More Output-Sensitive Geometric Algorithms

Ken Clarkson
AT&T Bell Labs
Murray Hill, NJ
http://netlib.att.com/netlib/att/cs/home/clarkson.ht

Finding Extrema

Given a set S of n sites,
find extremal $F \subset S$, $|F| = A < n$.

- vertices of $\mathbf{conv}S$;
- coordinate-wise minima of S;
- minima in a partial order of S;

An $O(nA)$ algorithm

Examine each site p in turn,
adding it to $E \subset F$ or throwing it out.

- if E proves $p \notin F$, throw out p;
- otherwise,
  - use p to find $q \in F \setminus E$;
  - add q to F;

First step: $O(|E|) = O(A)$ per point of S; Second step: $O(n)$ per point of F;

$O(nA)$ time overall.

Coordinatewise Minima

To prove $p \notin F$ using E,
find $z \in E$ dominating p.

To find $q \in F \setminus E$ using p:
look through $S \setminus E$,
maintaining t, initially p:
if a new point q is dominated by t, throw out q;
if q dominates t,
throw out t and replace it by q.

In the plane,
for sites coordinate-wise independent,

$$n + O(n^{5/8})$$

comparisons on average.

First member of E kills almost all sites.

Apparently nontrivial analysis.

Comparable to [BCL] on average,
plus worst-case bound.

Why the analysis is nontrivial

[[graphics could not be recovered]]

Four traces of t
that gives first point in E,
2^{20} points uniform in $[2^{20}, 2^{20}]$.

First intuition: arrive at $(2^{20}/\sqrt{n}, 2^20/\sqrt{n}) = (2^{10}, 2^{10})$.

Extreme Points

To check if p might be extremal: solve an LP to find *witness vector* v with

$$v \cdot p > \max_{x \in E} v \cdot x$$

if no such v, $p \in \mathbf{conv} E \subset \mathbf{conv} F$.

Use p to find $q \in F \setminus E$: q gives $\max_{x \in S} v \cdot x$. q cannot be in E, and must be in F.

$O(nA)$ time is apparently new.

Polynomial in dimension d,
up to linear programming.

Slightly sharper bounds are attainable,
replacing n^2 in bounds by nA.

Other Results

- Space-efficient convex hulls
  - apply idea inductively in d;
  - use lexicography for tie-breaking;
  - $O(n)$/edge of face graph.
- $O(n \log A)$ convex hulls when expected hull size is $O(r)$ for $R \in_R \binom{S}{r}$.
- Thinning for nearest neighbor classification.

$O(n \log A)$ convex hulls

Maintain hull of random subset R,
and visibilities of points in $S \setminus R$
(the *conflict graph*)

Also: maintain $\mathbf{conv}E$, $E \subset F$,
with $|E| \approx r/\log r$.

Use:

- $\mathbf{conv}R$ to update quickly,

- $\mathbf{conv}E$ to quit early.

Assume $A \ll n$.

Maintain E using previous algorithm,
but find witness vector
after building DK decomposition for $\mathbf{conv}E$,
so that $O(\log|E|)$ time/point needed.

Find $q \in F \setminus E$ by using
a few of the conflict lists for $\mathbf{conv}R$.

NN Thinning

Nearest Neighbor classification:
sites have colors;
given a query point,
color it by nearest site.

Thinning:
Delete all possible sites allowing color to be preserved.

A site is *redundant* and can be thinned, if
all its Delaunay neighbors are the same color.

Generating all Delaunay neighbors requires $O(n^3)$ time, as above.

Ideas here give an algorithm needing

$$O(n \log n) \sum_i A_i n_i$$

time, n_i sites of color i, A_i irredundant.

As before, maintain
E_i, a set of irredundant sites of color i.

For each p,
either E_i proves p redundant, or
there is site w with color not i,
and point x,
with x equidistant from p and w,
and closer to them than to any site in E_i.

A walk along $p \to x \to w$ gives a site $q \in F_i \setminus E_i$.

Concluding Remarks

- Algorithms not exponential in d, except maybe LP;
- A simpler $O(n \log A)$ algorithm should be possible.