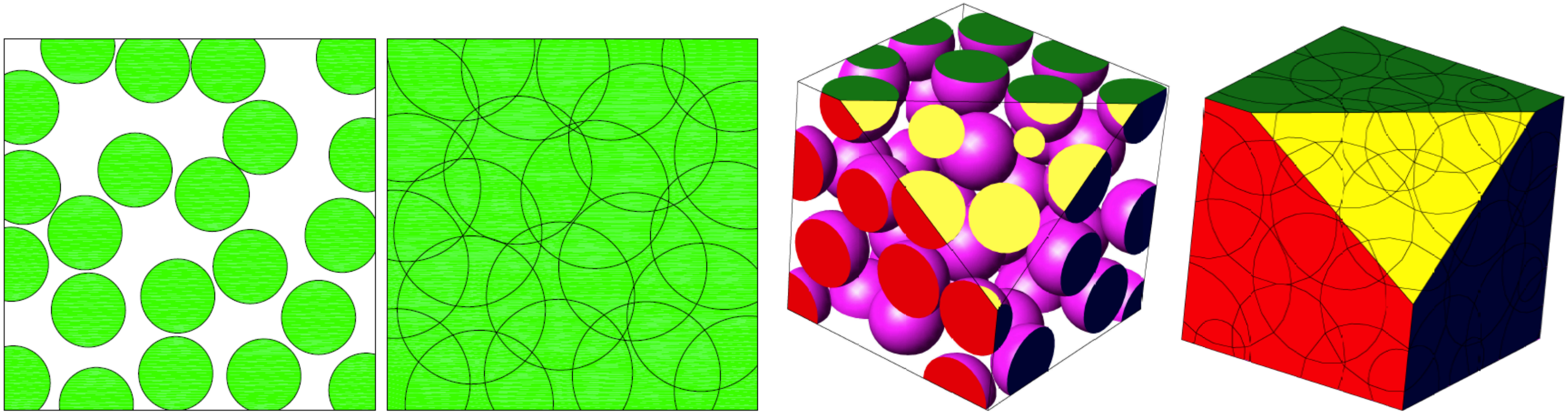


A Simple Algorithm for Maximal Poisson-Disk Sampling in High Dimensions

Mohamed S. Ebeida, Scott A. Mitchell, Anjul Patney,
Andrew A. Davidson, and John D. Owens

presenter = Scott

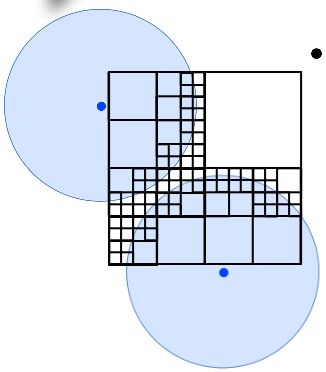
Eurographics 2012

UCDAVIS
UNIVERSITY OF CALIFORNIA

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company,
for the United States Department of Energy's National Nuclear Security Administration
under contract DE-AC04-94AL85000.

Sandia National Laboratories

Overview



- **Classic Dart throwing +**
  - Quadtree
  - Squares track remaining regions
  - Track misses for refinement decisions
  - Avoid refining too deep

[Wei08] Wei L.-Y.: Parallel Poisson disk sampling.
ACM Transactions on Graphics 27, 3 (Aug. 2008), 20:1–20:9.

[BWWM10] Bowers J., Wang R., Wei L.-Y., Maletz D.:
Parallel Poisson disk sampling with spectrum analysis on surfaces.
ACM Transactions on Graphics 29 (Dec. 2010), 166:1– 166:10.

"Make everything as simple as possible, but not simpler." – A. Einstein

- Flat quadtree – one level of squares active, pool of indices
  - Simpler Datastructure ☺ Less memory ☺
- Globally refine periodically, ignore local misses
  - Simpler Datastructure ☺ More parallel ☺
- Refine to machine precision,
  on average it is so rare that memory is not an issue
  - More Maximal ☺

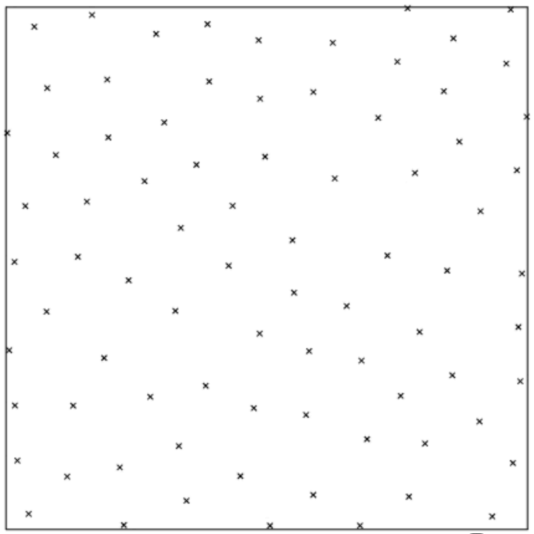
"This could be the current algorithm of choice for dart throwing." – Eurographics reviewer #2

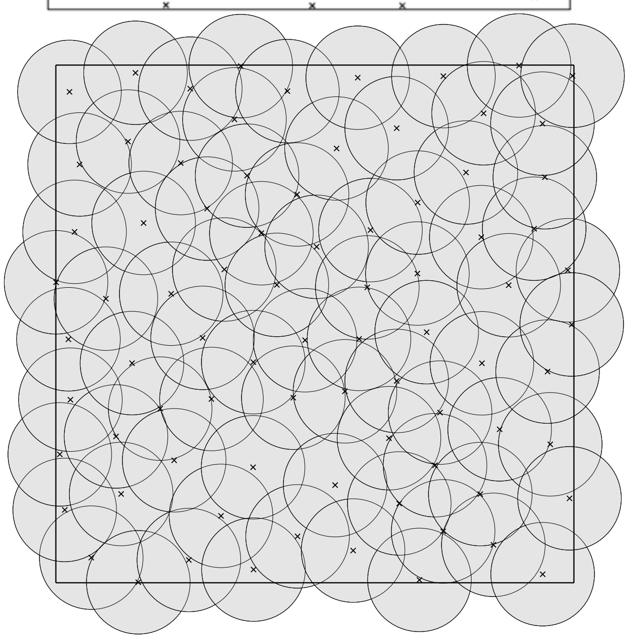
- Code works great but we can't prove the spatial stats theory.

Provable:
Ebeida M. S., Patney A., Mitchell S. A., Davidson A., Knupp P. M., Owens J. D.:
Efficient maximal Poisson-disk sampling.
ACM Transactions on Graphics 30, 4 (July 2011), 49:1–49:12

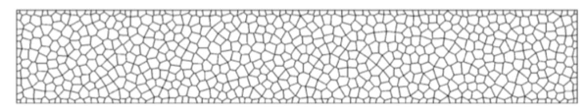
Sandia National Laboratories

Why MPS?
Maximal Poisson-disk Sampling



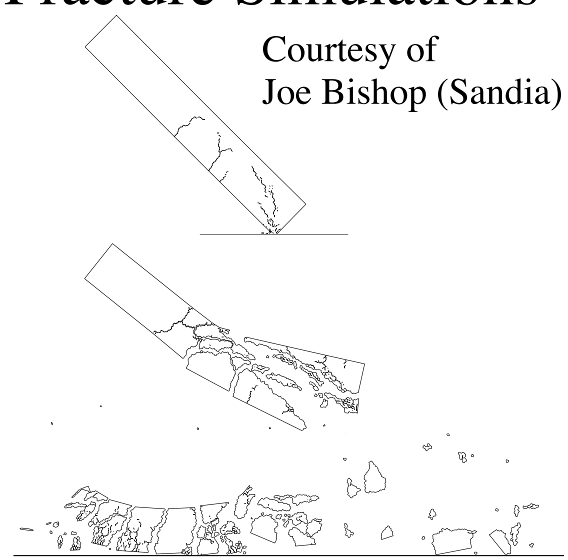


- **Properties**
  - **Random distribution**
    - **Without visible patterns, correlations**
    - **Blue noise spectrum**
  - **Separated-yet-dense**
    - **Efficient-yet-quality interpolation**
- **Graphics**
  - **texture synthesis**
- **Mesh generation**
  - **Random cracks, quality**
- **Design of computer exp.**
  - **high dimensions, 10-100**



Fracture Simulations

Courtesy of
Joe Bishop (Sandia)



