Graphs for Dynamic Computational Geometry

David Eppstein

Dept. of Information & Computer Science

University of California, Irvine

Some questions:

Many standard dynamic graph problems

(connectivity, shortest paths, planarity testing...)

But only one standard geometric graph problem

(Euclidean MST)

Where are the other interesting problems?

Euclidean MST defined on a complete graph
and is a subset of the Delaunay Triangulation

But complete graph $\Rightarrow$ uninteresting algorithms
and DT can be complete for dimension ≥ 3

What other geometric graph structures can we use?

Outline

Euclidean MST
- complete graph
- Delaunay triangulation
- Yao graph
- bichromatic closest pair graph

Dynamic width
- rotating caliper graph

Minimum diameter subset
- lune graph

Circle packing for quad meshing
- circle intersection graph

Robot motion planning
- quadtree dual graph

Kinetic connectivity of rectangles
- rectangle boundary graph

First, what is "Dynamic"?

- Dynamic graph: edge insert/delete
  but in most geometric problems, edges are part of solution technique not part of problem

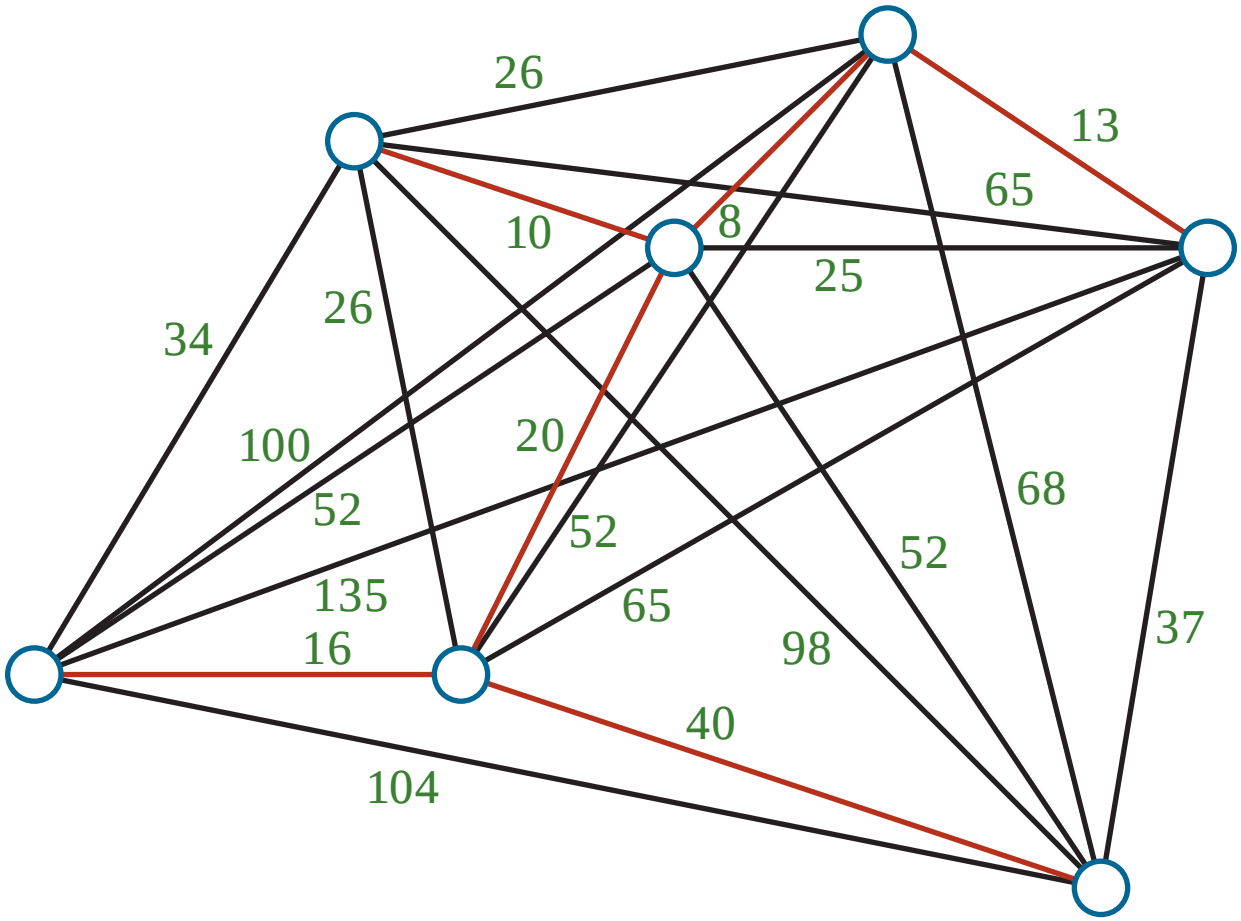
- Dynamic geometry: point insert/delete
  (discrete changes of point set
  similar to graph vertex insert/delete)

- Parametric geometry:
  points with static "flight plans"
  describing continuous change of point set

- Kinetic geometry: flight plan insert/delete

Euclidean MST:

Minimum spanning tree of
complete Euclidean graph



(shown with squared edge lengths)

Dynamic MST in complete graph

Insert/delete vertex

$\Rightarrow \Theta(n)$ edges

$\Rightarrow \mathcal{O}(n \log^4 n)$ time per update

(in any dimension)

To do better:

