Complexity and Computation of 3D Delaunay Triangulations

Nina Amenta (UC-Davis)

Large inputs

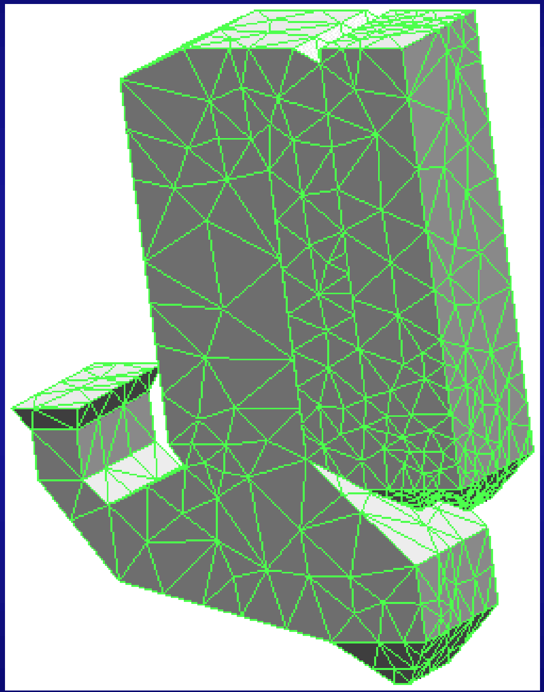
Input point set, produce surface:

20,000 - 20,000,000

Large inputs

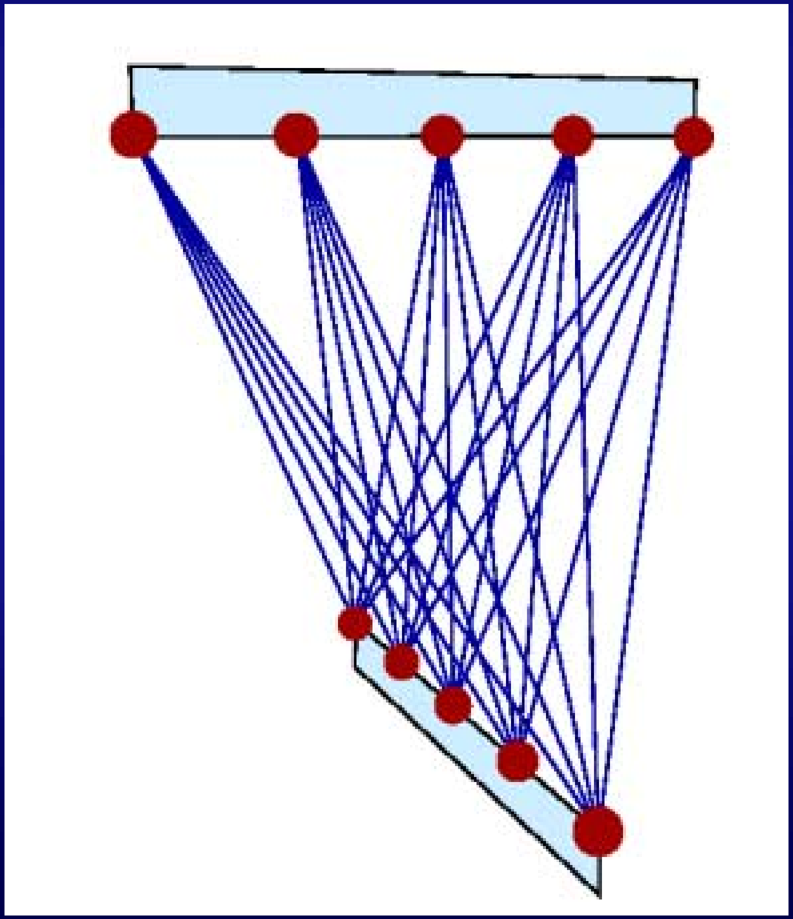
Input surface, produce tetrahedral mesh for finite element simulations; also medial axis.

Easily millions.



Shewchuk (98?)

3D Delaunay $O(n^2)$ ...

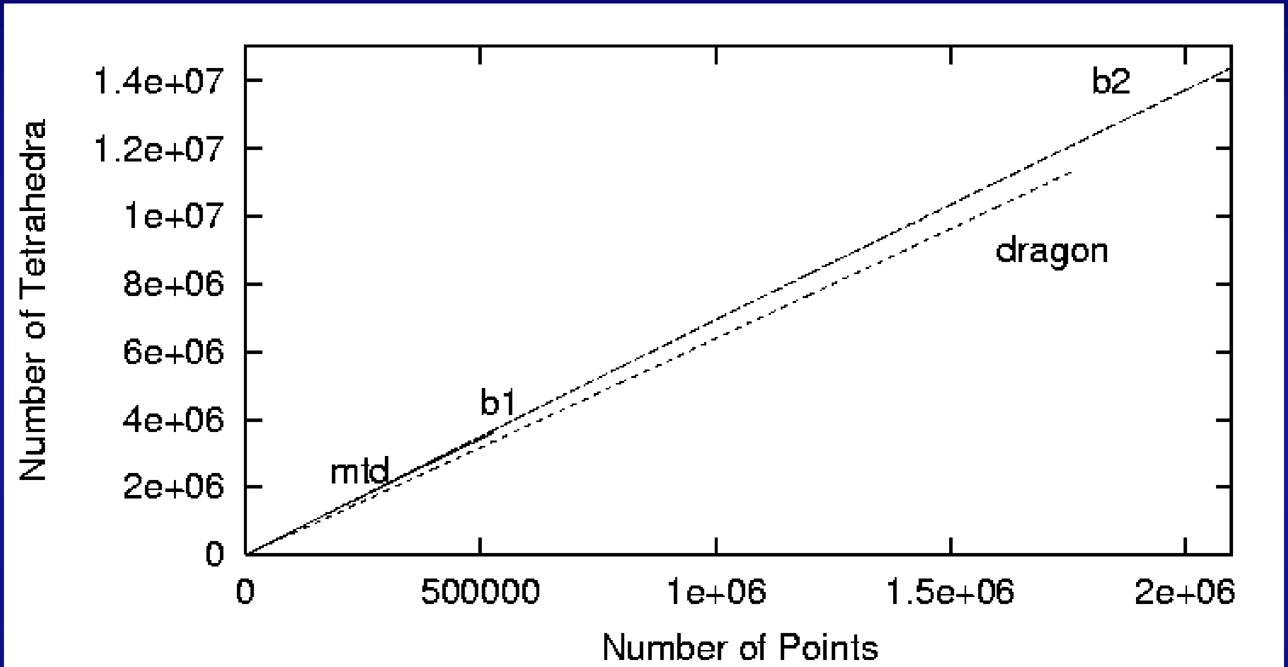


n/2 points on each of two skew lines.

