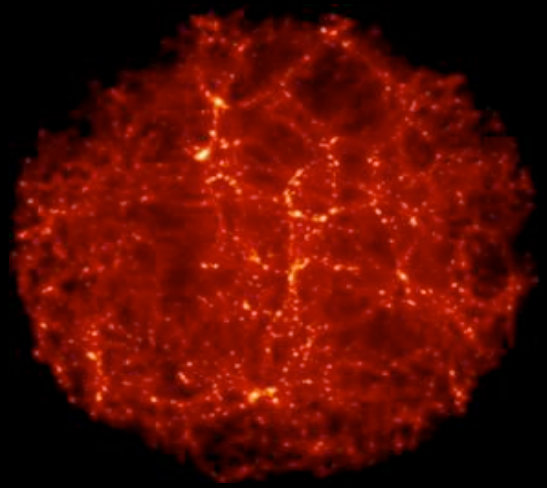
Recent Results in Astrostatistics



Alexander Gray

Georgia Institute of Technology

College of Computing

FASTlab: Fundamental Algorithmic and Statistical Tools Laboratory

The FASTlab

Fundamental **A**lgorithmic and **S**tatistical **T**ools Laboratory

www.fast-lab.org

1. Arkadas Ozakin: **Research scientist**, Math, Physics; PhD Physics
2. Nikolaos Vasiloglou: **Visiting scholar**, EE; PhD EE
3. Abhimanyu Aditya: **Affiliate**, CS; MS CS

4. Dong Ryeol Lee: **PhD student**, CS + Math
5. Ryan Riegel: **PhD student**, CS + Math
6. Sooraj Bhat: **PhD student**, CS
7. Wei Guan: **PhD student**, CS
8. Nishant Mehta: **PhD student**, CS
9. Parikshit Ram: **PhD student**, CS + Math
10. William March: **PhD student**, Math + CS
11. Hua Ouyang: **PhD student**, CS
12. Ravi Sastry: **PhD student**, CS
13. Long Tran: **PhD student**, CS
14. Ryan Curtin: **PhD student**, EE
15. Ailar Javadi: **PhD student**, EE

+ 5-10 **MS students**

The FASTlab

Fundamental **A**lgorithmic and **S**tatistical **T**ools Laboratory

www.fast-lab.org

1. Arkadas Ozakin: **Research scientist**, Math, Physics; PhD Physics
2. Nikolaos Vasiloglou: **Visiting scholar**, EE; PhD EE
3. Abhimanyu Aditya: **Affiliate**, CS; MS CS

4. Dong Ryeol Lee: **PhD student**, CS + Math
5. Ryan Riegel: **PhD student**, CS + Math
6. Sooraj Bhat: **PhD student**, CS
7. Wei Guan: **PhD student**, CS
8. Nishant Mehta: **PhD student**, CS
9. Parikshit Ram: **PhD student**, CS + Math
10. William March: **PhD student**, Math + CS
11. Hua Ouyang: **PhD student**, CS
12. Ravi Sastry: **PhD student**, CS
13. Long Tran: **PhD student**, CS
14. Ryan Curtin: **PhD student**, EE
15. Ailar Javadi: **PhD student**, EE

+ 5-10 **MS students**

Recent Results in Astrostatistics

1. **Fast algorithms:** n-point, friends-of-friends, theory
2. **Software:** databases
3. **Statistical methodology:** new ML methods, ML with measurement errors

Core methods of statistics / machine learning / mining

- **Querying:** spherical range-search **O(N),** orthogonal range-search **O(N),** spatial join **O(N^2),** nearest-neighbor **O(N),** all-nearest-neighbors **O(N^2)**
- **Density estimation:** mixture of Gaussians, kernel density estimation **O(N^2),** kernel conditional density estimation **O(N^3)**
- **Regression:** linear regression, kernel regression **O(N^2),** Gaussian process regression **O(N^3)**
- **Classification:** decision tree, nearest-neighbor classifier **O(N^2)**, nonparametric Bayes classifier **O(N^2)**, support vector machine **O(N^3)**
- **Dimension reduction:** principal component analysis, non-negative matrix factorization, kernel PCA **O(N^3)**, maximum variance unfolding **O(N^3)**
- **Outlier detection:** by density estimation or dimension reduction
- **Clustering:** by density estimation or dimension reduction, k-means, mean-shift segmentation **O(N^2),** hierarchical (FoF) clustering **O(N^3)**
- **Time series analysis:** Kalman filter, hidden Markov model, trajectory tracking **O(N^n)**
- **Feature selection and causality:** LASSO, L_1 SVM, Gaussian graphical models, discrete graphical models
- **2-sample testing and matching:** bipartite matching **O(N^3),** n-point correlation **O(N^n)**

Now pretty fast…

- **Querying:** spherical range-search **O(logN)*,** orthogonal range-search **O(logN)*,** spatial join **O(N)*,** nearest-neighbor **O(logN),** all-nearest-neighbors **O(N)**
- **Density estimation:** mixture of Gaussians, kernel density estimation **O(N),** kernel conditional density estimation **$O(N^{log^3})$***
- **Regression:** linear regression, kernel regression **O(N),** Gaussian process regression **O(N)***
- **Classification:** decision tree, nearest-neighbor classifier **O(N)**, nonparametric Bayes classifier **O(N)***, support vector machine **O(N)/$O(N^2)$**
- **Dimension reduction:** principal component analysis, non-negative matrix factorization, kernel PCA **O(N)***, maximum variance unfolding **O(N)***
- **Outlier detection:** by density estimation or dimension reduction
- **Clustering:** by density estimation or dimension reduction, k-means, mean-shift segmentation **O(N),** hierarchical (FoF) clustering **O(NlogN)**
- **Time series analysis:** Kalman filter, hidden Markov model, trajectory tracking **$O(N^{logn})$***
- **Feature selection and causality:** LASSO, L_1 SVM, Gaussian graphical models, discrete graphical models
- **2-sample testing and matching:** bipartite matching **O(N)**,** n-point correlation **$O(N^{logn})$***

4 main computational bottlenecks:

N-body, graphical models, linear algebra, optimization

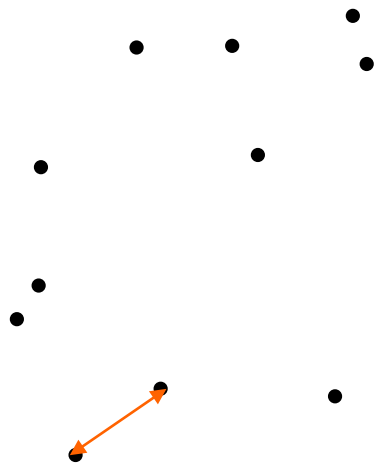
- **Querying:** spherical range-search **O(N),** orthogonal range-search **O(N),** spatial join **O(N^2),** nearest-neighbor **O(N),** all-nearest-neighbors **O(N^2)**
- **Density estimation:** mixture of Gaussians, kernel density estimation **O(N^2),** kernel conditional density estimation **O(N^3)**
- **Regression:** linear regression, kernel regression **O(N^2),** Gaussian process regression **O(N^3)**
- **Classification:** decision tree, nearest-neighbor classifier **O(N^2),** nonparametric Bayes classifier **O(N^2),** support vector machine **O(N^3)**
- **Dimension reduction:** principal component analysis, non-negative matrix factorization, kernel PCA **O(N^3),** maximum variance unfolding **O(N^3)**
- **Outlier detection:** by density estimation or dimension reduction
- **Clustering:** by density estimation or dimension reduction, k-means, mean-shift segmentation **O(N^2),** hierarchical clustering **O(N^3)**
- **Time series analysis:** Kalman filter, hidden Markov model, trajectory tracking **O(N^n)**
- **Feature selection and causality:** LASSO, L_1 SVM, Gaussian graphical models, discrete graphical models
- **2-sample testing and matching:** bipartite matching **O(N^3),** n-point correlation **O(N^n)**

Characterization of an entire distribution?

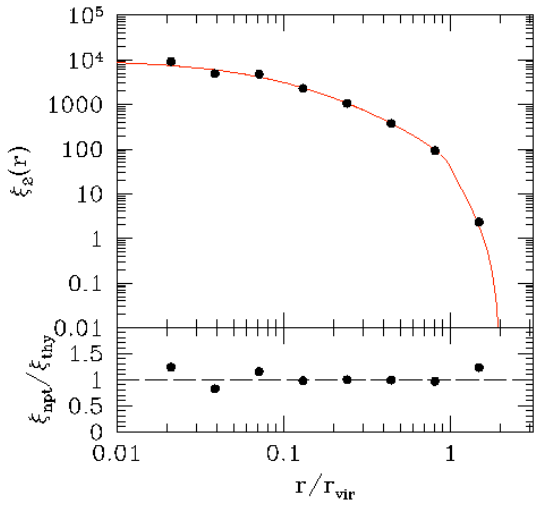
2-point correlation

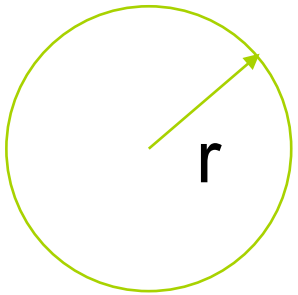
"How many pairs have distance < r ?"

$$\sum_{i}^{N}\sum_{j\neq i}^{N} I(\|x_i - x_j\| < r)$$



2-point correlation function





The *n*-point correlation functions

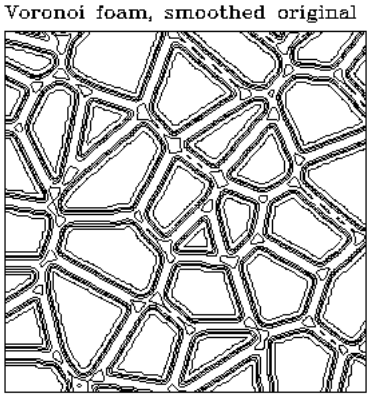
- Spatial inferences: filaments, clusters, voids, homogeneity, isotropy, 2-sample testing, …
- Foundation for theory of point processes [Daley,Vere-Jones 1972], unifies spatial statistics [Ripley 1976]
- Used heavily in biostatistics, cosmology, particle physics, statistical physics

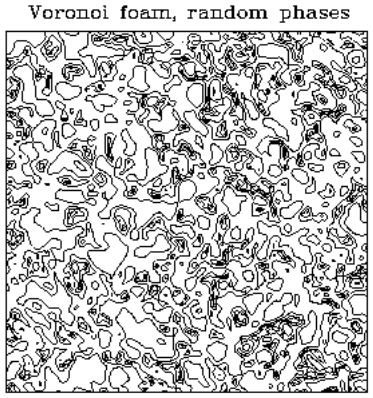
2pcf definition:

$$dP = \lambda^2 dV_1 dV_2 [1 + \xi(r)]$$

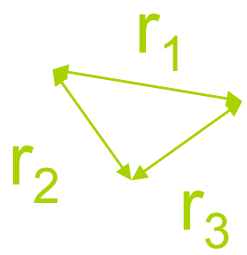
3pcf definition:

$$dP = \lambda^3 dV_1 dV_2 dV_3 \cdot [1 + \xi(r_{12}) + \xi(r_{23}) + \xi(r_{13}) + \zeta(r_{12}, r_{23}, r_{13})]$$



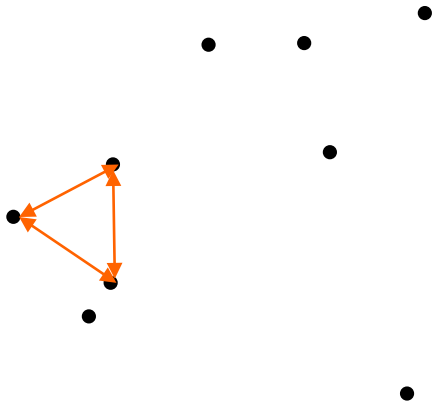


Standard model: n>0 terms should be zero!