Sandia National Laboratories

Maximal Poisson-Disk Sampling

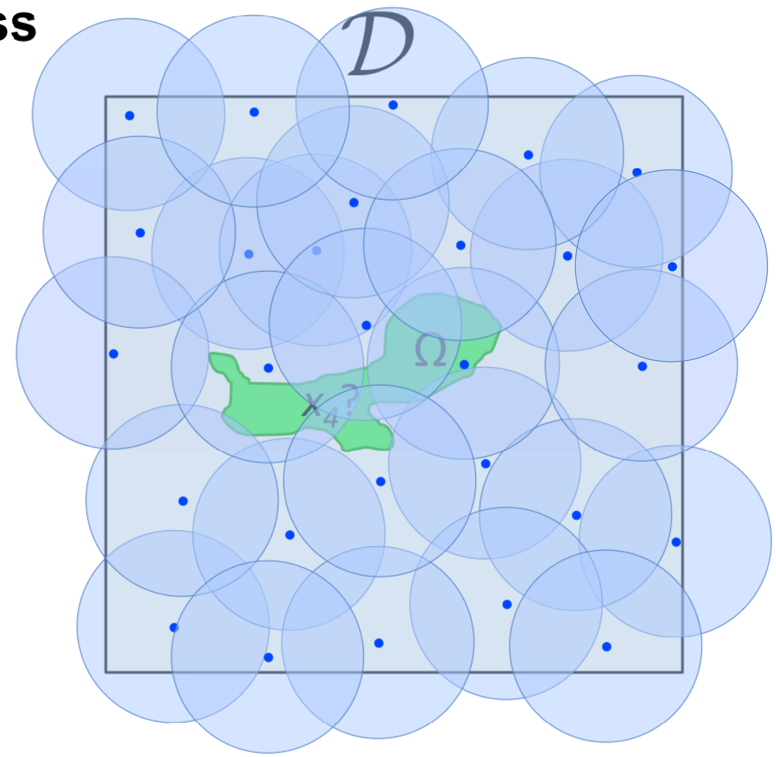
- **What is MPS?**
  - **Dart-throwing**
  - **Insert random points into a domain, build set X**
    - **With the "Poisson" process**

Empty disk: $\forall x_i, x_j \in X, x_i \neq x_j : ||x_i - x_j|| \geq r$

Bias-free: $\forall x_i \in X, \forall \Omega \subset \mathcal{D}_{i-1} :$

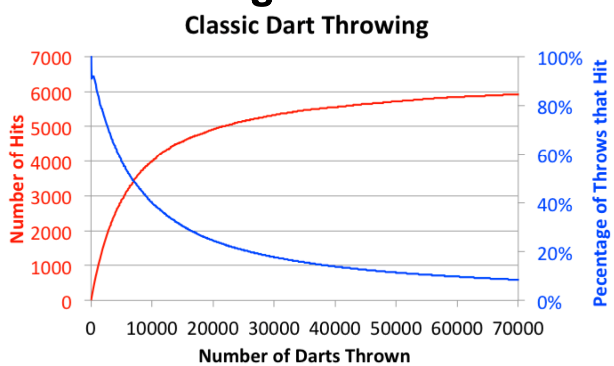
$$P\left(x_i \in \Omega\right) = \frac{\text{Area}(\Omega)}{\text{Area}(\mathcal{D}_{i-1})}$$

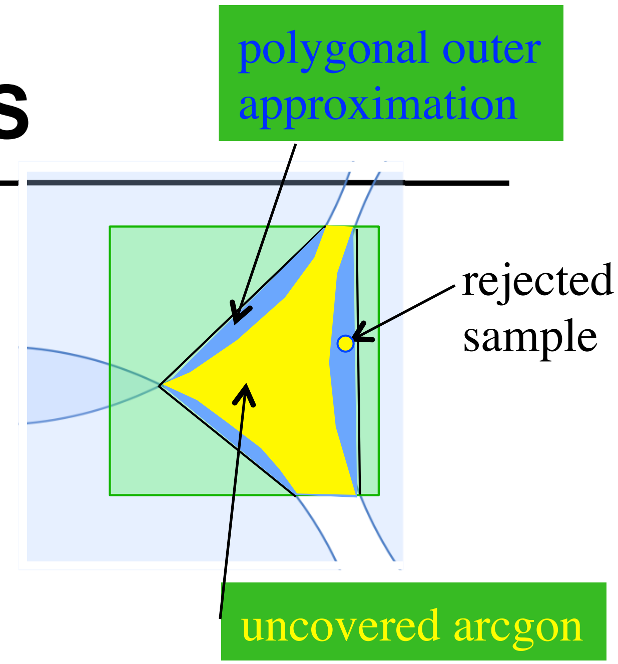
Maximal: $\forall x \in \mathcal{D}, \exists x_i \in X : ||x - x_i|| < r$



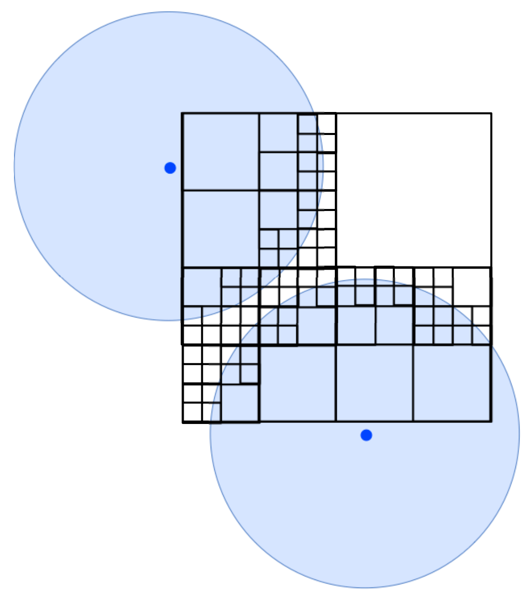
Algorithm for MPS

- **Classic algorithm**
  - **Throw a point, check if disk overlaps, keep/reject**
  - **Fast at first, but slows as uncovered area decreases**
    Can't get maximal.





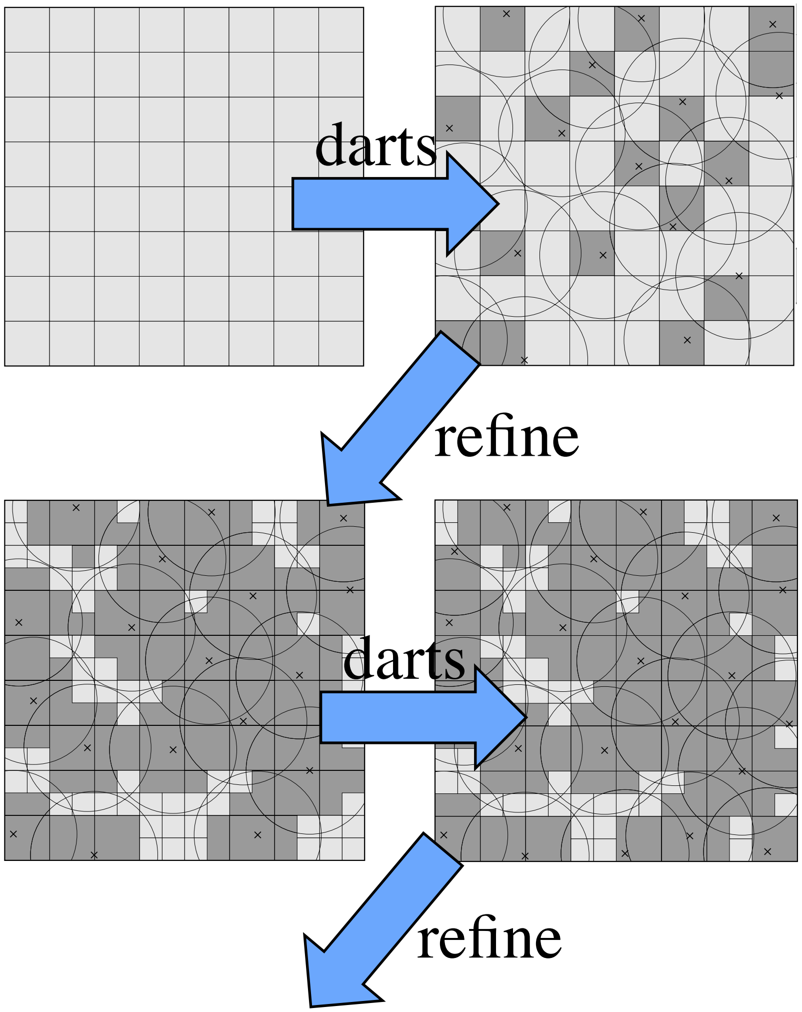
- **Speedup by targeting just the uncovered area**
  - **Polygons Ebeida et al. SIGGAPH 2011**
  - **Quadtrees to approximate the uncovered area**
    - **Discard covered squares**
    - **Uncovered squares: a sample is always acceptable**
    - **Partially covered squares: may need to refine**



Bias-free: $\forall x_i \in X, \forall \Omega \subset \mathcal{D}_{i-1}$:

$$P\left(x_i \in \Omega\right) = \frac{\text{Area}(\Omega)}{\text{Area}(\mathcal{D}_{i-1})}$$

