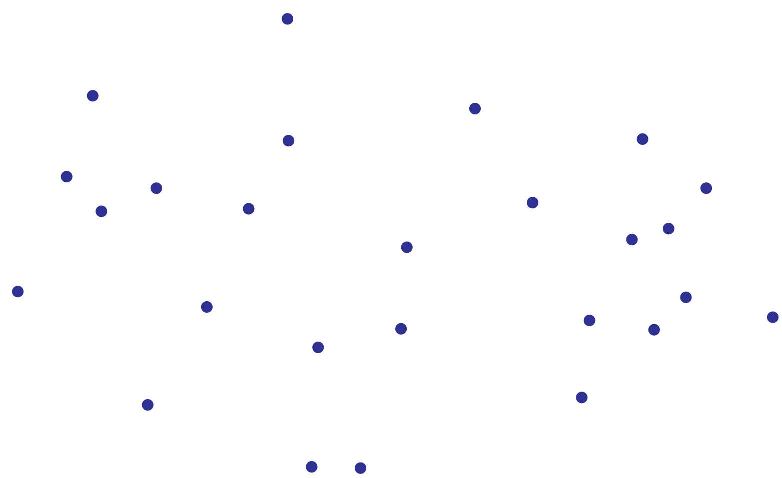
Faster Construction of Planar 2-Centers

David Eppstein

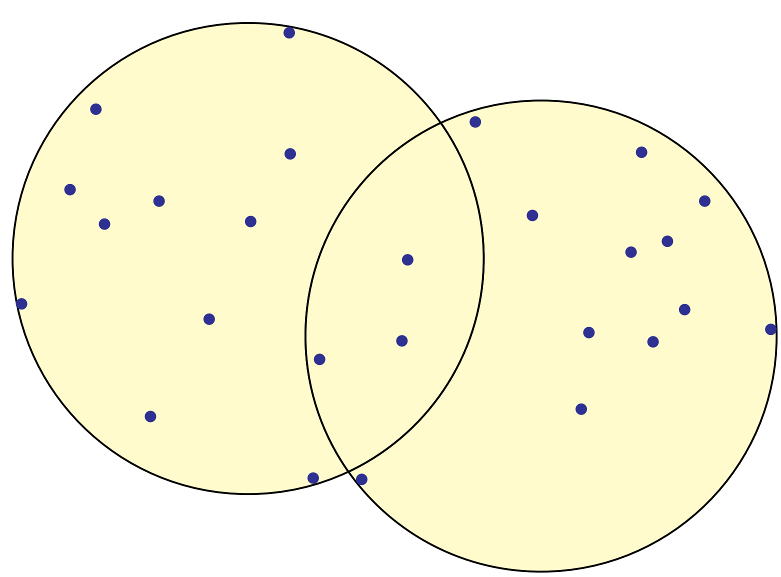
Dept. Information and Computer Science
Univ. of California, Irvine

http://www.ics.uci.edu/~eppstein/

The problem

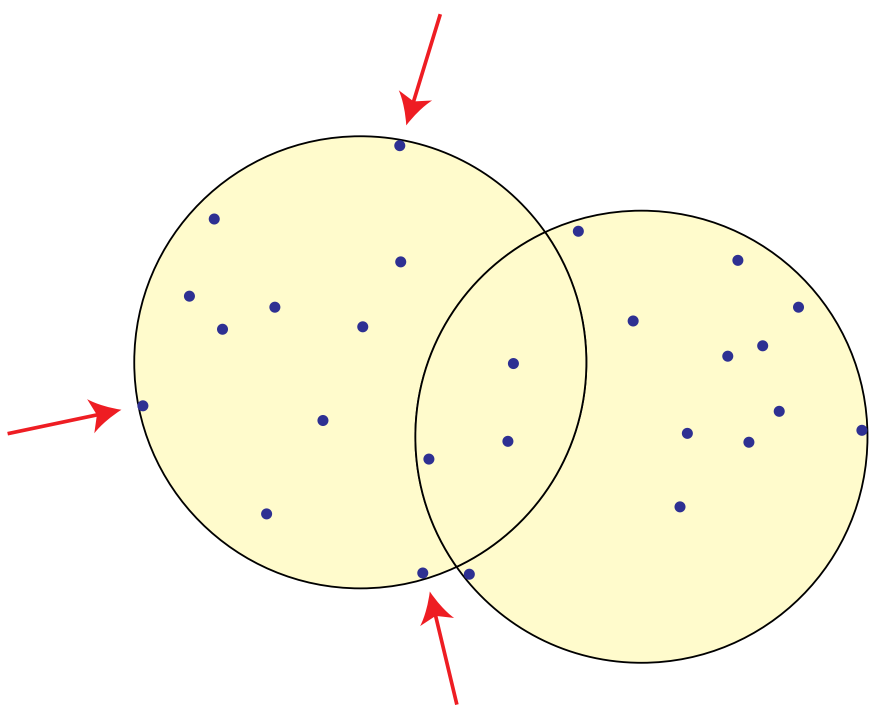


Cover *n* points w/two minimum-radius circles



It’s safe to assume:

