Kirkpatrick-Seidel's Prune-and-Search Convex Hull Algorithm

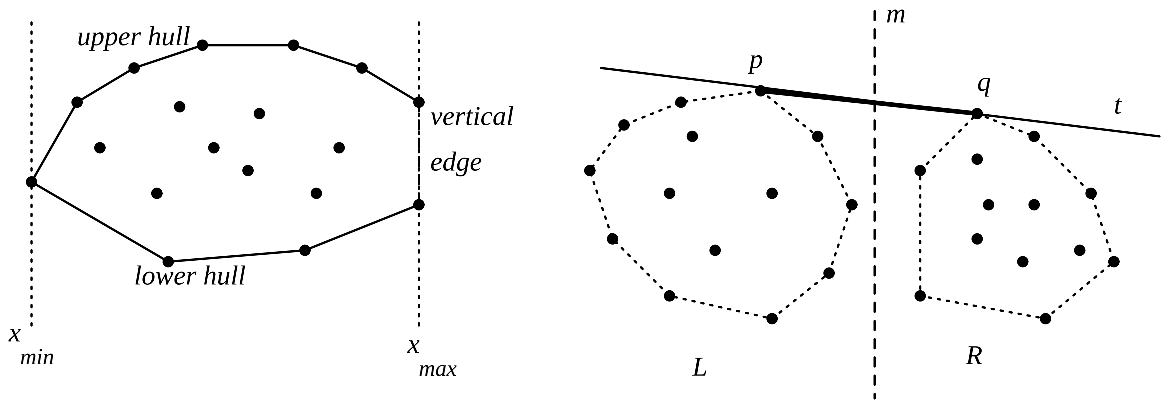
Introduction

This note concerns the computation of the convex hull of a given set $P = \{ p_1, p_2, \cdots, p_n \}$ of n points in the plane. Let h denote the size of the convex hull, ie the number of its vertices. The value h is not known beforehand, and it can range anywhere from a small constant to n. We have already seen that any convex hull algorithm requires at least $\Omega(n \; lg \; n)$ time in the worst case, and have studied a number of algorithms, such as Graham's scan algorithm, whose worst case time complexity is $O(n \; lg \; n)$. If n was the only measure of problem size, then these algorithms are optimal. However, we also know that the Jarvis march algorithm requires $O(nh)$ time. The latter can range anywhere from $O(n)$ to $O(n^2)$ depending on the value of h. Is there an algorithm which is asymptotically superior to both Graham scan and Jarvis march, for all possible values of h? Below, we will describe Kirkpatrick and Seidel's [KiS86] algorithm that requires $O(n \; lg \; h)$ time.

Kirkpatrick-Seidel's algorithm applies a design technique known as the *prune-and-search* method or *Megiddo's technique*. Nimrod Megiddo showed, eg, how this technique can be used to solve fixed dimensional linear programs in linear time [Meg83, Meg84], and how to compute the smallest circle that encloses a finite number of given points in the plane in linear time [Meg89]. Dyer [Dye84] independently discovered the same technique. Many other applications of this powerful algorithm design technique appear in the literature. Edelsbrunner's book [Ede87] also gives a brief description of the method in section 15.6 and shows its applications, eg, to linear programming in chapter 10, and to ham-sandwich cuts in section 14.1. Frances Yao in section 6, chapter 7 of van Leeuwen's book [vanL90] also discusses this technique. The prune-and-search technique can be traced back to the first linear time median finding algorithm of Blum-Floyd-Pratt-Rivest-Tarjan [BFP73]. The latter algorithm finds the median (and in general, the k-th smallest element) of a finite set of given numbers in linear time and is also described in section 10.3 of Cormen-Leiserson-Rivest [CLR91].

Kirkpatrick-Seidel's Algorithm

Consider the minimum and maximum x-coordinates of points in P, denoted $x_{\min}$ and $x_{\max}$. Convex Hull of P can be be viewed as a pair of convex chains called the *upper hull* and the *lower hull* of P (excluding the possible vertical edges at $x_{\min}$ or $x_{\max}$). (See Fig. 1(a).) The algorithm that computes the upper-hull of P is given below. The lower-hull can be computed in a similar manner and is omitted from further discussion.