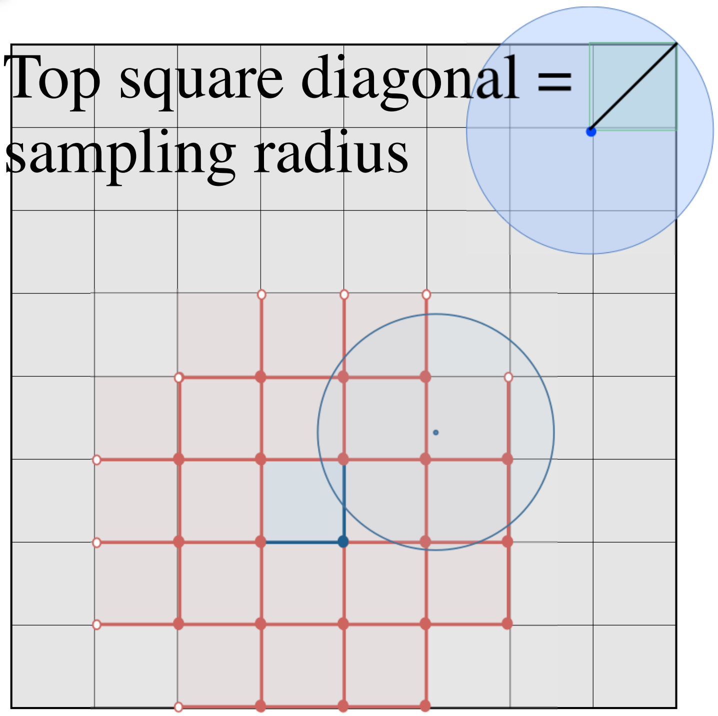
Our Algorithm - Basics



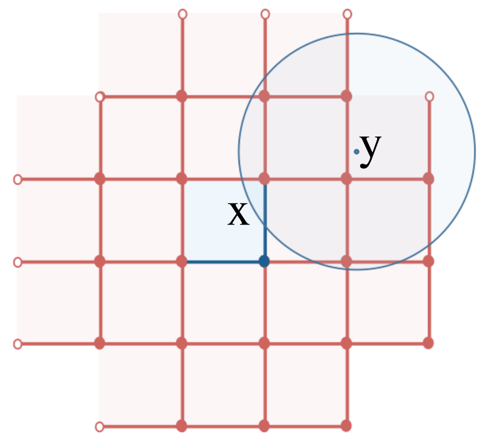
- **Datastructure:**
  - **Squares contain uncovered area**
- **Throw darts**
  - **Pick square, pick point in square**
  - **If dart is outside nearby circles**
    - **Accept dart as sample**
    - **Delete square**
- **Refine all squares**
  - **Discard subsquares covered by single disks**
- **Repeat**

Sandia National Laboratories

Datastructure: Quadtree Root



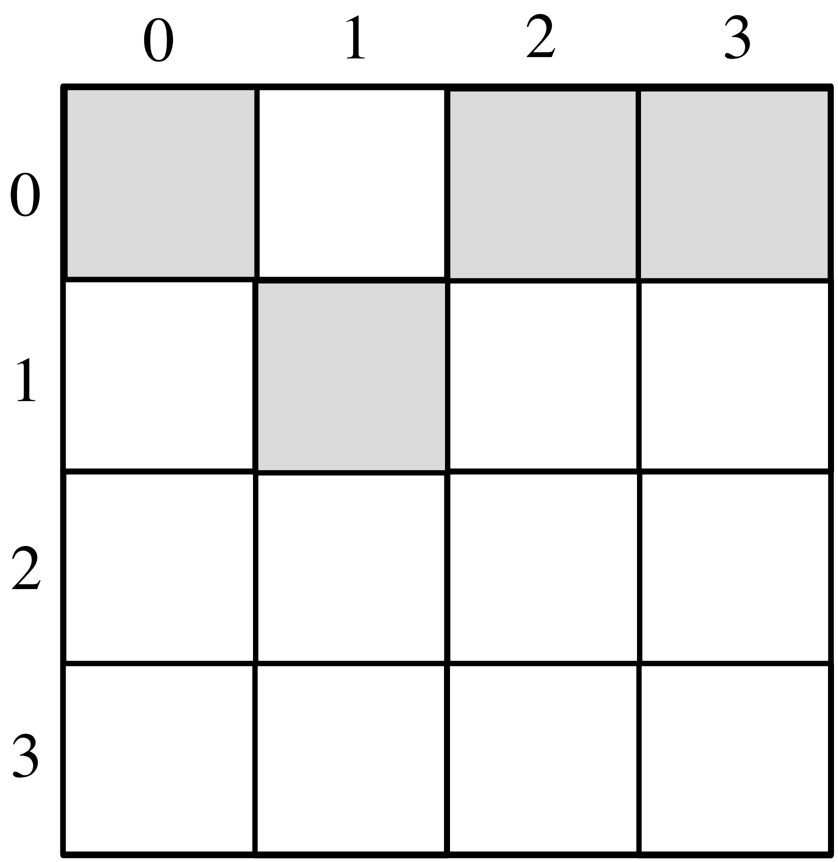
- Squares sized so
  - Can fit at most one sample
  - Nearby square template for "Point in disk?" conflict check
    - Pointer from square to its sample



Unpublished extension: use kd-tree for proximity…

Datastructure: Flat Quadtree Leaves

Flat: Only one level *i* is used at a time



- **Pool of squares**
  - **Global level *i***
  - **Squares that might accept a sample**
  - **Array of indices *C***