- Both circles are the same size
- One circle has three tangent points, or is diameter circle of two points
- (Other circle is less constrained)



History

Agarwal and Sharir, SODA 1991:

$O(n^2 \log^3 n)$

Eppstein, FOCS 1991:

$O(n^2 \log^2 n \log\log n)$ randomized

Katz and Sharir, SCG 1993:

$O(n^2 \log^3 n)$

Jaromczyk and Kowaluk, SCG 1994:

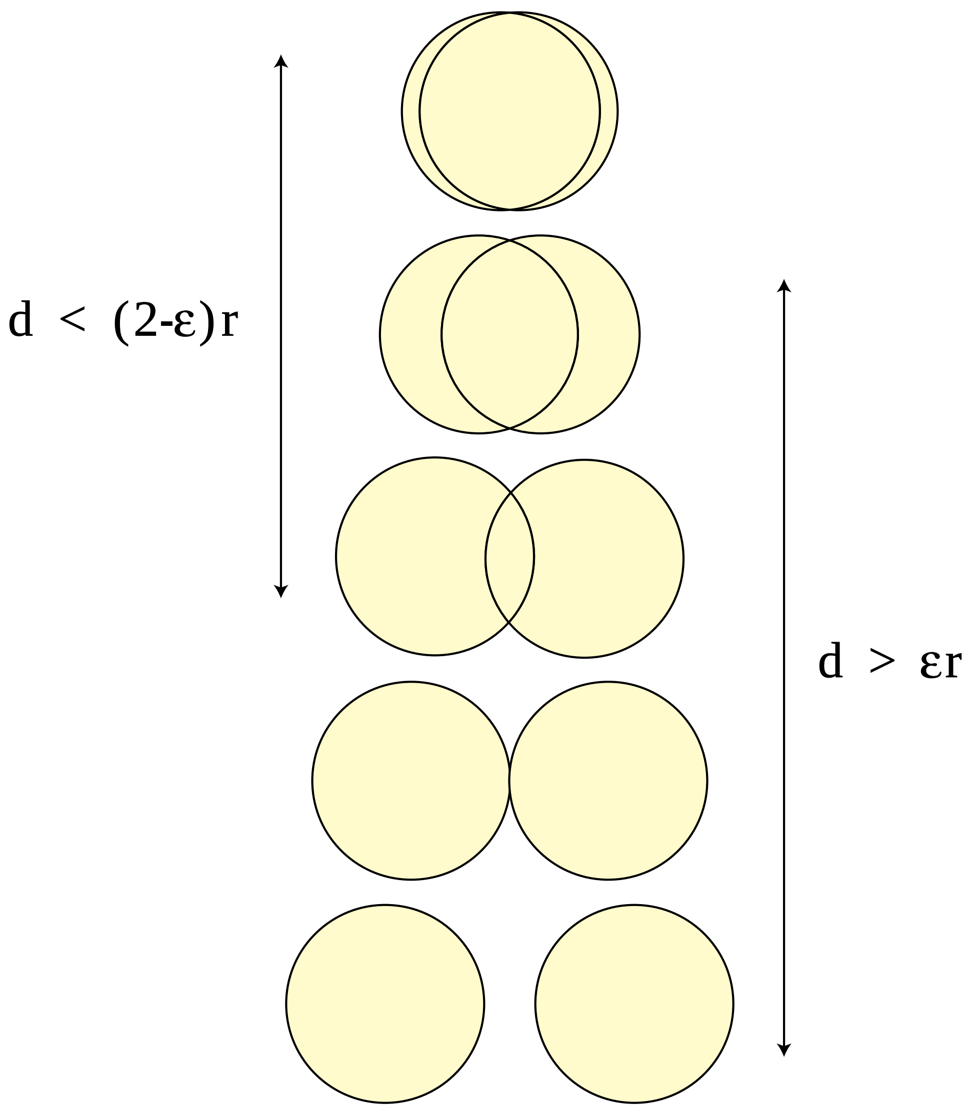
$O(n^2 \log n)$

Sharir, SCG 1996:

$O(n \log^9 n)$

New result: $O(n \log^2 n)$ randomized

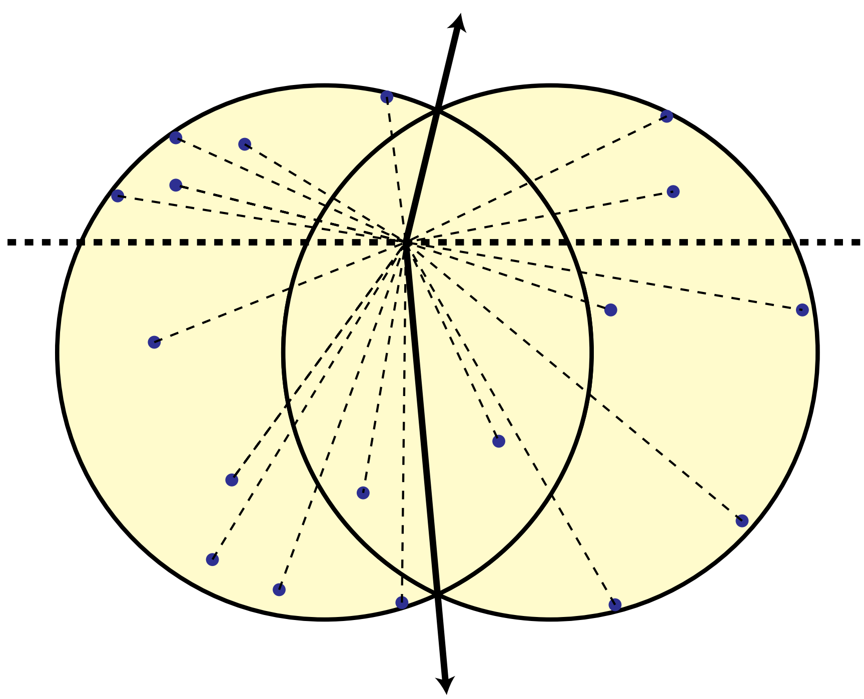
Two Cases (based on circle separation)



Overlapping Case → Matrix

Find point in intersection of disks
(by testing $O(1)$ candidates)

Look for partition by two rays



Form matrix representing possible partitions:

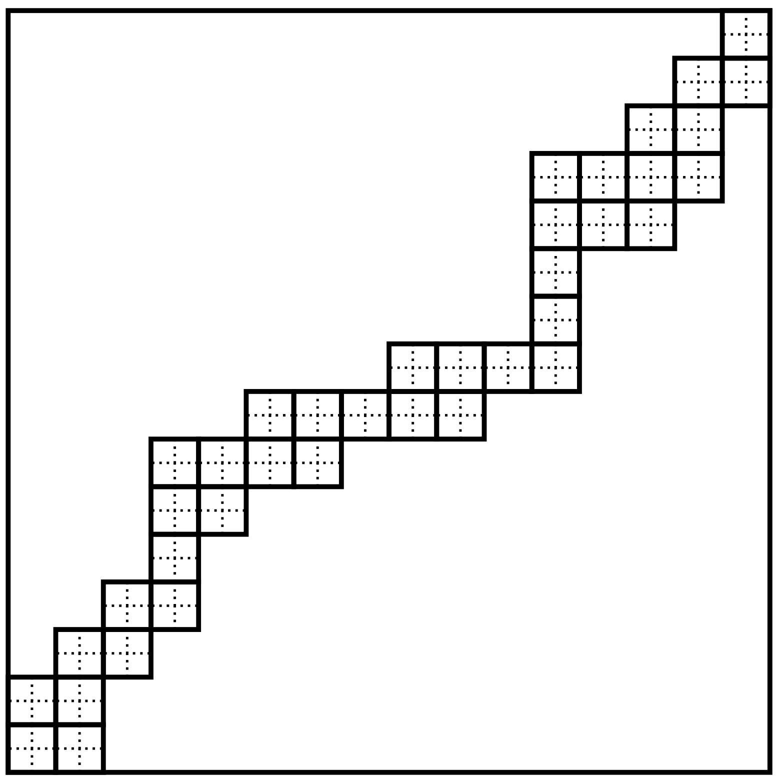
Row index = position of upper ray

Column index = position of lower ray

Overlapping Case: Quadtree Search

(based on matrix selection of Frederickson and Johnson)

Represent potential partitions as set of $\frac{n}{k} \times \frac{n}{k}$ squares (initially, one square for whole matrix)



For $O(\log n)$ stages:

Subdivide each square into four

Prune back down to $O(k)$ squares

Overlapping Case: Pruning

Too many squares $\rightarrow$ many interior corners

Pick a corner x randomly
and evaluate corresponding circumradii

Compare x against all other interior corners

If x better than y, eliminate one of y's squares
(above and to right of y, left circle only gets larger;
below and to left, right circle only gets larger.)

