc{pred}(p), p, L.\textsc{succ}(p))$
10 　$x \leftarrow \textsc{Node}(p, \Delta_p, \Theta_p, \text{false})$
11 　$T.\textsc{insert}(\Theta_p, x)$
12 　$H.\textsc{insert}(\Delta_p, x)$
13 **return** (T, H, c, k)

The structure L in Step 1 is a cyclic array and supports PRED and SUCC operations. The function $\textsc{Node}(p, \Delta_p, \Theta_p, \textsc{deleted})$ creates a new node

*Department of Computer Science, George Mason University, rrufai@gmu.edu
†SAP Labs, Inc., 111 Rue Duke, Montreal, QC H3C 2M1, Canada, raimi.rufai@sap.com
‡Department of Computer Science, George Mason University, richards@gmu.edu

(a 4-tuple), whose attributes can be accessed using the attribute names POINT, GOODNESS, POLAR, and DELETED. Clearly initialization takes constant time.

3.2 Process

Algorithm 2: PROCESS(T, H, c, k, p)

Input : T: balanced BST with $\leq k$ nodes;
H: min-heap of the nodes of T;
p: new point; k: memory budget
Output: T: a balanced BST updated with p,
H: a min-heap updated with p.

1 $x \leftarrow$ NODE$(p, 0, \text{POLAR}(p, c), \text{false})$
2 $(T, H) \leftarrow$ UPDATEHULL(T, H, c, x)
3 **if** $|T| > k$ **then**
4 $\quad (T, H) \leftarrow$ SHRINKHULL(T, H)
5 **return** (T, H)

Procedure PROCESS is invoked each time a new point arrives. A new node x is created and used to update the current hull by invoking procedure UPDATEHULL. The call to UPDATEHULL(T, H, c, x) in line 2 of Procedure PROCESS updates the structures T and H with x. If the point associated with x falls within the interior of the current convex hull or on its boundary, it is discarded. This test is done in Line 4 of UPDATEHULL. Further, x's goodness is computed and if it is smaller than H.MINIMUM, again x is discarded. Otherwise, the chain of vertices that lie between the two new neighbors of x on the hull are deleted from both T and H. This deletion is done in Lines 8 to 16 of UPDATEHULL. The goodness of x's neighbors are then updated. The directional extrema are also updated if required.

Whenever the number of nodes in T exceeds k, the procedure SHRINKHULL is called to choose one vertex for deletion. This is done by calling DELETEMIN() on the min-heap structure H to obtain the node q that should be deleted. The procedure then updates the GOODNESSes of q's neighbors and deletes q from T.

This algorithm is sensitive to the order in which the points arrive in the stream. Consider the six points A, B, C, D, E, F shown in Figure 1 and Figure 2 below.

3.3 Query

Algorithm QUERY is invoked to obtain the current hull at any point in the streaming process. It simply traverses T to return the hull vertices in a cyclic list, and runs in linear time.

Algorithm 3: UPDATEHULL(T, H, c, x)

Input : T: balanced BST with $\leq k$ of conv(S);
H: min-heap of $\leq k$ nodes;
x: new node.
Output: T: balanced BST updated with x;
H: min-heap updated with x.

1 T.INSERT(x)
2 $y \leftarrow T$.PRED(x)
3 $z \leftarrow T$.SUCC(x)
4 **if** *not* CONTAINS$(\triangle ycz, x)$ **then**
5 $\quad (s, t) \leftarrow$ TANGENTS(T, x)
6 $\quad x.\Delta \leftarrow$ GOODNESS(s, x, t)
7 $\quad$ **if** $x.\Delta \geq H$.MINIMUM() **then**
8 $\quad\quad w \leftarrow T$.SUCC$(s)$
9 $\quad\quad$ **while** $w \neq t$ **do**
10 $\quad\quad\quad w$.DELETED $\leftarrow$ *true*
11 $\quad\quad\quad H$.CHANGEKEY$(w, -\infty)$
12 $\quad\quad\quad T$.DELETEKEY(w)
13 $\quad\quad\quad w \leftarrow T$.SUCC$(s)$
14 $\quad\quad q \leftarrow H$.MINIMUM()
15 $\quad\quad$ **while** q.DELETED **do**
16 $\quad\quad\quad q \leftarrow H$.DELETEMIN()
17 $\quad\quad H$.INSERT(x)
18 $\quad\quad H$.CHANGEKEY$(s, \text{GOODNESS}(T.\text{PRED}(s), s, x))$
19 $\quad\quad H$.CHANGEKEY$(t, \text{GOODNESS}(x, t, T.\text{SUCC}(t)))$
$\quad\quad$ ▷ Update extrema if needed ◁
20 $\quad\quad (N, W, S, E) \leftarrow$ UPDATEEXTREMA(T, c, x)
21 $\quad\quad$ **foreach** $n \in (N, W, S, E)$ **do**
$\quad\quad\quad$ ▷ To prevent the deletion of an extremum ◁
22 $\quad\quad\quad H$.CHANGEKEY(n, ∞)
23 $\quad$ **else**
24 $\quad\quad T$.DELETEKEY(x)
25 **else**
26 $\quad T$.DELETEKEY(x)
27 **return** (T, H)