3-point correlation

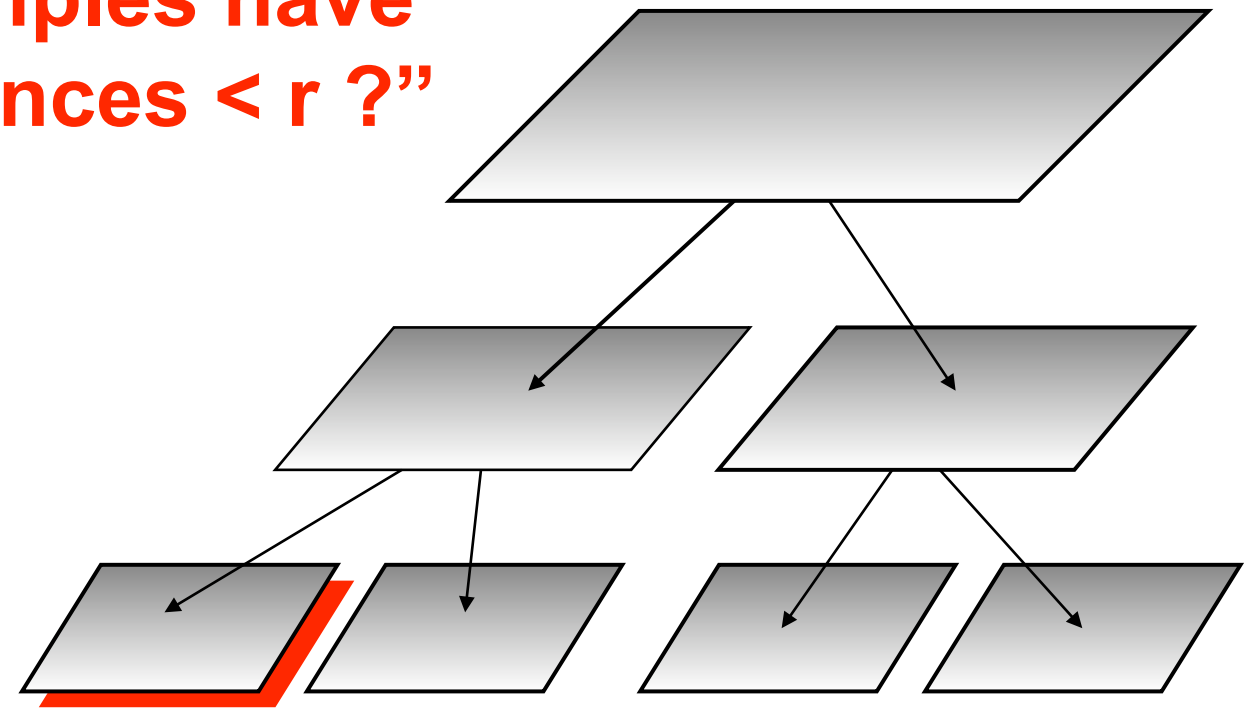
"How many triples have pairwise distances < r ?"



$$\sum_{i}^{N}\sum_{j\neq i}^{N}\sum_{k\neq j\neq i}^{N} I(\delta_{ij} < r_1) I(\delta_{jk} < r_2) I(\delta_{ki} < r_3)$$

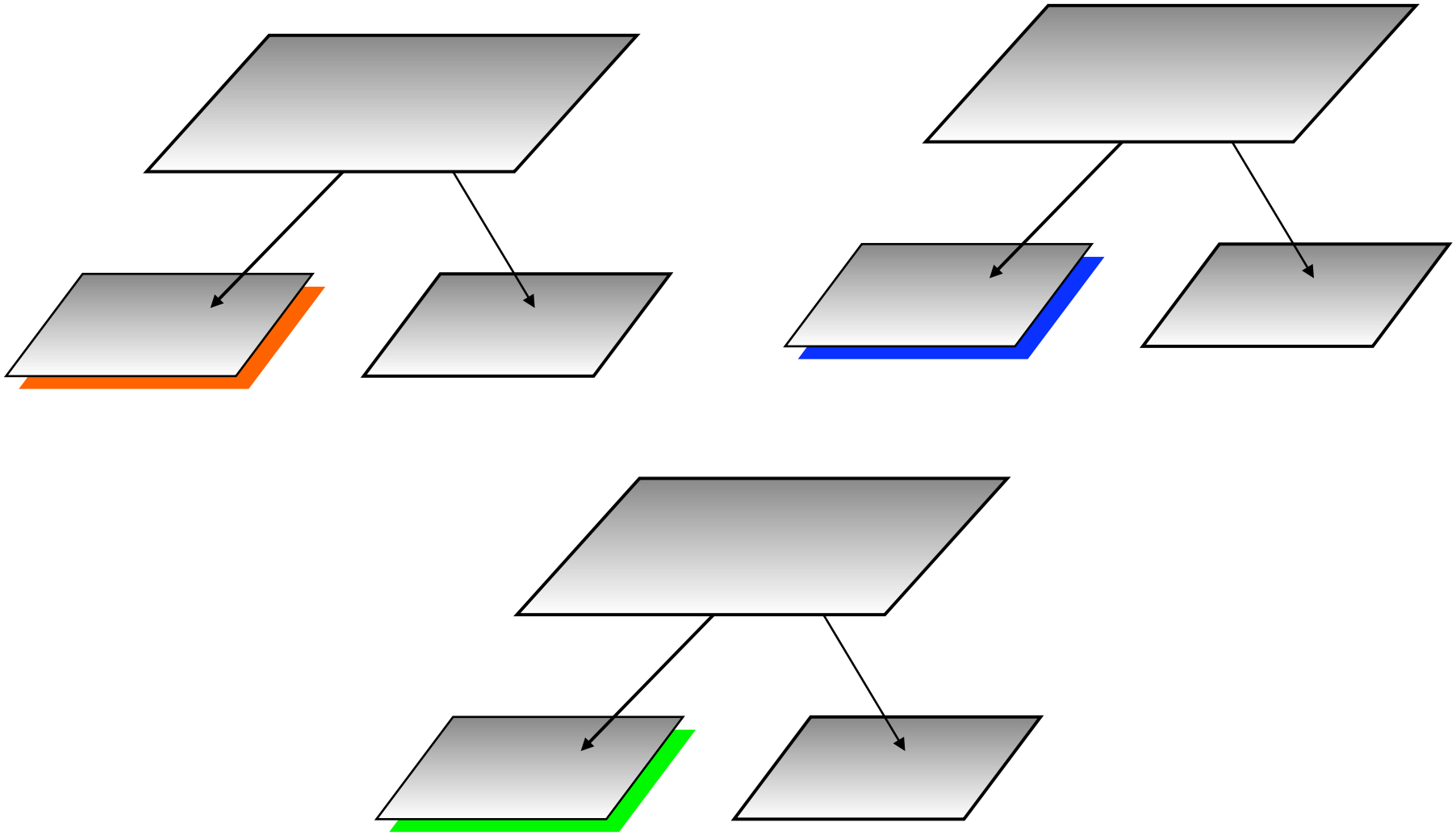
How can we count *n*-tuples efficiently?

"How many triples have pairwise distances < r ?"

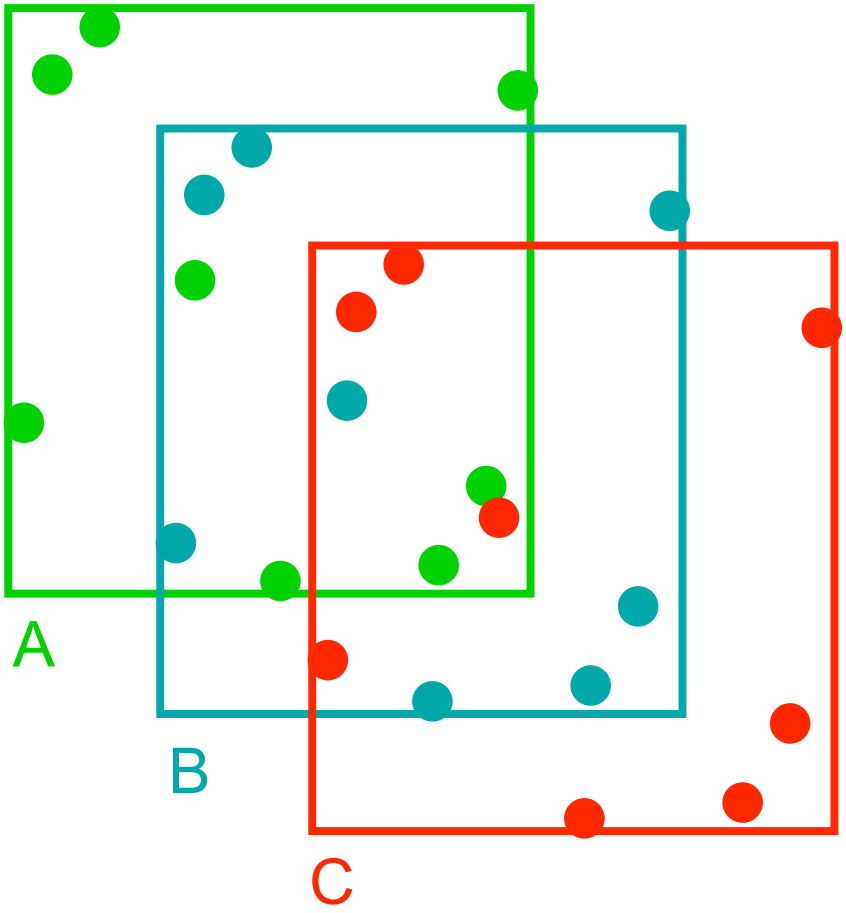


Use *n* trees!

[Gray & Moore, NIPS 2000]



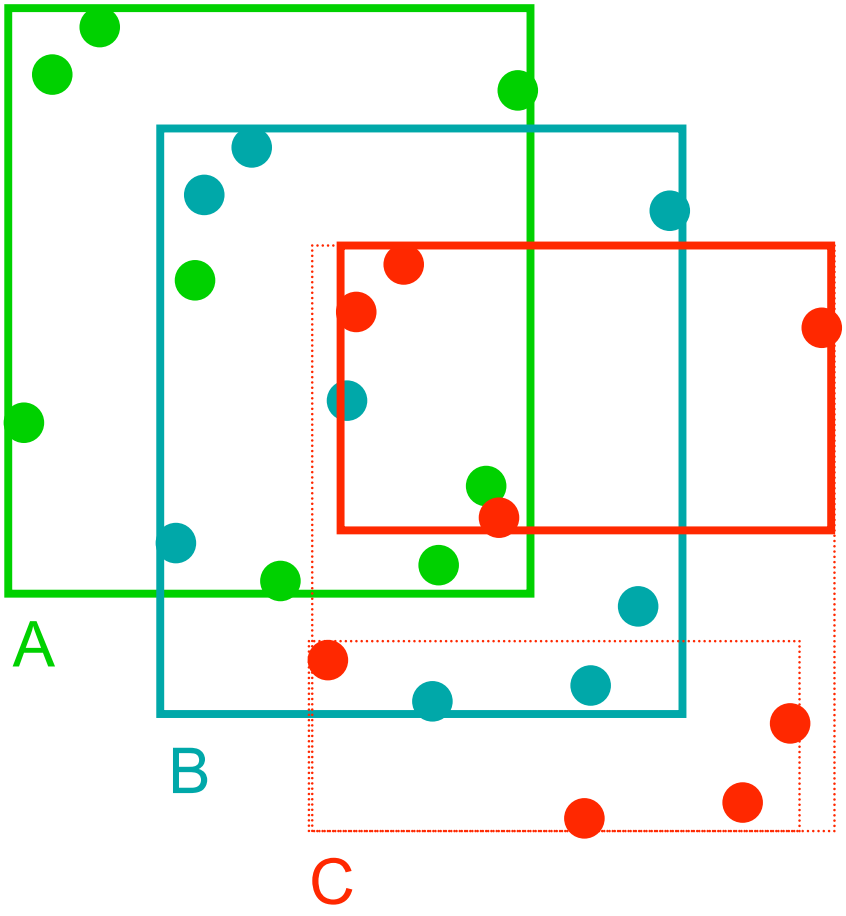
“How many valid triangles a-b-c (where $a \in A, \quad b \in B, \quad c \in C$) could there be?



count{**A**,**B**,**C**} =

?

"How many valid triangles a-b-c (where $a \in A, \quad b \in B, \quad c \in C$) could there be?