Expect 50% of interior corners to become exterior

Repeat $O(1)$ times until few interior corners left

Overlapping Case: Analysis

$O(\log n)$ stages.

Only slow part: compare corners to random choice ($=$ test circumradii from corresponding partitions)

Connect into path, use Hershberger-Suri offline circumradius decision algorithm:

$O(n \log n)$ per stage

Use exact circumradius data structure:

$O(k \log^c n)$ per stage

Combine both methods:

Exact circumradius when $k = O(n/log^c n)$

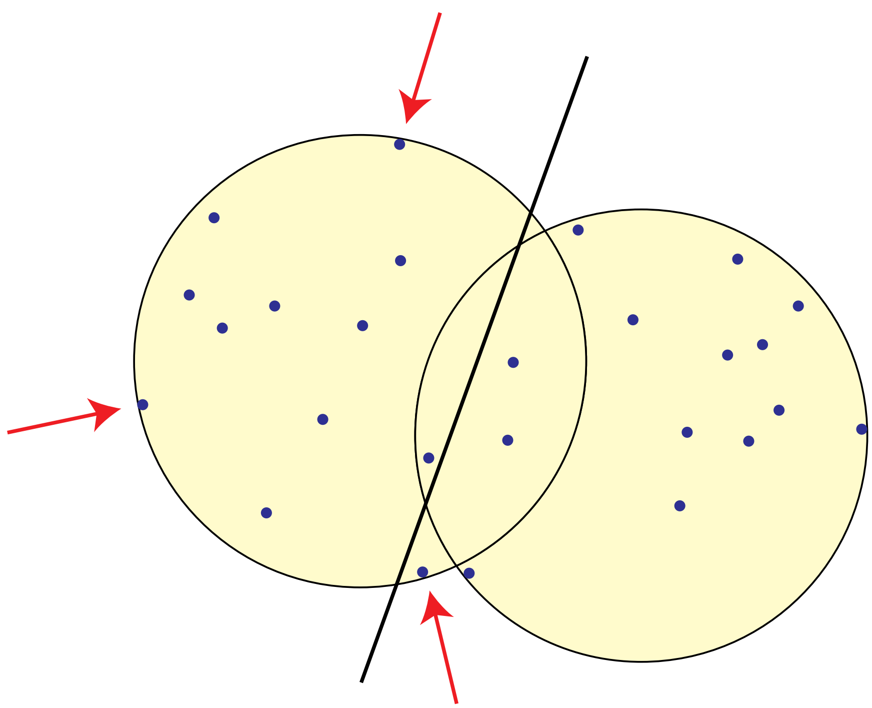
Offline alg for remaining $O(\log \log n)$ stages

Total: $O(n \log n \log \log n)$

Separated Case: Cut Line

Find halfspace containing only points of constrained disk, including at least one tangent point

(by testing $O(1)$ candidates)



Separated Case: Main Idea

Parametric search

Let A_1 be decision algorithm
(compare given radius against optimum)

Let A_2 be any algorithm that is discontinuous at the optimal value (e.g. the decision algorithm again)

Simulate $A_2(r^*)$ by replacing each comparison in A_2 with a call to A_1.