Algorithm 4: SHRINKHULL(T, H)

Input : T: BST with $k+1$ vertices of conv(S);
H: min-heap of $k+1$ vertices of conv(S).
Output: T: BST with k vertices of conv(S);
H: min-heap of k vertices of conv(S).

$q \leftarrow H$.DELETEMIN()
$p \leftarrow T$.PRED(q)
$r \leftarrow T$.SUCC(q)
T.DELETEKEY(q)
H.CHANGEKEY$(p, \text{GOODNESS}(T.\text{PRED}(p), p, r))$
H.CHANGEKEY$(r, \text{GOODNESS}(p, r, T.\text{SUCC}(r)))$
return (T, H)

3.4 Complexity Analysis

Lemma 1 *Procedure* UPDATEHULL *runs in* $\mathcal{O}(\log k)$ *amortized time on the input stream* S.

Proof. The initial steps take $\mathcal{O}(\log k)$ time using standard BST techniques. Step 4 takes $\mathcal{O}(1)$ time. The

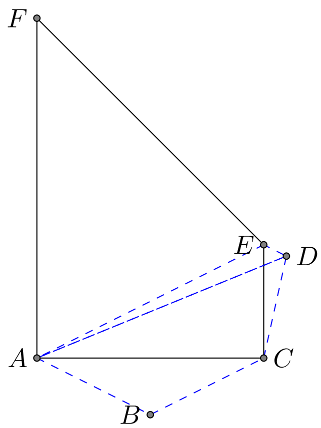

Figure 1: $k = 4$, arrival sequence: A, B, C, D, E, F. D is deleted after E arrives, and B after F.

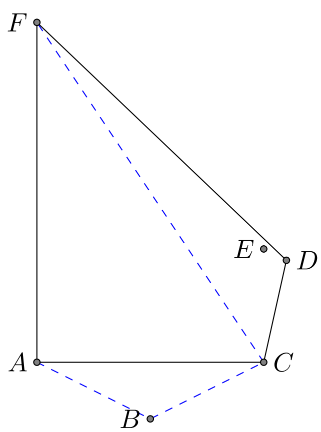

Figure 2: $k = 4$ with arrival sequence: A, B, C, D, F, E. B is deleted after F arrives. E is deleted since it is an interior point.

call to TANGENTS takes $\mathcal{O}(\log k)$ time [5]. The rest of the procedure — Steps 8 through 16 — deletes a vertex chain that no longer belongs to the hull. Since these vertices are only deleted once per point in S, the total cost over all invocations of the procedure UPDATEHULL is $\mathcal{O}(n \log k)$, where n is the length of S. □

Lemma 2 *Procedure* SHRINKHULL *runs in time* $\mathcal{O}(\log k)$.

Proof. Every step of Procedure SHRINKHULL takes $\mathcal{O}(\log k)$. □

Lemma 3 *Procedure* PROCESS *runs in time* $\mathcal{O}(\log k)$ *time.*

Proof. Each invocation of PROCESS makes a single call to UPDATEHULL and at most a single call to SHRINKHULL. Thus PROCESS also runs in $\mathcal{O}(\log k)$ time. □

Lemma 4 *Let T_{i-1} be the convex hull computed before invoking Algorithm* UPDATEHULL*, and let T_i be the resulting hull after it returns. Then the following invariant holds*

$$|T_{i-1}| \leq |T_i|. \tag{3.1}$$

Proof. Consider the invocation of UPDATEHULL on an arbitrary point p_i. The fate of p_i is one of the following two cases.