Points on moment curve, etc.

All examples distributed on 1D curve?

… but linear in practice.



A & Choi, 01

Adding samples from surfaces in random order, # tetrahedra grows linearly.

Linear special cases

Dwyer 91 - Uniform random in ball (any constant dimension)

Erickson 02 - Nicely sampled solid, “spread” $O(n^{1/3})$

Golin & Na 00 - Uniform random on surface of convex polyhedron

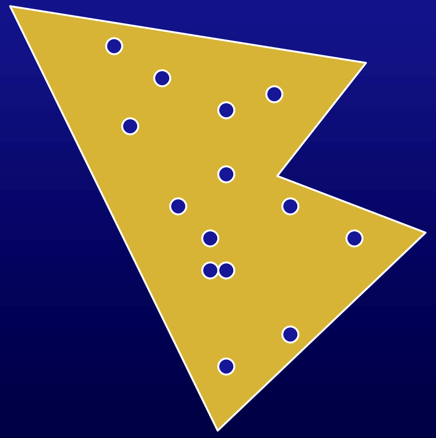
Attali & Boissonnat 02 - Nicely sampled polyhedral surface

Sampling models

Consider behavior as n->infinity.

Every point has a point within distance ε and no point within distance δ.

Every point has $1 \le m \le k$ samples within distance ε.



Almost linear

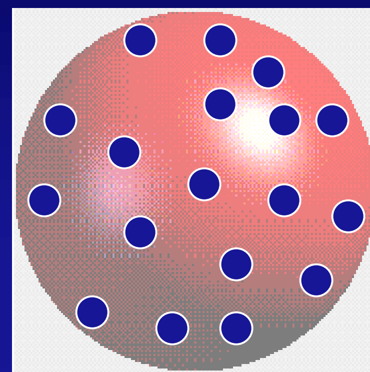
Golin & Na 02 - Uniform random on polyhedral surface, O(n $\log^4 n$)

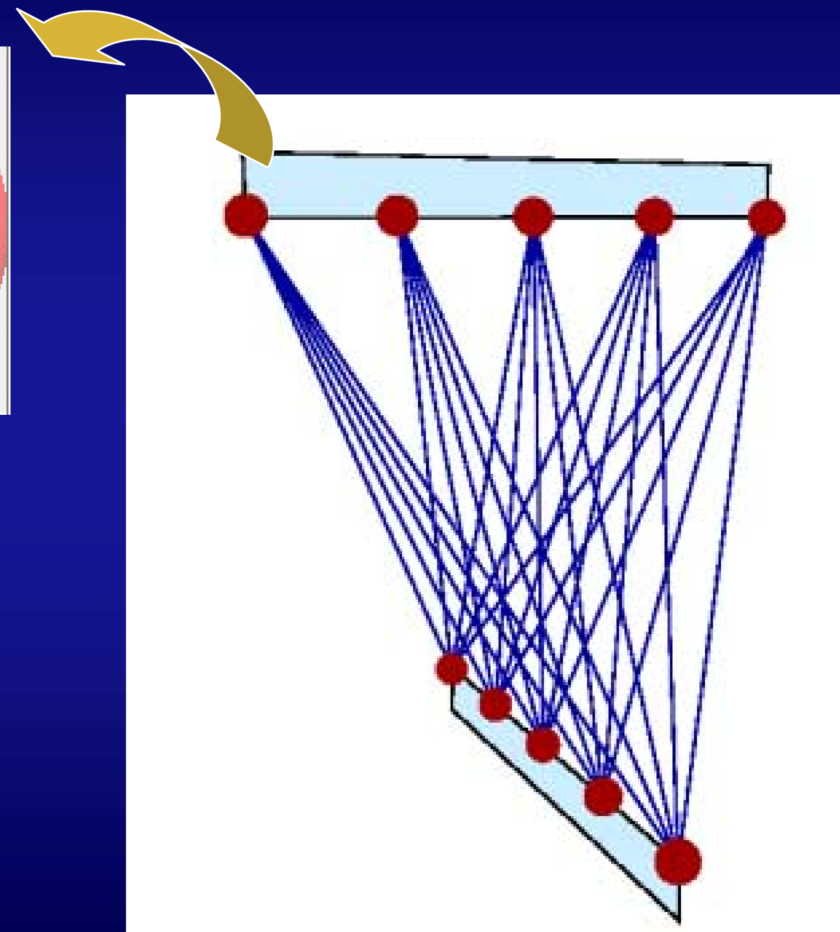
Attali, Boissonnat & Lieuter 03 -Nice sampling, “generic” smooth surface S: singular points (with osculating maximal tangent balls) form a 1D set with fixed length, O(n lg n)

Lower bounds

Jeff Erickson (by Howard Sun)

Lower bounds





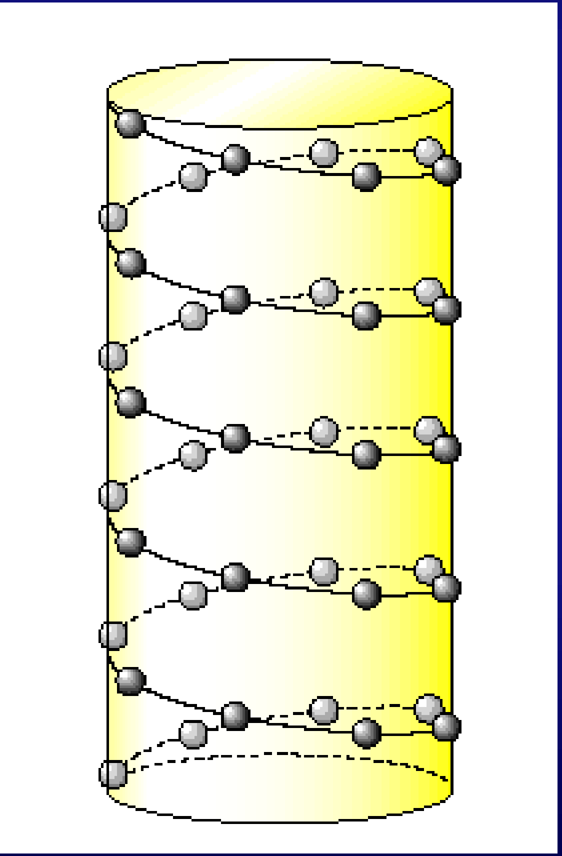
O(εn) balls

Given n, ε , can construct a suface and an ε–sample with O($n^2\varepsilon^2$) triangulation.