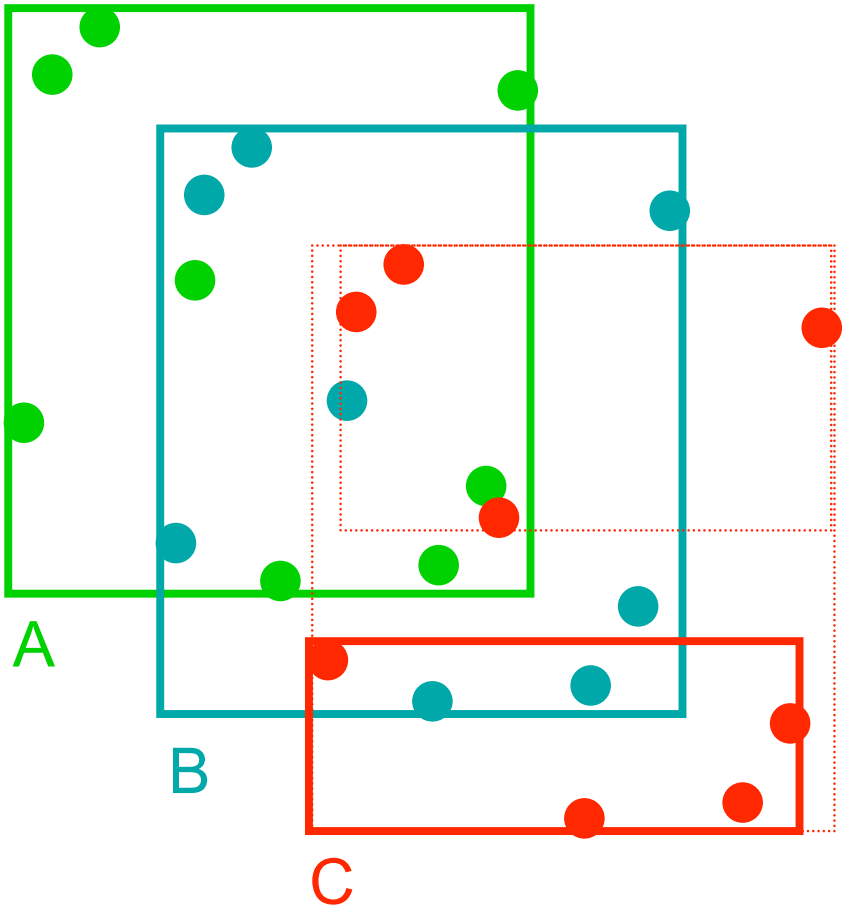
count{**A**,**B**,**C**} =

count{**A**,**B**,***C.left***}

+

count{**A**,**B**,**C.right**}

"How many valid triangles a-b-c (where $a \in A, \quad b \in B, \quad c \in C$) could there be?



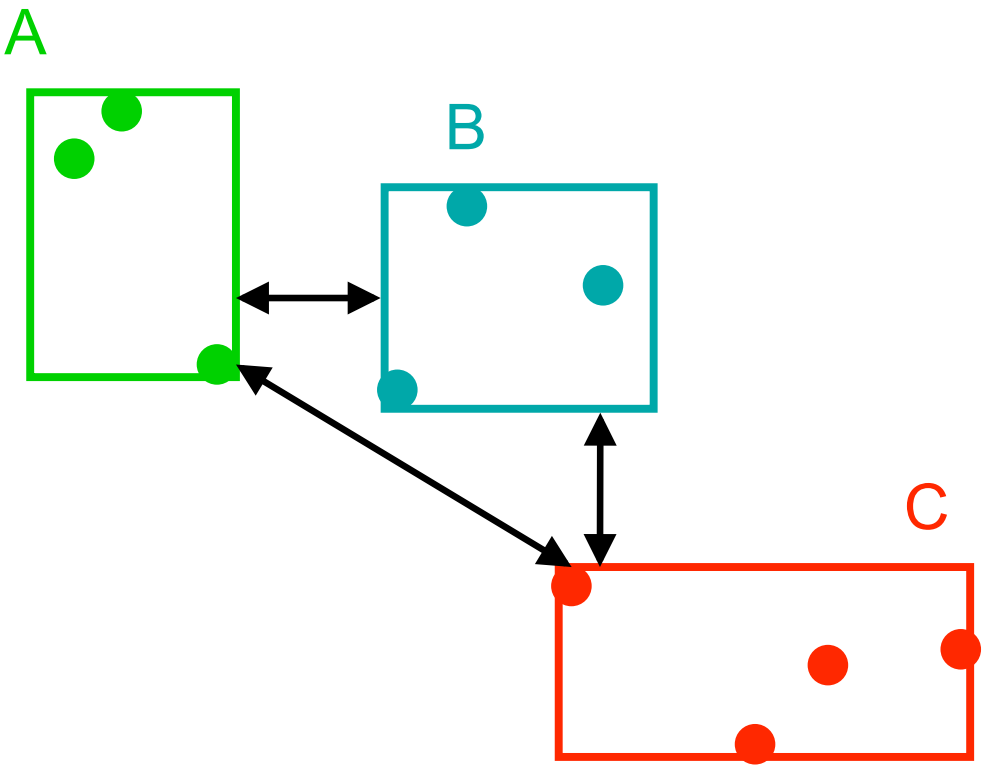
count{**A**,**B**,**C**} =

count{**A**,**B**,**C.left**}

+

count{**A**,**B**,***C.right***}

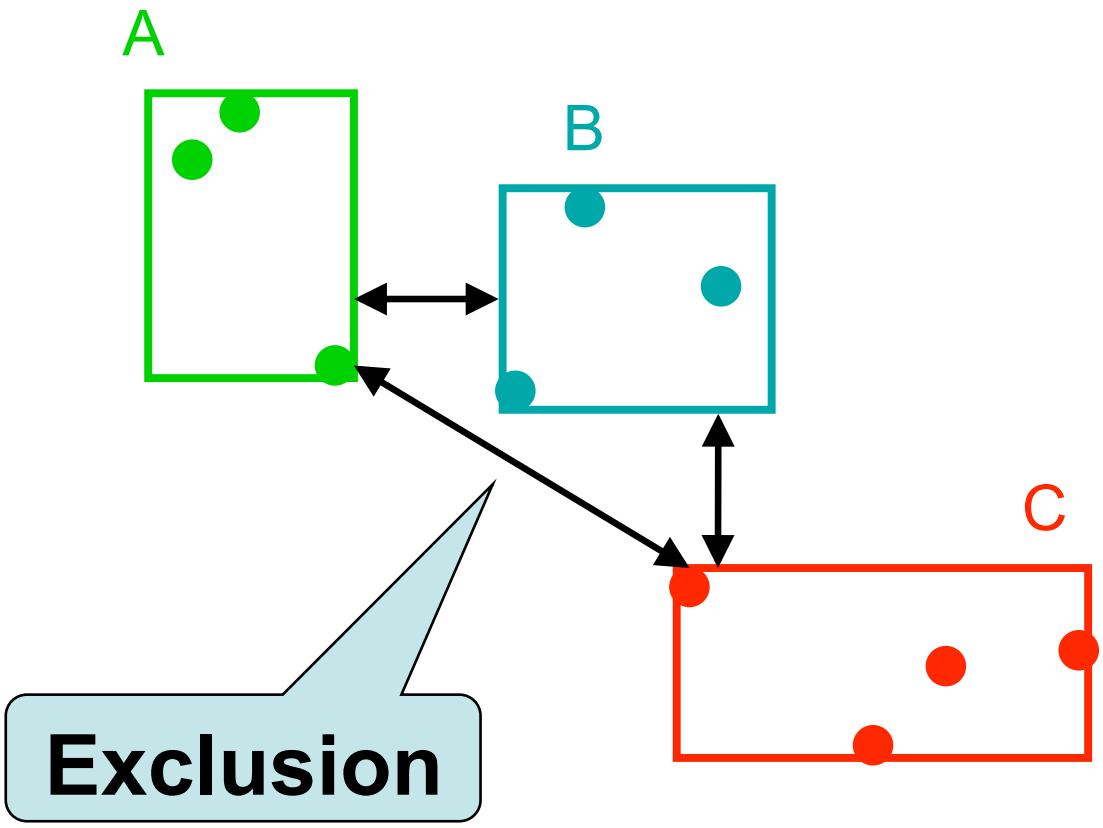
"How many valid triangles a-b-c (where $a \in A, \quad b \in B, \quad c \in C$) could there be?



count{**A**,**B**,**C**} =

?

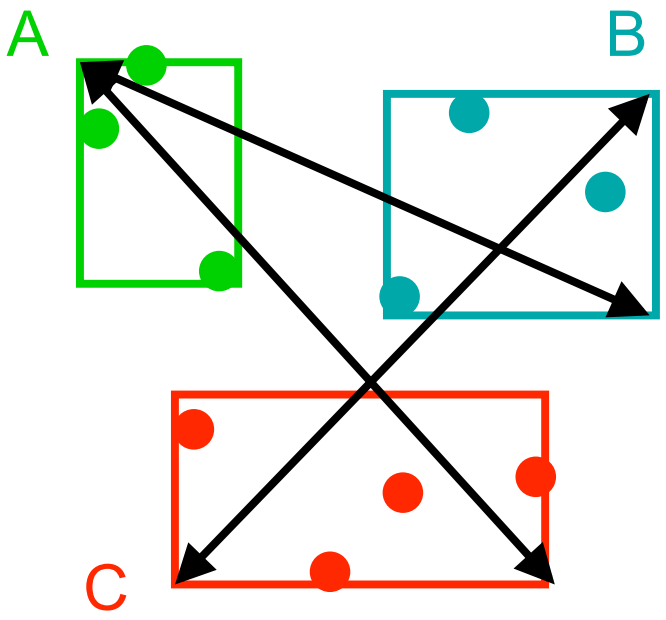
"How many valid triangles a-b-c (where $a \in A, \quad b \in B, \quad c \in C$) could there be?

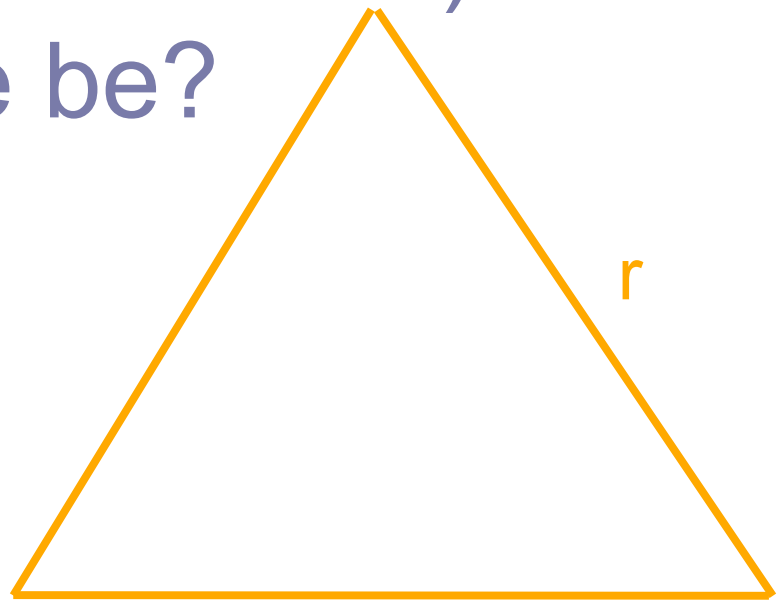


count{**A**,**B**,**C**} =

0!

“How many valid triangles a-b-c (where $a \in A, \quad b \in B, \quad c \in C$) could there be?