Case 1 **(p_i lies in the interior of T_{i-1}.)**
UPDATEHULL *ignores p_i, in which case the hull does not grow and $T_i = T_{i-1}$.*

Case 2 **(p_i lies in the exterior of T_{i-1}.)**
UPDATEHULL *expands T_{i-1} by adding p_i to the hull and therefore T_i has a bigger area than T_{i-1}.*

□

Lemma 5 *When $k \geq |\operatorname{conv}(S)|$ the algorithm computes the exact convex hull of S.*

Proof. The algorithm then is equivalent to that of Preparata [5]. □

3.5 Error Analysis

We only discuss in this section the relative area error, which is defined as

$$err_{\text{area}}(P, P') = \frac{|\text{area}(P) - \text{area}(P')|}{\text{area}(P)} \tag{3.2}$$

where P denotes the vertex set of the true convex hull, and P' that of the approximate convex hull.

Lemma 6 *Each deletion from a convex $(k+1)$-gon by Algorithm* SHRINKHULL *introduces an error no worse than $\mathcal{O}(1/k^3)$.*

Proof. Let $m = k+1$. Let Q be a convex m-gon and let $e_1, e_2, ..., e_m$ be its ears. Let $|e_i|$ denote the area of e_i. Let $Q_i' = Q - e_i$ denote the k-gon that would result if e_i were deleted. Therefore, the ratio $|e_i|/|Q|$ represents the area error that would result from deleting e_i. Further, let R_m denote a regular m-gon with unit area, and let R be the circumradius of R_m.

Renyi and Sulanke [6] proved that

$$\frac{1}{|Q|^m} \prod_{i=1}^{m} |e_i| \leq |r|^m, \tag{3.3}$$

whenever r is an ear of R_m.

By taking logarithms and invoking the mean-value theorem, it is clear that there must exist at least one ear e_j in Q such that $\frac{|e_j|}{|Q|} \leq |r|$. Since

$$|r| = 4R^2 \frac{\pi^3}{m^3}\left[1 - \frac{\pi^2}{m^2} + \mathcal{O}\left(\frac{1}{m^4}\right)\right], \tag{3.4}$$

it follows that

$$\frac{|e_j|}{|Q|} < 4R^2 \frac{\pi^3}{m^3} \tag{3.5}$$

$$= \mathcal{O}\left(\frac{1}{k^3}\right). \tag{3.6}$$

□

Lemma 7 *Let $e_1, e_2, ..., e_m$ denote the sequence of ears deleted by the streaming algorithm. Then*

$$|e_i| \leq |e_{i+1}| < H.\text{MINIMUM} \text{ for all } i = 1, 2, ..., m-1. \tag{3.7}$$

Proof. Recall that Algorithm UPDATEHULL only inserts a new node if its goodness is greater than $H.$MINIMUM. By definition, $H.$MINIMUM increases with each deletion. Before the the ith deletion, $H.\text{MINIMUM} = |e_i|$, but becomes $|e_{i+1}|$ afterwards. □

Note that the computed hull consists of four (x-y monotone) chains: from W to N, from N to E, from E to S, and from S to W. Our discussion will only be for the chain from W to N. Suppose that chain is s_1, s_2, $\ldots s_l$. Let s_0 be a short vertical side below W and s_{l+1} be a short horizontal side to the right of N. (The reason for these two additional sides is to automatically take into account the fact that all points seen will be in a bounding box, as indicated by the next lemma. If we do not maintain the bounding box, then our chains are not monotone and the definitions below would be more complex.) Let p_i be the vertex common to s_i and s_{i+1}.

Lemma 8 *After processing the points in S, the directional extrema (N, W, S, E), maintained by Algorithm* UPDATEHULL *define an axis-parallel bounding box B that contains* $\text{conv}(S)$.

Proof. Note that these directional extrema are extreme over all of S in the four axis-parallel directions. Suppose there were some point p in S not contained in B. Further suppose, without loss of generality, that p lies above B. Then p must be more extreme than N in the positive y direction, a contradiction. □

The *outer ear* for side s_i is the triangle formed by s_i and the extensions of the sides s_{i-1} and s_{i+1}. The *flap* for side s_i is a trapezoidal subset of its outer ear: it is the region $\square p_i p_{i+1} q_1 q_2$ where $\overline{q_1 q_2}$ is parallel to s_i and q_1 and q_2 are on the boundary of the outer ear. The height of the flap, h_i, perpendicular to s_i will be chosen to be the minimum value that maintains an invariant. The h_i is used in the analysis and is not calculated by the algorithm.

We will choose h_i after each deletion that creates the side s_i so that this invariant holds (if s_i was not created by a deletion, then $h_i = 0$).

