Computing a Single Cell in the Overlay of two Simple Polygons ⋆

Mark de Berg [a] Olivier Devillers [b,1] Katrin Dobrindt [a,2]
Otfried Schwarzkopf [a,3]

[a] *Dept. of Computer Science, Utrecht University, P.O. Box 80.089, 3508 TB Utrecht, the Netherlands.*

[b] *INRIA, B.P. 93, 06902 Sophia-Antipolis Cedex, France.*

This note combines the lazy randomized incremental construction scheme with the technique of "connectivity acceleration" to obtain an $O(n(\log^\star n)^2)$ time randomized algorithm to compute a single face in the overlay of two simple polygons in the plane.

Key words: computational geometry, randomized algorithms, arrangement.

1 Introduction

The arrangement of n line segments in the plane can be computed using a randomized incremental algorithm in expected time $O(n \log n + A)$, where A is the number of intersection points of the segments. If we are only interested in a single face of the arrangement, marked by a given point—the origin, say—the lazy cleaning scheme of de Berg et al. [BDS94] can be used, resulting in an $O(n\alpha(n) \log n)$ algorithm. [4] If, on the other hand, the line segments are

⋆ This research was supported by the Netherlands' Organization for Scientific Research (NWO) and was partially supported by the ESPRIT III Basic Research Action 6546 (PROMotion).

[1] Part of this work has been done while this author was visiting Utrecht University.

[2] Current address is INRIA.

[3] Current address: Dept. of Computer Science, Pohang University of Science and Technology, Hyoja-Dong, Pohang 790-784, South Korea.

[4] We denote by $\alpha(n)$ the pseudo-inverse of Ackermann's function; α grows to infinity, but very slowly, for $n \leqslant 2^{2^{2^{\cdot^{\cdot^{\cdot^{2}}}}}}$ (a tower of twos of height 2^{65536}) we have $\alpha(n) \leqslant 3$.

connected—for instance if they form two simple polygons—the $O(n \log^\star n)$ "connectivity acceleration" scheme [Sei91,CCT92,Dev92] can be used to obtain a randomized incremental algorithm with expected running time [5] $O(n \log^\star n + A)$.

In this note we consider the problem of computing a single face in the overlay of two simple polygons. This is a special case of both of the situations mentioned above, and so we can either use lazy randomized construction to construct the face in $O(n\alpha(n) \log n)$ expected time; or, alternatively, "connectivity acceleration" to compute the whole overlay in time $O(n \log^\star n + A)$, and can then extract the interesting face in time $O(n)$.

In the following we will show how to combine the two techniques to obtain a randomized algorithm with expected time complexity $O(n(\log^\star n)^2)$. Although we will briefly summarize the two techniques we are using, we will have to assume that the reader is somewhat familiar with them.

2 Incremental randomized algorithms

Randomized incremental construction has become one of the fundamental tools of computational geometry [CS89,Mul94,BDSTY92]. A geometric structure defined by a set S of given geometric objects is computed incrementally, adding the objects in random order while maintaining the structure. In the following we summarize the main ideas and results in our context.

Consider a set S of n line segments in the plane. The *trapezoidation* induced by this set is a subdivision of the plane into *trapezoids* and is obtained by extending vertical segments from every endpoint of a segment or intersection point of two segments upwards and downwards, until we reach another segment—see Figure 1. The randomized incremental algorithm for computing the trapezoidation maintains a *history graph* [BDSTY92,GKS92] of the construction. The history graph is a directed acyclic graph, its nodes are the trapezoids that have been created during the incremental construction. The trapezoids of the current trapezoidation are leaf nodes in the history graph. When a new segment s is added to the structure, the trapezoids of the current trapezoidation that are intersected by s are *located* by a partial traversal of the history graph. The history graph is then *updated* as follows. Each leaf node corresponding to an intersected trapezoid is split by s into at most four new trapezoids. These new trapezoids are added to the history graph as leaves below the intersected

---

[5] We define $\log^{(i)} n$ as $\overbrace{\log \log \cdots \log}^{i \text{ times}} n$; $\log^\star n = \inf\{k \mid \log^{(k)} n \leqslant 1\}$. Thus for any imaginable data set $n \leqslant 2^{65532}$ and thus $\log^\star n \leqslant 5$.