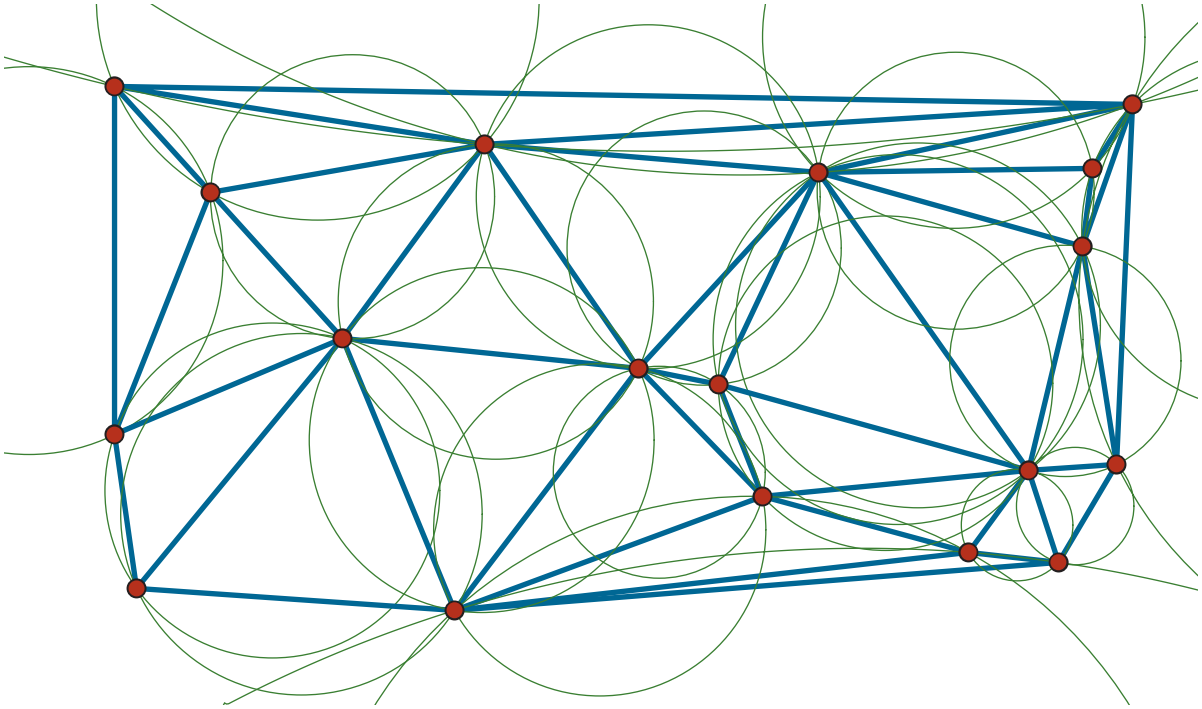
need a sparse graph

that doesn't change much

and can be updated quickly

Delaunay Triangulation

Form edges for each chord of an empty circle



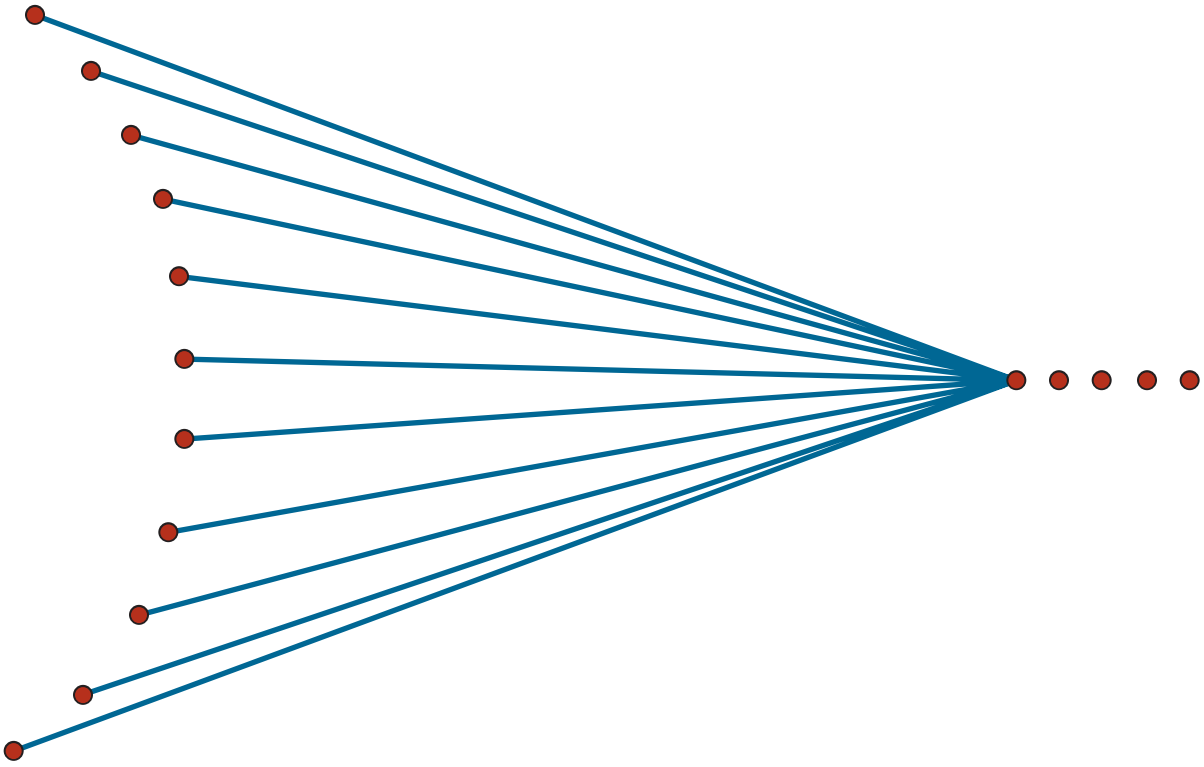
Contains Euclidean MST

For $d = 2$, planar $\Rightarrow \mathcal{O}(n)$ edges

For $d \geq 3$ can be a complete graph
e.g., points (t, t^2, t^3)

The Problem with Delaunay

One insertion can change $\ominus(n)$ edges



So unusable for dynamic algorithms?

Average Case Delaunay

If all insertion orders equally likely
then expected change $= \mathcal{O}(1)$

Proof:

$E[\#\text{ edge insertions}] =$ avg degree
$\#\text{ deletions} = \#\text{ insertions} + \Delta(\#\text{ edges})$

The Δ terms telescope

Similarly, if all points equally likely to be deleted,
expected change $= \mathcal{O}(1)$

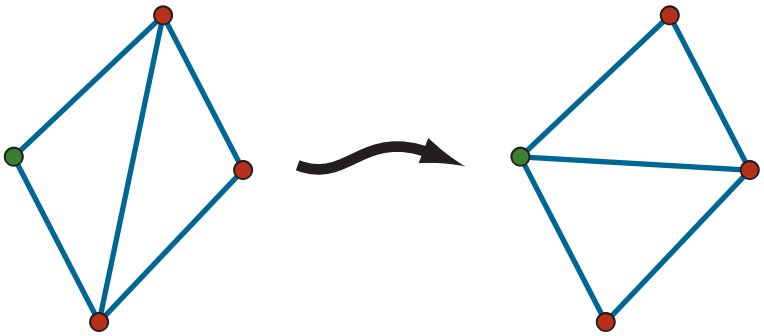
Average Case Delaunay continued:

How to insert a new point

Find triangle containing new point

(dynamic planar point location)

Flip adjacent diagonals
until locally Delaunay



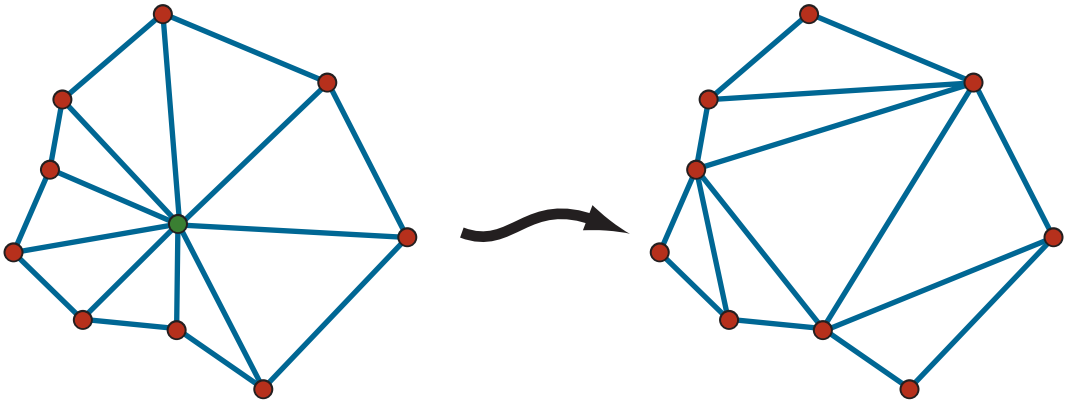
Time $= \mathcal{O}(\log n + \#\text{ changes})$

Average Case Delaunay continued:

