Dynamic Planar Convex Hull

Ph.D. Defense Riko Jacob

31st May 2002

BRICS, University of Aarhus
Outline of the talk

- Planar convex hull
- Duality: Lower Envelope
- Application: $k$-level
- Overall structure of the data structure
- Some key ingredients
- Lower bounds
Planar Convex Hull

Input
A set of points $S \subseteq \mathbb{R}^2$

Output
The points on the convex hull $\text{CH}(S)$ in clockwise order

$n = |S| \quad h = |\text{CH}(S)|$

Known results

Optimal $O(n \log n)$
Graham 1972; ...

Output-sensitive $O(n \log h)$
Kirkpatrick, Seidel 1986; Chan 1996
Graham’s Scan

Andrew’s variant for upper hull
Dynamic Planar Convex Hull

Updates
Insert and delete points

Queries
(a) The extreme point in a direction
(b) Does a line intersect $\text{CH}(S)$?
(c) Is a point inside $\text{CH}(S)$?
(d) Neighbor points on $\text{CH}(S)$
(e) Tangent points on $\text{CH}(S)$
(f) The edges of $\text{CH}(S)$ intersected by a line
## Dynamic Planar Convex Hull Results

<table>
<thead>
<tr>
<th>Insertions only</th>
<th>Update</th>
<th>Query</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preparata 1979</td>
<td>$O(\log n)$</td>
<td>$O(\log n)$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Deletions only</th>
<th>Update</th>
<th>Query</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hershberger, Suri 1992</td>
<td>$O_A(\log n)$</td>
<td>$O(\log n)$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Offline</th>
<th>Update</th>
<th>Query</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hershberger, Suri 1996</td>
<td>$O_A(\log n)$</td>
<td>$O(\log n)$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Fully dynamic</th>
<th>Update</th>
<th>Query</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overmars, van Leeuwen 1981</td>
<td>$O(\log^2 n)$</td>
<td>$O(\log n)$</td>
</tr>
<tr>
<td>Chan 1999</td>
<td>$O_A(\log^{1+\epsilon} n)$</td>
<td>$O(\log n)$</td>
</tr>
<tr>
<td>Brodal, Jacob 2000</td>
<td>$O_A(\log n \cdot \log \log n)$</td>
<td>$O(\log n)$</td>
</tr>
<tr>
<td>Kaplan, Tarjan, Tsioutsouliklis ’01</td>
<td>$O_A(\log n \cdot \log \log n)$</td>
<td>$O(\log n)$</td>
</tr>
</tbody>
</table>

$O_A$=Amortized Query=Queries (a)–(e)
### Dynamic Planar Convex Hull Results

<table>
<thead>
<tr>
<th>Insertions only</th>
<th></th>
<th>Update</th>
<th>Query</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preparata 1979</td>
<td></td>
<td>$O(\log n)$</td>
<td>$O(\log n)$</td>
</tr>
</tbody>
</table>

**Deletions only**

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>Update</th>
<th>Query</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hershberger, Suri 1992</td>
<td></td>
<td>$O_A(\log n)$</td>
<td>$O(\log n)$</td>
</tr>
</tbody>
</table>

**Offline**

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>Update</th>
<th>Query</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hershberger, Suri 1996</td>
<td></td>
<td>$O_A(\log n)$</td>
<td>$O(\log n)$</td>
</tr>
</tbody>
</table>

**Fully dynamic**

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>Update</th>
<th>Query</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overmars, van Leeuwen 1981</td>
<td></td>
<td>$O(\log^2 n)$</td>
<td>$O(\log n)$</td>
</tr>
<tr>
<td>Chan 1999</td>
<td></td>
<td>$O_A(\log^{1+\epsilon} n)$</td>
<td>$O(\log n)$</td>
</tr>
<tr>
<td>Brodal, Jacob 2000</td>
<td></td>
<td>$O_A(\log n \cdot \log \log n)$</td>
<td>$O(\log n)$</td>
</tr>
<tr>
<td>Kaplan, Tarjan, Tsioutsoulikis ’01</td>
<td></td>
<td>$O_A(\log n)$</td>
<td>$O(\log n)$</td>
</tr>
<tr>
<td>this Thesis</td>
<td></td>
<td>$O_A(\log n)$</td>
<td>$O(\log n)$</td>
</tr>
</tbody>
</table>

$O_A = $ Amortized Query = Queries (a)–(e)
Duality Transformation

\[ p = (a, b) \in \mathbb{R}^2 \quad \text{maps to} \quad p^* := (a \cdot x - b = y) \]
Dynamic Lower Envelope

**Updates**
Insert and delete lines

**Queries**
(a) Vertical line intersection
(b) Is a point above $LE(S)$
(c) Is a line above $LE(S)$
(d) Next segments on $LE(S)$
(e) The segments of $LE(S)$ intersected by a line
(f) The extreme point of $LE(S)$ in some direction
Application: $k$-level in the plane

Sweep-line algorithm Edelsbrunner and Welzl 1986
Previously $O((n + m)\alpha(n) \log n)$ expected time Har-Peled 1998
Now $O((n + m) \log n)$ for $m$ segments on the $k$-level

The 3-level of the 6 lines is depicted in thick red.
Overall Structure

Updates

Logarithmic Meth.