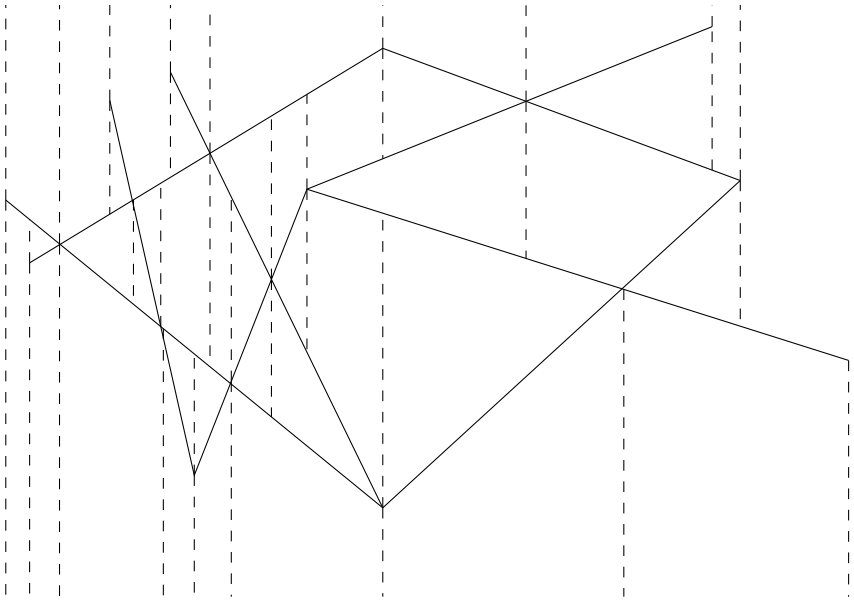

Fig. 1. Example of trapezoidal map

trapezoid (which is now no longer a leaf).

If the segments of S are inserted in random order, the leaf nodes of the history graph intersected by a new segment can be found in $O(\log n)$ expected time by traversing the graph from the root, and the history graph can be updated in expected $O(1 + A/n)$ time, where A is the total number of intersection points.

Theorem 1 [BDSTY92, Proposition 3.4] *An arrangement of n* line segments *can be computed in $O(n \log n + A)$ expected time, where A is the complexity of the arrangement.*

3 Lazy randomized incremental algorithms

While the entire arrangement induced by a set of n line segments might have quadratic combinatorial complexity, the combinatorial complexity of any of its faces is only $O(n\alpha(n))$ [GSS89]. In the following we are interested in computing a single face in the arrangement of line segments, say the face containing the origin, instead of the entire arrangement. To be more precise, we compute the trapezoidation of this face.

However, single faces in arrangements do not fit into the usual framework of randomized incremental construction, because it cannot be decided locally whether a trapezoid belongs to the current structure [BDS94]. More precisely, a newly inserted segment may cut off parts of the current face by separating them from the origin. It is difficult to determine the trapezoids that are cut

off, as it depends on the complete configuration of the segments. *Lazy* randomized incremental construction solves this problem by postponing this decision: When inserting a segment we simply split the intersected trapezoids, and do not attempt to identify and discard the parts that have been cut off from the relevant face. This way we would, of course, end up constructing the full arrangement of segments. Therefore the structure is cleaned after inserting the 2^i-th segment, $1 \leqslant i \leqslant \log n$. To perform these clean-up steps the current trapezoidation is traversed, and the trapezoids outside the relevant face are marked. These "outside" trapezoids remain as leaves in the history graph, but need no further refinement.

The expected size of the structure induced by r segments that is maintained by the algorithm is $O(r\alpha(r))$, and thus is asymptotically not larger than the face itself. Between the clean-ups in step 2^i and 2^{i+1}, $1 \leqslant i \leqslant \log n$, the history graph can be updated in $O(2^i\alpha(2^i))$ expected time. The clean-up in step 2^i takes time proportional to the number of trapezoids in the current structure, which is $O(2^i\alpha(2^i))$. The expected time to compute a single face in an arrangement of n segments is therefore $O(n\alpha(n)\log n)$.