How to delete a point

Remove deleted point

Retriangulate resulting hole



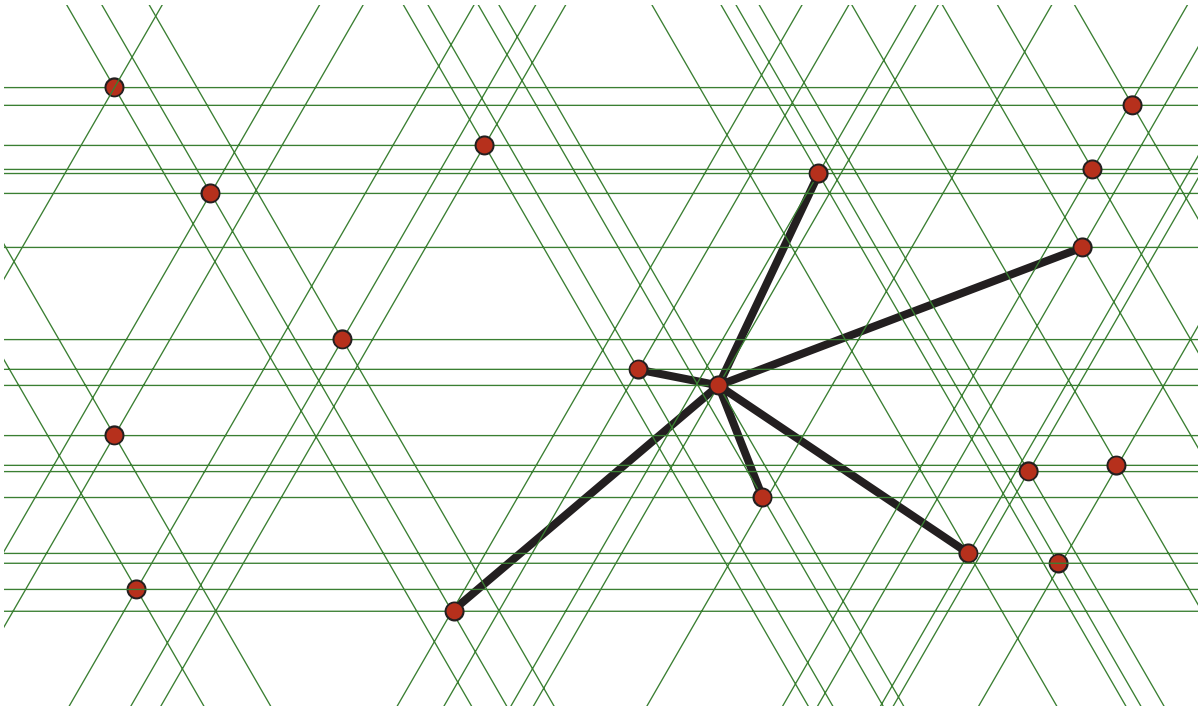
Time $= \mathcal{O}(\#\text{ changes})$

Incremental/Offline MST: Yao graph

Lemma: If $(u, v) \in$ MST, then v must be nearest neighbor in any 60° wedge from u

Proof: Nearer neighbor w would also be nearer to v, allowing shortcut u-w-v

Yao graph: connect vertex to neighbors from each of six 60° wedges



Sequence dependence for Yao graph

Graph defined above contains MST
but may still change too much

Modification: connect each vertex to
nearest previously inserted neighbors

Still contains MST

Six $= \mathcal{O}(1)$ new edges per insertion

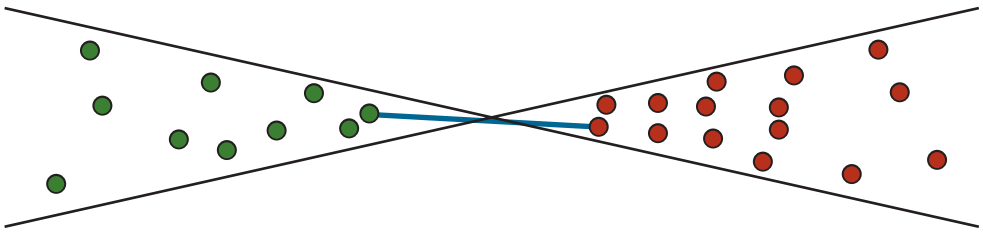
$\mathcal{O}(\log^2 n)$ time (from nearest neighbor queries)

Fully dynamic MST

Key insight:

[Agarwal, Edelsbrunner, Schwarzkopf, Welzl 1990]

Let (u, v) be an edge in the MST
and let W be a *double wedge*
with sufficiently small angle ($< 19.08°$)
containing u and v in opposite wedges



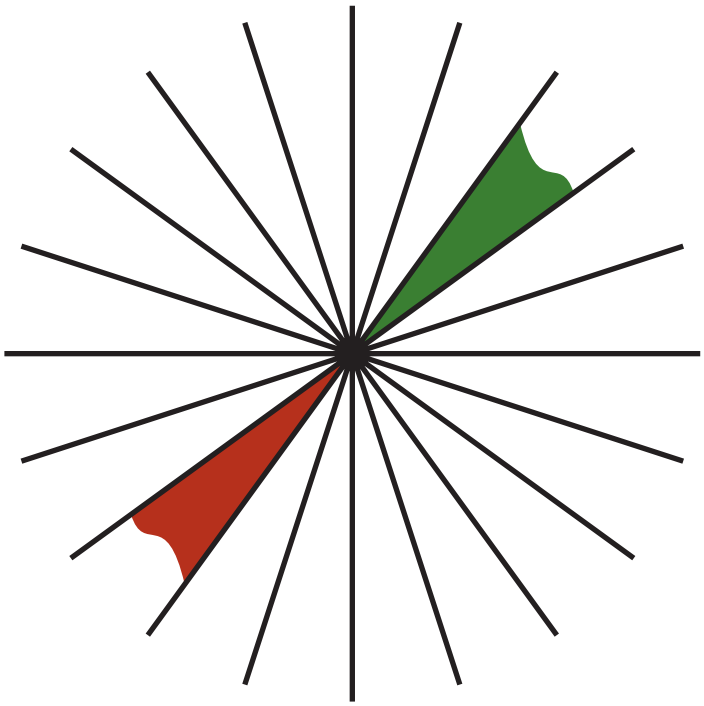
Then (u, v) is the closest pair
of points from opposite wedges

if not, we could find shortcut u-u'-v'-v

So, to solve MST, find set of double wedges
containing all edges of complete graph