(a) (b)

Fig. 1 (a) The upper and lower hulls, (b) The upper bridge $\overline{pq}$.

Algorithm *UpperHull*(P)

0. **if** $|P| \le 2$ **then return** the obvious answer
1. **else begin**
2. Compute the *median* x_{med} of x-coordinates of points in P.
3. Partition P into two sets L and R each of size about $n/2$ around the median x_{med}.
4. Find the *upper bridge* $\overline{pq}$ of L and R, $p \in L$, and $q \in R$
5. $L' \leftarrow \{ r \in L \mid x(r) \le x(p) \}$
6. $R' \leftarrow \{ r \in R \mid x(r) \ge x(q) \}$
7. $LUH \leftarrow UpperHall(L')$
8. $RUH \leftarrow UpperHall(R')$
9. **return** the concatenated list LUH, $\overline{pq}$, RUH as the upper hull of P.
10. **end**

Analysis

This is a divide-&-conquer algorithm. The key step is the computation of the *upper bridge* in step 4 which is based on the prune-&-search technique. (See Fig. 1(b).) In the next section we will show that this step can be done in $O(n)$ time. We also know that step 2 can be done in $O(n)$ time by the linear time median finding algorithm. Hence, steps 3-6 can be done in $O(n)$ time. For the purposes of analyzing algorithm *UpperHall*(P), let us assume the upper hull of P consists of h edges. Our analysis will use both parameters n (input size) and h (output size). Let $T(n, h)$ denote the worst-case time complexity of the algorithm. Suppose LUH and RUH in steps 7 and 8 consist of h_1 and h_2 edges, respectively. Since $|L'| \le |L|$ and $|R'| \le |R|$, the two recursive calls in steps 7 and 8 take time $T(n/2, h_1)$ and $T(n/2, h_2)$ time. (Note that $h = 1 + h_1 + h_2$. Hence, $h_2 = h - 1 - h_1$.) Therefore, the recurrence that describes the worst-case time complexity of the algorithm is

$$T(n,h) = \begin{cases} O(n) + \max_{h_1} \{ T(\frac{n}{2}, h_1) + T(\frac{n}{2}, h-1-h_1) \} & \text{if } h > 2 \\ O(n) \quad \text{if } h \le 2 \end{cases}$$

Theorem: $T(n, h) = O(n \ lg \ h)$.

Proof: Suppose the two occurences of $O(n)$ in the above recurrence are at most cn, where c is a suitably large constant. We will show by induction on h that $T(n, h) \le cn \ lg \ h$ for all n and $h \ge 2$. For the base case where $h = 2$, $T(n, h) \le cn \le cnlg2 = cn \ lg \ h$. For the inductive case,

$$\begin{aligned} T(n, h) &\le cn + \max_{h_1} \{ \ c\frac{n}{2} \ lg \ h_1 + c\frac{n}{2} \ lg \ (h - 1 - h_1) \ \} \\ &= cn + c\frac{n}{2} \cdot \max_{h_1} lg \ (\ h_1(h - 1 - h_1) \ ) \\ &\le cn + c\frac{n}{2} \ lg \ (\frac{h}{2} \cdot \frac{h}{2}) \\ &= cn + c\frac{n}{2} \ 2 \ lg \ \frac{h}{2} \\ &= cn \ lg \ h \ . \end{aligned}$$

Finding the Upper Bridge in Linear Time

The problem is this: we are given a collection P of n points in the plane, which is separated into two non-empty subsets L and R by a known vertical line m, with L on the left and R on the right. We wish to find a line t passing through one point from each subset, such that none of the given points lies above t. See Fig 1(b). In other words, we want the upper exterior common tangent (the upper bridge) of the convex hulls of L and R. If the convex hulls of L and R were known, the common tangents could easily be found in linear time (see the merge step in the divide-&-conquer convex hull algorithm discussed earlier in the course). However, computing the convex hull of $\Theta(n)$ points costs $\Theta(n \ lg \ n)$ time in the worst case.