Lower bounds

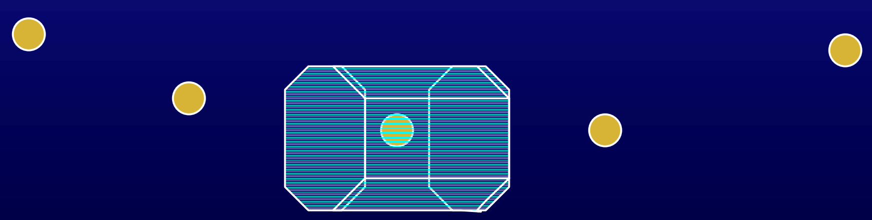
Helix with sqrt(n) turns, sqrt(n) samples per turn.

Fact (Erickson, Bochis & Santos): ball tangent to cylinder at 2 samples in same turn contains no other samples -> $O(n^{3/2})$ Delaunay edges.



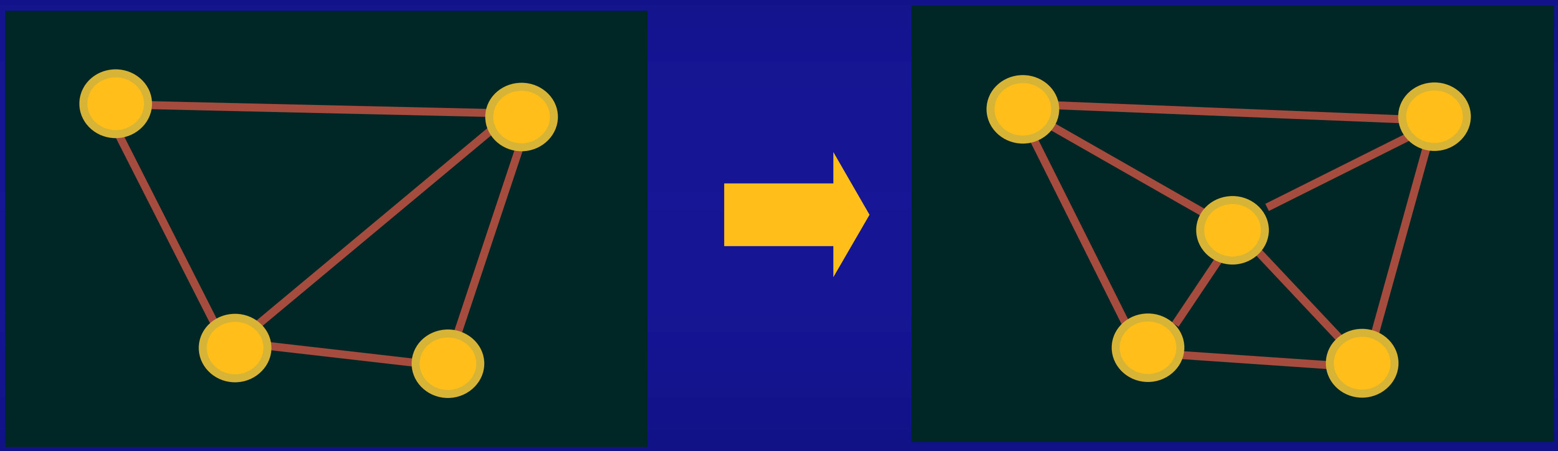
Higher dimensions?

Conjecture: Nice distribution of samples from surface of co-dimension c has Delaunay triangulation of complexity $O(n^{\lfloor c/2 \rfloor + 1})$?



Compute only "in-manifold" linear part?

Randomized incremental algorithm



Add points one by one in random order, update triangulation with flips. Simple, optimal (worst-case expected time).

Implementations

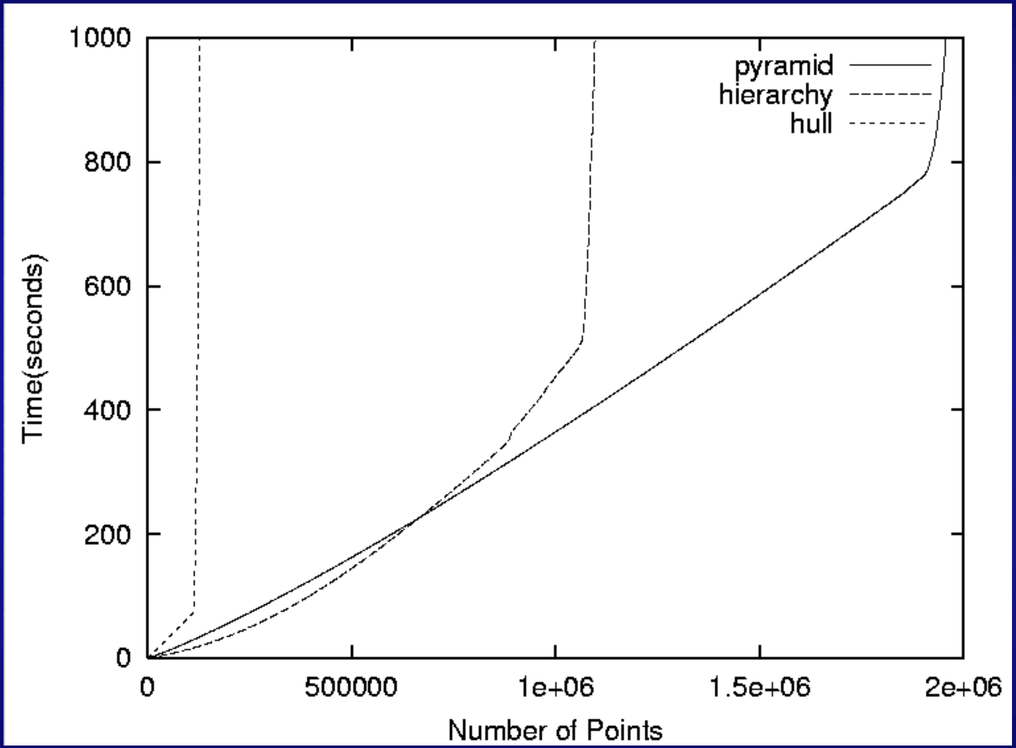
delcx - Edelsbrunner, Muecke, Facello 92,96

hull - Clarkson 96

CGAL Delaunay hierarchy - Devillers, Teillaud, Pion 01

pyramid - Shewchuk, unreleased

Memory usage



Performs great…until !