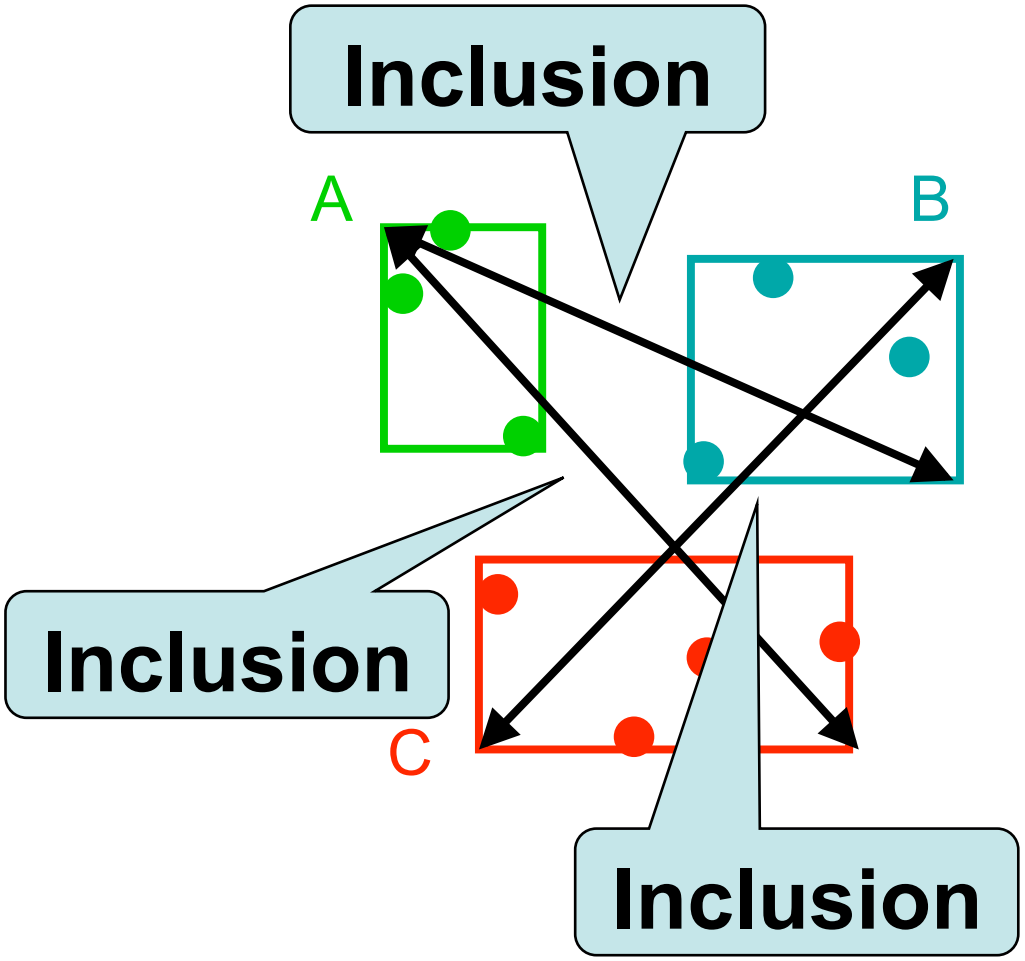


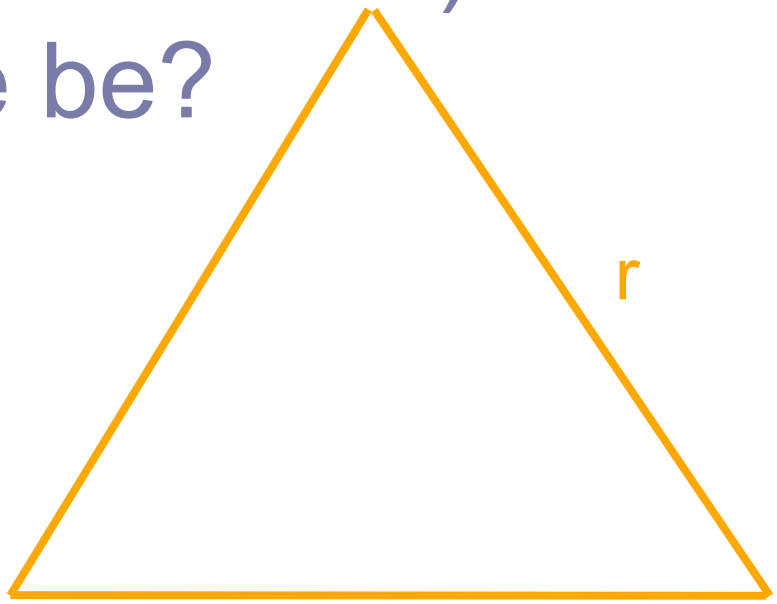


count{**A**,**B**,**C**} =

?

“How many valid triangles a-b-c (where $a \in A, \quad b \in B, \quad c \in C$) could there be?



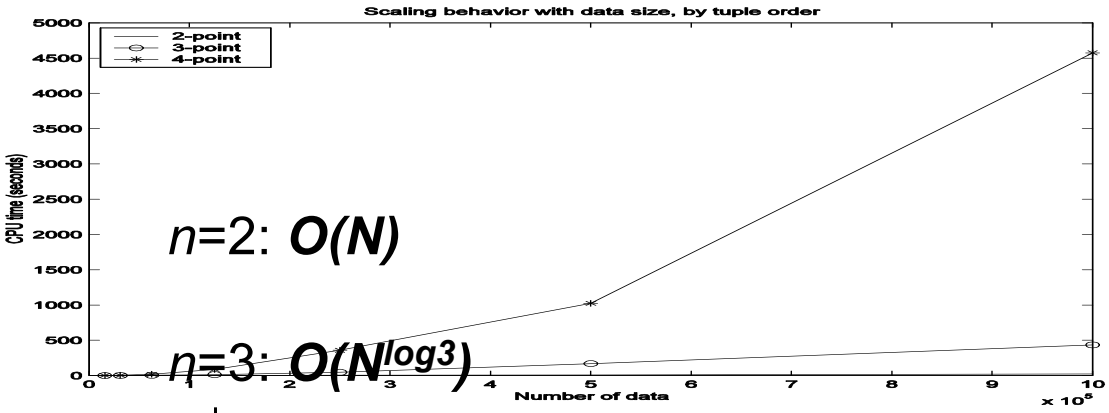


count{**A**,**B**,**C**} =

|A| x |B| x |C|

3-point runtime

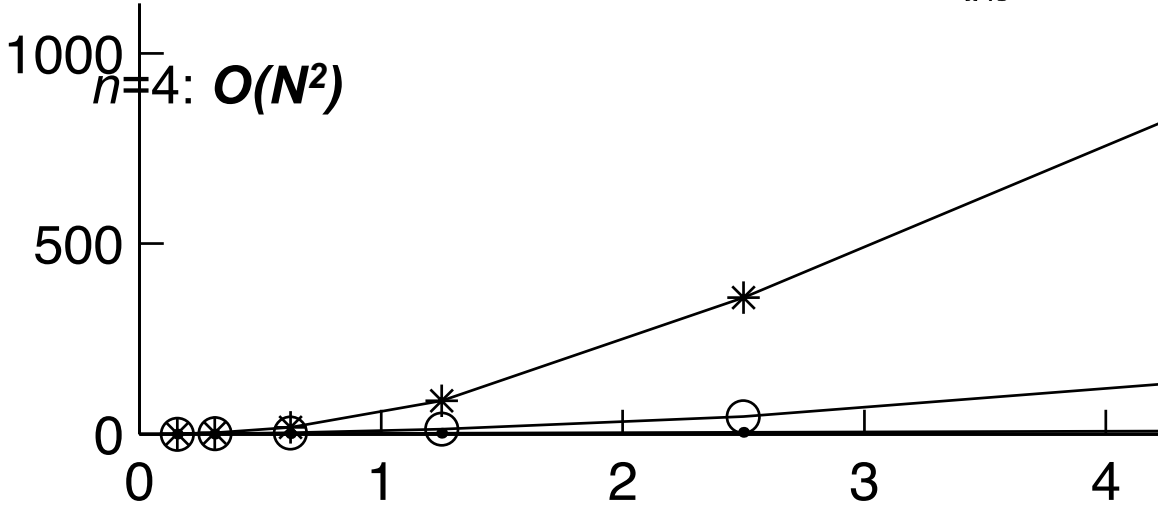
(biggest previous: 20K)



n=2: ***O(N)***

n=3: ***O(N^{log3})***

n=4: ***O(N^2)***



VIRGO simulation data,

N = **75,000,000**

naïve: 5x10^9 sec.
(~150 years)

multi-tree: **55 sec.**

(exact)

New: Compute All Matchers Simultaneously!

	Naive - $O(N^n)$ (estimated)	Single bandwidth [Gray & Moore 2000, Moore *et al.* 2000]	Multi-bandwidth [March & Gray in prep 2010] *new*
2 point cor. 100 matchers	2.0×10^7 s	352.8 s 56,000	4.96 s 71.1
3 point cor. 243 matchers	1.1×10^{11} s	891.6 s 1.23×10^8	13.58 s 65.6
4 point cor. 216 matchers	2.3×10^{14} s	14530 s 1.58×10^{10}	503.6 s 28.8

10^6 points, galaxy simulation data

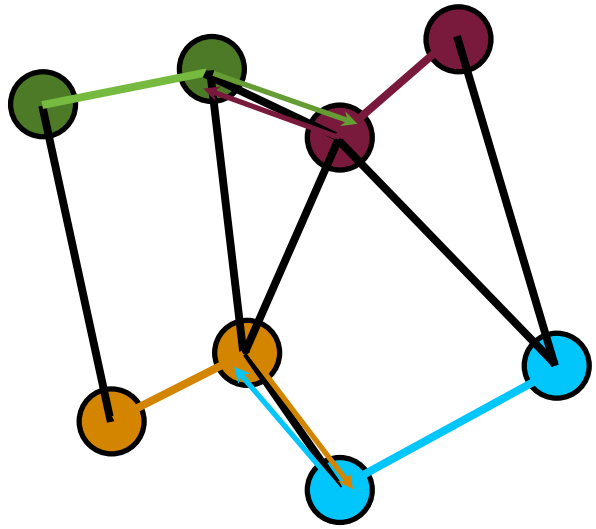
Other Approaches

- This is *exact*
- General – the only approach that can scale to 3-point and beyond
- FFT and others – approximate with no clear error bound
- GPU – constant factor only, can in principle be combined with trees
- Coming soon: large-radius solution

Friends-of-Friends
(aka Hierarchical Clustering, aka Euclidean Minimum Spanning Tree)

Boruvka's strategy:

- *Boruvka step:* Add the nearest neighbor of each component to the spanning forest
  - At most *log N* Boruvka steps to form the MST
- *O(T(N) log N)* time
  - *T(N)* is time to perform Boruvka step



DualTreeBoruvka

- Simply Boruvka's algorithm
- Requires at most *log N* iterations
  - Number of components is halved in each iteration
- Use a dual-tree method to find nearest neighbor pairs

Algorithm 1 Dual-Tree Borůvka (Tree root q)

$E = \emptyset$
while $|E| < N - 1$ **do**
  FindComponentNeighbors(q, q, e)
  $E \leftarrow E \cup e$
  UpdateTree(q)
end while

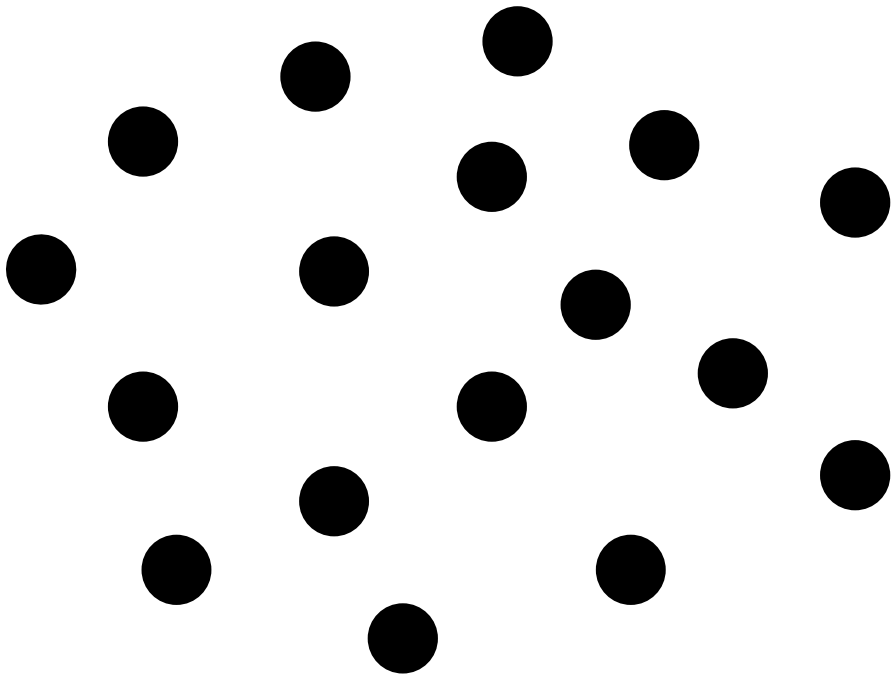
Dual-Tree FindComponentNeighbors

- All query algorithm
  - Amortize work by searching for many queries
- Finds the nearest neighbor pair for the component containing *q*, and components containing descendants of *q*

Algorithm 2 **FindComponentNeighbors**(Cover tree node q_j, Reference Set R_i, Edge set e)