Theorem 2 [BDS94, Theorem 4] *Given a set of n line segments in the plane, the face in the arrangement containing the origin can be computed in $O(n\alpha(n)\log n)$ using $O(n\alpha(n))$ storage.*

4 Accelerated randomized incremental algorithms

If the input segments are somehow connected—for instance, if they form a simple polygon—this can be exploited to accelerate the trapezoidal map algorithm [Sei91,CCT92,Dev92]. The basic idea of this "connectivity acceleration" is that we do not traverse the history graph from the root, but from a previous stage where we precomputed a *conflict graph*, storing all the intersections between the not yet inserted segments and the current trapezoidation. This computation is done by traversing the trapezoidation along the edges of the polygon. By choosing the steps where a conflict graph is computed carefully, namely at stage $n/\log^{(h)} n$, where $0 \leqslant h \leqslant \log^\star n$, the final running time is $O(n \log^\star n)$.

This technique allows to compute the intersection of m simple polygons with a total of n vertices and A intersection points in $O(A + m\log n + n\log^\star n)$. This works similarly to the case of one polygon, but during the conflict graph computation, one vertex of each polygon must be localized in the history graph, adding an overall cost of $O(\log n)$ time for each polygon.

We now summarize results on accelerated randomized incremental construc-

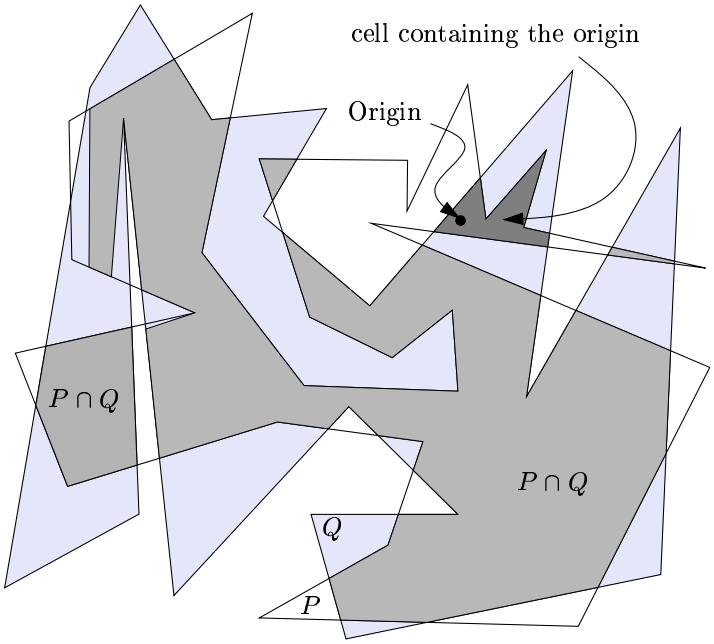

Fig. 2. Example of the overlay of two simple polygons

tion for line segments, specializing [Dev92, Theorem 1] for the case of $A = O(n\alpha(n))$.

Theorem 3 *Given a set of n line segments in the plane,*

(i) The expected size of the conflict graph at stage k is $O((n-k)\alpha(k))$.
(ii) The expected number of edges of the conflict graph created at stage k is $O\left(\frac{n-k}{k}\alpha(k)\right)$.
(iii) The expected size of the history graph at stage k is $O(k\alpha(k))$.
(iv) The expected cost of inserting the rth object in the history graph is $O(\alpha(r)\log r)$.
(v) The expected cost of inserting the rth object in the history graph knowing the conflicts at stage k is $O(\alpha(r)\log(r/k))$.