Bichromatic Closest Pair Graph: Step I

Find $\mathcal{O}(1)$ double wedges
with small angles

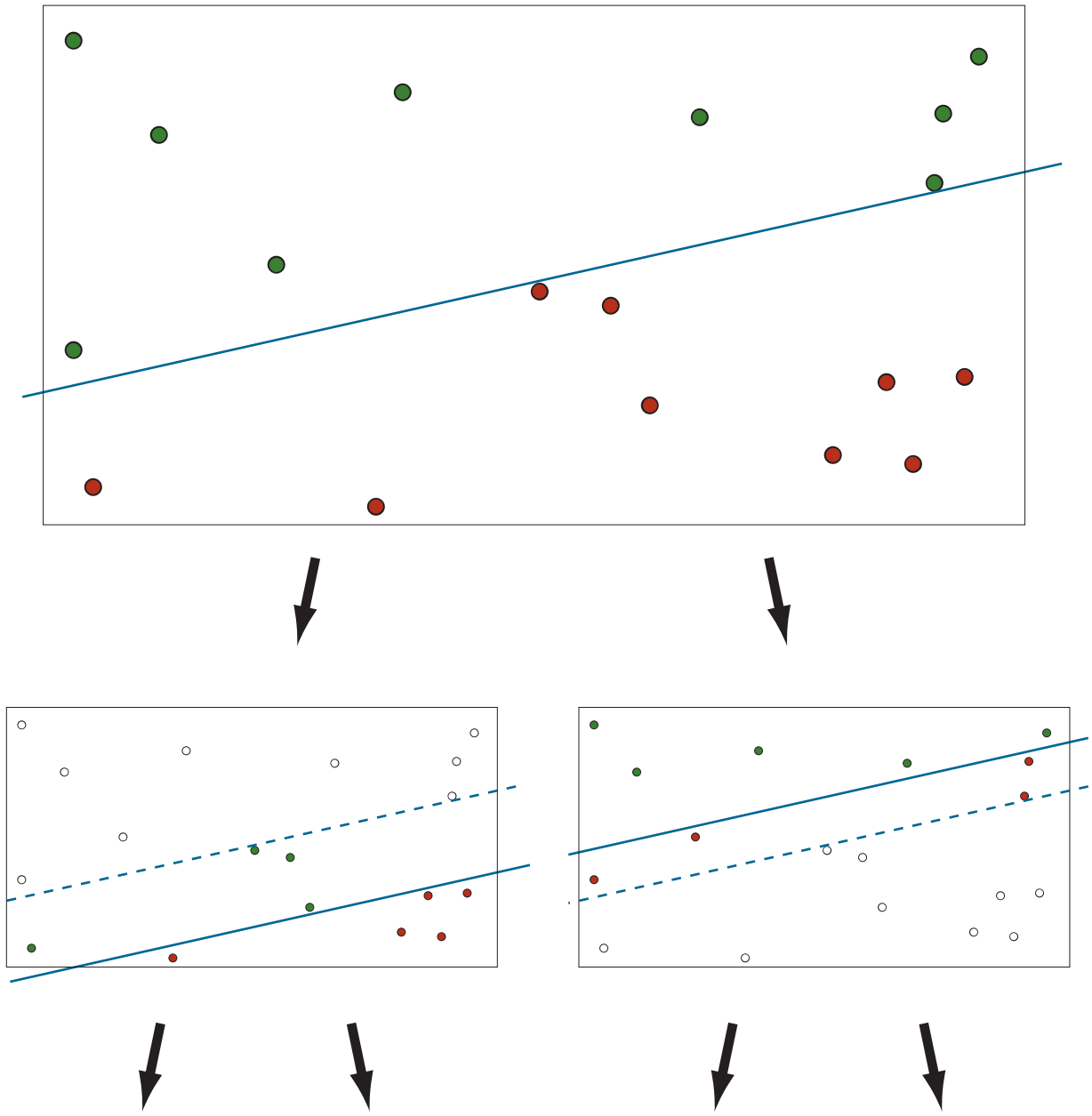


Such that any edge is contained in a translate of one of the double wedges

Then, for each double wedge. . .

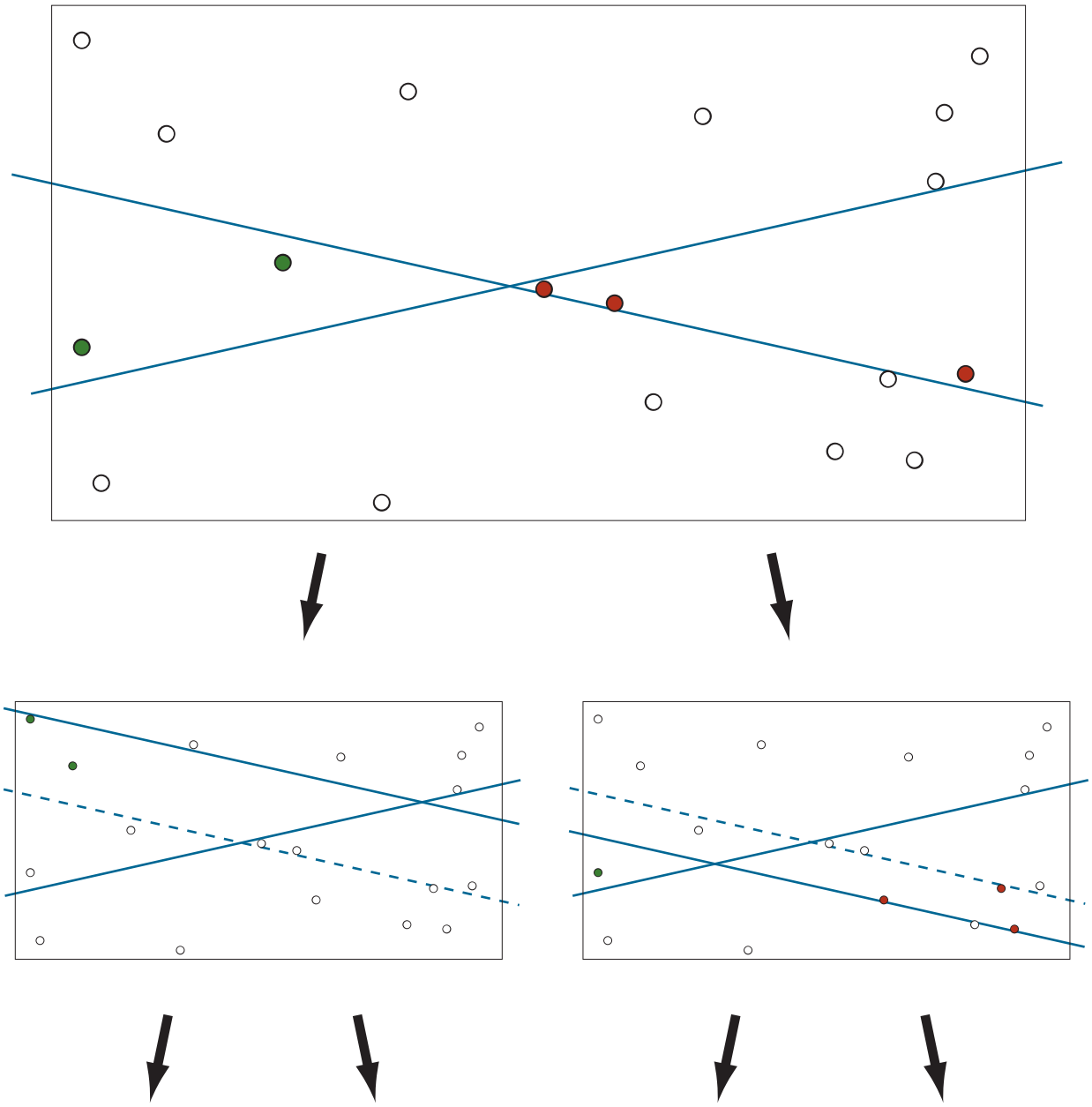
Bichromatic Closest Pair Graph: Step II

Recursively subdivide parallel to wedge boundary



Bichromatic Closest Pair Graph: Step III

Recursively subdivide parallel to other boundary



Bichromatic Closest Pair Graph: Analysis

d stages of recursive subdivision
$\log_2 n$ levels of recursion per stage

$\mathcal{O}(n \log^{d-1} n)$ subproblems
$\Rightarrow$ graph with $\mathcal{O}(n \log^{d-1} n)$ edges

Each point in $\mathcal{O}(\log^d n)$ subproblems
$\Rightarrow$ few graph changes per update

How to maintain BCP Graph?