Point location strategies

Theoretical bottleneck.

O(log n) per location possible with search data structure, but is it worth the effort in practice?

CGAL, hull - data structures

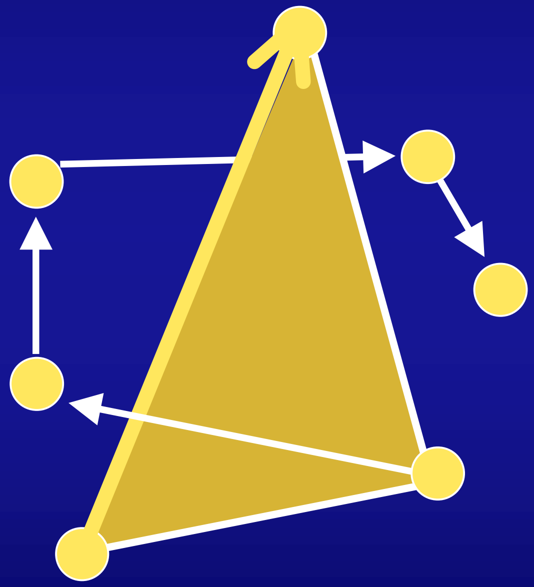
delcx, pyramid - no data structures

Idea

Blelloch, Blandford, Cardoze, Kadow 03

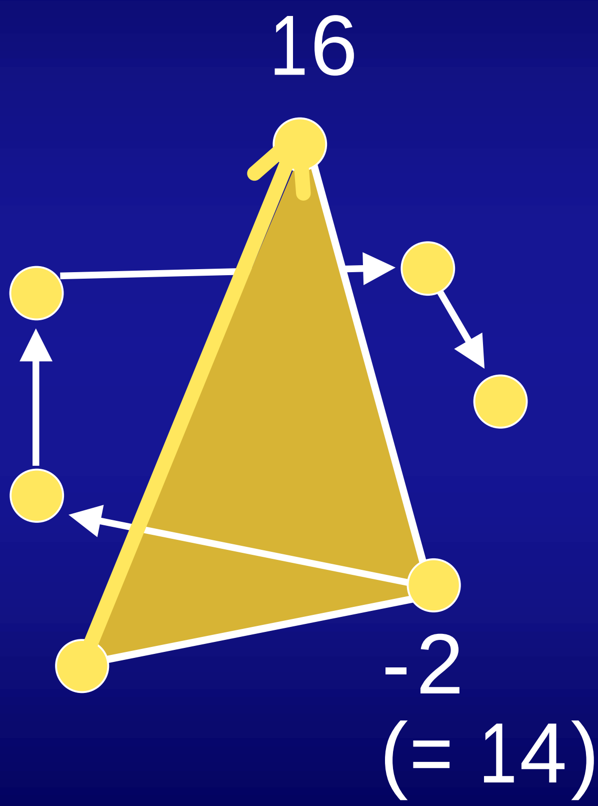
Compress representation of DT, while allowing updates.

Representation



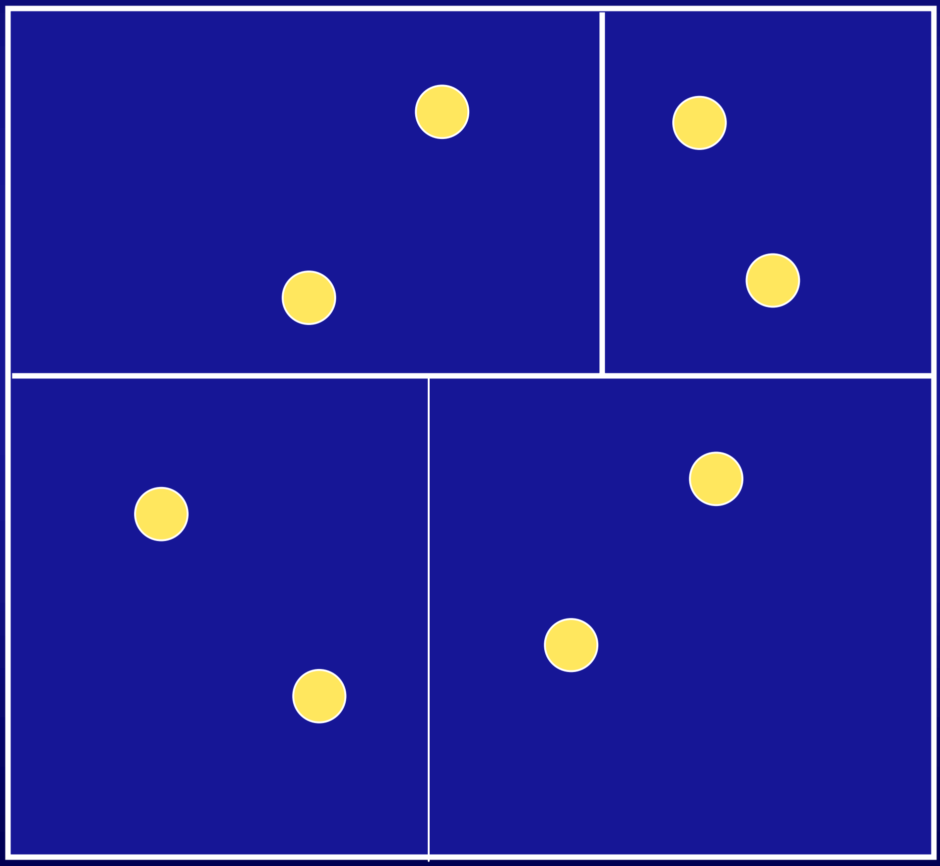
List vertex indecies around each edge (with tricks to reduce redundancy).