C^i

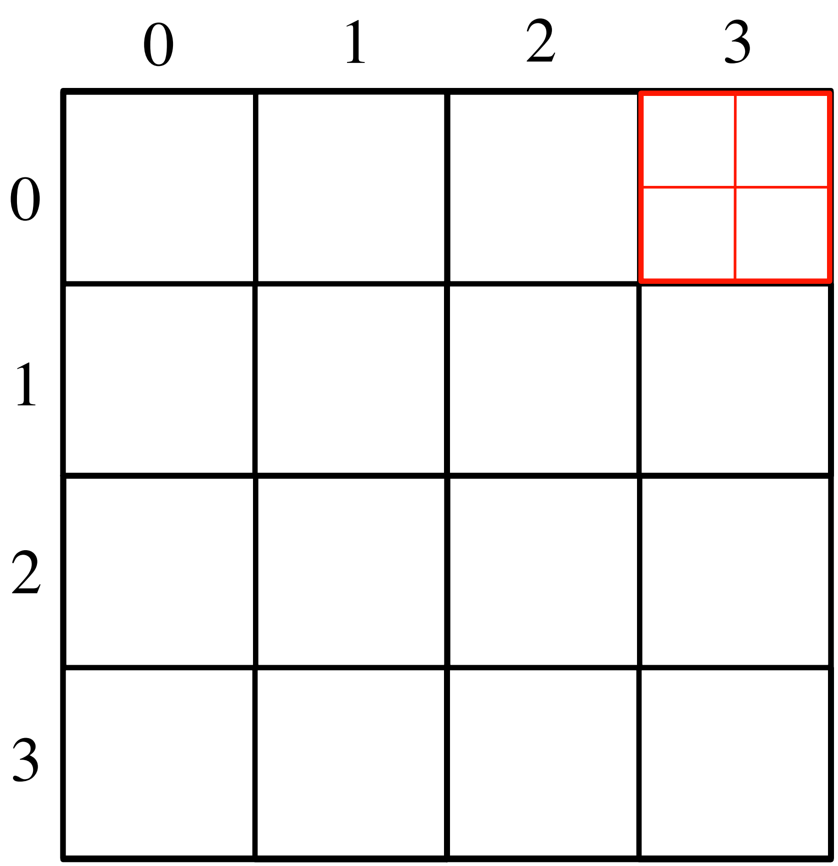
(0,0)
(0,2)
(0,3)
(1,1)
end
…

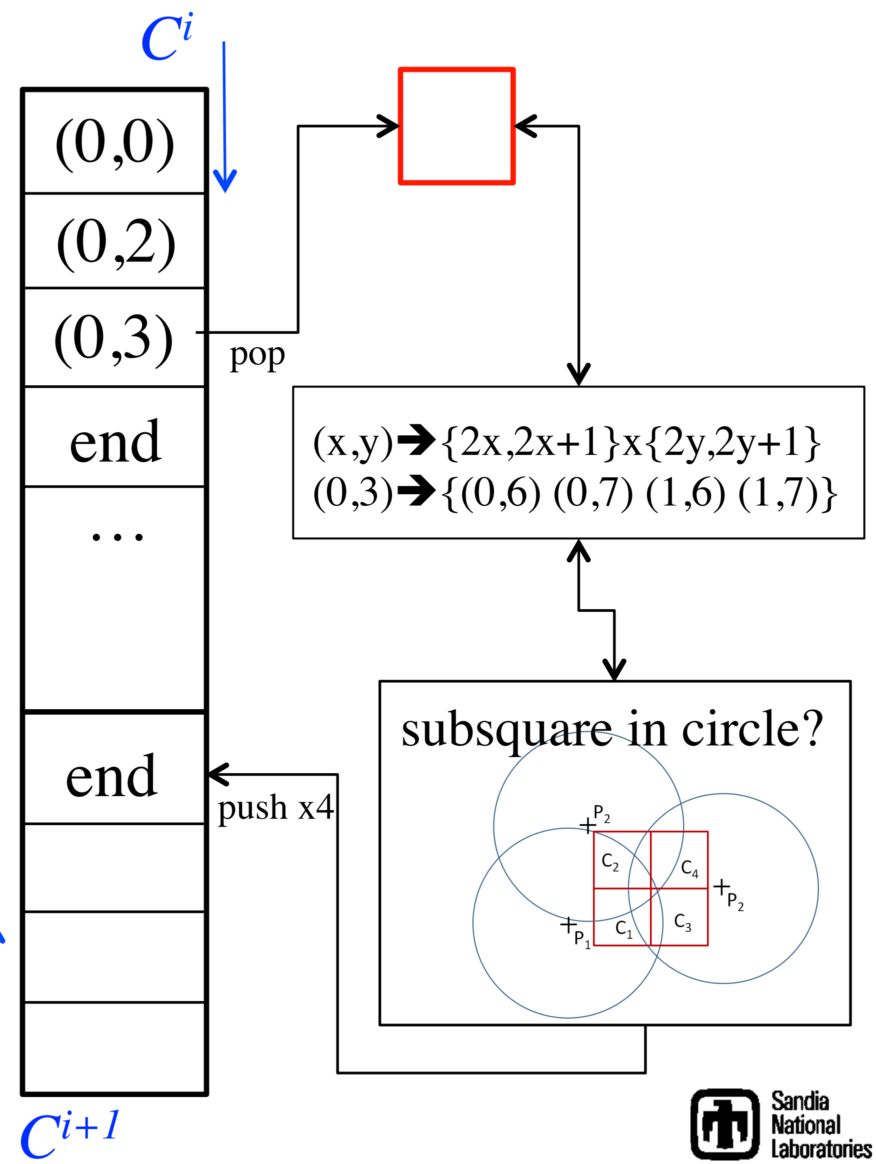
i=3
i.e. initial $\times 2^i$
squares per side

Flat Quadtree Refinement

Update in place.

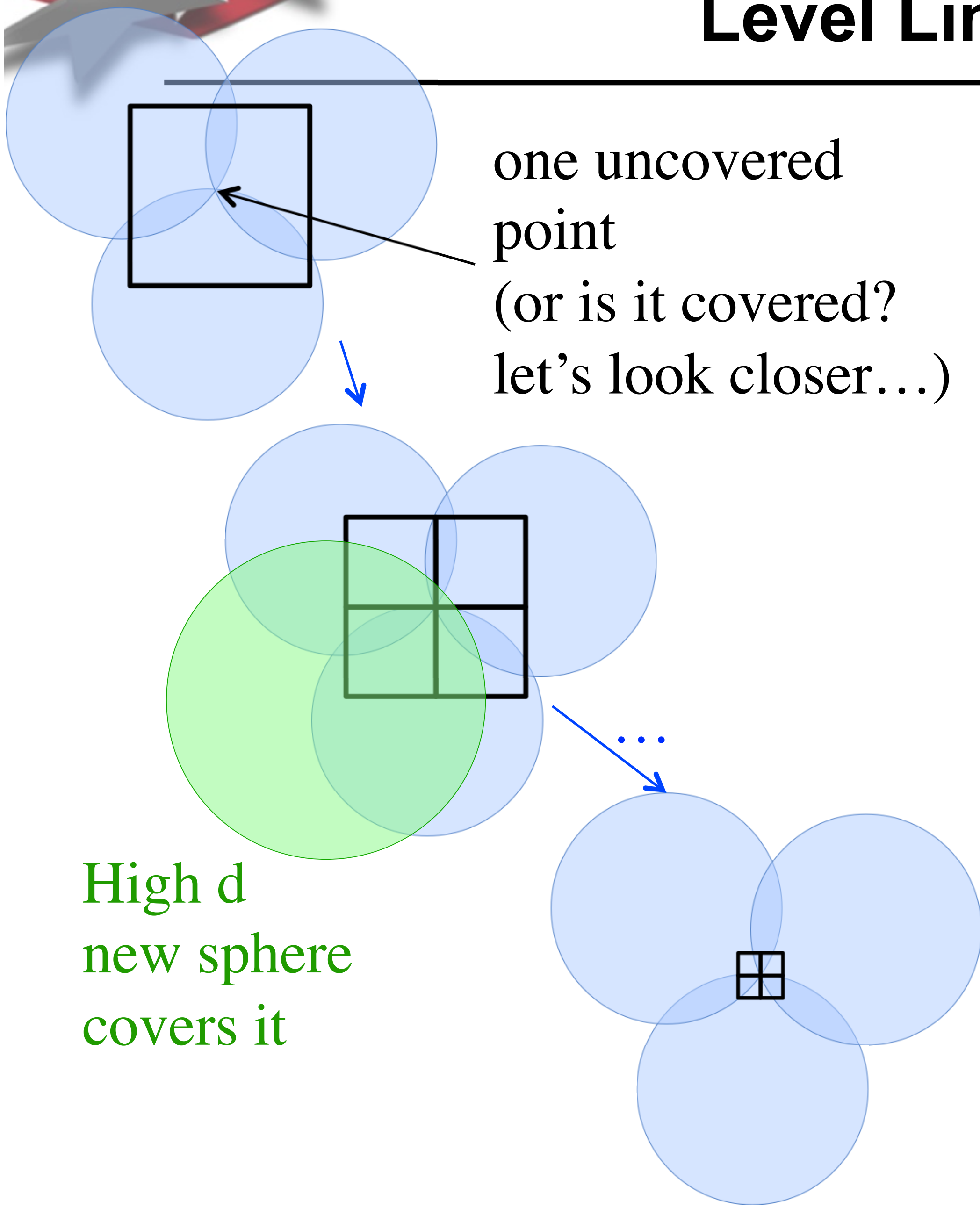
i++





Sandia National Laboratories

Level Limit?



- **Problem**
  - **Test if square in single circle**
  - **Small voids require infinite refinement**
- **Solution: [Wei08], [BWWM10**
  - **Stop early**
    to avoid memory blow-up
- **Solution: Us**
  - **Refine to finite-precision**
  - **Small voids happen rarely on average so**
  - **Memory is fine in practice**
  - **Benefit: maximal**

Algorithm – outer loop parameters

Algorithm 1 Simple MPS algorithm, CPU.

initialize $\mathcal{G}^o$, $i = 0$, $\mathcal{C}^i = \mathcal{G}^o$
while $|\mathcal{C}^i| > 0$ **do**
  {throw darts}
  for all $A|\mathcal{C}^i|$ (constant) dart throws **do**
    select an active cell $\mathcal{C}^i_c$ from $\mathcal{C}^i$ uniformly at random
    if $\mathcal{C}^i_c$'s parent base grid cell $\mathcal{G}^o_c$ has a sample **then**
      remove $\mathcal{C}^i_c$ from $\mathcal{C}^i$
    else
      throw candidate dart c into $\mathcal{C}^i_c$, uniform random
      if c is disk-free **then**
        {promote dart to sample}
        add c to $\mathcal{G}^o_c$ as an accepted sample p
        remove $\mathcal{C}^i_c$ from $\mathcal{C}^i$ {additional cells might be covered, but these are ignored for now}
      end if
    end if
  end for
  {iterate}
  for all active cells $\mathcal{C}^i$ **do**
    if $i < b$ subdivide $\mathcal{C}^i_c$ into 2^d subcells
    retain uncovered (sub)cells as $\mathcal{C}^{i+1}$
  end for
  increment i
end while

Tuning parameter choices: A, B

C^o = number initial cells

C^i = number current squares

How many throws before refining?

Throws = A | C^i |

How big does array C need to be to hold all the refined grid cells?

