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. For more convex objects, we prove a (non optimal) asymptotic bound of $\mathcal{O}(\delta \sum_{i=1}^{k} \log(n_i/\delta))$ on the worst case computational complexity of the convex hull of the union of k convex objects of respective sizes $(n_1, \ldots, n_k)$ requiring δ orientation tests to certify the answer. Our algorithms are deterministic, they use portions of the convex hull of input objects to describe the final convex hull, and take 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, 6, 7], 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 [13], over instances of size n and output size h [9], and over instances formed by polygonal chains with a parameterized number of self intersections [11]. Nielsen and Yvinec [12] 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 [12, 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 $I = \{A_1, \ldots, A_k\}$ be a set of k upper hulls in the Euclidean plane, of respective sizes $n_1, \ldots, n_k$ 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.

We express the asymptotic performance of our algorithms in the worst case over instances composed of k hulls of sizes $(n_1, \ldots, n_k)$, but also for a particular measure (defined in Section 2.1) of the difficulty δ of the instance: as the asymptotic worst case complexity for parameters k and $(n_1, \ldots, n_k)$ is easily obtained by taking the worst value for δ, our analysis can only be more precise. In particular, our algorithms compute the merged hull of a finite number of hulls ($k \in \mathcal{O}(1)$) in linear time in the worst case over δ, and in sublinear time for many instances. On the other hand, on instances with many objects of finite size (i.e. $k = \Theta(n)$) our algorithm for general k can perform up to $\mathcal{O}(nh)$ operations, which is worse than previously known algorithms [4], which compute the convex hull in time $\mathcal{O}(n \log h)$.

In this paper, we describe our algorithms, the adaptive analysis of their complexity and one 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 first algorithm (Section 3) computes optimally the merged hull of two convex hulls, and our second algorithm (Section 4) computes the merged hull of k convex hulls. Both algorithms are taking advantage of the ordered representation of the upper hulls in

*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

sorted arrays, and their analysis 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 5.

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 others. 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.*

2.2 Basic Operations

An easy type of instances (i.e. of small size of certificate δ) is one 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.

$\delta = 2$ $\delta = O(n)$

Figure 1: The same output, different difficulty

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 algorithm [3] 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 [10], but even in time adaptive in the *positions* of the endpoints of the common tangent (e.g. 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 [10],

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

Before presenting our general algorithm in Section 4, we present a simpler version to compute the union of two convex objects (case where $k = 2$). Beside the pedagogical interest to expose a simpler algorithm before the more complicated one, the algorithm for the union of two convex objects 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 $1, \ldots, n_1$ and $1, \ldots, n_2$, 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:

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 (including b).
        Update a to a' (not ignoring its predecessor on
        A).
    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 $a = a'$, 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.

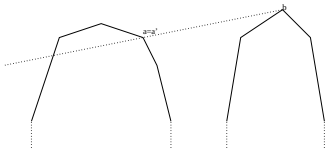

Figure 2: $a = a'$

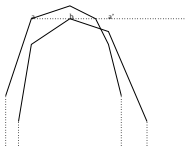

Figure 3: a' is on the right of b

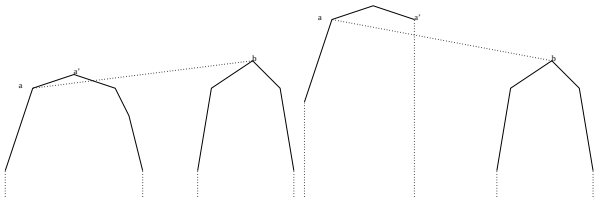

Figure 4: a' is on the left of b

- 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 intervals of points from the input hulls to describe the merged hull, performing exactly δ iterations as 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 overlaps, 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 the Certificate Computed

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, one from each iteration. We proof that any certificate verifying the merged upper hull requires $\Omega(\delta)$ orientation tests.

The proof is by adversary strategy. Consider the three cases in algorithm 1. In first case, the orientation test is defined by a tangent from one hull to the other. In the second case, it is defined by an arc on the outer hull. In the third case, it is defined by a common tangents between two hulls. By each of them, we can march a vertical range on each hull. We group every 4 adjacent orientation tests, which defines a vertical strip over two hulls. Among the 4 orientation tests in one strip, we consider the third orientation test. The relationship of this part of the hulls cannot be determined by orientation tests outside the vertical strip. Without any of these $\delta/4$ orientation tests, we can force the output wrong. Therefore, we need $\Omega(\delta)$ orientation tests. □

4 Convex Hull of the Union of k objects

4.1 Adaptive Algorithm

Using the techniques described in the Section 2.2, our algorithm can take advantage of the order of the representation of the hulls to compute the description of the merged hull.