Data structure for nearest neighbor queries
$\mathcal{O}(n^{1-\frac{2}{\lceil d/2 \rceil + 1}+\epsilon})$ per operation
[Agarwal and Matoušek 1992]

Data structure for bichromatic closest pairs
$\mathcal{O}(\log^2 n)$ nearest neighber ops per update
[Eppstein 1992]

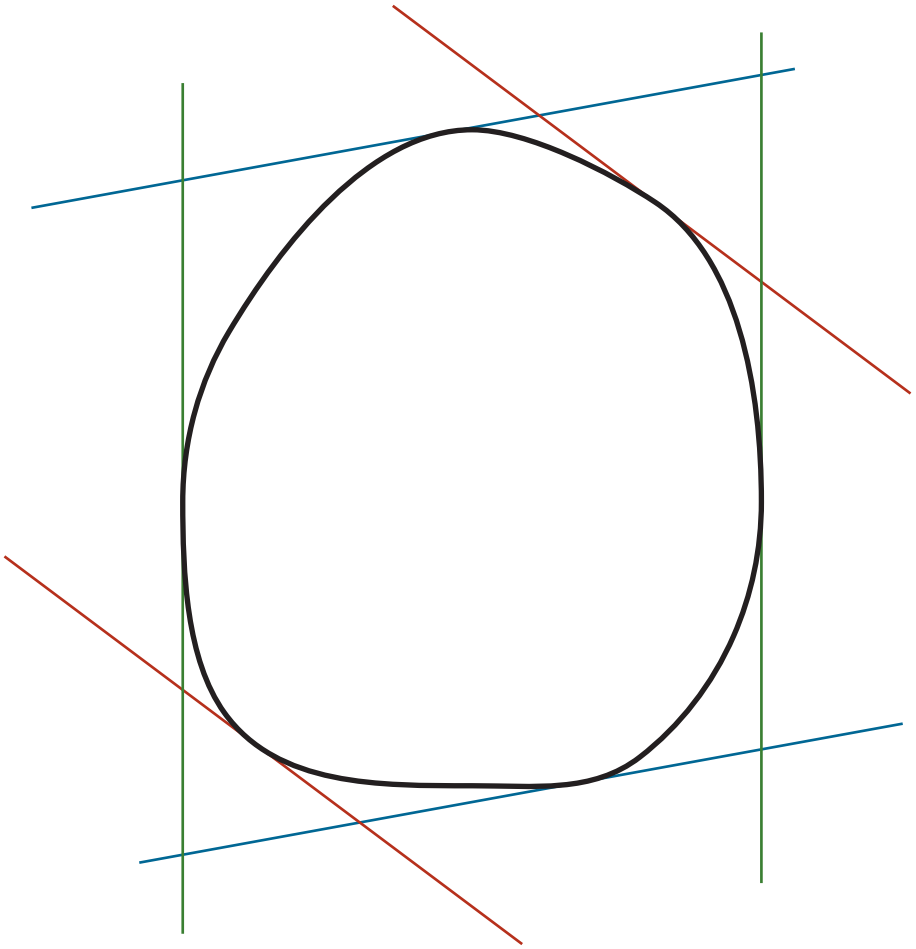
Maintenance of recursive subdivision:
Weight-balanced trees
Rebuild when tree grows too unbalanced

Total: $\mathcal{O}(n^{1-\frac{2}{\lceil d/2 \rceil + 1}+\epsilon})$ per update

$\mathcal{O}(n^{\epsilon})$ for $d = 2$
$\mathcal{O}(n^{1/3+\epsilon})$ for $d = 3$ or $d = 4$
$\mathcal{O}(n^{1/2+\epsilon})$ for $d = 5$ or $d = 6 \ldots$

Diameter and Width

Convex body has two parallel tangents for any slope



Diam = max distance between parallel tangents

Width = min distance between parallel tangents

Diameter and Width Algorithms

Computable in $\mathcal{O}(n \log n)$ time
("Rotating caliper algorithm"):
[Toussaint 1983]

Build convex hull

Find vertical parallel tangents

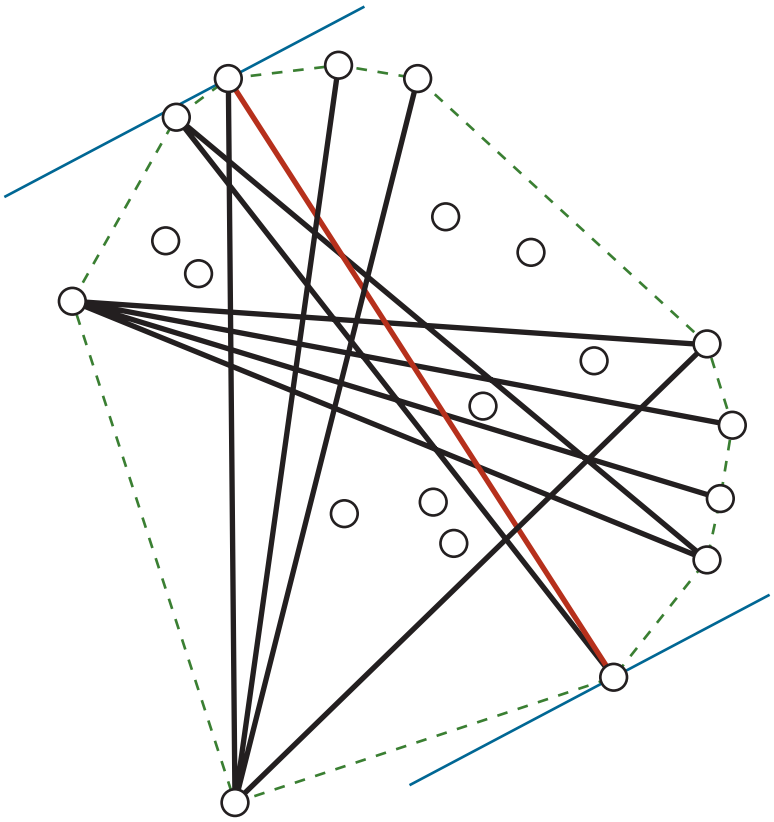
Repeatedly advance one of two tangent points around hull until vertical again

Dynamic diameter similar to BCP

Dynamic width?

Rotating Caliper Graph

Connect convex hull vertices by an edge whenever they are the contact points of two parallel tangents



Forms a thrackle (all edges cross)
with one edge per hull vertex
making an odd cycle with side branches

Average Case Dynamic Width

[Eppstein 1993]

Maintain:

Average-case-efficient dynamic convex hull

Rotating Caliper Graph

Priority queue of vertices and opposite edges
(adjacent neighbors of the same vertex)

Expect $\mathcal{O}(1)$ change to graph
(because it's sparse)

$\Rightarrow \mathcal{O}(\log n)$ expected time per update

Incremental or Decremental Width

[Eppstein 1999]

Maintain:

Dynamic convex hull

Pointer hull vertex $\rightarrow$ best opposite edge

Data structure for finding best edge
$\mathcal{O}(n^\epsilon)$ per update
similar to nearest neighbor structure

Priority queue of vertex-edge pairs

$\mathcal{O}(n^\epsilon)$ time per hull change

For insertions only (or deletions only)
total hull change over all updates $= \mathcal{O}(n)$
so $\mathcal{O}(n^\epsilon)$ amortized time per update

Minimum-diameter k-point subset

[Eppstein and Erickson 1992]

Static problem: given n points
choose a subset of k points
w/ min diameter among all such subsets

Main content of paper:
$\mathcal{O}(n/k)$ subproblems of $\mathcal{O}(k)$ points each
Transforms $nf(n)$ time $\Rightarrow nf(k)$

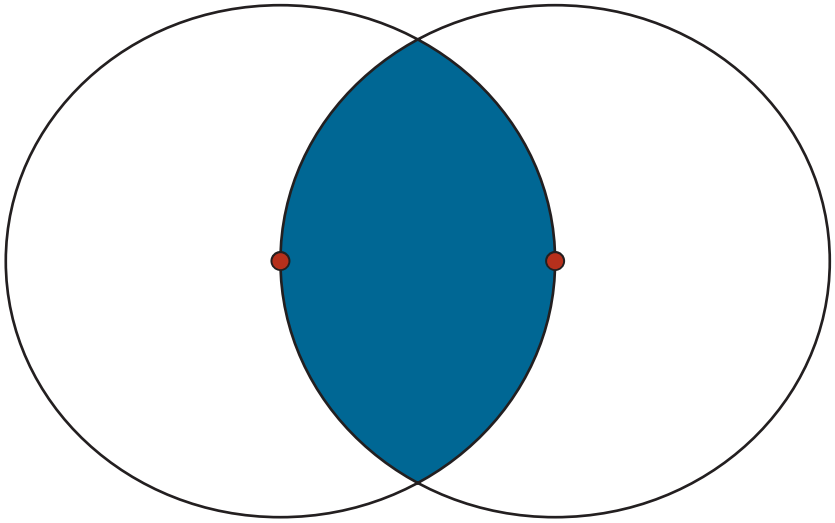
So, assume $k = \Theta(n)$

Binary search among all pairwise distances
calling Test(v, D) for each vertex

Test(v, D):
is there a D-lune of v containing
a k-point subset with diameter $\leq D$?