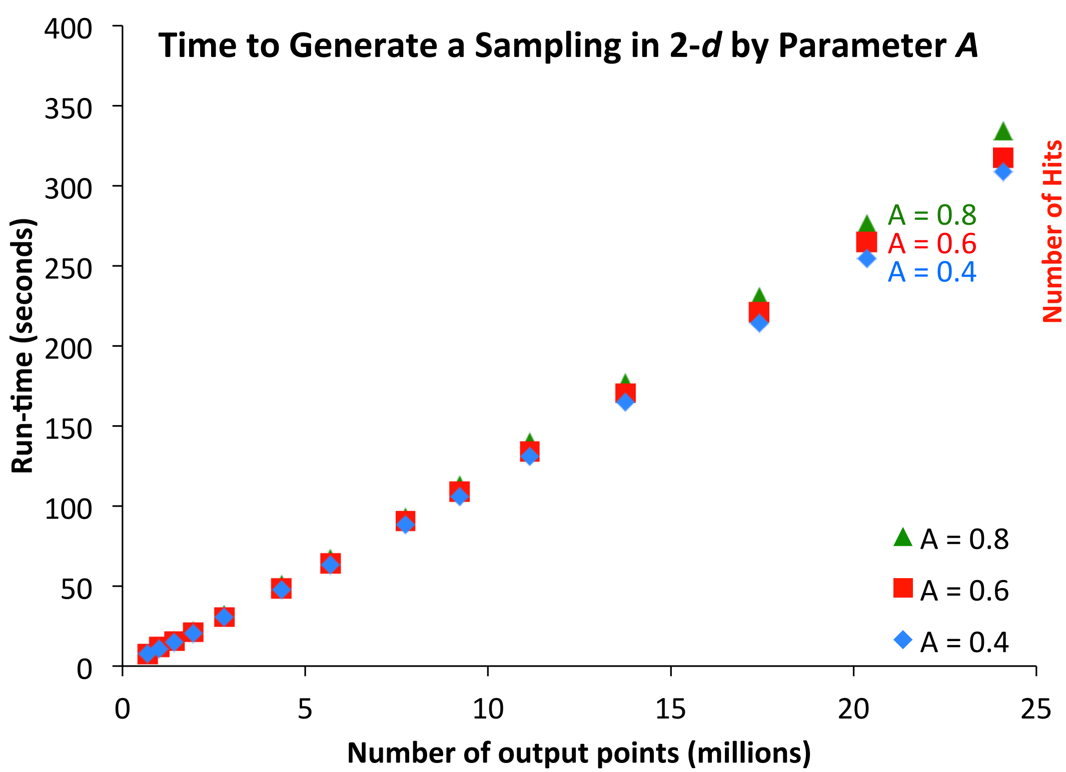
C = B | C^o |

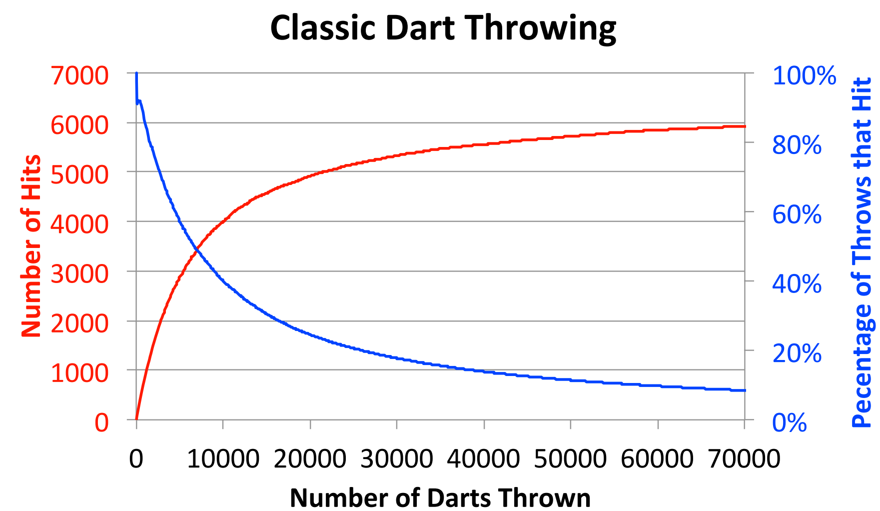
Big A ←→ more time, smaller memory B

Sandia National Laboratories

A (time) and B (memory) parameters

- **Big A ←→ more time, smaller memory B**
  - **A≈1, B≈dimension. (A increases for d>4)**
  - **Insensitive to value of A above a threshold**
    - **Intuition: as classical dart throwing, most hits happen early, no benefit to more throws**

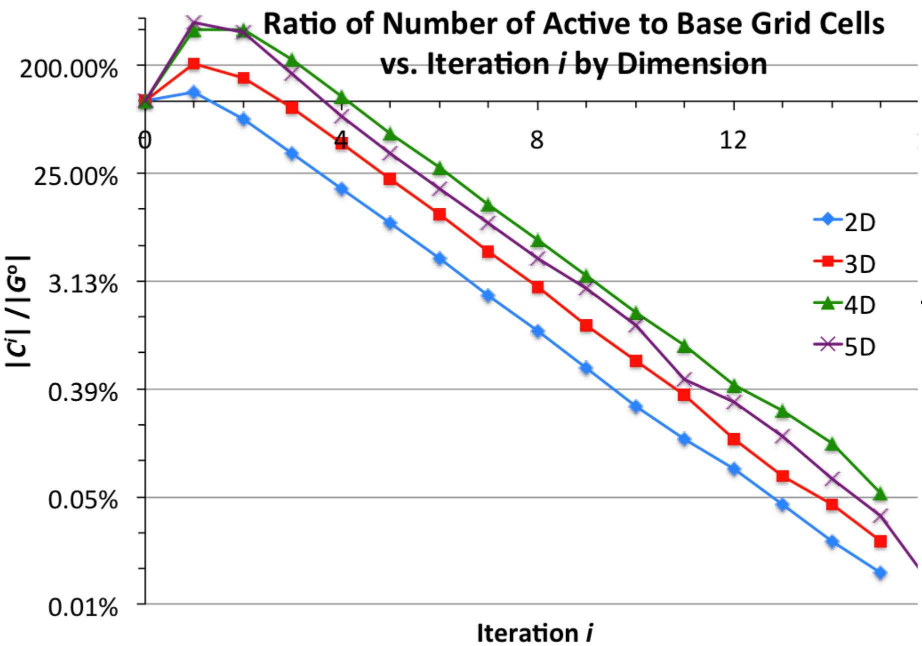


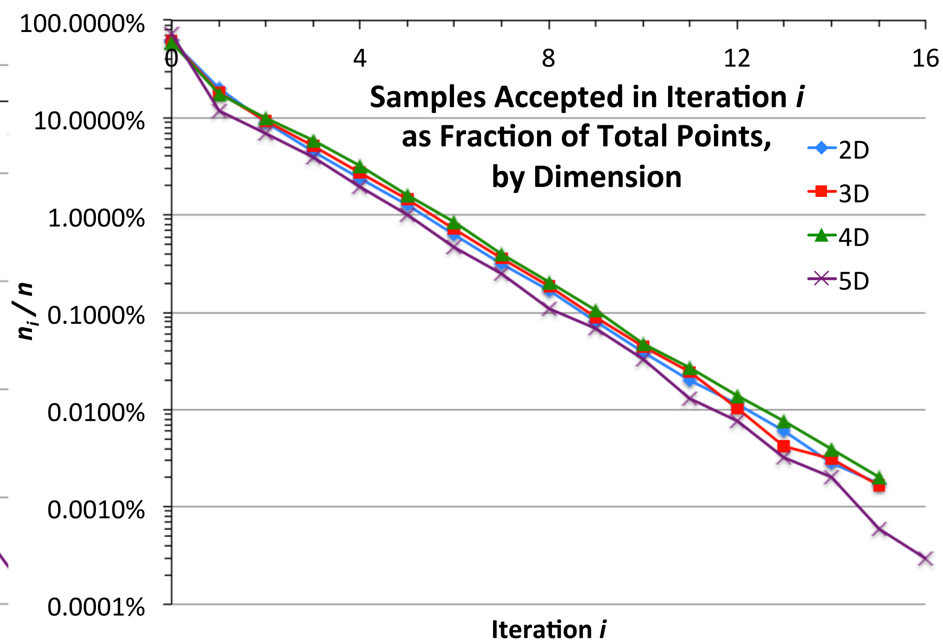


Sandia National Laboratories

Time and Memory
Experimental results

- **Memory and time peaks in early interations**
  - **Exponential convergence thereafter**
  - **Log y scale**