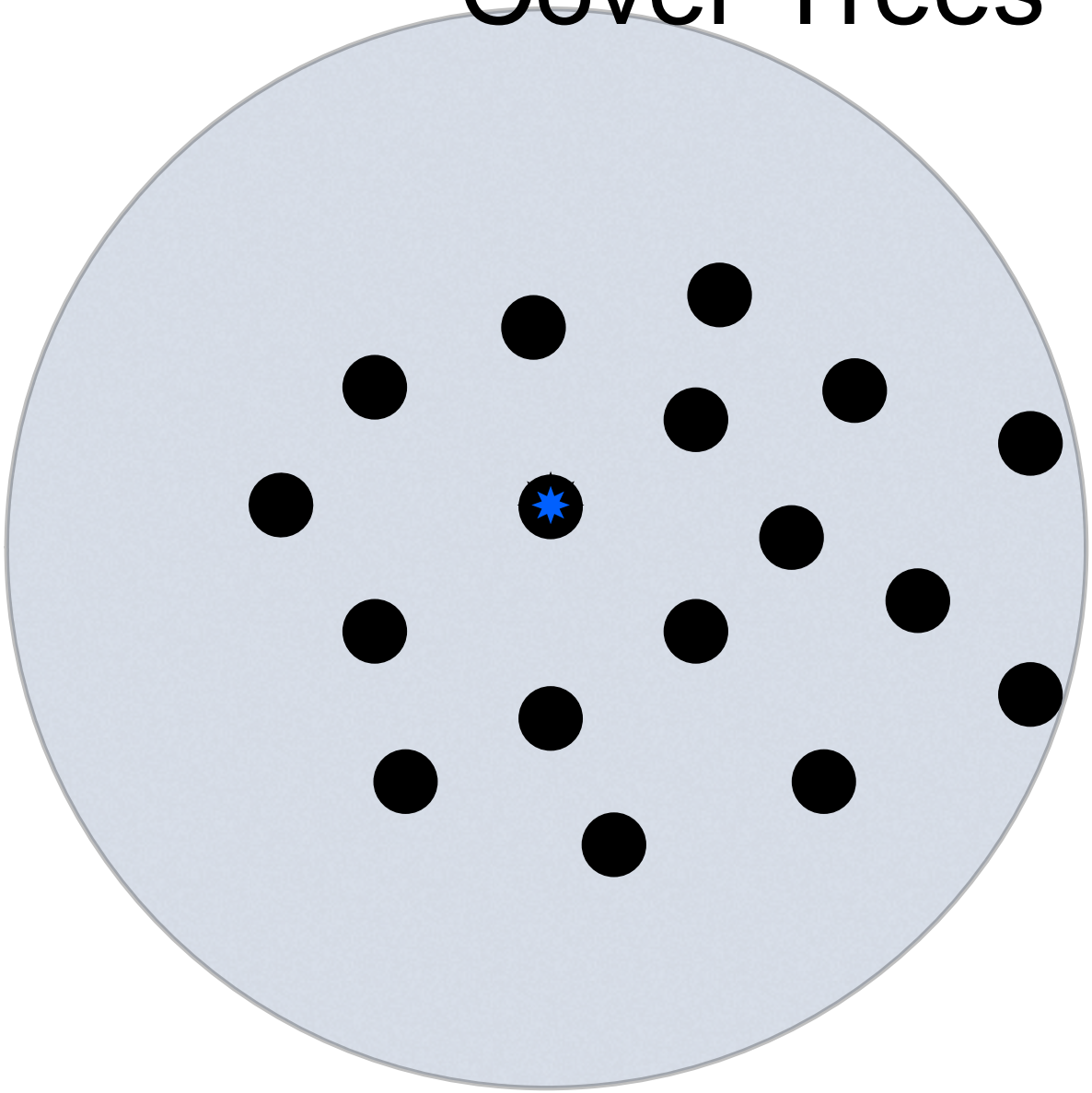
```
if i = −∞ then
  // base case
  for all q that are descendants of q_j and r ∈ R_i with r ≁ q do
    if d(q, r) < d(C_q) then
      d(C_q) = d(q, r), e(C_q) = (q, r)
    end if
  end for
else if j < i then
  // reference descend
  R = {r ∈ Children(r') : r' ∈ R_i and r ⋪ q_j}
  d = min { d(C_q), min_{r∈R, r∼q_j} {d(q_j, r) + 2^i}, min_{r∈R, r≁q_j} {d(q_j, r)} }
  R_{i−1} = {r ∈ R : d(q_j, r) ≤ d + 2^i + 2^{j+2}}
  d(C_q) = d
  FindComponentNeighbors(q_j, R_{i−1}, e)
else
  // query descend
  for all p_{j−1} ∈ Children(q_j) do
    FindComponentNeighbors(p_{j−1}, R_i, e)
  end for
end if
```

Cover Trees



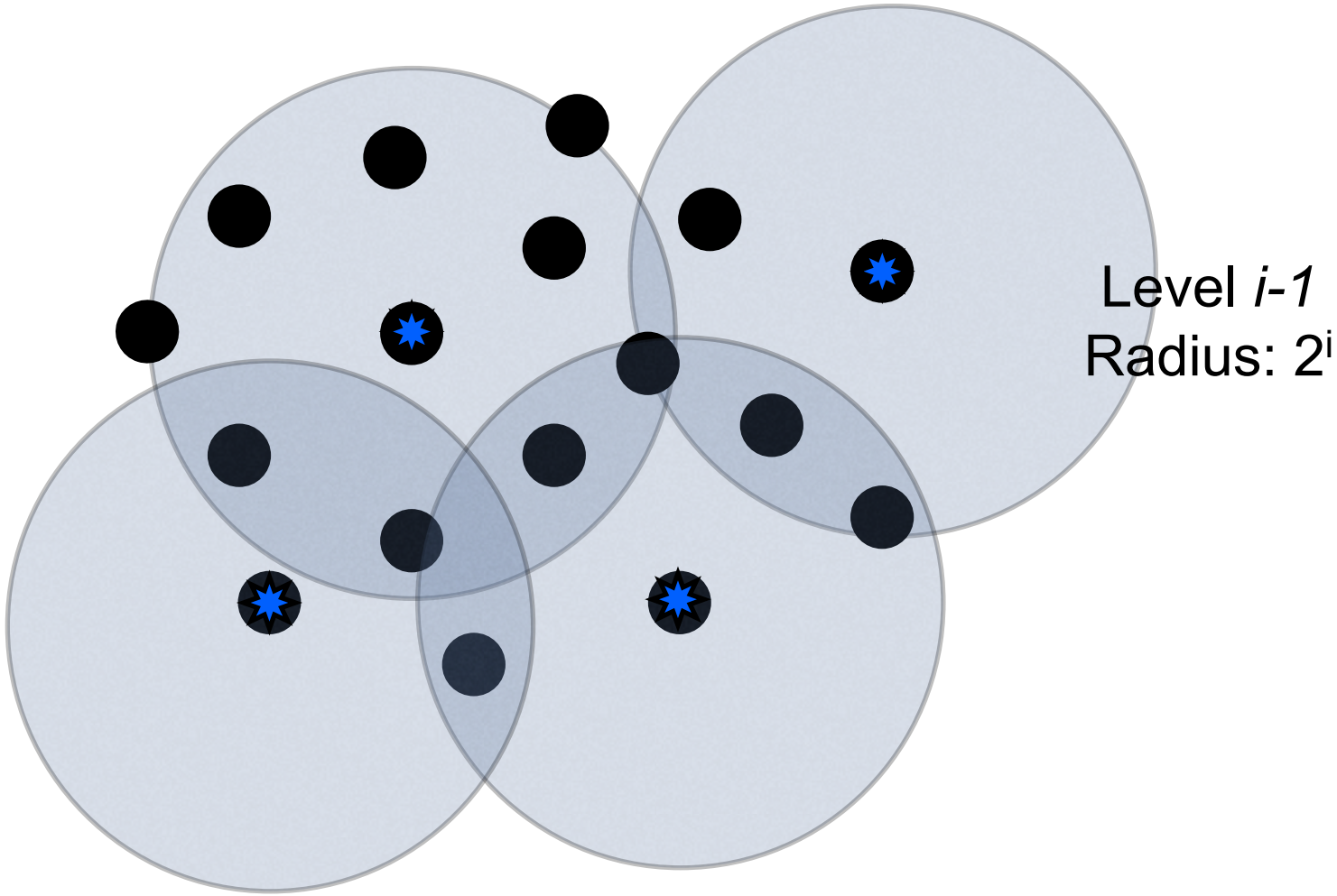
From [BKL06]