Invariant 1 *Each deleted point, not in the hull itself, is from one of the flaps. Further, the area of the corresponding ear is contained in the flap.*

When p_i is deleted and a new side $s = \overline{p_{i-1}p_{i+1}}$ created, the corresponding h is calculated: it is minimized subject to the constraint that the new flap includes the flaps from s_{i-1} and s_i. Let h' be the height of p_i in $\triangle p_{i-1}p_i p_{i+1}$. Then $h \leq 2h'$, by similar triangles. Recall that h corresponds to a triangle (ear) chosen because it had minimum area. Hence we get the following lemma.

Lemma 9 *The area of any flap is $\leq 4H.$*MINIMUM().

Proof. Suppose s_i is the side for a given flap. Let $a = H.\text{MINIMUM}()$. The height h satisfies

$$h \leq 2h' \leq 2\frac{2a}{s_i}. \tag{3.8}$$

Since the top of the trapezoid is less than its base, it fits within a parallelogram M of base s_i and height h. □

The following theorem gives an upper bound on the area error for processing $n \gg k$ points.

Theorem 10 *The total area error incurred in the streaming process is bounded above by $\mathcal{O}\left(1/k^2\right)$.*

Proof. A deleted point contributes to the error if it is outside the computed hull. Some or all of its ears may not be in the computed answer. We know that each such ear is from some flap. A single flap may cover many such (overlapping) ears, but the total missed area of all such ears is bounded by the area of that flap. Hence the total area error is bounded by the total area of all the flaps.

By Lemma 6, $H.$MINIMUM is at most $\mathcal{O}\left(1/k^3\right)$ and since there are k outer ears, the total error is $\mathcal{O}\left(1/k^2\right)$. □

Note that, in general, not all deletions will have an impact on the final k-gon returned after processing all the points in the stream. However, when an adversary could provide a stream of points that all lie on the convex hull, such as the vertices of a regular n-gon, the above error bound, being a worst-case bound, will still apply.

Theorem 11 *(Lopez and Reisner [4]) Given an adversarial input, the total area error accumulated by all the deletions is at least*

$$2\pi^2\left[\frac{1}{k^2}-\frac{1}{n^2}\right]. \tag{3.9}$$

Proof. This bound was obtained by [4], but in their case, they had access to all the vertices offline, as mentioned earlier in Section 2. □

4 Empirical Results

A stream S of 10,000 random points lying on a common circle was generated. We then fed 33 random shuffles of S to the streaming algorithm and computed the mean distance and area relative errors. These were then used to compute the lower and upper bounds as defined in Theorem 10 and Theorem 11. The empirical area error is neatly sandwiched between the two bounds, as expected.

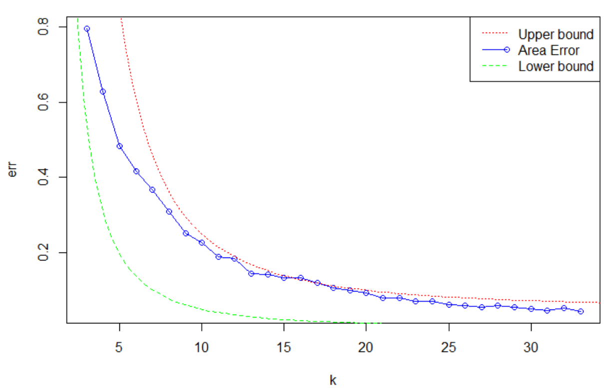

Figure 3: Empirical area error sandwiched between the curves of the lower and upper bounds

The relative distance measure between the set P of vertices of the true convex hull and the set P' of vertices of the approximate hull is defined as

$$err_{\delta,\mathrm{diam}}(P,P')=\delta(P,P')/\operatorname{diam}(P), \tag{4.1}$$

where $\delta(\cdot,\cdot)$ stands for the Hausdorff distance[1].

Figure 4 and Figure 5 show the distance and area relative errors using three goodness measures: the area of an ear, the height of the ear, and the angle made by the ear with the centroid. What is clear from these results is that the measure of goodness based on the area and that based on the distance (height of the ears) are both very effective. The results for the angle of an ear were not as good, indicating that the relation between the measure of goodness and the error measure is important.

[1]The Hausdorff distance between a finite point set P and another Q is defined as $\delta(P,Q) = \max(\max_{p\in P}\min_{q\in Q}\|p-q\|, \max_{q\in Q}\min_{p\in P}\|q-p\|)$.