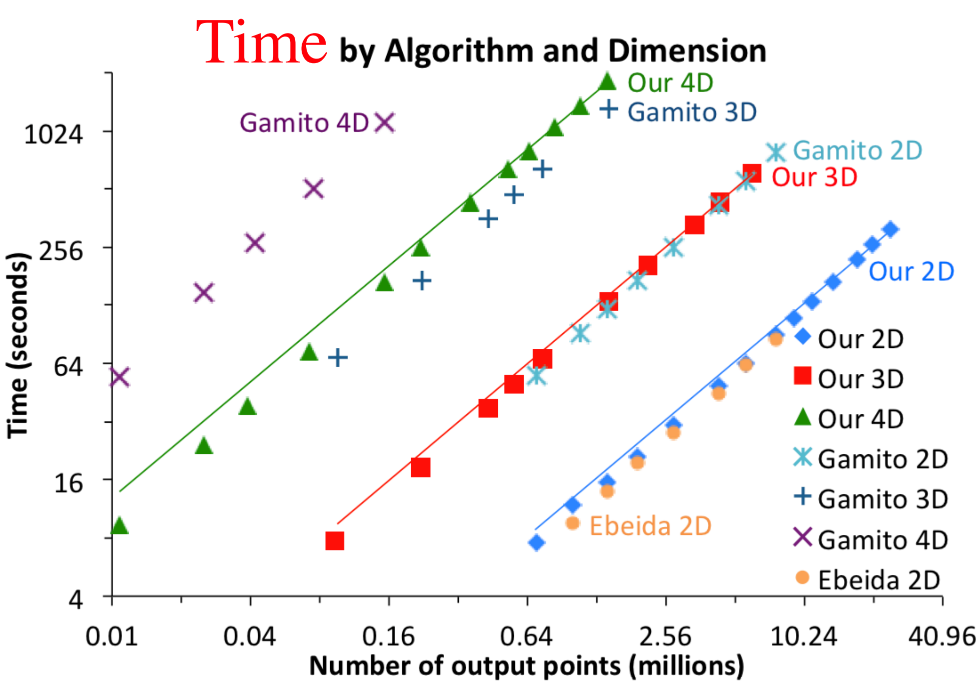
#boxes ≈ time, memory,

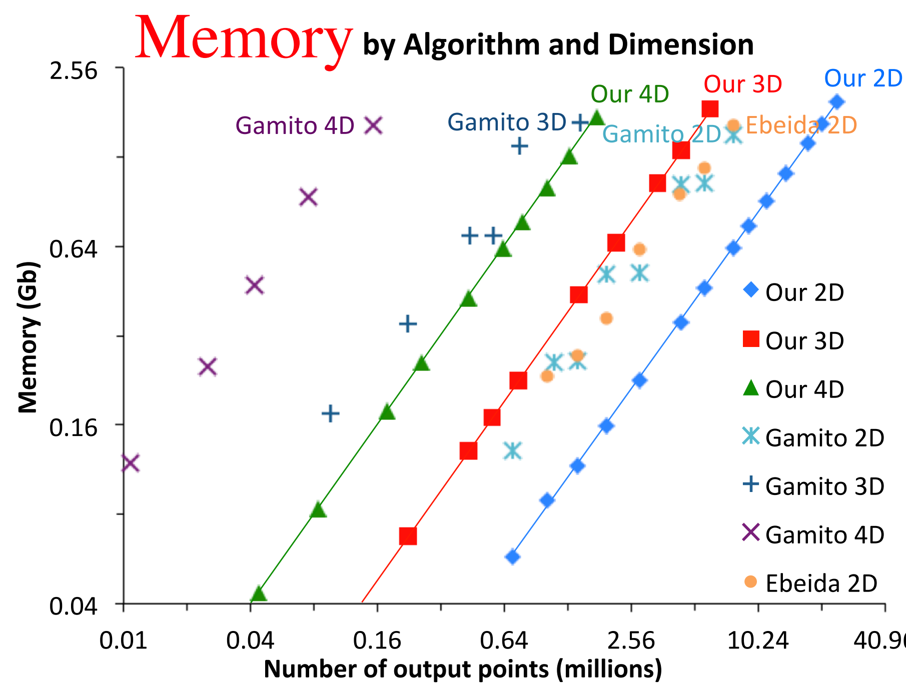
Time and Memory

vs. true quadtrees (Gamito), polygons (Ebeida 2D)

all linear in both, but constants matter

log-log scales





Memory savings from simpler datastructure

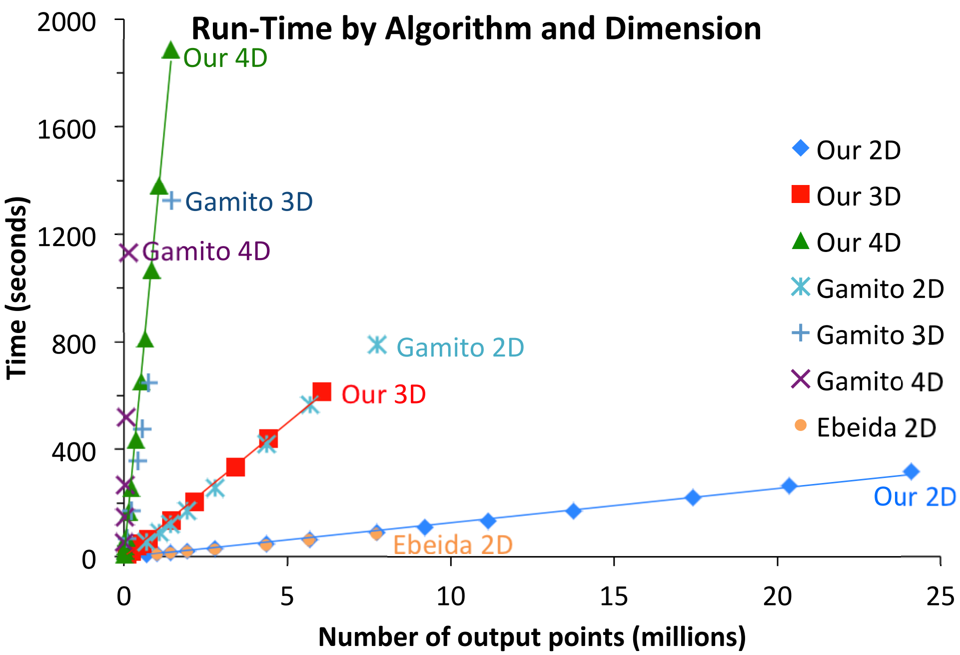
Time savings from that + simpler/fewer checks

Sandia National Laboratories

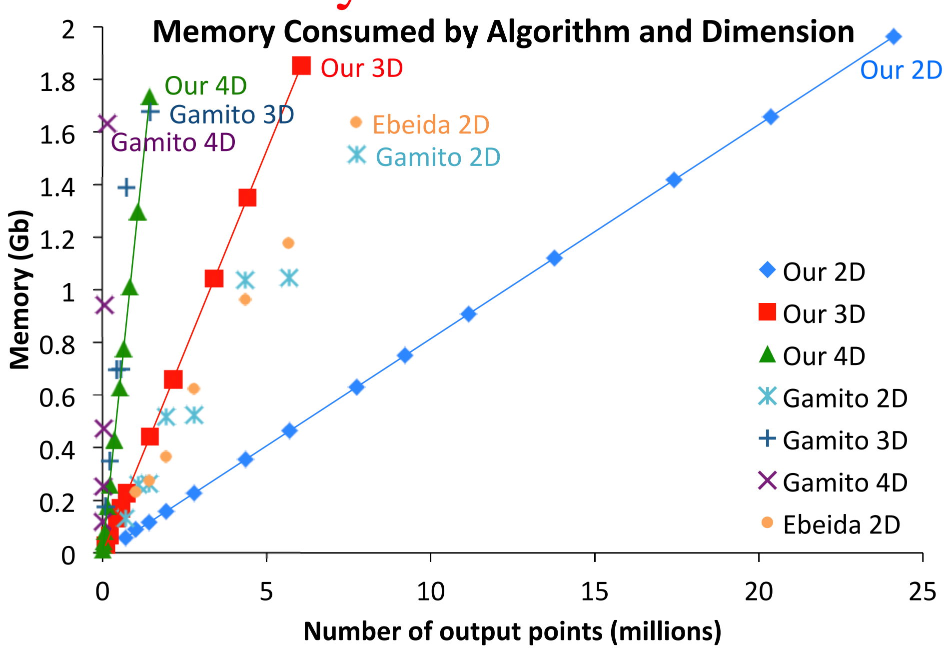
Time and Memory Theory

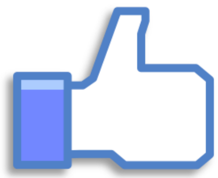
- **Run-time**
  - **Practice: linear in #points, grows by dimension**
  - **Proof: not available**
    - **Spatial statistics, expected area fraction of cells? And where?**
- **Memory**
  - **Linear in #points**
  - **No dynamic memory allocation**