5 A single face in the intersection of two polygons

At first glance it seems that one should be able to combine the two previous techniques to compute a single face in the overlay of two simple polygons with in total n vertices in $O(n\alpha(n)\log^\star n)$. (Any face in the overlay of two simple polygons is a simple polygon and has $O(n)$ vertices—see Figure 2.) But this does not work out so easily. The main problem is the computation of the conflict graph at stage r: We would like to trace one polygon through the current trapezoidation. This trapezoidation consists of trapezoids of the current relevant face, C_r, and of "outside" trapezoids. By the theorem on higher moments [CS89,Mul94] the number of intersections between the polygon and

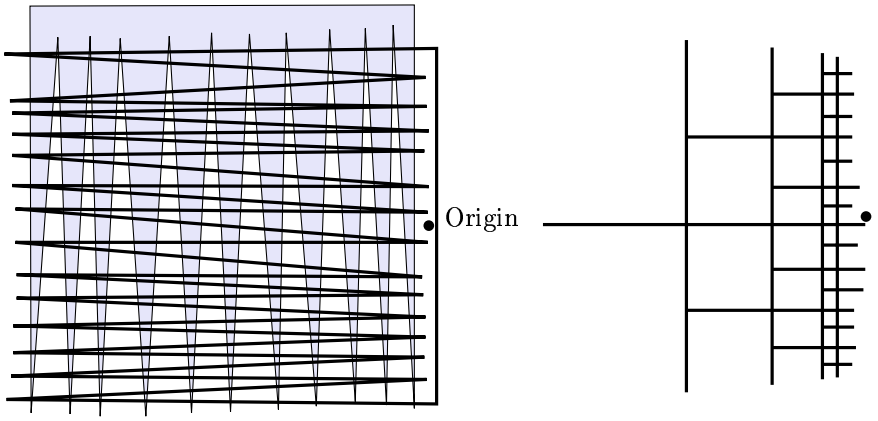

Fig. 3. The number of conflicts may be $\Omega(n \log r)$.

trapezoids of C_r is only $O(n\alpha(r))$. However, there can be $\Omega(n \log r)$ intersections points between the polygon and the "outside" trapezoids. Since a conflict graph construction has to take place when $r > n/\log n$, we cannot afford to trace the polygon through all these "outside" trapezoids.

An example of the $\Omega(n \log r)$ bound is given in Figure 3. The left part shows the two polygons and the position of the origin. The right part shows the "average" shape of the trapezoidal map at some stage r. The overlap of the shaded polygon with this map has linear size, but the overlap of the map with the remaining polygon produces an expected number of $n \log r$ intersection points, most of which concern "outside" trapezoids. The number of intersection with "inside" trapezoids is only linear.

Fortunately we can cope with this problem by separately computing the intersection of the two polygons with the current face C_r and then tracing the polygons only inside C_r. The algorithm is as follows.

Algorithm *ComputeSingleFace*

1. Let $s_1, s_2, \ldots, s_n$ be a random permutation of S;
2. Generate the initial conflict graph $\mathcal{G}_1$ for $s_2, \ldots, s_n$ and the trapezoidation of s_1;
3. $last\_cleanup := 1$; $last\_log := n$;
4. **for** $r = 2$ **to** n
5. **do** Locate s_r in the history graph starting at the conflict graph $\mathcal{G}_{n/last\_log}$;
6. Update the history graph;
7. **if** $r = 2 \cdot last\_cleanup$
8. **then** perform a clean-up step to obtain C_r; $last\_cleanup := r$;
9. **if** $r = n/\log(last\_log)$
10. **then** $last\_log := \log(last\_log)$;
11. Perform a clean-up step to obtain C_r;
12. Compute the intersections points between C_r and both of the two polygons by running the standard accelerated

algorithm twice;

13. Compute $\mathcal{G}_{n/last\_log}$ by tracing the relevant parts of the two polygons through the trapezoidation of C_r;

It remains to analyse this algorithm:

Line 5. By Theorem 3*(v)*, locating the rth segment costs $O(r\alpha(r)\log(r/last\_log))$. Summing over all n segments we get $O(n\alpha(n)\log^\star n)$.

Lines 6 and 8. As in [BDS94, Section 5.2], the update of the history graph in line 6 and the clean-up in line 8 can be done in $O(n\alpha(n))$ time in total.

