Convex Hull of the Union of Convex Objects in the Plane: an Adaptive Analysis

Jérémy Barbay* Eric Y. Chen†

Abstract

We prove a tight asymptotic bound of $\Theta(\delta \log(n/\delta))$ on the worst case computational complexity of the convex hull of the union of two convex objects of sizes summing to n requiring δ orientation tests to certify the answer. Our algorithm is deterministic, it uses portions of the convex hull of input objects to describe the final convex hull, and it takes advantage of easy instances, such as those where large parts of two objects are horizontally or vertically separated.

1 Introduction

An adaptive analysis of the computational complexity of a problem considers more parameters than the mere size n of the instance to be solved, such as the size of the result, or more sophisticated measures of the difficulty of the instance. A particular case of this approach has been applied to some fundamental problems in computational geometry, under the name of "output-sensitive" complexity analysis, where instances with a small output are considered easier, such as for the computation of the intersection points of a set of line segments [1, 2, 5, 6], or of the convex hull, discussed here.

The computational complexity of the convex hull has been studied in the worst case over instances of size n [11], over instances of size n and output size h [7], and over instances formed by polygonal chains with a parameterized number of self intersections [9]. Nielsen and Yvinec [10] studied the computation of the convex hull of the union of convex objects in the plane such that the convex hull of any pair of objects can be computed in constant time (such as discs or simple convex objects). Under those conditions, they proposed an algorithm which complexity is expressed as a function of the output size and of the maximum number of intersections of each object with others [10, page 4].

We also consider the computational worst case complexity of the convex hull of the union of convex objects in the plane, but in a different context than Nielson and Yvinec, where the convex hull of any pair of objects can be much more difficult to compute, sometime as much as to require linear time. Since the computation of the planar convex hull can be reduced to the computation of the lower and upper hulls, which can then be merged in constant time, we focus on the computation of the planar upper hull of the convex hull of the union of upper hulls. Let $I = \{A_1, A_2\}$ be an instance composed of two upper hulls in the Euclidean plane, of respective sizes n_1 and n_2 and each given as an array containing the coordinates of its points, ordered in clockwise order: we consider the problem of computing the upper hull of the minimal convex hull containing every point from the instance I, which we call the *merged hull* of I for short.

In this paper, we describe our algorithm, the adaptive analysis of its complexity and the matching adaptive computational lower bound, all in a model where only orientation tests (testing which side of a line is a point, in clockwise order) are allowed on the input. Our algorithm (Section 2) computes the merged hull taking advantage of the ordered representation of the upper hulls in sorted arrays. The analysis of this algorithm introduces the notion of *certificate* of an instance, from which we define our measure of difficulty over the instances. We highlight other application of our techniques in Section 4.

2 Convex Hull Problem

Before introducing our algorithm and its analysis, we introduce the concept of *certificate* of an instance, and some basic techniques on upper hulls.

2.1 Notion of Certificate

An *orientation test* is a constant time operation, which can determine which side a point lies to a line. Whereas an algorithm will perform many orientation tests to solve an instance, only a few are required to check that the result is correct. There is a similar situation in the comparison model: the binary search algorithm performs $\lceil \lg n \rceil + 1$ comparisons to search for the insertion rank of an element in a sorted array, whereas at most two comparisons are required to check the validity of the answer.

Given two upper hulls A and B, the merged hull is easier to compute for some instances than for some oth-

*DCC (Departamento de Ciencias de la Computación), Universidad de Chile, Santiago, Chile, jeremy.barbay@dcc.uchile.cl

†CSCS (Cheriton School of Computer Science) University of Waterloo, Canada. y28chen@uwaterloo.ca

ers. For example, Figure 1 show two instances, each composed of two hulls A and B at various distances from each other. The two instances have exactly the same input size n, and exactly the same output (the hull A), but they are much different in difficulty: one can *verify* that the edges of A exactly form the merged hull in two orientation steps in the first case, while n orientation steps are required in the second case.