Geom. Merging

$\log^2 n$ explicit hulls, deletion-only

Queries

Interval Tree

Secondary Struct.

(Bootstrapping)

$2 \times$ Bootstrapping:
Insert and Query $\log n$
Delete: $n \rightarrow \log^5 n \rightarrow \log n$
Logarithmic Method

Bentley and Saxe 1980

Bentley and Saxe 1980
Static Geometric Merging

Difficult: maintain under deletions of points
Combining Queries: Interval Tree

Task: combine the search on several lower envelopes into one search.
Follows ideas from Chan 1999; different choice of parameters, save some work by relaxed placement and lazy movements:
exploit knowledge about the (dynamic of the) intervals
Semidynamic Merging

Create Set($p$) Create singleton set

Merge($A, B$) Combine data structures for $A$ and $B$ into one new for $A \cup B$

Delete($r$) Delete $r$ from all merging structures

- Maintains list of points on the upper hull
- Works on binary merging forest
- Performance: $O_A(1)$ per element in the set
Core Problem

Maintain the equality points of the merging

An equality oracle allows $O_A(1)$ per element hull maintenance
Separation Certificate

vertical certificates (sweep line): $O(n)$ per deletion
parallel tangent search: $O(\log n)$ per deletion
suspended search: in some variant $O(1)$ per deletion
Greedy Separation

Greedily choosing tangent lines

Seems too rigid and sensitive for changes
Truss Bridge

We call the construction Truss
Shortcuts: Reducing the complexity of the outer hull
Dangling search

One deletion affects only constantly many strong rays
Splitters

Data structure(s) keeping (family of) sorted sequences

Elements $e_i$ from a totally ordered universe

- **Build**$(e_1, \ldots, e_n)$ $O_A(n)$
- **Split**$(t)$ $O_A(1)$
- **Extend**$(e_{n+1})$ $O_A(1)$

Hoffmann, Mehlhorn, Rosenstiehl, Tarjan 1986

Split includes searching;

Dangling searches are suspended searches;
promise to split when finishing the search.
Splitters

Data structure(s) keeping (family of) sorted sequences

Elements $e_i$ from a totally ordered universe

$$\text{Build}(e_1, \ldots, e_n) \quad O_A(n)$$
$$\text{Split}(t) \quad \text{suspended} \quad O_A(1)$$
$$\text{Extend}(e_{n+1}) \quad O_A(1)$$

Hoffmann, Mehlhorn, Rosenstiehl, Tarjan 1986

Split includes searching;

Dangling searches are suspended searches;
promise to split when finishing the search.
The geometric situation of losing equality points. We join the splitter over a dangling search: Feasible because we promise to split.
Account: Life-cycle of a point $p \in A$

1. $p$ becomes part of $UH(A)$. $p$ in replacement splitter.
2. we realize $p \in UC_0(B)$. $p$ in lasting splitter.
3. we decide to select $p$.
4. Delete on $B$: $p \notin UC(B)$. $p$ in surfacing splitter.
5. $p$ is hidden by a shortcut.
6. $p$ is hidden by a bridge.
7. $p$ gets on $UH(C)$.
8. The point $p$ gets deleted.

Schematic: the two hulls after one deletion
New Techniques

**Splitter:** suspended (dangling) searches, restricted join over search

**Dynamization:** Reuse of existing data structures

**Geometric Merging:** Focus on equality points, selected points, dangling search, over-approximation, shortcuts, truss

**Interval Tree:** Relaxed placement of intervals, lazy movement, location justifier

**Linear Space:** Separators
Tight Lower Bounds

\[ q(n) \] be (amortized) query time
\[ I(n) \] amortized insertion time

\[ q(n) = \Omega(\log n) \quad \text{and} \quad I(n) = \Omega \left( \log \frac{n}{q(n)} \right). \]

on algebraic real-RAM, off-line usage of data structure, reduction based.

Applies to Membership and Predecessor as well.
Decision Problem

For $k < n$

$$(x_1, \ldots, x_n, y_1, \ldots, y_k) \in \text{DISJOINTSET}_{n,k} \subset \mathbb{R}^{n+k}$$

$\iff$

for all $i, j$ we have $x_i \neq y_j$

\text{DISJOINTSET}_{n,k} has $\Omega(k^n)$ connected components

An algebraic computation tree has height $\Omega(n \log k)$.

(Ben-Or 83)
Summary and Open Problems

Presented

Dynamic Planar Convex Hull Data structure

- Query $O(\log n)$
- Insert $O_A(\log n)$
- Delete $O_A(\log n)$

and a matching lower bound.

Open Problems

- Make it simple
- worst-case instead of amortized bounds
- more general queries
- explicit maintenance of the hull