CONVEX HULLS

Chan's optimal output sensitive algorithms for convex hulls

Alberto Parravicini
Université libre de Bruxelles

December 15, 2016

What's on the menu?

→ Basic notions

→ Algorithms for convex hulls

→ Optimal 2D algorithm

→ Optimal 3D algorithm

Crash course on convex hulls

A *convex set* S is a set in which, $\forall\, x, y \in S$, the segment $xy \subseteq S$.

Given a set of points P in d dimensions, the **Convex Hull CH(P)** of P is:

$\rightarrow$ the *minimal convex set* containing P.

$\rightarrow$ the union of *all convex combinations* of points in P, i.e. the points $CH(P)$ are s.t.

$$\sum_{i=1}^{|P|} w_i \cdot x_i,\ \forall x_i \in P,\ \forall w_i :\ w_i \geq 0 \text{ and } \sum_{i=1}^{|P|} w_i = 1$$

Output sensitive algorithm

The complexity of an *output-sensitive* algorithm is a function of both the *input size* and the *output size*.

Jarvis's march

- **Core idea:** given an edge $\boldsymbol{pq}$ of the convex hull, the next edge $\boldsymbol{qr}$ to be added will maximize the angle $\angle \boldsymbol{pqr}$

- At each step we scan all the $\boldsymbol{n}$ points.
- How many steps? $\boldsymbol{h}$, size of the hull.
- **Complexity:** $\boldsymbol{O(nh)}$

One step, visually

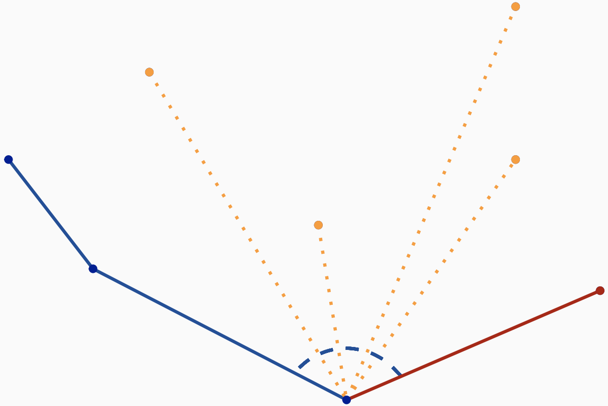


Figure: *A step of the Jarvis March. The red line is the next segment that will be added to the hull.*

Jarvis's march

Algorithm 1: Jarvis March

Input: a list S of bidimensional points.
Output: the convex hull of the set, sorted counterclockwise.

hull =[]
$x_0 =$ the leftmost point.
hull.push(x_0)
Loop *hull.last() != hull.first()*
    candidate = S.first()
    foreach *p in S* **do**
        if *p != hull.last() and p is on the right of the segment "hull.last(), candidate"* **then**
            candidate = p
    if *candidate != hull.first* **then** hull.push(candidate) **else** break
return hull

Graham's scan

- Sort the points in clockwise order around the leftmost point - **Cost:** $O(n \log n)$
- Keep the current hull in a stack $H = \{h_0, \ldots, h_{curr}\}$.
- Inspect each point p in order - **Cost:** $O(n)$
  - While $h_{curr-1}h_{curr}p$ is a right turn, pop h_{curr}.
  - Push p to H.
- **Overall complexity:** $O(n \log n)$

Graham's scan

Algorithm 2: Graham Scan

Input: a list S of bidimensional points.
Output: the convex hull of the set, sorted counterclockwise.
hull =[]
$x_0 =$ the leftmost point.
Put x_0 as the first element of S.
Sort the remaining points in counter-clockwise order, with respect to x_0.
Add the first 3 points in S to the hull.
forall *the remaining points in S* **do**
  while *hull.second_to_last(), hull.last(), p form a right turn* **do**
    hull.pop()
  hull.push(p)
return hull

Can we do better?

So far:

→ **Jarvis's march:** $O(nh)$
→ **Graham's scan:** $O(n \log n)$

→ How do we compare their complexity?

→ We would like to do even better!

Can we get to $O(n \log h)$?

Chan's 2D algorithm

- **Idea:** Some points will never be in the hull! Discard them to speed up *Jarvis's march*.
- Combine *Jarvis's march* and *Graham's scan*.

- Algorithm (**Chan 2D**):
  - Split the n points in groups of size m ($\lceil n/m \rceil$ groups).
  - Compute the hull H_i of each group
    Cost: $\boldsymbol{O((n/m) \cdot (m \log m)) = O(n \log m)}$
  - Until the hull is complete, repeat:
    - Given the current hull $H_{tot} = \{h_0, \ldots, h_{curr}\}$, find for each H_i the point right tangent to h_{curr} (binary search, $\boldsymbol{O(\log m) \lceil n/m \rceil}$).
    - Pick the tangent point p that maximizes $\angle h_{curr-1} h_{curr} p$.

One step, visually

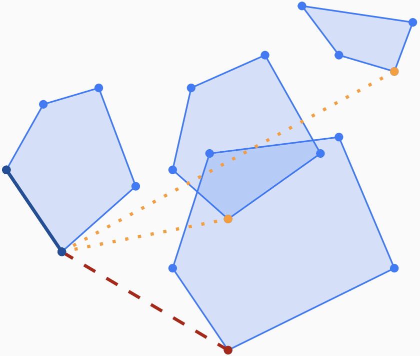


Figure: *A step of Chan's algorithm. In blue, the existing hull, in orange, the tangents, in red, the new edge that will be added.*

Chan's 2D algorithm, part 2

Complexity: $\boldsymbol{O(n \log m + (h(n/m) \log m))}$

What about m? Let's pretend the size h of the final hull is known a-priori.