Theorem 5 *The description of the merged hull of k upper hulls of respective sizes $n_1, \ldots, n_k$ can be computed in $\mathcal{O}(\delta \sum \log(n_i/\delta))$ orientation tests (i.e. $\mathcal{O}(\delta k \log(n/\delta k))$ when $n = \sum_i n_i$, by concavity of the log), for an instance of certificate size δ.*

Proof. Similar to Jarvis's march, Algorithm 2 finds in k orientation tests the first point a of the desired upper hull, and "wraps around" the objects to find the other points. The successors of a in its hull of origin A are then candidates to contribute to the final upper hull.

The convexity of the objects yields two major improvements. First, using Observations 2 and 3, the algorithm can identify a whole range of consecutive edges in the hull A which will contribute to the final upper hull, in logarithmic time. Second, using Lemma 2, the algorithm can identify where the hull A ceases to contribute to the final upper hull, again in logarithmic time.

The technical difficulty of optimally merging $k > 2$ upper hulls is to perform only sequential doubling searches in A, rather than many doubling searches over the same elements of A. This is first achieved by sorting the tangent points of a with other hulls by slope in order to search for the corresponding intersection points in A in sequence, which cost can be amortized. Then, for the hulls which intersect A, it is achieved by performing a single doubling search in A while searching the other hulls for a common tangent, stopping the doubling search in A as soon as a doubling search in another hull B ends, in a way similar to Demaine *et al.* [8]'s intersection algorithm. Even though all the tangents are not computed, the elements covered by doubling search in each hull are covered by some tangent, and can be further ignored, hence the amortization of the cost of those searches.

Once both the leftmost leftend of a common tangent c and the leftmost certified point a' of A have been

computed, the leftmost of those is taken as the last point of A known to be in the final upper hull, and a new iteration can begin.

The complexity of the algorithm is amortized over the δ doubling searches in each upper hull: in a hull of size n_i the algorithm performs at most $\mathcal{O}(\delta)$ doubling searches, of total cost $\mathcal{O}(\delta \log(n_i/\delta))$ (by concavity of the logarithm). Summing over the k hulls yields the result. □

5 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 by itself recursively to merge k convex objects, and one algorithm to compute the description of the convex hull of the union of k convex objects in a more holistic way. Both our algorithms take advantage of the relative positions of the objects.

One could also to consider the intermediate relaxation of those problems: given k convex objects and a parameter $t \leq k$, describe the region covered by at least t of those objects (clearly this is the intersection for $t = k$ and the union for $t = 1$, and a relaxation in between).

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 [5], 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.

Algorithm 2 Convex Upper Hull of k Objects

Identify the starting point a of the merged hull, and its hull A.
repeat
  for each other hull B **do**
    Search in B for the tangent (a, b).
    Add b to a Max-heap H_1, ordered by the slope of (a, b).
  end for
  for each point b of H_1 (in order) **do**
    Search in A for its rightmost intersection a' with (a, b), starting from where the last search in A ended.
    if a' is on the segment $[a, b]$ **then**
      Add B to a queue Q.
    else
      Add a' to a Min-heap H_2, ordered by the x-coordinate of a'.
    end if
  end for
  Search in A and in parallel in all the hulls which are in Q for the common tangent (c, d) with the leftmost point c in A, of right extremity named d in hull B.
  if c is to the left of the leftmost point a' of H_2 **then**
    Output and further ignore points of A left of c.
    Switch (a, A) to (d, B).
  else
    Update a to a' (but do not ignore its predecessors).
  end if
until no point is left in any other hull than A
Output all remaining points of A.

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. Optimal output-sensitive convex hull algorithms in two and three dimensions. *GEOMETRY: Discrete & Computational Geometry*, 16, 1996.

[5] 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.

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

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

[8] E. D. Demaine, A. López-Ortiz, and J. I. Munro. Adaptive set intersections, unions, and differences. In *Proceedings of the 11th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 743–752, 2000.

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

[10] 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.

[11] 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.

[12] 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.

[13] 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.