Line 11. One iteration of line 11 takes $O(r\alpha(r))$ time ([BDS94, Section 5.2]).

Line 12. The intersection points between the simple polygon C_r and one of the input polygons are computed using the connectivity-accelerated algorithm in $O(A+m\log^\star m)$ expected time. Here, A is the number of intersection points between C_r and the polygon, and m is the total size of the two polygons. We have $A = O(n\alpha(r))$, and $m = O(n + r\alpha(r))$, since the complexity of C_r is at most $O(r\alpha(r))$ [BDS94, Theorem 5]. It follows that one iteration of line 12 takes $O(r\alpha(r) + n\log^\star n)$ time.

Line 13. The conflict graph can be computed in time proportional to its size. The expected size is $O(n\alpha(r))$ [BDS94, Theorem 5].

Lines 9–13. Since this loop is executed $\log^\star n$ times, $r < n$ and $\alpha(n) < \log^\star n$, the total expected time complexity of *ComputeSingleFace* is $O\Big(n\alpha(n)\log^\star n + \log^\star n(n\alpha(n) + n\log^\star n\Big)$ which reduces to $O(n(\log^\star n)^2)$.

Theorem 4 *A single face in the overlay of two simple polygons having a total of n vertices can be computed in expected $O(n(\log^\star n)^2)$ time.*

6 Open problems

We would obtain an algorithm with $O(n\alpha(n)\log^\star n)$ expected running time if the construction of the conflict graph could be done in linear time. We do

not know how to achieve this, since the original polygons cannot be traced in the current complete subdivision (the cost of tracing it through the "outside" trapezoids may be $\Omega(n \log n)$).

It is not clear that the face defined by a sample of r edges of two simple polygon actually has complexity $\Theta(r\alpha(r))$ when $r < n/\alpha(n)$. After all, the final result has only linear size.

An interesting generalization is the case of m polygons with a total of n vertices. This problem has applications to path planning in an environment of m polygons of total complexity n. A straightforward divide-and-conquer construction based on our algorithm for pairs of polygons yields $O(n(\log^* n)^2 \log m)$ complexity, which almost matches the $\Omega(n \log m)$ lower bound.

References

[BDSTY92] J.-D. Boissonnat, O. Devillers, R. Schott, M. Teillaud, and M. Yvinec. Applications of random sampling to on-line algorithms in computational geometry. *Discrete Comput. Geom.*, 8:51–71, 1992.

[CCT92] K. L. Clarkson, R. Cole, and R. E. Tarjan. Randomized parallel algorithms for trapezoidal diagrams. *Internat. J. Comput. Geom. Appl.*, 2(2):117–133, 1992.

[CS89] K. L. Clarkson and P. W. Shor. Applications of random sampling in computational geometry, II. *Discrete Comput. Geom.*, 4:387–421, 1989.

[BDS94] M. de Berg, K. Dobrindt, and O. Schwarzkopf. On lazy randomized incremental construction. In *Proc. 26th Annu. ACM Sympos. Theory Comput.*, pages 105–114, 1994.

[Dev92] O. Devillers. Randomization yields simple $O(n \log^* n)$ algorithms for difficult $\Omega(n)$ problems. *Internat. J. Comput. Geom. Appl.*, 2(1):97–111, 1992.

[GKS92] L. J. Guibas, D. E. Knuth, and M. Sharir. Randomized incremental construction of Delaunay and Voronoi diagrams. *Algorithmica*, 7:381–413, 1992.

[GSS89] L. J. Guibas, M. Sharir, and S. Sifrony. On the general motion planning problem with two degrees of freedom. *Discrete Comput. Geom.*, 4:491–521, 1989.

[Mul94] K. Mulmuley. *Computational Geometry: An Introduction Through Randomized Algorithms.* Prentice Hall, Englewood Cliffs, NJ, 1994.

[Sei91] R. Seidel. A simple and fast incremental randomized algorithm for computing trapezoidal decompositions and for triangulating polygons. *Comput. Geom. Theory Appl.*, 1:51–64, 1991.