Diameter and Lunes

lune = intersection of two circles
each circle center on the other circle



D-lune of v: circle radius is D, one center on v

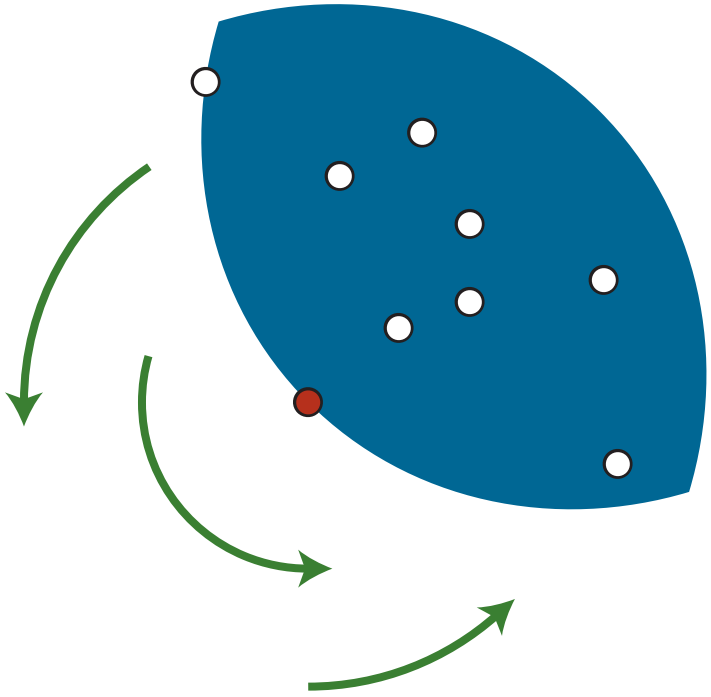
If $\text{diam}(S) = uv$, $|uv| \leq D$ then $S \subset$ a D-lune of v

So $\text{Test}(v, D)$ suitable for use in binary search

Static to Dynamic Transformation

To test all D-lunes of v:

Rotate D-lune around v

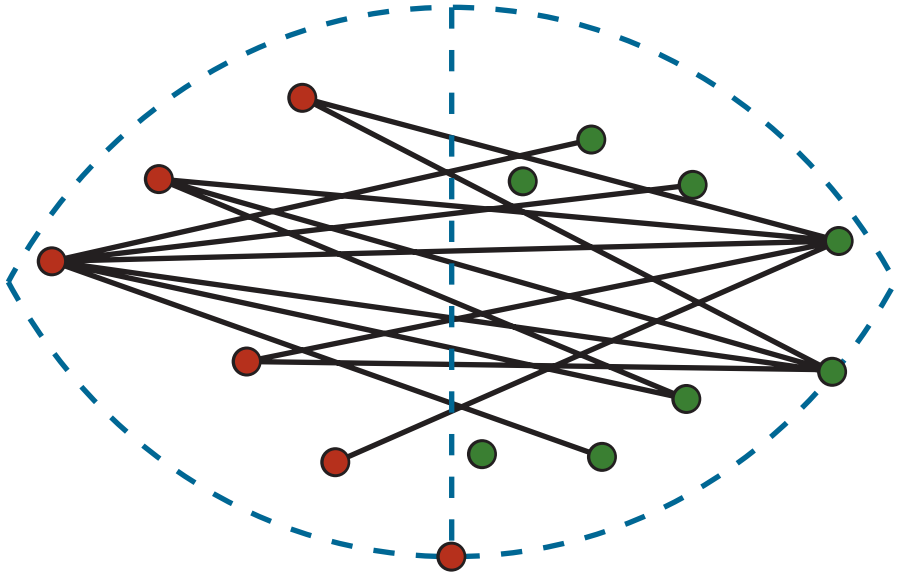


Maintain dynamic set of points in lune

Maintain maximum subset w/diam $\leq D$

Lune Graph

Form set of edges w/length $> D$



Max subset w/diam $\leq D$ is max independent set

Bipartite $\Rightarrow$ solve by matching

Update by single alternating path search

Data structure for finding graph neighbors
(circular hull, Hershberger and Suri 1989)
$\Rightarrow \mathcal{O}(n \log n)$ time per update
$\Rightarrow \mathcal{O}(n^2 \log n)$ per test

New Randomized Improvement

Let $D = \infty$

For each v in random order:

> If Test$(v, D - \epsilon)$ returns true:
>
> > Binary search among all distances
> > for min D: Test(v, D) returns true

Binary search happens only for $\mathcal{O}(\log n)$ points (in expectation)

Remaining points only have one call to Test

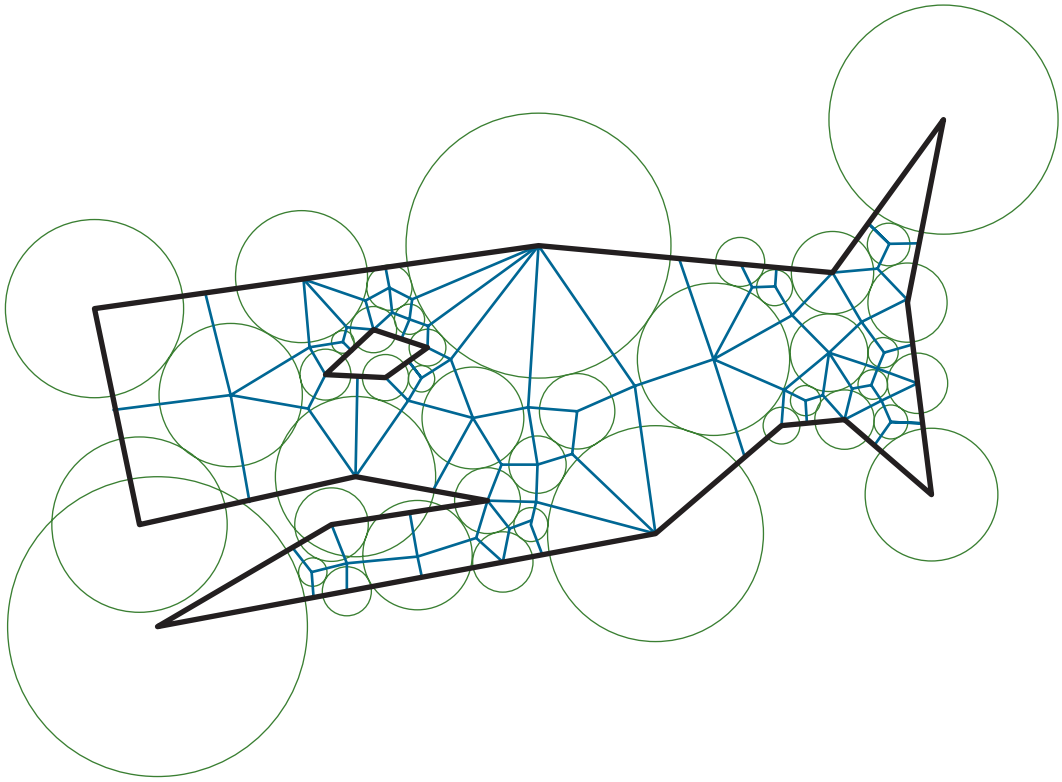
Total expected time $\mathcal{O}(n^3 \log n) \Rightarrow \mathcal{O}(nk^2 \log k)$