Representation



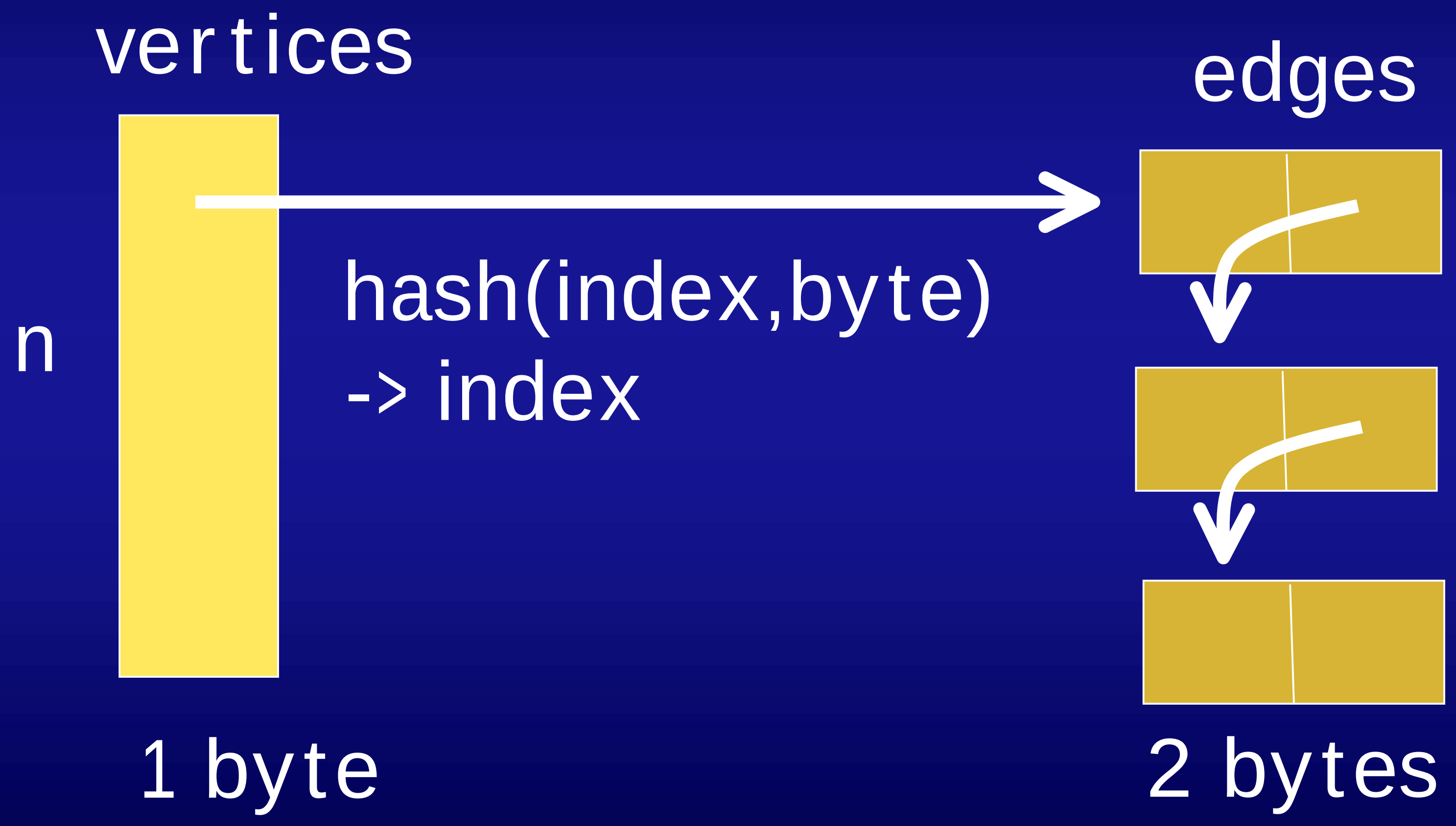
Use difference coding so each index is just a few bits.