If we put $m = h$, we get complexity $O(n \log h + (h(n/h) \log h)) = \boldsymbol{O(n \log h)}$

But h is not known!

- → Reiterate the algorithm multiple times, with $m = 2^{2^i}$
- → Iteration cost: $\boldsymbol{O(n \log H) = O(n 2^i)}$
- → $O\left(\sum_{i=1}^{\lceil \log \log h \rceil} n 2^i\right) = O(n 2^{\lceil \log \log h \rceil + 1}) = \boldsymbol{O(n \log h)}$

Chan's 2D pseudo-code

Algorithm 3: ChanHullStep, a step of Chan's algorithm

Input: a list S of bidimensional points, the parameters m, H
Output: the convex hull of the set, sorted counterclockwise, or an empty list, if H is $< h$

Partition S into subsets $S_1, \ldots, S_{\lceil n/m \rceil}$.
for $i = 1, \ldots, \lceil n/m \rceil$ **do**
    Compute the convex hull of S_i by using Graham Scan, store the output in a counter-clockwise sorted list.
$p_0 = (0, -\infty)$
$p_1 =$ the leftmost point of S.
for $k = 1, \ldots, H$ **do**
    for $i = 1, \ldots, \lceil n/m \rceil$ **do**
        Compute the points $q_i \in S$ that maximizes $\angle p_{k-1} p_k q_i$, with $q_i \neq p_k$, by performing binary search on the vertices of the partial hull S_i.
    $p_{k+1} =$ the point $q \in \{q_1, \ldots, q_{\lceil n/m \rceil}\}$.
    if $p_{k+1} = p_t$ **then** return $\{p_1, \ldots, p_k\}$
return *incomplete*

Chan's 2D pseudo-code, reprise

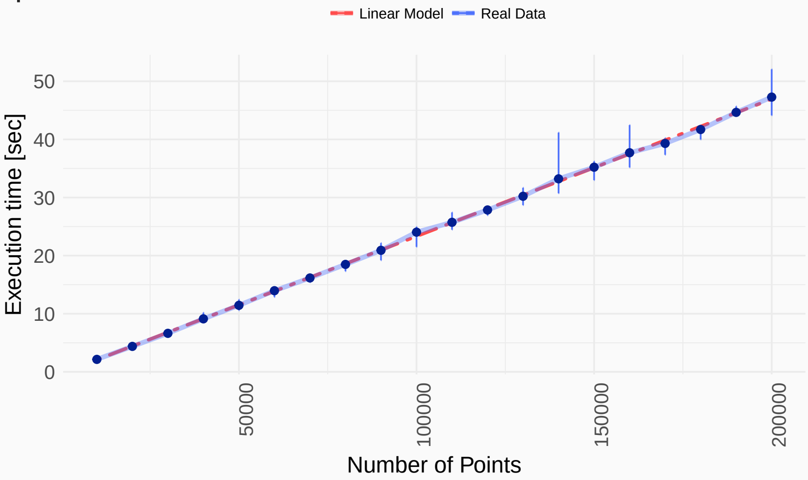
Algorithm 4: Chan's algorithm

Input: a list S of bidimensional points
Output: the convex hull of the set
for $i = 1, 2, \ldots$ **do**
    L = ChanHullStep(S, m, H), where $m = H = min\{|S|, 2^{2^i}\}$
    if $L \neq incomplete$ **then** return L

Chan's 2D - Empirical analysis

Is a real implementation ***O*(*n log h*)**?

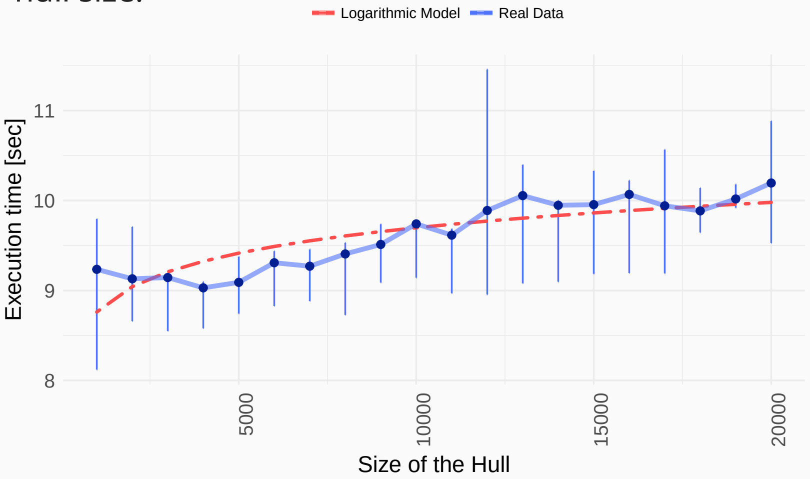
Test 1: fixed hull size (1000), increasing number of points.



Chan's 2D - Empirical analysis

Is a real implementation ***O*(*n log h*)**?

Test 2: fixed number of points (40000), increasing hull size.



THANK YOU!