Computation of the upper bridge of L and R can be formulated as a 2-variable linear program with n linear constraints, and hence, can be solved in $O(n)$ time by Megiddo's linear-programming algorithm. The linear program formulation is as follows. Suppose the equation of the (non-vertical) bridge line t is $y = \alpha x + \beta$. The two coefficients α (the slope) and β (the y-intersept) are the two unknowns that we have to compute. Suppose the x-coordinate of the vertical separator line m between L and R is $x = a$. (See Fig. 1(b).) Then, the y-coordinate of the intersection of t and m is $y_o = \alpha a + \beta$. Clearly any line that is at or above every point of $L \cup R$ cannot intersect m at a y-coordinate lower than y_o. This gives us the desired 2-variable linear program; find α and β to:

minimize $\quad \alpha \ a + \beta$
subject to:

$$\alpha x(p_i) + \beta \ge y(p_i) \qquad \text{for all } p_i \in L \cup R \ .$$

Instead of discussing Megiddo's solution of this linear program, we will discuss Kirkpstrick-Seidel's direct method. The key to their algorithm is a simple prune-&-search criterion that in linear time allows us to eliminate a good many of the points that do not define the upper bridge.

Let us fix for the moment our attention on lines of a particular slope α. We can compute in linear time a supporting line of L of slope α. Suppose this line is tangent to L at some point $p \in L$. We can do the same for R and obtain a supporting line of R of slope α tangent to R at some point $q \in R$. Now if the line $\overline{pq}$ has slope less than α, then so must the common tangent t; similarly, if $\overline{pq}$ has slope greater than α then so does t; and if $\overline{pq}$ has slope α then $t = \overline{pq}$. See Fig 2(a,b).

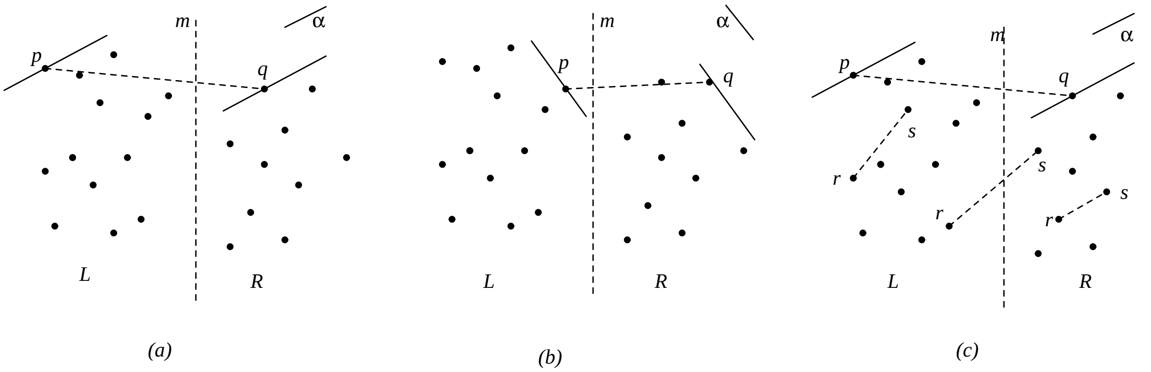


Fig 2.

Now suppose the first case holds, so slope of t is less than α. Let r, s be any two points of $L \cup R$, such that r is to the left of s and the line $\overline{rs}$ has slope *greater* than α. Then we can conclude that t cannot pass through r, because a line of slope less than α through r must pass below s. See Fig 2(c). The second case, where the slope of t is greater than α, is entirely symmetrical: we can eliminate the second member of any pair r, s, with r to the left of s, if the line $\overline{rs}$ has slope less than α.