Cover Trees

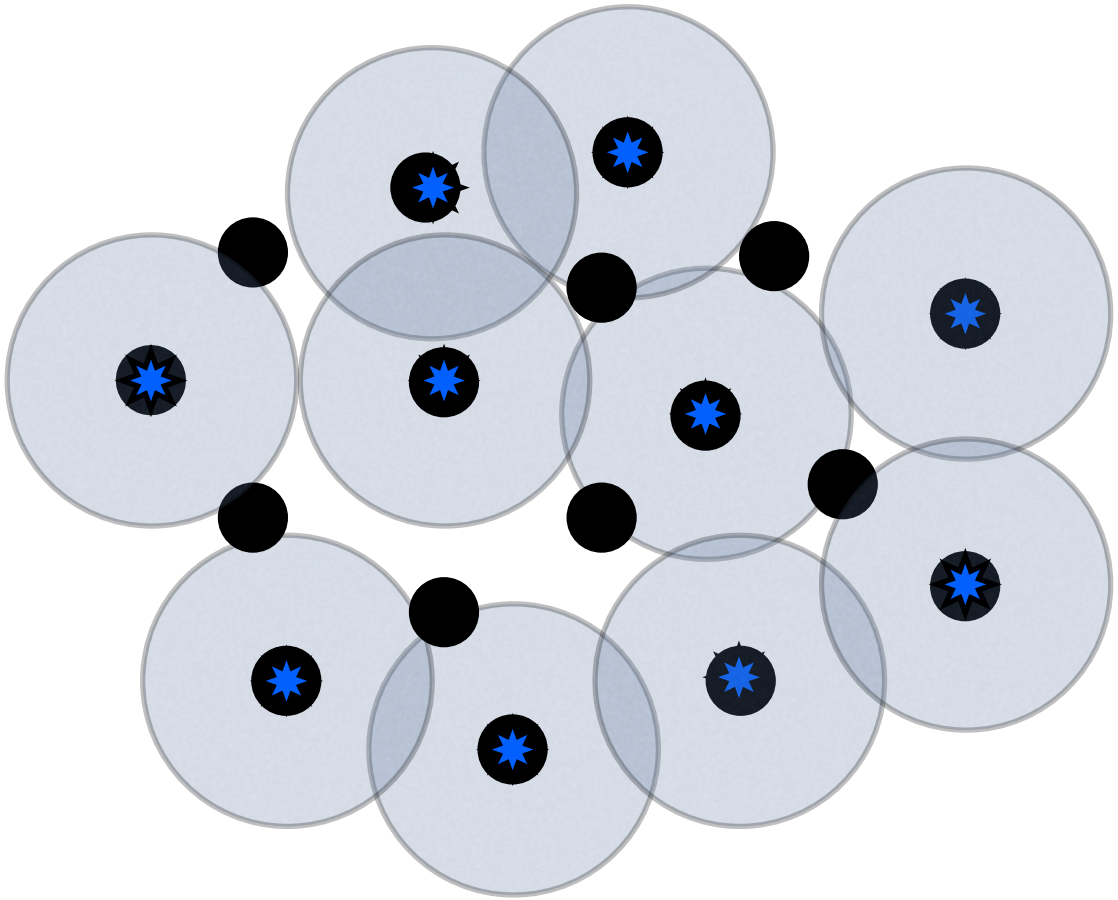


Level *i*
Radius: 2^{i+1}

Cover Trees

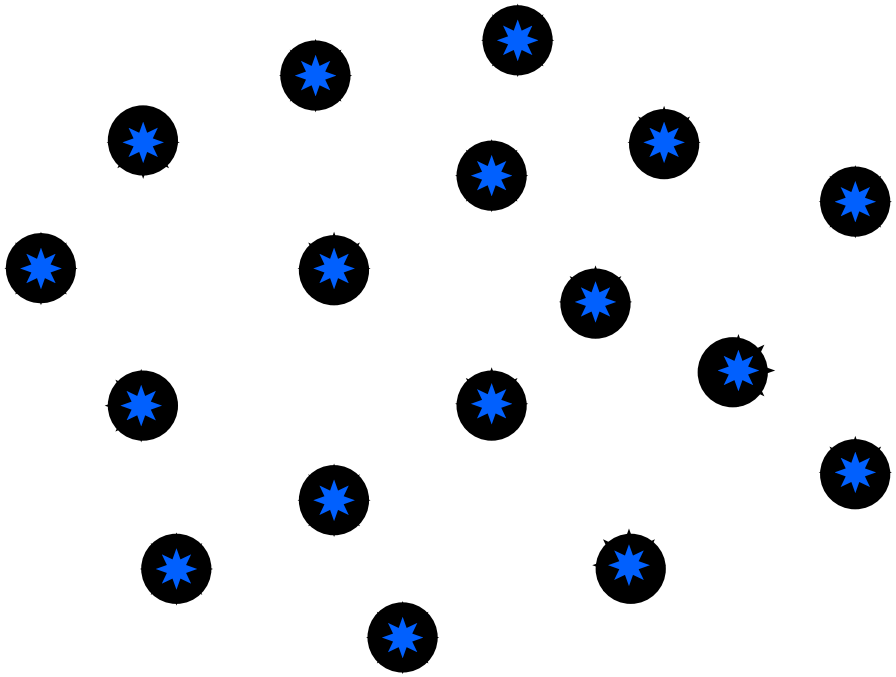


Cover Trees



Level *i-2*
Radius: 2^{i-1}

Cover Trees

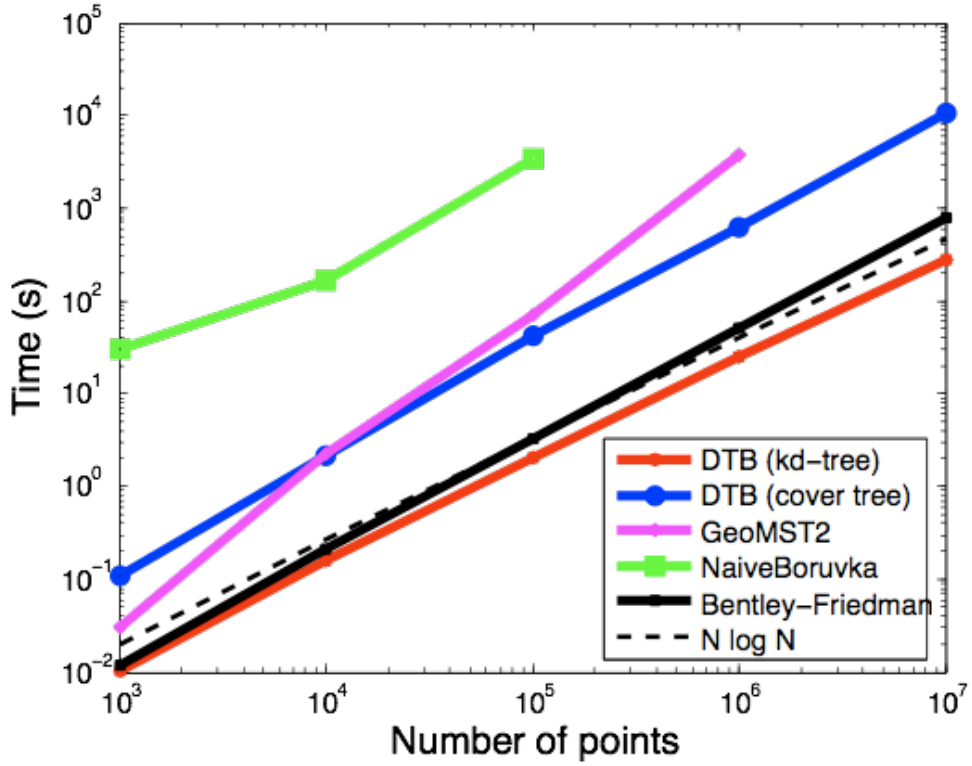


Space: *O(N) nodes*
Construction: *O(N log N)*

Experiments

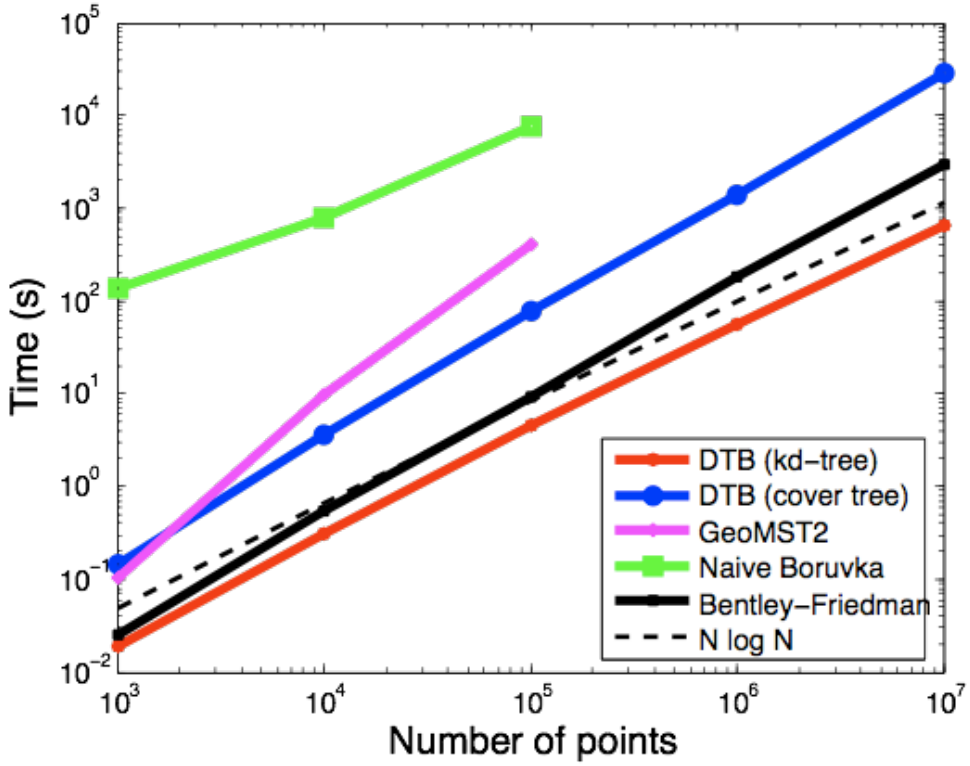
- Implementations:
  - DualTreeBoruvka on *kd*- and cover trees
  - Bentley & Friedman's [BF78] single tree version of Prim's algorithm
  - GeoMST2 [NZZ00]: WSPD-based implementation of Kruskal's algorithm
  - Naive Boruvka: Boruvka's algorithm with neighbors found by brute force

Results



Mixture of 10 Gaussians in 3 dimensions

Results



Subsets of Sloan Digital Sky Survey, 4 dimensional spectra

Results

N	dim	BF78	DTB kd	DTB cover
40,000	3840	45780.43	45825.18	**15791.37**
1,000,000	3	42.54	**17.39**	333.45

Top: High-dimensional SDSS spectra
Bottom: *(x,y,z)* coordinates from
galaxy formation simulation

Theoretical analysis

- The **fastest experimental results**, as compared to the current best algorithms
- The first application of **adaptive algorithm analysis** to the EMST problem
- The **tightest runtime bound** on the EMST problem: achieves lower bound, *O(N log N)* to within nearly a constant factor

Runtime Proof

- **Theorem:** For a set *S* of *N* points in a metric space with expansion constant *c*, cluster expansion constant c_p, and linkage expansion constant c_l, the DualTreeBoruvka algorithm on a cover tree runs in time:

$$O(N \log N \, \alpha(N)) \approx O(N \log N)$$

  - α(N) is inverse of Ackermann function
    - Comes from Disjoint-Set data structure
    - Grows *very* slowly: $\alpha(10^{80}) \leq 5$

FindComponentNeighbors Runtime

- **Theorem:** Under the assumptions of the previous theorem, the FindComponentNeighbors algorithm on a cover tree finds the nearest neighbor pair for each component in time bounded by:

$$O(N\,\alpha(N)) \approx O(N)$$

Algorithm Analysis by Parts

- Base Case:
  - *O(N)* query nodes,
    Work per query:
- Reference Recursion: $O(\max_i |R_i| \, \alpha(N))$
  - Duplication of references bounded by tree's width and depth bound [cite],
    Max duplication:

$$O(c^4 N + c^6 \max_i |R_i| \log N \alpha(N))$$

- Query Recursion:
  - *O(N)* explicit query nodes
- *TOTAL TIME:*

$$O(\max_i |R_i| \cdot N \cdot \alpha(N))$$

Algorithm 2 **FindComponentNeighbors**(Cover tree node q_j, Reference Set R_i, Edge set e)

```
if i = −∞ then
  // base case
3:  for all q that are descendants of q_j and r ∈ R_i with r ≁ q do
      if d(q, r) < d(C_q) then
        d(C_q) = d(q, r), e(C_q) = (q, r)
6:    end if
    end for
  else if j < i then
9:  // reference descend
    R = {r ∈ Children(r') : r' ∈ R_i and r ⋪ q_j}
    d = min { d(C_q), min_{r∈R, r∼q_j} {d(q_j, r) + 2^i}, min_{r∈R, r≁q_j} {d(q_j, r)} }
12: R_{i−1} = {r ∈ R : d(q_j, r) ≤ d + 2^i + 2^{j+2}}
    d(C_q) = d
    FindComponentNeighbors(q_j, R_{i−1}, e)
15: else
    // query descend
    for all p_{j−1} ∈ Children(q_j) do
18:   FindComponentNeighbors(p_{j−1}, R_i, e)
    end for
  end if
```

Recent Results in Astrostatistics

1. **Fast algorithms:** n-point, friends-of-friends, theory
2. **Software:** databases
3. **Statistical methodology:** new ML methods, ML with measurement errors

Software

- **MLPACK (C++)**
  - First scalable comprehensive ML library
- **MLPACK-db (C# →C++)**
  - fast data analytics in relational databases (SQL Server; on-disk)
- **FastML (commercial: Analytics 1305)**
  - Very-large-scale data (parallel, streaming)

MLPACK-db

- Modification of fastlib (library) to allow for loadDB and saveDB to access databases.
- We use a library UnixODBC which lets you connect to databases(MySQL, MSSQL, Postgres etc) using a neat API
- UnixODBC provides a neat set of abstractions that lets you talk to databases from C, C++, Python et. al in Unix environments.

Fast Tree-based Algorithms in Databases

Framework Essentials:

- Build kD-Tree index on your table
- Data isn't copied
- Complexity hidden from user
- Simple queries to the server for ML functions
  - Exec dbo.createKdTree(<tableName>, <treeName>);
  - Exec dbo.getNearestNeighbors(<treeName>, <results>);
- Extensible – Easy to write new ML methods.

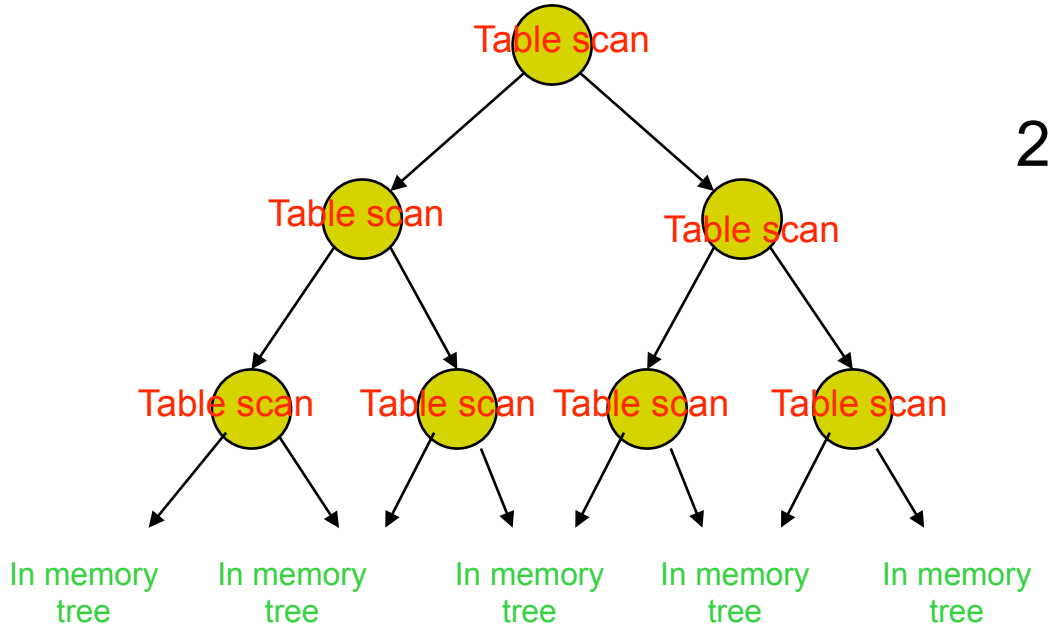
Tree Strategy

- Form a cluster index on a spatial key
- Use it to quickly access spatially local data
- Also store tree nodes in a relational table

- Problem:
  - How to find the spatial keys?
  - E.g. how to build the tree?