We define the notion of *certificate* of an instance as a set of orientation tests which permit to check the validity of the result of the algorithm, not only justifying the presence of each point of the output, but also the exclusion of the other points:

Definition 1 *Consider k upper hulls $A_1, \ldots, A_k$ of respective sizes $n_1, \ldots, n_k$ and their merged hull A, expressed as intervals on $A_1, \ldots, A_k$. A* certificate *of A is a set of orientation tests "$A_i[p]$ is right to the line $\overrightarrow{A_j[q]A_j[k]}$", such that the convex hull of any instance satisfying those orientation tests is given by the description of A. The* size *δ of a certificate is the number of orientation tests composing it.*

Figure 1: The same output, different difficulty

2.2 Basic Operations

An easy type of instances (i.e. of small size of certificate δ) is that where a large section of one of the components of the instance can be "eliminated" by a simple orientation test, in the sense that the corresponding points will not contribute to the merged hull.

Observation 1 *Given a line $\overrightarrow{l}$ and an upper hull A, if the point $A[p]$ is right to $\overrightarrow{l}$ and the slope of $A[p]A[p+1]$ is smaller than the slope of $\overrightarrow{l}$, then all points right of $A[p]$ are right to $\overrightarrow{l}$; if the point $A[p]$ is right to $\overrightarrow{l}$ and the slope of $A[p-1]A[p]$ is greater than the slope of $\overrightarrow{l}$, then all points left of $A[p]$ are right to $\overrightarrow{l}$.*

Another characteristic which can make instances easier, is that it is not necessary to perform a binary search for the two edges of a hull intersecting a line on the whole upper hull at each search: a doubling search [3] algorithm permits to amortize the cost of each search over the whole hull:

Observation 2 *Given a line $\overrightarrow{l}$ and an upper hull A, the edge $(A[p], A[p+1])$ which intersects $\overrightarrow{l}$, if any, can be found in $\mathcal{O}(\log p)$ orientation tests.*

The same holds when searching for the tangent of a hull passing by a specific point: once again a doubling search [3] algorithm permits to amortize the costs of each search over the whole hull.

Observation 3 *The tangent $(x, A[p])$ of a point x with an upper hull A, if any, can be found in $\mathcal{O}(\log p)$ orientation tests.*

Many convex hull algorithms depend on computing the *common tangent* between two convex hulls in linear time. Using the previous observations, we show that if the upper hulls are horizontally separated, this common tangent can be found in time logarithmic not only in the size of the convex hulls [8], but even in time adaptive in the *positions* of the endpoints of the common tangent (i.e. closer to the left extremity is easier).

Lemma 2 *Given upper hulls A and B of respective sizes n and m, horizontally separated by a vertical line l, the common tangent $(A[p], B[q])$ from A to B can be computed in $\mathcal{O}(\log p + \log q)$ orientation steps.*

Proof. This idea is inspired by the prune-and-search method proposed by Kirkpatrick and Snoeyink [8], where the common tangent between two non-intersecting hulls can be computed in $O(\log n_1 + \log n_2)$ time.

Without loss of generality, suppose that A is to the left of l and that B is to its right. The basic operation performed by the algorithm considers the edge in clockwise order $(a, a') \in A$ and an edge $(b, b') \in B$:

1. If both a' and b' are left of the line $\overrightarrow{ab}$, then the points of B at the left of b (inside points of B) can be ignored.

2. Symmetrically, if both a' and b' are right the line $\overrightarrow{ab}$, then the points of A at the right of a (inside points of A) can be ignored.

3. If a' is right to $\overrightarrow{ab}$ and b' is left to it, then the points of A at the right of a and the points of B at the left of b (inside points) can be ignored.

4. If a' is left to $\overrightarrow{ab}$ and b' is right to it, which hull gets reduced depends of the slopes of (a, a') and (b, b'). If the line (a, a') cuts l above the line (b', b) then the points of A at the left of a can be ignored. Otherwise, the points of B at the right of b' can be ignored.

□

The result of Lemma 2 is still useful when only large parts of the upper hulls (and not the upper hulls themselves) are horizontally separated. Such parts which are separated from each other can be found using Observations 2 and 3.