Assigning indices

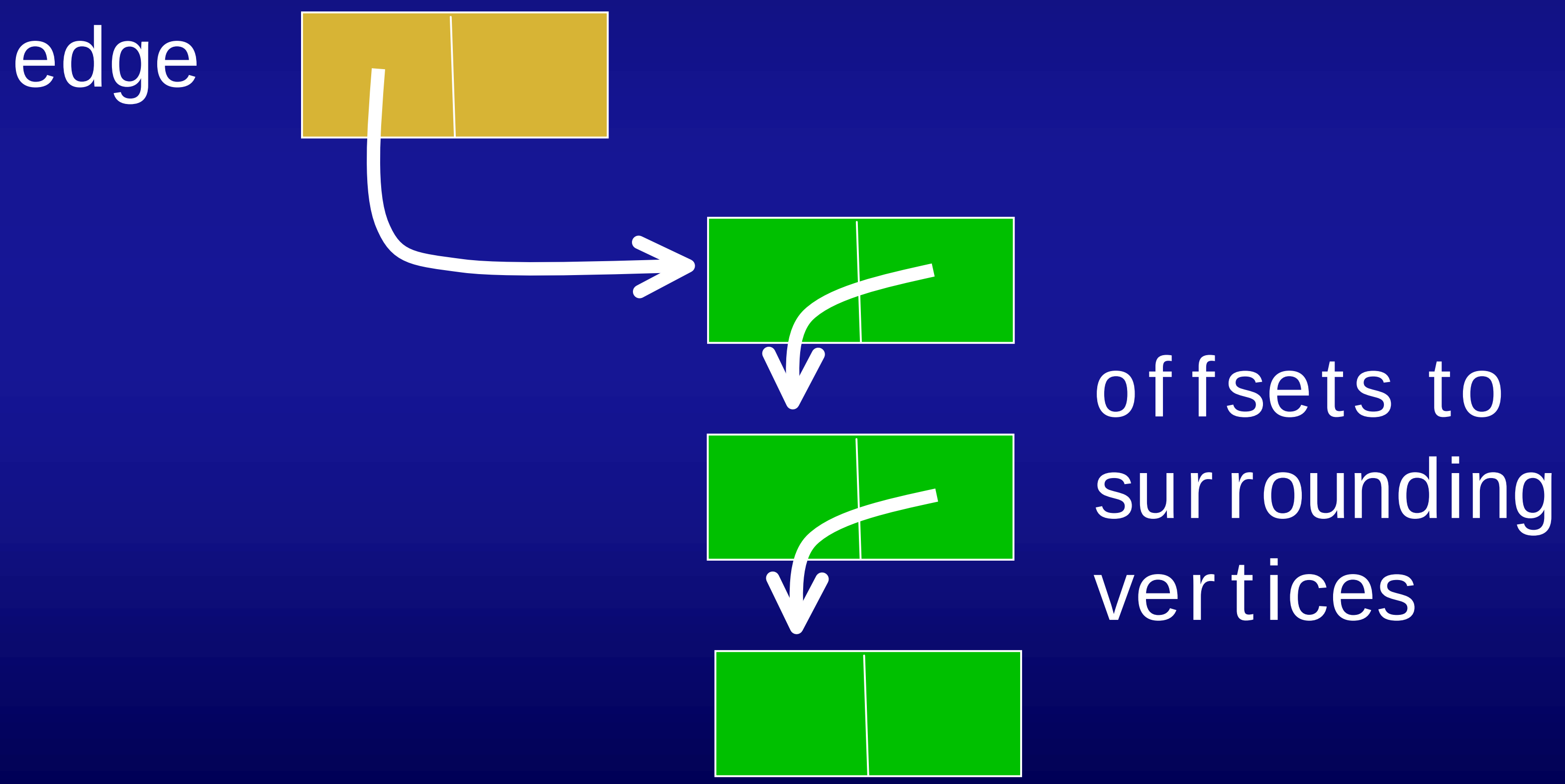


Use kd-tree to assign similar indices to points (hopefully) near each other in Delaunay triangulation.

Data structures

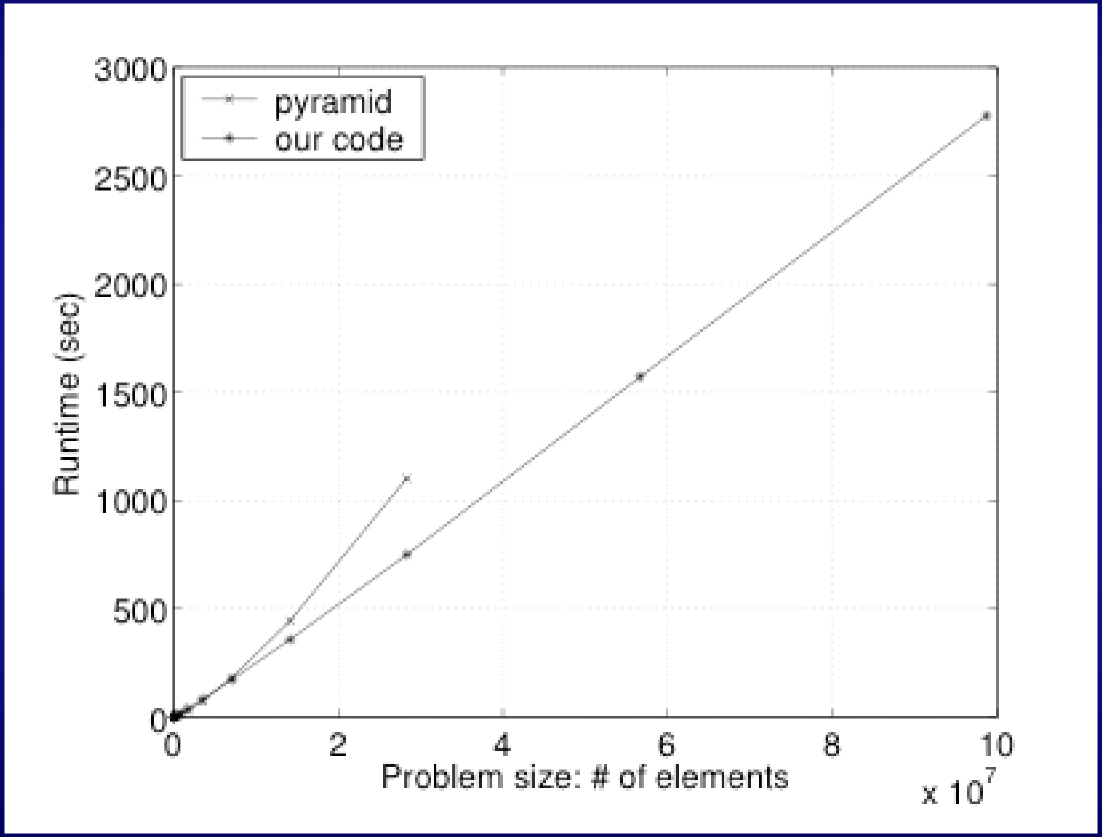


Data structures



Compression results

50min



100M tets = 10M pts

2 Gig, 2.4 GHz

Idea

Partially randomized insertion order

- increase locality of reference, especially as data structure gets large
- retain enough randomness to guarantee optimality

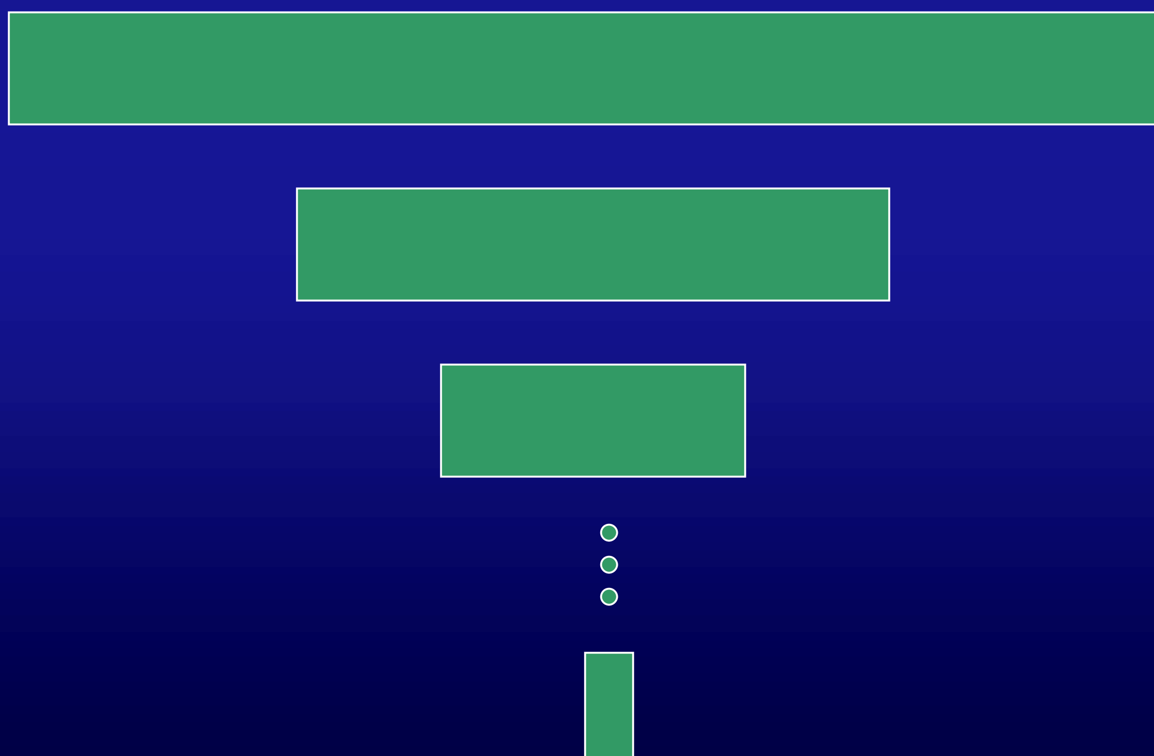
Biased Randomized Insertion Order (BRIO)