kD-tree construction

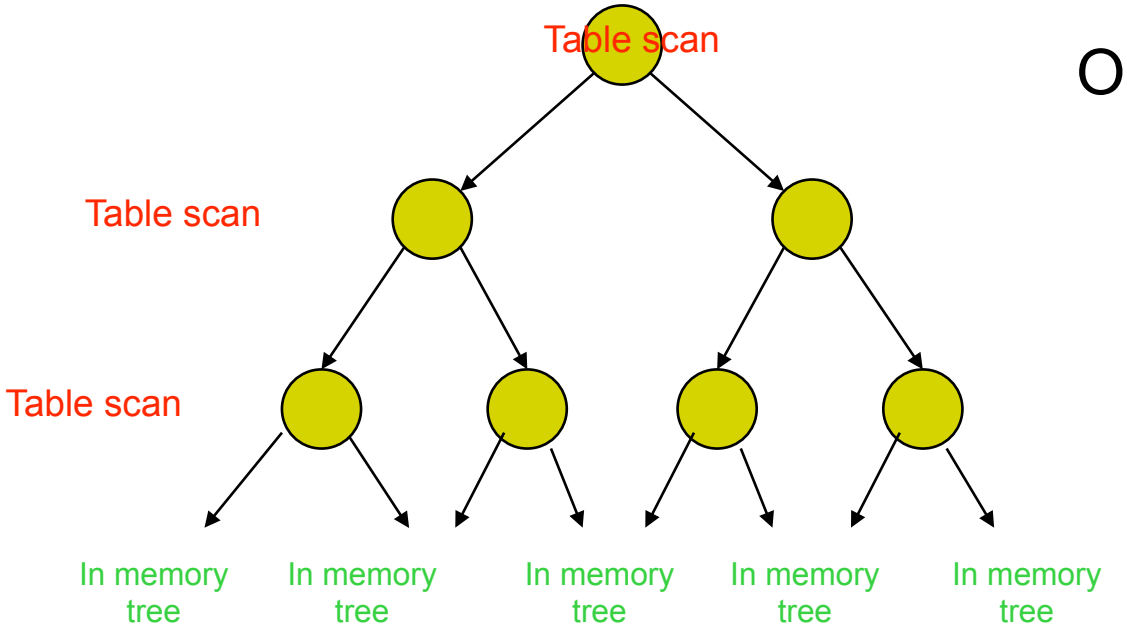
- Take 1



$2\ N/M\ O(N) = O(N^2)$

kD-tree construction

- Take 2



O(N log^2 N)

kD-tree construction

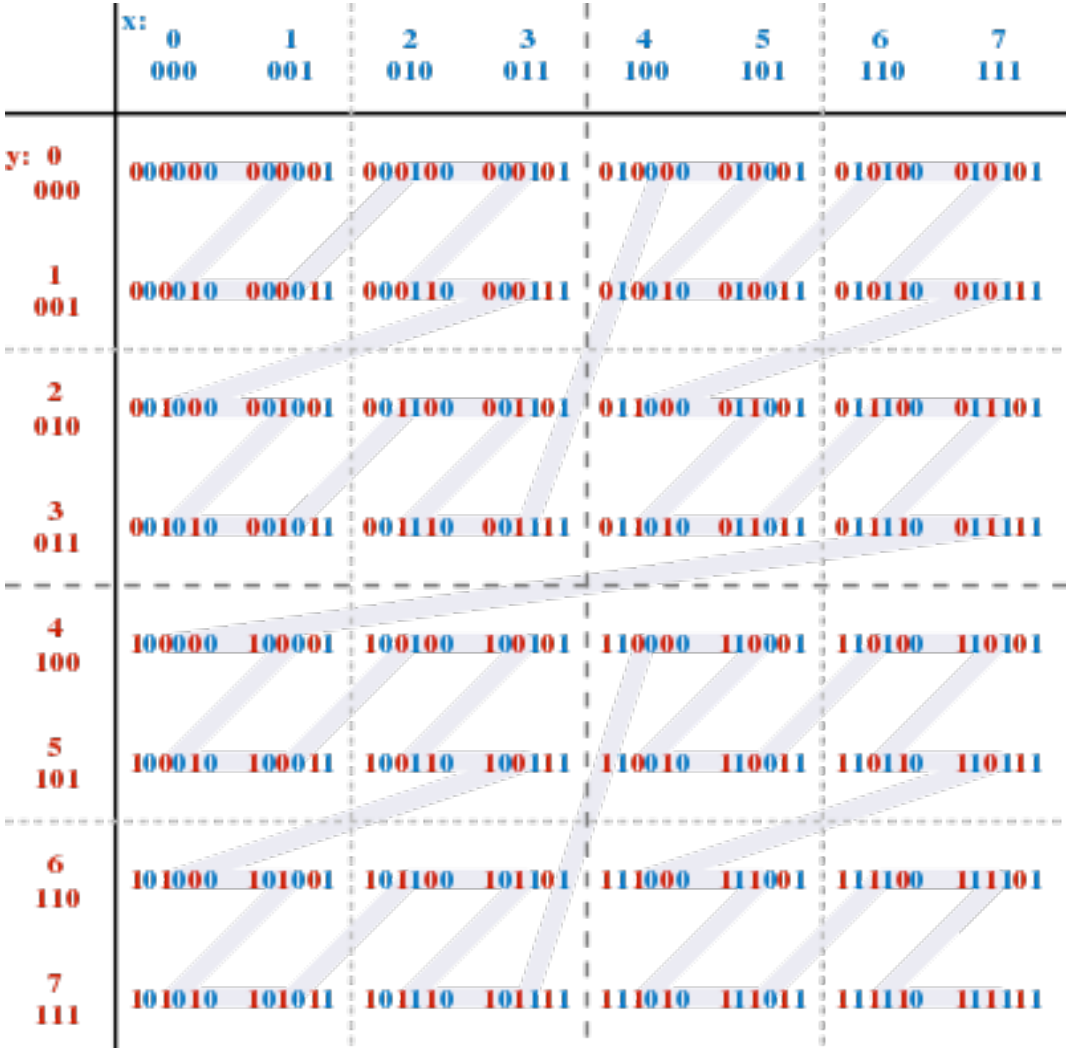
- Take 3

- Bottleneck: Table scans during building top part of the tree

Solution: Hybrid tree structure

- Use Morton Z-curve to structure data (computed in one pass)
- Load large amounts of spatially local data into memory
- Build a in-memory kD-tree for each chunk
- Build the (small) top tree by combining these sub-trees

Morton Z-order



http://en.wikipedia.org/wiki/File:Z-curve.svg

Tree Construction – Assign and Index by Z-Curve

Data set

Id	x	y
1	2.4	3.4
2	3	5
3	5.3	3.3
4	1.1	2.2
5	3.7	9.3

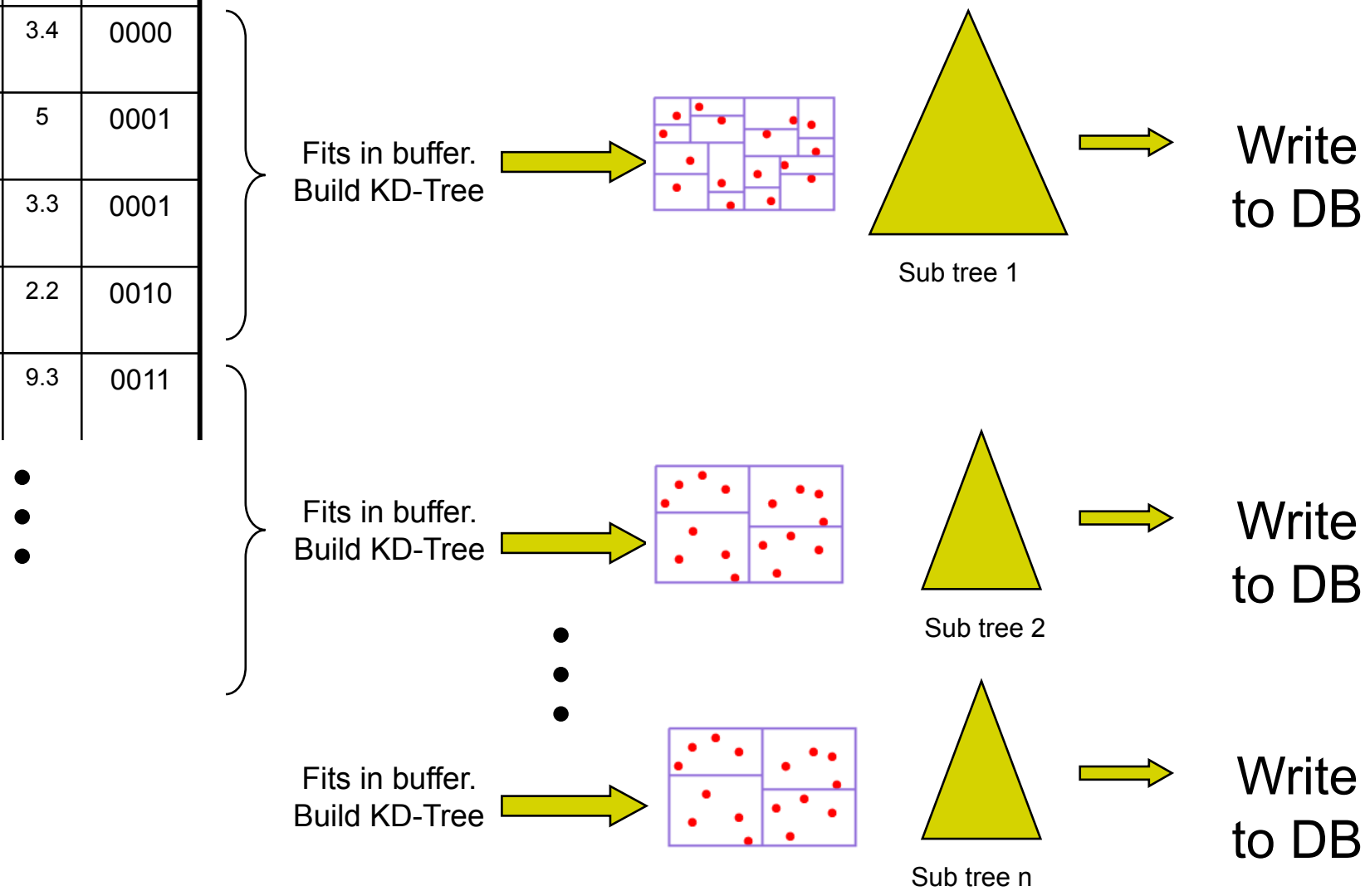
Assign Morton Z-order ids and build clustered index on Z-id

Clustered Index Sorts by Z-Curve Value

Id	x	y	Z-id
4	2.4	3.4	0000
26	3	5	0001
23	5.3	3.3	0001
1	1.1	2.2	0010
93	3.7	9.3	0011

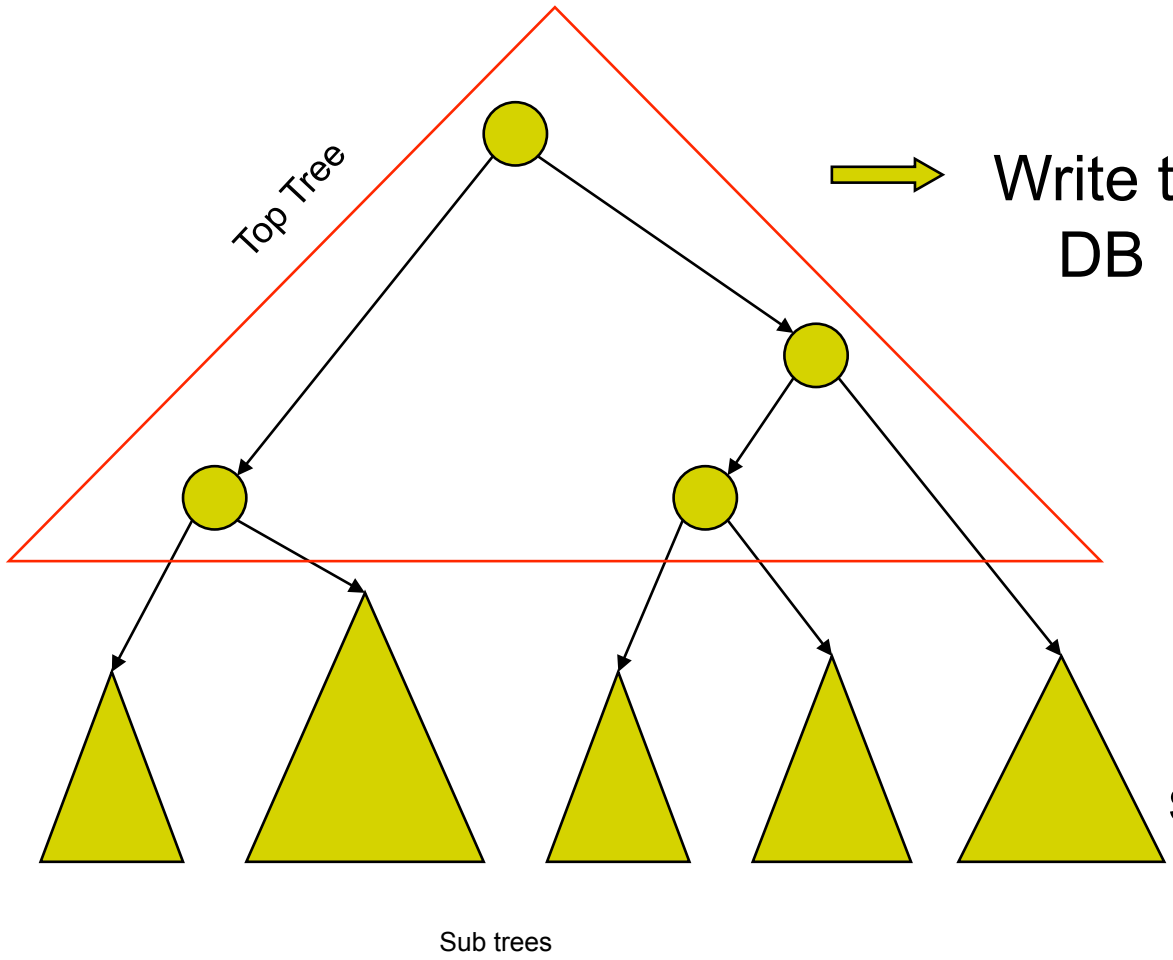
Tree Construction – Sub Trees

Id	x	y	Z-id
4	2.4	3.4	0000
26	3	5	0001
23	5.3	3.3	0001
1	1.1	2.2	0010
93	3.7	9.3	0011



http://graphics.stanford.edu/courses/cs368-00-spring/TA/manuals/CGAL/ref-manual2/SearchStructures/kdtree.gif