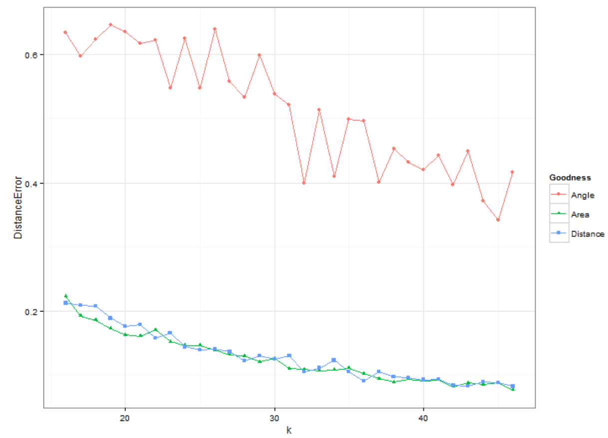

Figure 4: Distance Relative Errors

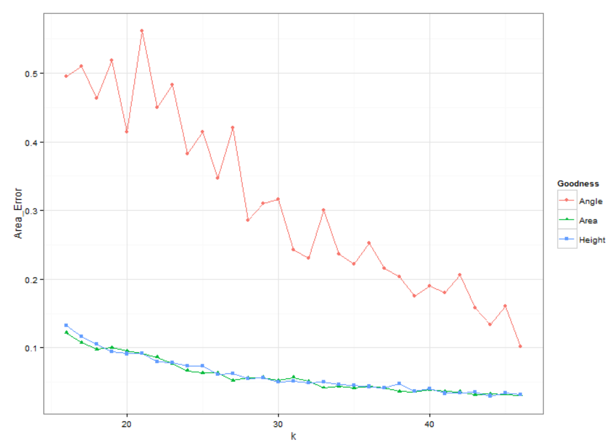

Figure 5: Area Relative Errors

5 A More General Approach

We propose a refinement of Algorithm 2, which uses the idea from Lopez and Reisner [4]. The essential difference is that rather than invoke SHRINKHULL every time the k-gon grows into a $(k+1)$-gon, the algorithm waits until the k-gon grows into an mk-gon for some small constant m before invoking SHRINKHULL. The algorithm PROCESS2 shows the details. This only works, of course, if the memory constraint allows the use of $(m-1)k$ extra memory for processing. The main benefit of this enhancement is that it reduces the effect of the order of the point sequence (illustrated earlier in

Figure 1 and Figure 2), while keeping the same overall asymptotic time bounds. We hope to analyze this approach both analytically and empirically.

Algorithm 5: PROCESS2(T, H, c, k, p)

Input : T: balanced BST with $\leq k$ of conv(S); H: min-heap of $\leq k$ of conv(S); p: new point; k: memory budget

Output: T: a height-balanced BST update with p if on the hull, H: a binary min-heap updated with p if on the hull.

$x \leftarrow$ NODE$(p, 0,$ POLAR(p, c), false$)$
$(T, H) \leftarrow$ UPDATEHULL(T, H, c, x)
if $|T| > mk$ **then**
    while $|T| > k$ **do**
        $(T, H) \leftarrow$ SHRINKHULL(T, H)
return (T, H)

6 Conclusion

We have presented a new streaming algorithm for the convex hull and analyzed its runtime and error bounds. We have proven that it is optimal for the area error measure. We have empirically shown that it is robust with respect to different goodness and error measures. Further analytic results are being studied.

The generalization of this approach to three or higher dimensions is conceptually straightforward. Each new point that is outside the current hull subtends a volume analogous to an ear, which can be given a goodness measure. Points which can be stored in memory are deleted according to this measure. However, we have not explored the computational complexity of these steps.

Acknowledgement

We thank the anonymous referees for their feedback and comments, which have helped us improve the readability of this paper.

References

[1] J. Hershberger and S. Suri. Convex hulls and related problems in data streams. In *Proc. of the ACM/DIMACS Workshop on Management and Processing of Data Streams*, 2003.

[2] J. Hershberger and S. Suri. Adaptive sampling for geometric problems over data streams. *Computational Geometry*, 39(3):191–208, 2008.

[3] J. Hershberger and S. Suri. Simplified planar coresets for data streams. In *Algorithm Theory—SWAT 2008*, pages 5–16. Springer, 2008.

[4] M. A. Lopez and S. Reisner. Efficient approximation of convex polygons. *International Journal of Computational Geometry & Applications*, 10(05):445–452, 2000.

[5] F. Preparata. An optimal real-time algorithm for planar convex hulls. *Communications of the ACM*, 22:402–405, 1979.

[6] A. Rényi and R. Sulanke. Über die konvexe Hülle von n zufällig gewählten Punkten. *Probability Theory and Related Fields*, 2:75–84, 1963. 10.1007/BF00535300.