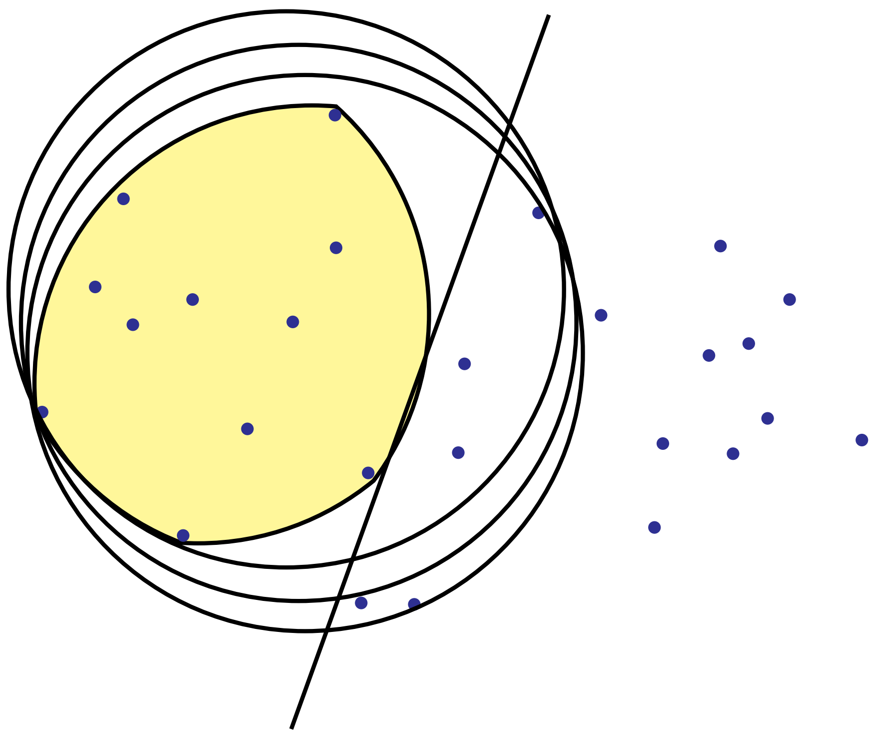
Because of discontinuity, calls must include $A_1(r^*)$

Separated Case: Efficiency Considerations

To make parametric search efficient:

- Simulate a parallel algorithm
- Batch calls to decision alg using binary search
- Remove as much as possible from simulation
  - preproc. not depending on parameter
  - postproc. after already discontinuous

Separated Case: Decision Algorithm



Swing circle around circular hull of pts in half-plane, testing circumradius of remaining points

Total: $O(n \log n)$

Separated Case: Simulated Algorithm

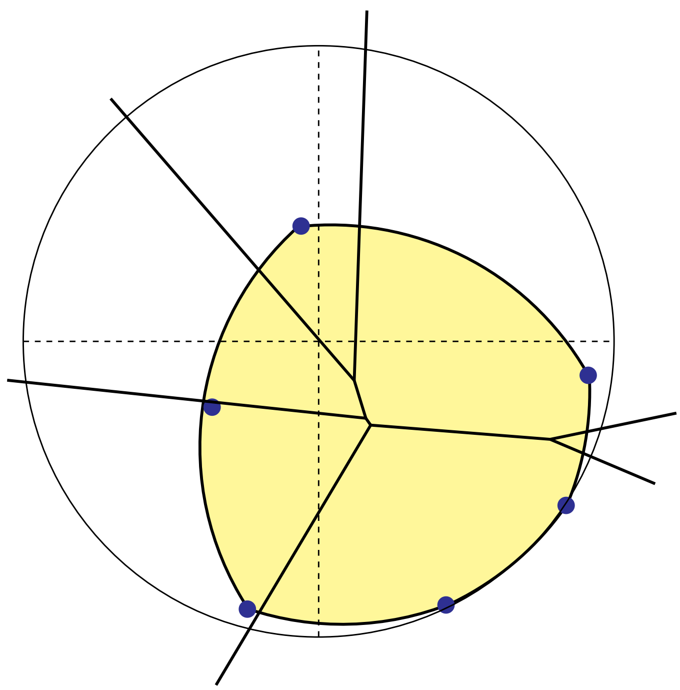
- Compute circular hull of points
- Sweep circle around hull
- Find sequence of point sets swept by circle

Already discontinuous – don't have to apply offline decision algorithm to sequence

(Proof: 4 cases. Is optimal circle supported by two or three tangent points, and are one or two of them on circular hull?)

Separated Case: Fast Circular Hull

Circular hull arcs correspond to certain edges of the farthest point Voronoi diagram



Compute Voronoi diagram (preproc.)

Test which Voronoi edges give hull arcs
    ($O(n)$ processors, $O(1)$ time)

Connect the dots (independent of param.)

Separated Case: Swinging the Circle

For each point p not in the hull
find hull vertex v the circle is pivoting on
when it crosses p (binary search)

For each hull pivot v
sort the associated points by sweep time

$O(n)$ processors, $O(\log n)$ time
but suitable for Cole's speed-up

Separated Case: Analysis

Preprocessing:

Voronoi diagram $O(n \log n)$

Simulate:

Finding hull arcs

$O(n)$ binary searches

One sorting algorithm

Total:

$O(n \log^2 n)$

Open Problems

Derandomize

(only uses $O(\log n \log\log n)$ random bits!)

Improve time bound

(only slow case: nearly tangent circles)

Make simple enough to be practical

(most complicated part: parametric sort)