3 Convex Hull of the Union of two objects

Beside the pedagogical interest to expose a simpler algorithm before the more complicated one, the algorithm for the union of two convex object is of independent interest:

- We prove that the certificate it finds is always optimal (which is not the case for the more general algorithm).
- It can be used as a building block for other union algorithms.

3.1 Adaptive Algorithm

Theorem 3 *The description of the merged hull of two upper hulls of respective sizes n_1 and n_2 can be computed in $\mathcal{O}(\delta(\log(n_1/\delta)+\log(n_2/\delta)))$ orientation tests, for an instance of certificate size δ.*

Proof. Let be A and B the two hulls forming the instance, of respective sizes n_1 and n_2. Without loss of generality, we prolongate each hull at each extremity by one vertical edge going to $-\infty$: hence all lines intersecting exactly once a hull are tangents to it, and other lines intersecting a hull once will intersect it a second time.

The algorithm 1 computes a description of the merged hull as a sequence of intervals over $1,\dots,n_1$ and $1,\dots,n_2$, representing consecutive points in A and B. To compute this description, the algorithm traverses the two hulls from left to right through doubling searches, searching for crossing points and discarding whole intervals of points in each hull, after certifying that they cannot contribute to the merged hull.

The invariant is quite simple: each iteration of the loop reduces the instance by discarding more points in each hull, and identifies the rightmost point yet certified to be in the merged hull, a, and its hull of origin A. In particular, all the points on the left of a in A which have not been output yet are certified to be part of the merged hull. Figures 2, 3 and 4 illustrate how the algorithm reduces the instance depending on the position of the intersection of A with the tangent from a to B:

- If $a = a'$, as all the points left of a in A are certified to be in the merged hull, and the points immediately on the right of a in A are right to $\overrightarrow{ab}$, the next point confirmed to be in the merged hull is b, hence reducing the instance by one point.

Algorithm 1 Convex Upper Hull of Two Objects

```
Identify the starting point a, and its hull A.
repeat
  Search in the other hull B for the tangent (a, b).
  Search in the hull A for its rightmost intersection
  a' with the line (a, b).
  if a = a' then
    Output and further ignore points of A left of a.
    Switch (a, A) to (b, B).
  else if a' is on the right of b then
    Further ignore points of B left of b.
    Update a to a' (not ignoring its predecessors).
  else
    Find the common tangent (c, d) between the
    points of A left of a' and the points of B right of
    b, separated by the vertical line passing by b.
    Output and further ignore points of A left of c.
    Switch (a, A) to (d, B).
  end if
until no point is left in any other hull than A.
Output all remaining points of A.
```

- If a' is on the right of b, as B is right to its tangent $\overrightarrow{ab}$, it is below the arc from A between a and a': the points of B between a and a' (or at least those between a and b, which have already been identified) can be further ignored, and all points on the left of a' (included) are certified to be part of the merging hull.
- If a' is on the left of b, A crosses and goes to the right of $\overrightarrow{ab}$ before potentially crossing B: some points in the right of b in B will contribute to the merged hull, and Lemma 2 indicates how to find them.

The algorithm outputs interval of points from the input hulls to describe the merged hull, performing exactly δ iterations because it computes the shortest certificate.

Each iteration corresponds to at most two doubling searches in each hull. As none of the δ doubling searches ever overlap, the number of orientation test in each hull of size n sums up to less than $\delta\log(n/\delta)$, by concavity of the log, hence the result. □

3.2 Optimality of Certificate

Lemma 4 *To determine the upper hull of two upper hulls, $\Omega(\delta)$ orientation tests are required.*

Proof. The certificate used in Algorithm 1 contains δ orientation tests. We prove that any certificate verifying the merged upper hull requires $\delta/4$ orientation tests.