These remarks suggest the following prune-&-search method. Pair up the n given points in an arbitrary way, and find the *median* slope α of the $n/2$ lines defined by those pairs. Now compute the supporting lines of, respectively, L and R of slope α and assume these lines, respectively, are tangent to L and R at points $p \in L$ and $q \in R$. It should be obvious why the median is a good choice: since half the pairs have slope less than α, and half have slope greater than α. Therefore, half the pairs will satisfy the criterion, no matter whether the slope of $\overline{pq}$ is greater or less than α. So, in either case we eliminate one point from half the pairs, for a total of at least $n/4$ points. Of course if $\overline{pq}$ has slope α then we are done.

It is possible to find the median slope in time linear in n using the same median finding algorithm mentioned in the previous section. The other operations clearly take $O(n)$ time. Therefore, in linear time we either stop, or eliminate at least $n/4$ of the original points. If we repeatedly apply this elimination (or pruning) process on the remaining points, we are guaranteed to find the common tangent, at a total cost of $O(n + (3/4)n + (3/4)^2 n + \cdots) = O(n)$ time. Note that we may eliminate a different number of points from L and from R, but this does not affect the analysis. Now we can state the following results.

Theorem: *The upper bridge of two vertically separated point sets can be computed in linear time.*

Corollary: *Kirkpatrick-Seidel's convex hull algorithm takes* $O(n\ lg\ h)$ *time.*

Kirkpatrick-Seidel also showed that in terms of the two parameters n and h, $\Omega(n\ lg\ h)$ is a worst-case lower bound to compute the convex hull (using a general computational model known as the algebraic decision tree model). Therefore, their algorithm is worst-case optimal.

3D Convex Hulls

The convex hull of n points in R^3 can also be computed in $O(n\ lg\ n)$ time by a divide-&-conquer algorithm. Recently, Edelsbrunner and Shi [EdS91] have shown that 3D convex hull can be computed in $O(n\ lg^2\ h)$ time, where h is the number of hull vertices (extreme points). Furthermore, Kenneth Clarkson and Peter Shor [ClS89] give a randomized 3D convex hull algorithm with $O(n\ lg\ h)$ *expected* time. Chazelle and Matousek [ChM93] have reported that derandomizing an algorithm of [ClS89] gives an $O(n\ lg\ h)$ time deterministic algorithm. See also [CSY95].

References

[BFP73] Blum, M., W. Floyd, V. Pratt, R. Rivest, R.E. Tarjan, *"Time bounds for selection"*, J. of Computer and System Sciences, vol. 7, pp. 448-461, 1973.

[CSY95] Chan, T.M.Y., J. Snoyink, C.K. Yap, *"Output-sensitive construction of polytopes in four dimensions and clipped Voronoi diagrams in three"* Proc. SODA'95, pp. 282-291, 1995.

[ChM93] Chazelle, B., and J. Matousek, *"Derandomizing an output-sensitive convex hull algorithm in three dimensions"*, Manuscript, 1993.

[ClS89] Clarkson, K., and P.W. Shor, *"Applications of random sampling in computational geometry, II"*, Discrete & Computational Geometry, vol. 4, no. 5, pp. 387-421, 1989.

[Dye84] Dyer, M.E., *"Linear time algorithms for two- and three-variable linear programs"*, SIAM J. Computing, vol. 13, pp. 31-45, 1984.

[EdS91] Edelsbrunner, H., and W. Shi, *"An $O(n \log^2 h)$ time algorithm for the three-dimensional convex hull problem"*, SIAM J. Computing, vol. 20, no. 2, 259-269, 1991.

[KiS86] Kirkpatrick, D.G., and R. Seidel, *"The ultimate planar convex hull algorithm?"*, SIAM J. Computing, vol. 15, no. 1, pp. 287-299, 1986.

[Meg83] Megiddo, N., *"Linear-time algorithms for linear programming in R^3 and related problems"*, SIAM J. Comp. 12, pp. 759-776, 1983.

[Meg84] Megiddo, N., *"Linear programming in linear time when the dimension is fixed"*, J. Association for Computing Machinery (JACM) 31, pp. 114-127, 1984.

[Meg89] Megiddo, N., *"On the ball spanned by balls"*, Discrete & Computational Geometry, vol. 4, no. 6, pp. 605-610, 1989.