(Matches result of [Bhattacharya and ElGindy 1997] but with a much simpler algorithm)

Circle Packing

Given polygon (with holes)

Place circles so gaps have $\leq$ 4 arcs



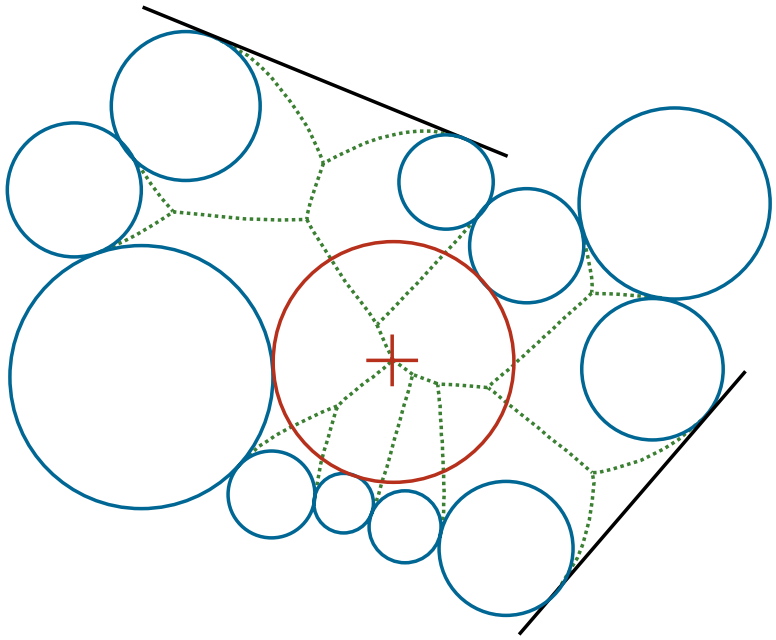
Useful for nonobtuse triangulation [Bern et al 1994]
quadrilateral mesh generation [Bern and Eppstein 1997]

Circle Packing Steps

- Protect vertices
  by placing 1 or 2 nearby circles

- Connect boundary components
  so interior becomes simply-connected

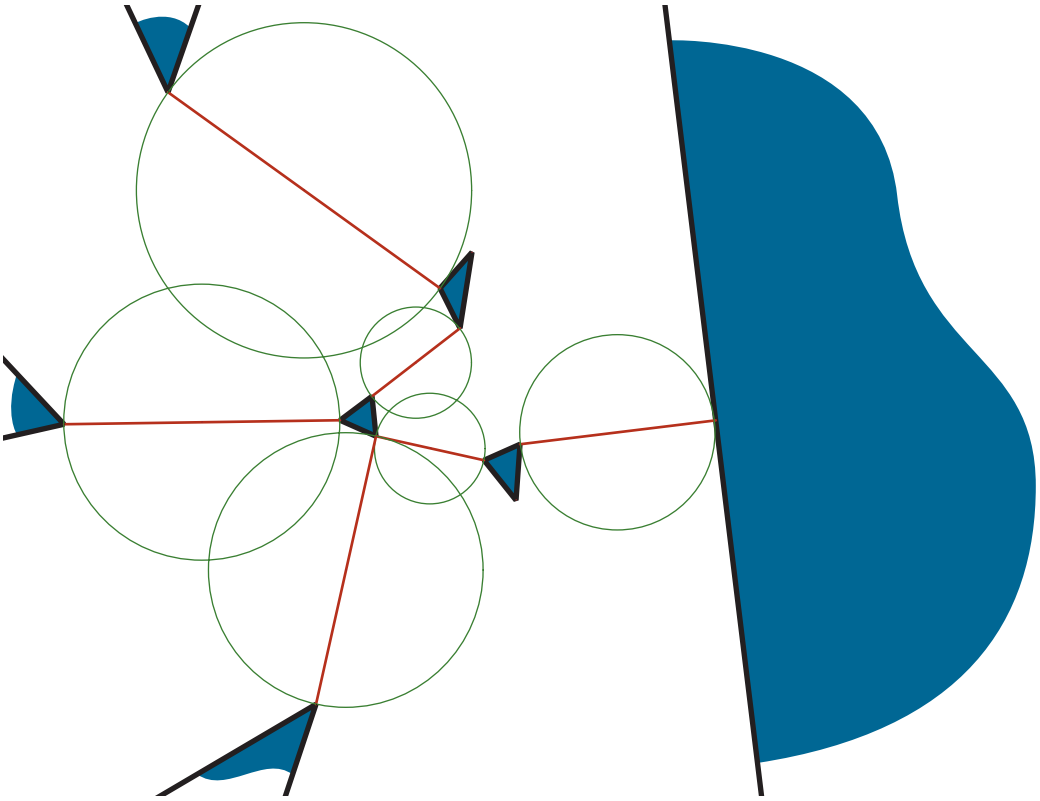
- Repeatedly subdivide interior into simpler pieces
  by centering circles on medial axis



To Connect Boundary Components

- Compute minimum spanning tree
  (using medial axis)

- Form diameter circles



- Shrink overlapping circles

Neighborhood Graph

[Miller, Teng, Vavasis 1991]

Any point in the plane $\in \mathcal{O}(1)$ circles

$\Rightarrow \mathcal{O}(1)$ larger circles cross any circle

$\Rightarrow \mathcal{O}(n)$ crossing pairs

List all crossings in $\mathcal{O}(n \log n)$ time
using geometric separator divide-and-conquer

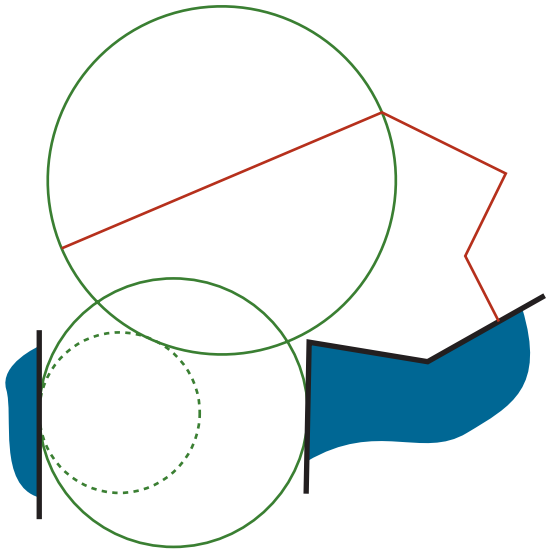
[Eppstein, Miller, Teng 1993]

Shrinking to Eliminate Crossings

Maintain dynamic tree [Sleator, Tarjan 1983]
vertices = polygon holes, edges = circles

For each circle *C*, larger to smaller,
and each crossing with a larger circle *D*:

- Remove *C* from packing and tree
- Find which subtree contains *D*
- Reconnect to *C*'s tangency on other subtree