(A, Choi, Rote, 03)

- Choose each point with prob = 1/2.
- Insert chosen points recursively con BRIO.
- Insert the remaining points in arbitrary order.

BRIO

log n rounds of insertion



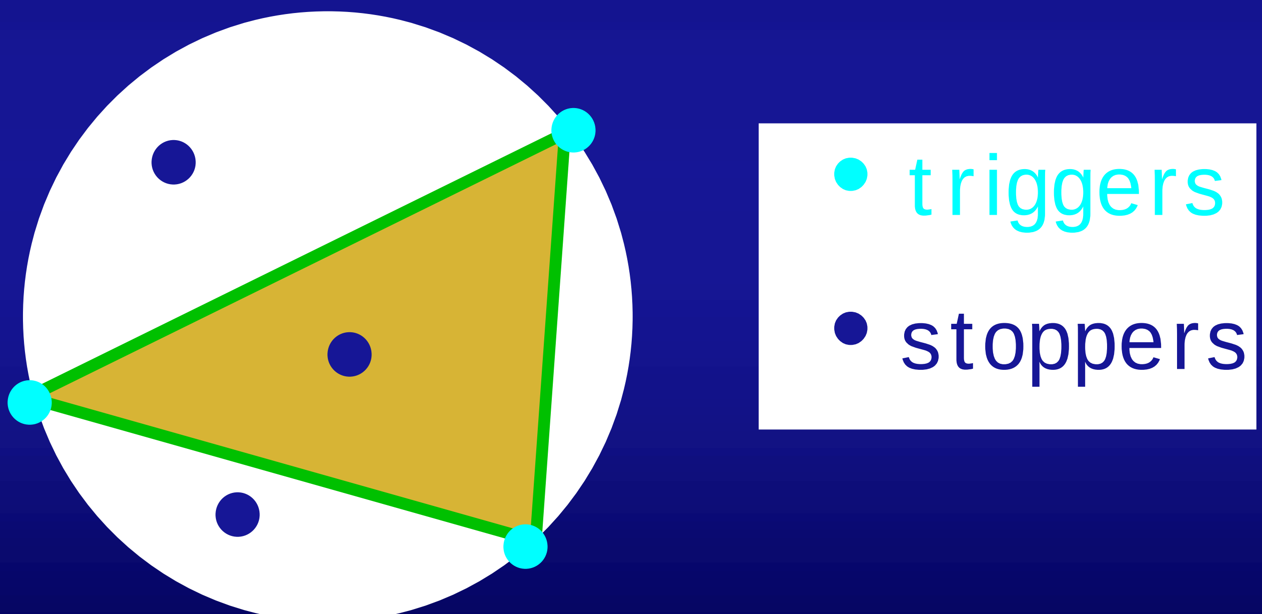
Analysis

Randomness has two benefits:

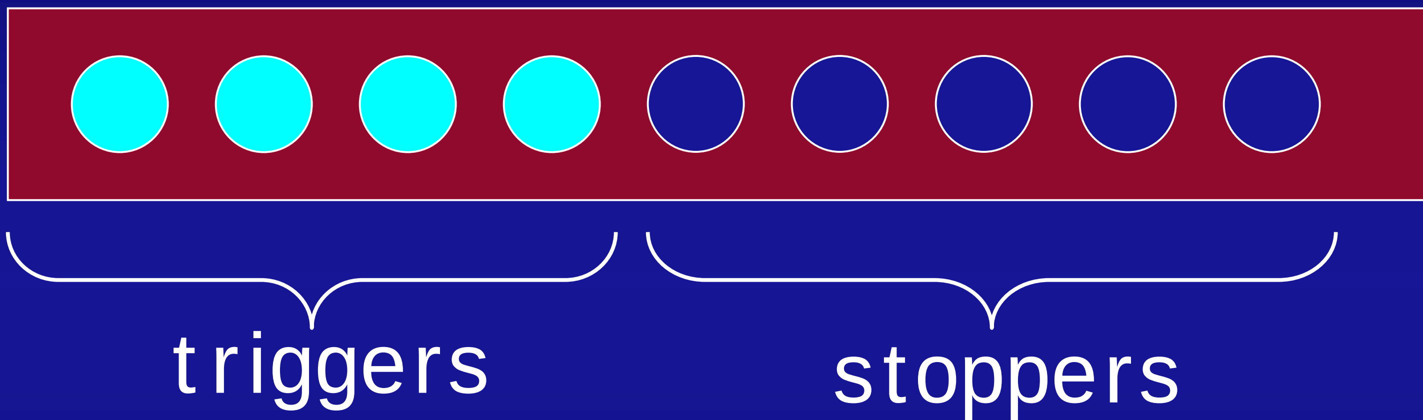
- Bound total number of tetrahedra
- Bound time required for locating new points in triangulation

Analysis

Adapted from Clarkson and Shor, Mulmuley



Triggers and stoppers



A tetrahedron appears during construction if all its triggers are inserted before any of its stoppers.

Probability tet appears

<= P[the round where all triggers are chosen is <= the first round where any stopper is chosen]

= P [s+4 random numbers, the first 4 <= others]
= $O(1/s^4)$

Analysis of optimality goes through directly.