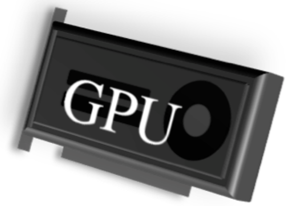
Time



Memory





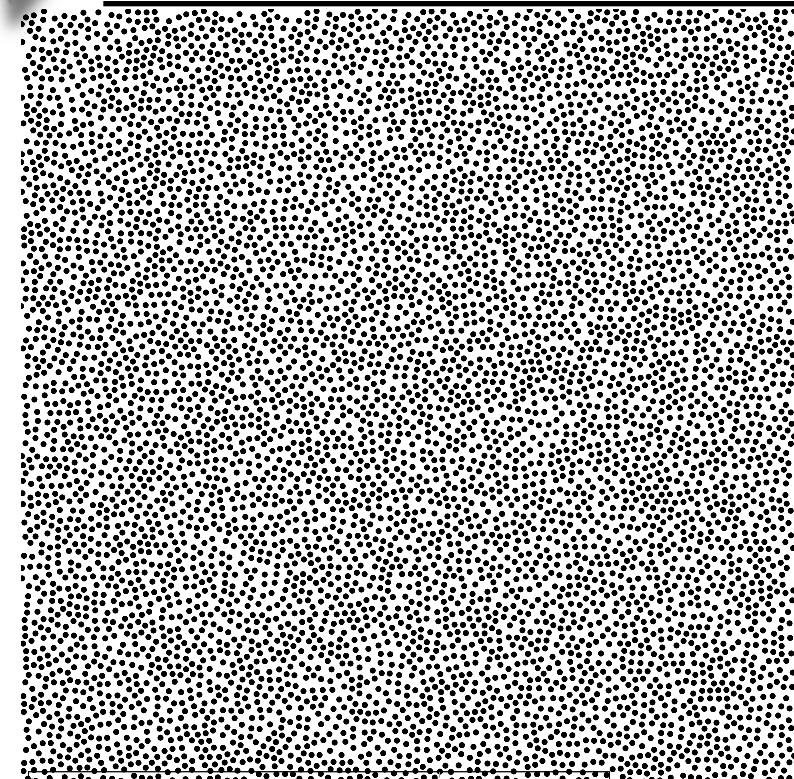


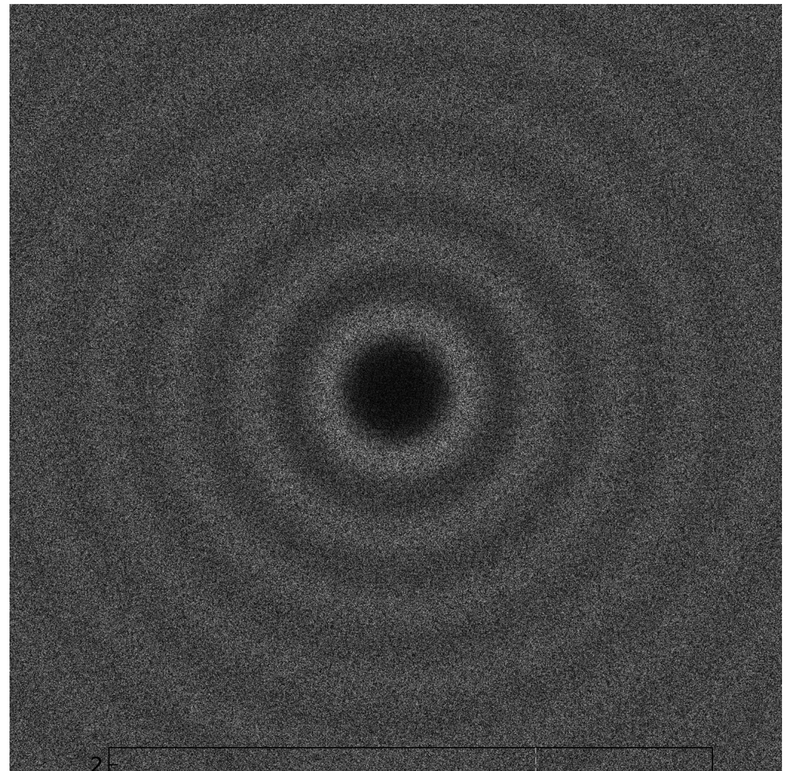
- Rejection sampling is great on a GPU
  - Nothing to communicate for a dart miss!

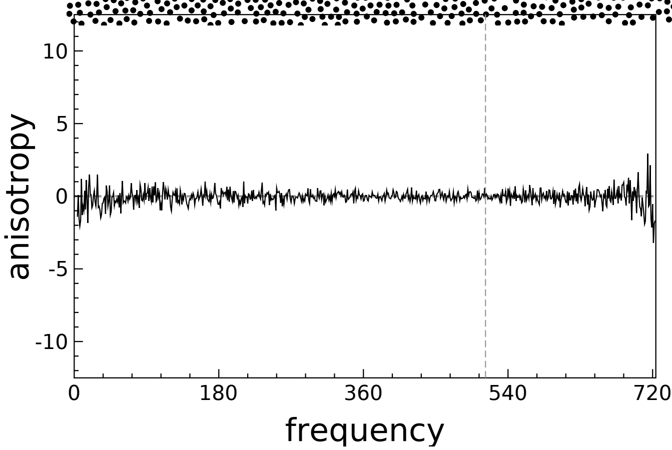
- 10x speedup on NVIDIA GTX 460
  - Memory-limited to 600k points 2d, 200k in 3d

Point Cloud Quality?

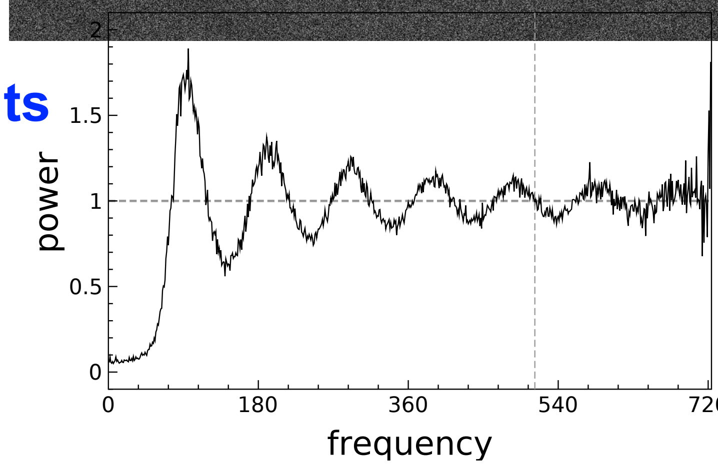
Provably correct bias-free, maximal up to precision





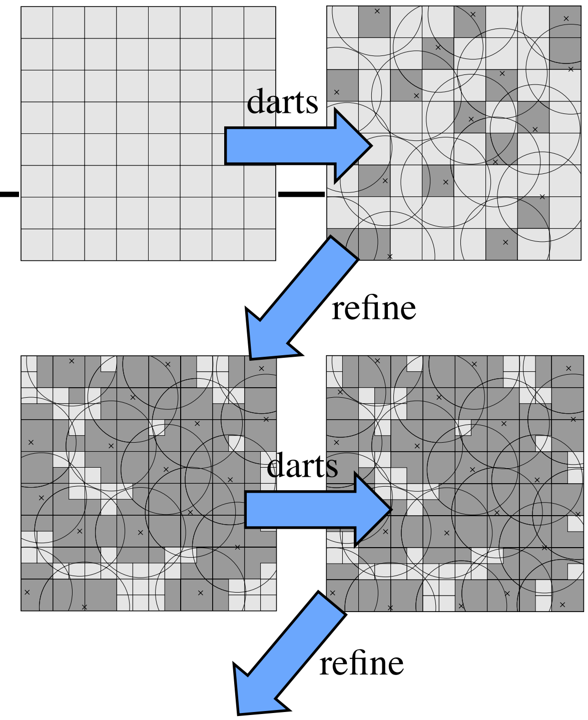


Experiments confirm (GPU)



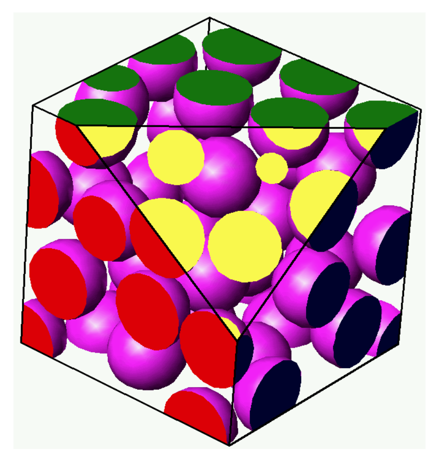
Sandia National Laboratories

Conclusions



- MPS Maximal Poisson-disk sampling
  - Simpler, faster, less memory
  - Three simple ideas
    - Flat quadtree
    - Constant # throws / ignore misses
    - Global refinement
  - CPU and GPU

Reviewer #0: "The paper is yet another one about faster Poisson-sampling, but I see that it is significantly faster, uses less memory, is just simpler, easier to implement, and works well for higher dimensions."