Based on three testing cases in Algorithm 1, we categorize orientation tests into three types. In case 1 and

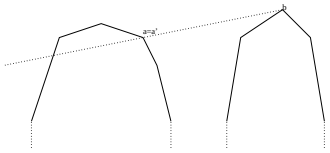

Figure 2: $a = a'$

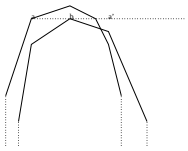

Figure 3: a' is on the right of b

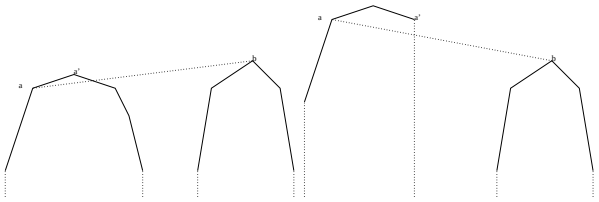

Figure 4: a' is on the left of b

3, the line in the orientation test is from a common tangent between A and B, and, in case 2, it is a tangent from a vertex in A to B. Grouping every 4 adjacent orientation tests together, we can have $\delta/4$ groups.

Then we prove the result by an adversary argument. Consider the vertical strip covered by 4 orientation tests in a group. The merged upper hull between A and B cannot be verified with only two orientation tests. We need at least one extra orientation tests in each group. Therefore, we need at least $\delta/4$ orientation tests. □

4 Conclusion

Convex hull instances with a very large set of points will not appear "out of nowhere": most likely, they will be formed of several objects from a library, for each of which a convex hull can be precomputed. In this context, we have given an algorithm to compute a description of the convex hull of the union of two convex objects which can be used recursively to merge k convex objects. Our algorithm takes advantage of instances where the relative positions of the objects makes the convex hull easier to compute.

As the basic operations are clearly identified in each algorithm, our results are easily generalizable to the transdichotomous computational model as well: each of the basic operation can be supported in time $\mathcal{O}(\log n/\log\log n)$ using a precomputed index [4], changing the complexity of the algorithm to $\mathcal{O}(\delta k \log n/\log\log n)$.

Acknowledgements: Many thanks to Alejandro López-Ortiz for discussing this direction of research, and to Timothy Chan and Alejandro Salinger for pointing to previous works.

References

[1] I. J. Balaban. An optimal algorithm for finding segments intersections. In *Proc. 11th Sympos. on Comput. Geom.*, pages 211–219, 1995.

[2] J. L. Bentley and T. A. Ottmann. Algorithms for reporting and counting geometric intersections. *IEEE Trans. Comput.*, C-28:643–647, 1979.

[3] J. L. Bentley and A. C.-C. Yao. An almost optimal algorithm for unbounded searching. *Information processing letters*, 5(3):82–87, 1976.

[4] T. M. Chan. Point location in o(log n) time, voronoi diagrams in o(n log n) time, and other transdichotomous results in computational geometry. In *Proc. 47th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 333–342, 2006.

[5] B. Chazelle and H. Edelsbrunner. An optimal algorithm for intersecting line segments in the plane. *J. ACM*, 39:1–54, 1992.

[6] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry, Algorithms and Applications*. Springer, 1997.

[7] D. G. Kirkpatrick and R. Seidel. The ultimate planar convex hull algorithm? *SIAM J. Comput.*, 1986. 15(1):287–299.

[8] D. G. Kirkpatrick and J. Snoeyink. Computing common tangents without a separating line. In *WADS '95: Proceedings of the 4th International Workshop on Algorithms and Data Structures*, pages 183–193, London, UK, 1995. Springer-Verlag.

[9] C. Levcopoulos, A. Lingas, and J. S. B. Mitchell. Adaptive algorithms for constructing convex hulls and triangulations of polygonal chains. In *SWAT '02: Proceedings of the 8th Scandinavian Workshop on Algorithm Theory*, pages 80–89, 2002.

[10] F. Nielsen and M. Yvinec. Output-sensitive convex hull algorithms of planar convex objects. *International Journal of Computational Geometry and Applications*, 8(1):39–66, 1998.

[11] F. P. Preparata and S. J. Hong. Convex hulls of finite sets of points in two and three dimensions. *Commun. ACM*, 20:87–93, 1977.