Circle Packing Analysis

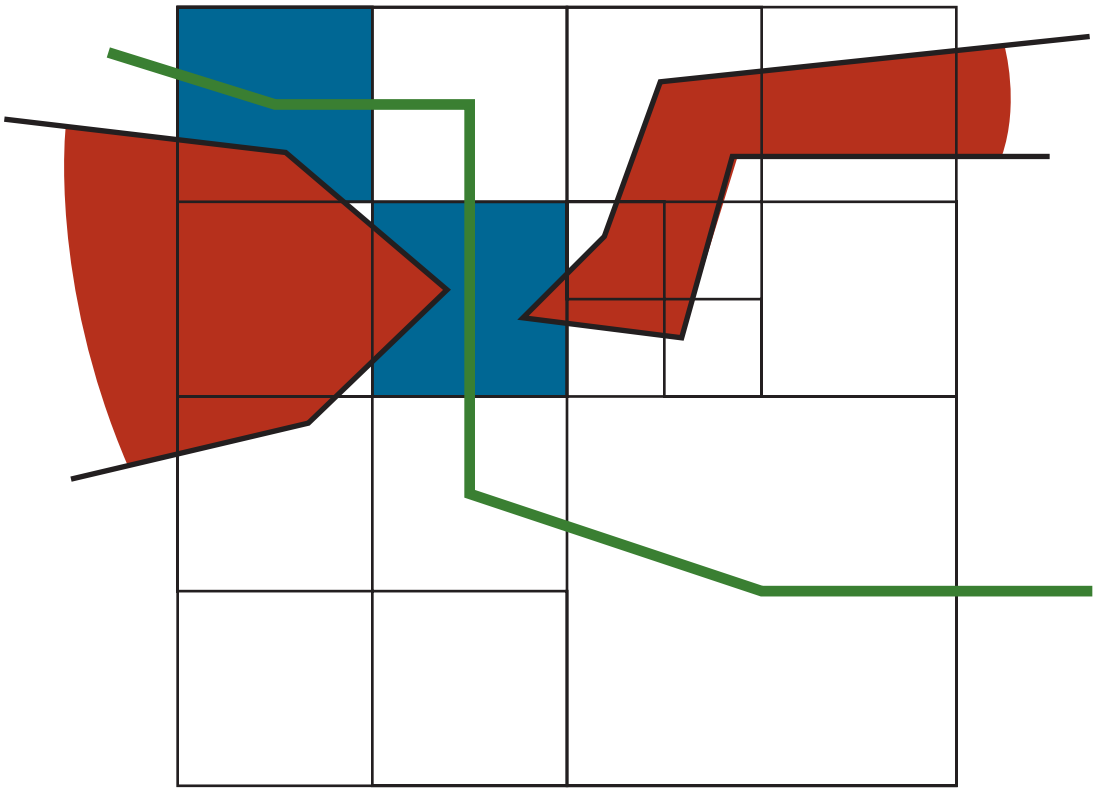
- $\mathcal{O}(n \log n)$ to find MST
- $\mathcal{O}(n \log n)$ to compute intersection graph
- $\mathcal{O}(\log n)$ per shrinkage operation
- $\mathcal{O}(n \log n)$ to finish the packing

Robot Motion Planning

[Barbehenn and Hutchinson 1995]

Problem: find path of robot through configuration space, avoiding obstacles

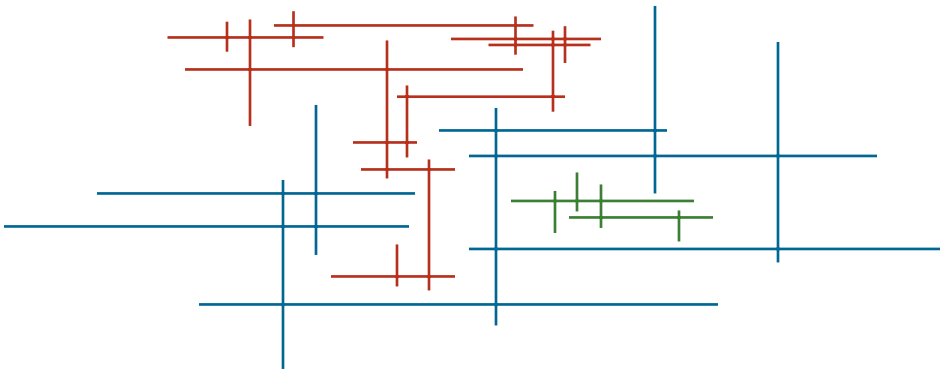
Solution idea: build quadtree over space
classify squares as open, blocked, unknown
find path through unblocked squares
refine unknown squares in path



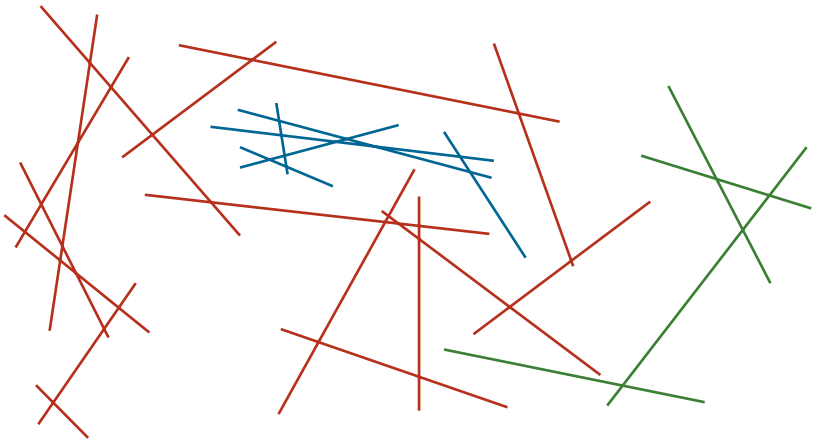
Incremental Connectivity of Line Segments

[Agarwal and van Kreveld 1993]

Axis-aligned segments: $\mathcal{O}(\log^2 n)$

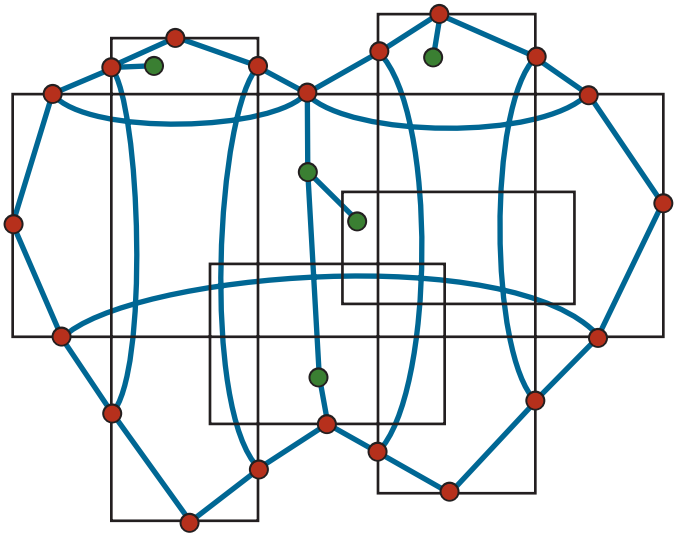


Unaligned segments: $\mathcal{O}(n^{1/3+\epsilon})$



Kinetic Connectivity of Rectangles

[Hershberger and Suri 1999]



Graph w/vertex per rectangle or boundary segment

Edges:

- Adjacent segments of union
- Adjacent segments of same rectangle
- Rectangle to one boundary segment
- Rectangle to the rectangle that contains its top left corner

Some Open Problems

MST of dynamic point set with arbitrary distance function in $\mathcal{O}(n \log^{\mathcal{O}(1)})$ space and update time

Polylogarithmic 2d Euclidean MST

Fully dynamic width

Derandomize min diameter subset improvement

Min diameter subset for $d \geq 3$

Kinetic Euclidean MST
(needs nonlinear kinetic graph algorithm)