- Future, dimensions > 4?
  - Not so great, quadtrees too big
- Two bonus thoughts…

UCDAVIS UNIVERSITY OF CALIFORNIA

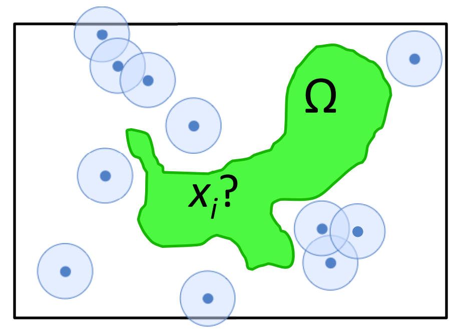
Sandia National Laboratories

Two bonus thoughts

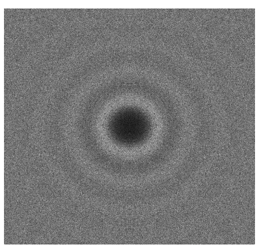
“Unbiased” Opinion

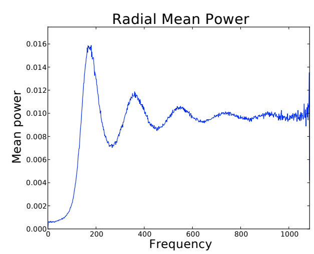
- Unbiased as a description of (serial) process
  - insertion probability independent of location

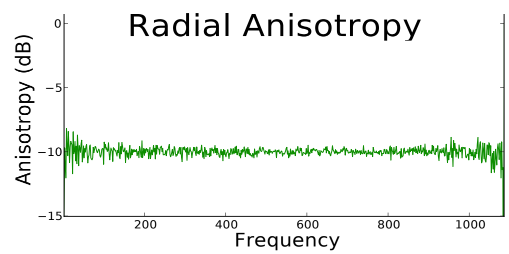
$$P(x_i \in \Omega) \propto \text{Area}(\Omega)$$



- Unbaised as a description of outcome
  - pairwise distance spectra, blue noise





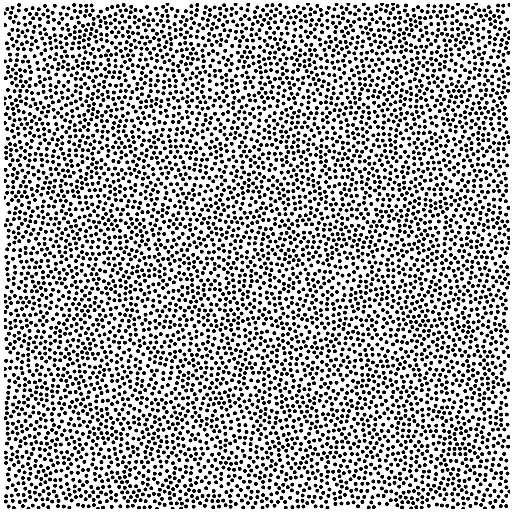


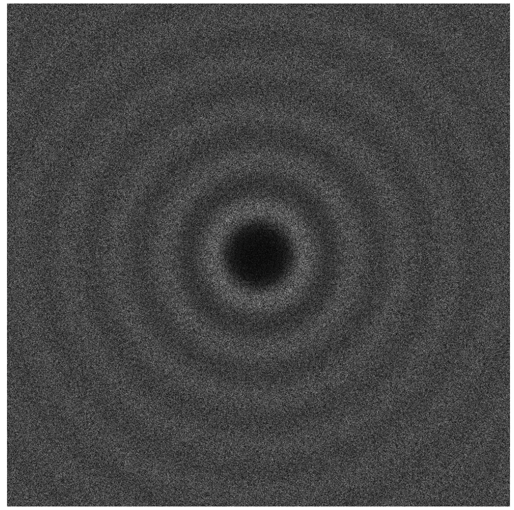
PSA code great for standard pictures

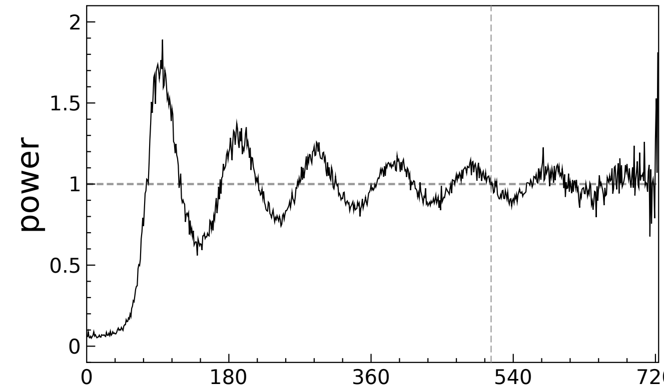
- Unbiased process leads to unbiased outcome, but so might other processes
  - Opinion: need something beyond “viewgraph norm”
  - Need metrics for “how unbiased is it”
    - Define spectrum S that is the limit distribution of unbiased sampling, and standard deviations.
    - Our process generated S’, and |S-S’| < 0.4 std dev (S)

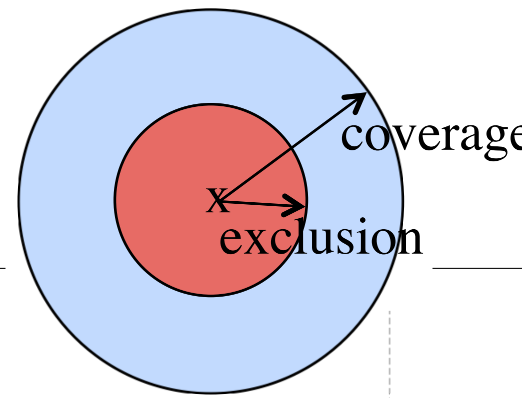
What is the real goal?

- **Classic MPS – a lot of effort to get maximal**

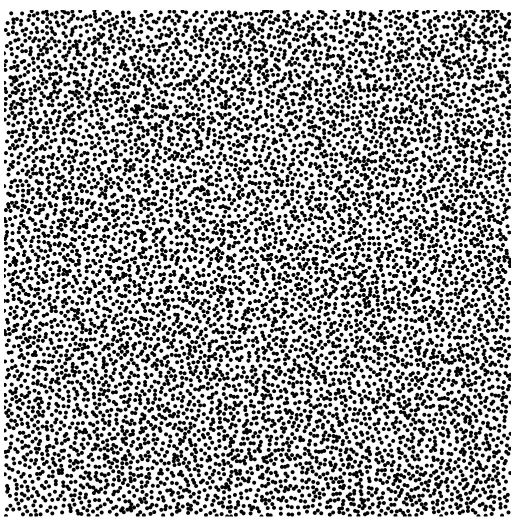


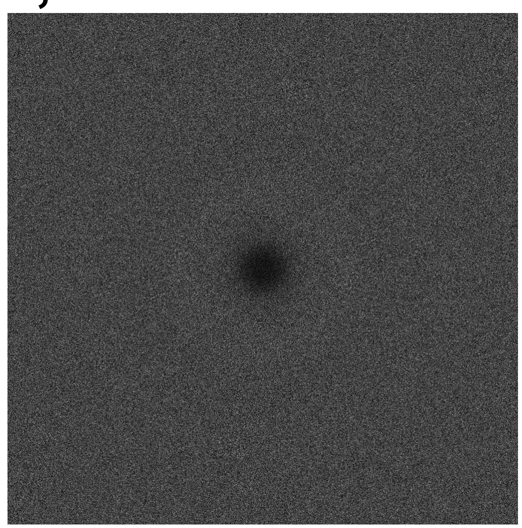


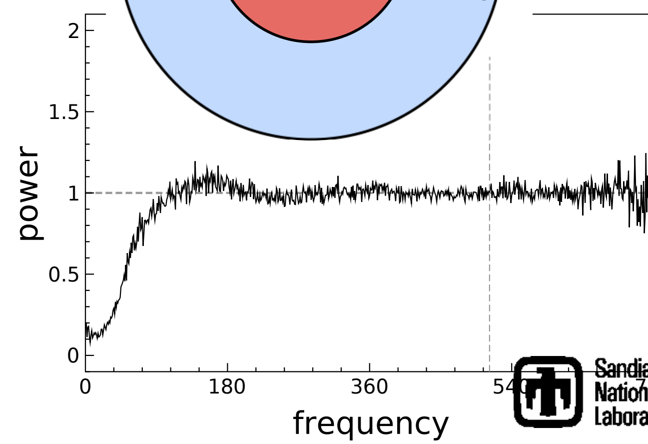




- **Two-radii MPS, submitted to CCCG**







Sandia National Laboratories