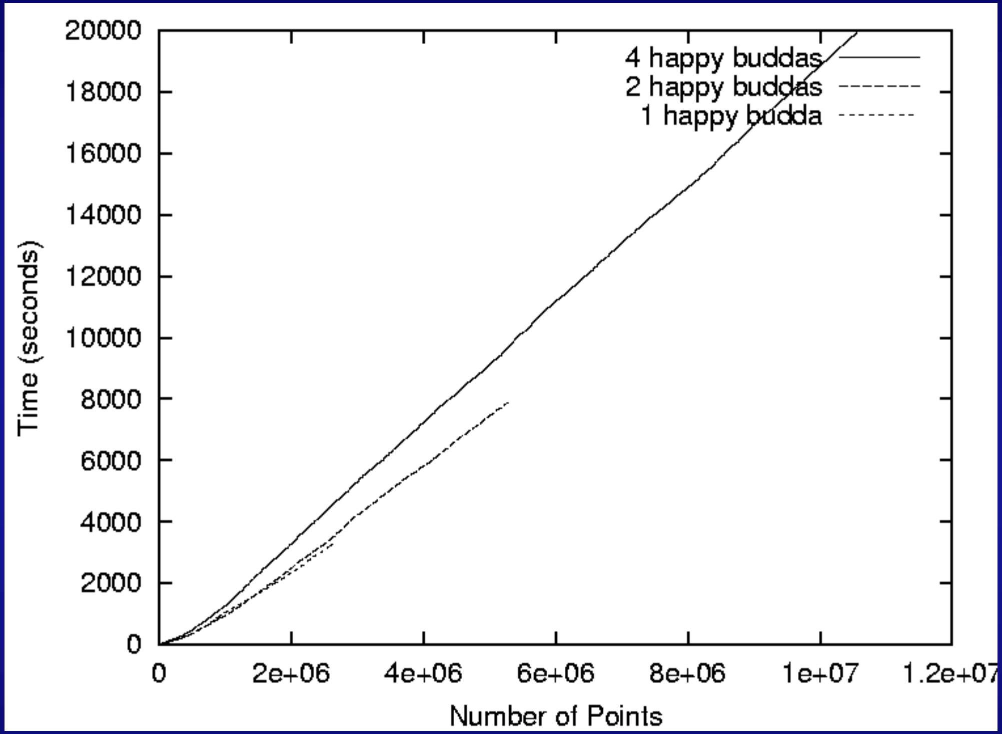
Experiments - pyramid

Point location: “walk” from last inserted point.

Multiple “Happy buddha”. 4096 kd-cells.

360 MHz, 128 M RAM, 4 GB Virtual memory

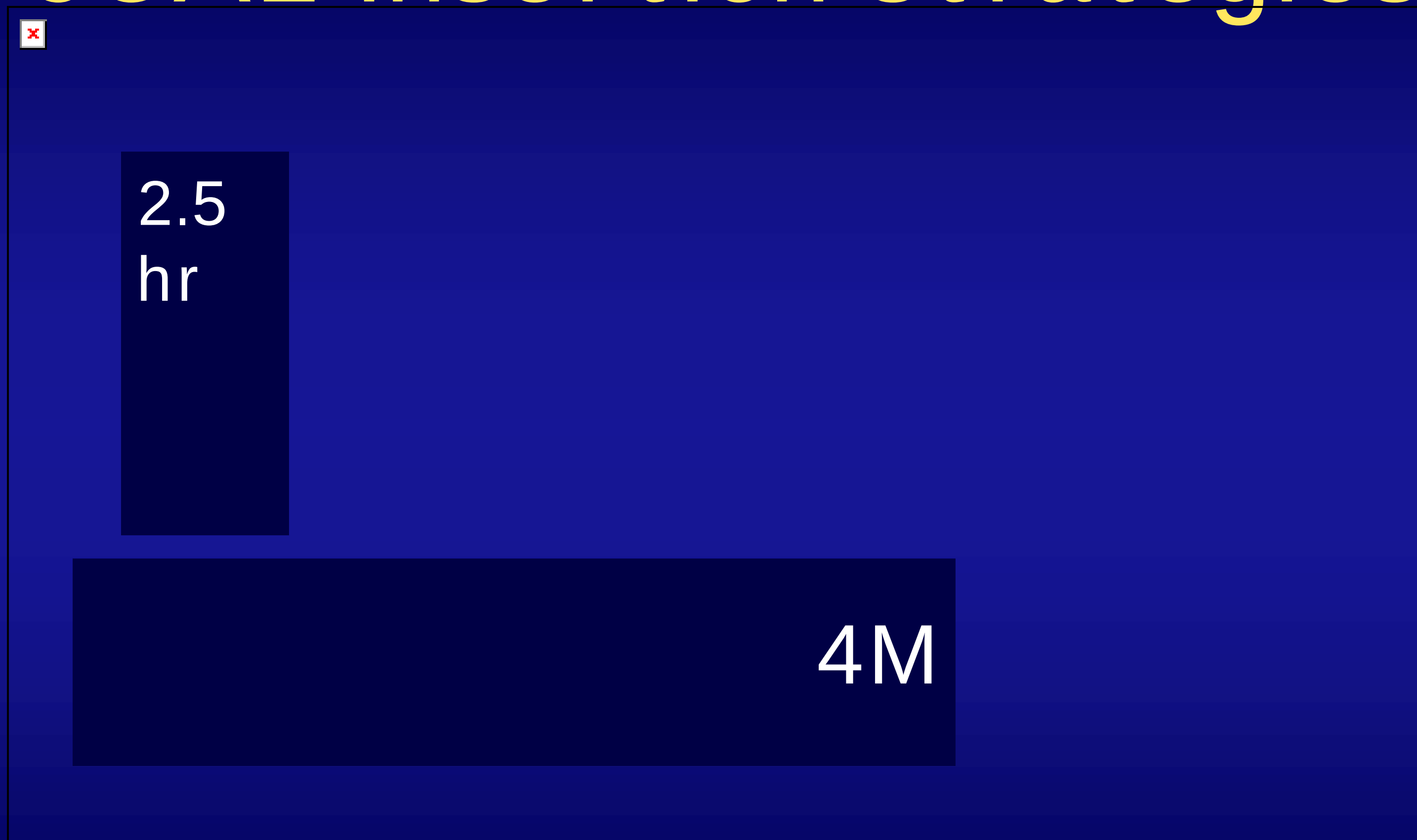
Pyramid



10 million points on tiny machine

1/2 hour on reasonable machine

CGAL insertion strategies



Thanks to Monique Teillaud and Ian Bowman.

Conclusion

Think of 3D Delaunay triangulation as essentially linear time, fairly efficient.

Really independent subproblems would help.