Tree Construction – Top Tree



Other Concerns

- Assignment of node keys

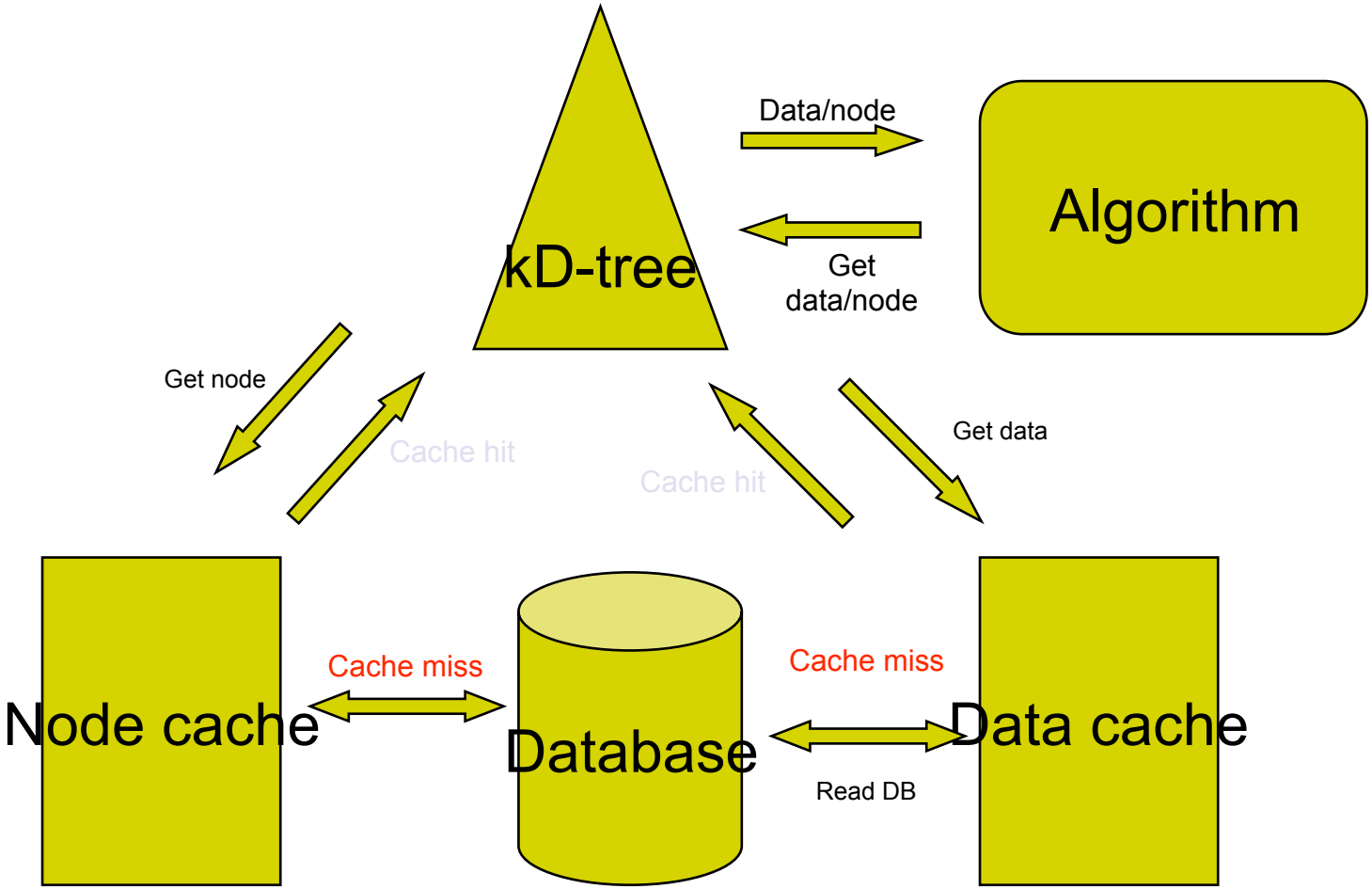
- Algorithm-specific statistics computed for each node

Complexity: 3 table scans

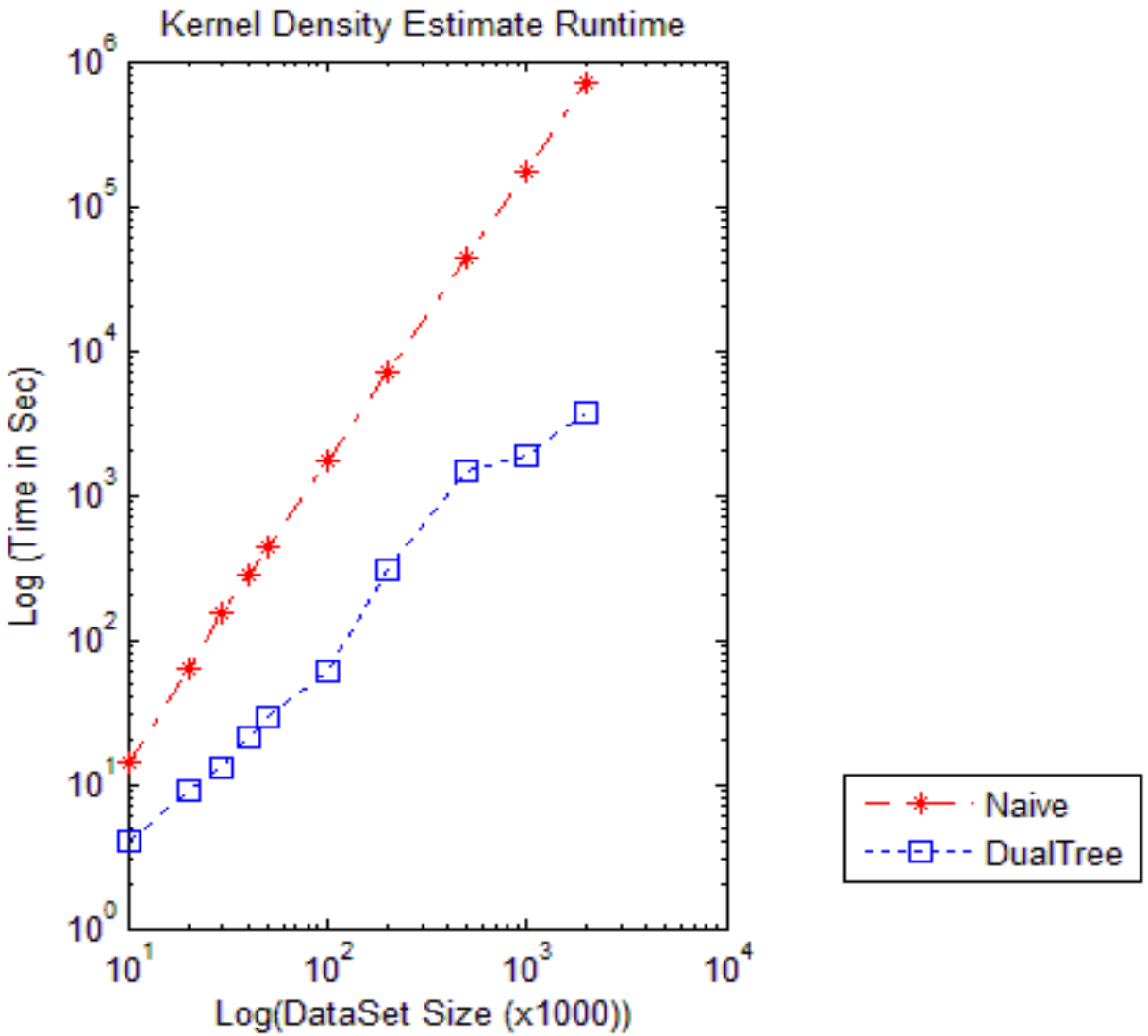
KD-tree API

- Tree provides an API for writing new algorithms
  - Manages memory with cache for indexed data and tree nodes
- Objectives:
  - Minimize the number of SQL queries
  - Users don't manage memory themselves

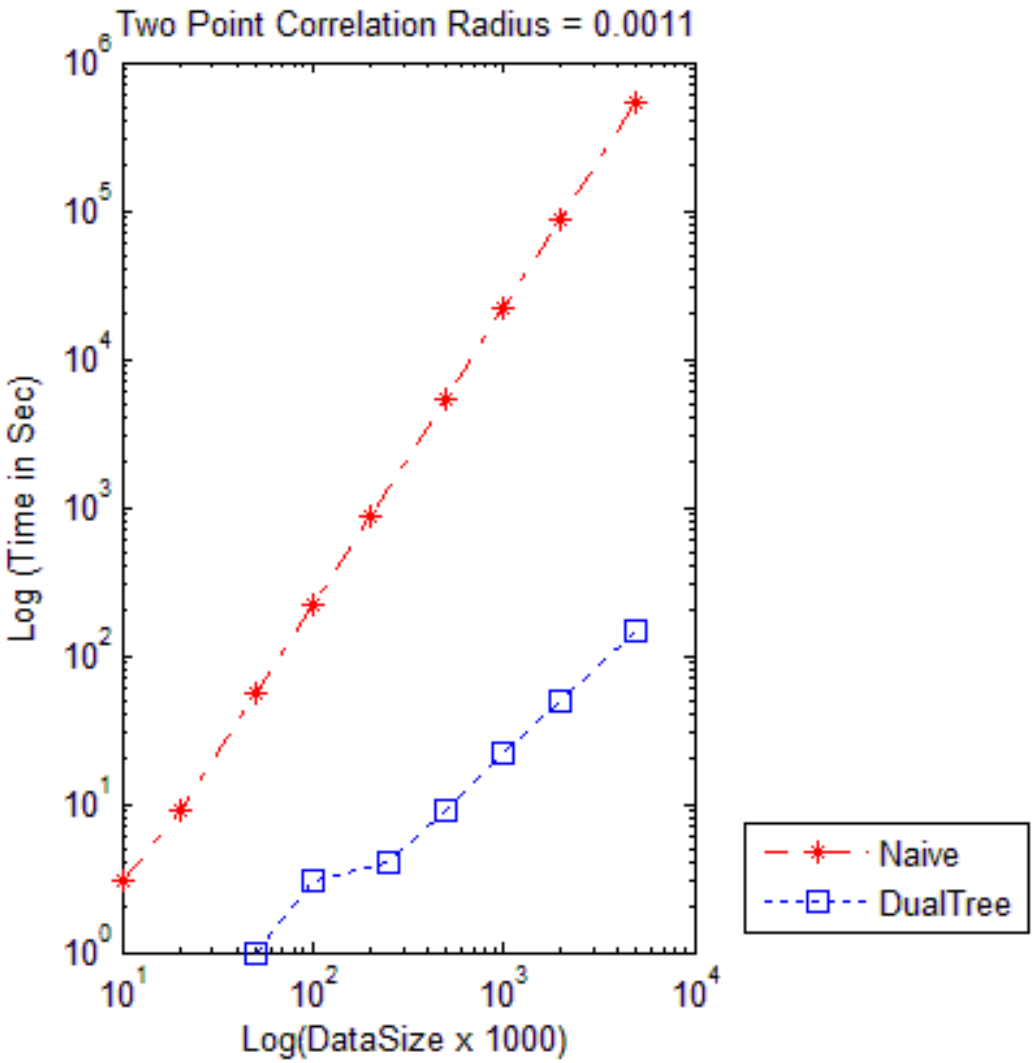
Framework


kD-tree
Data/node
Get
data/node
Algorithm
Get node
Cache hit
Cache hit
Get data
Node cache
Cache miss
Database
Cache miss
Read DB
Data cache

KDE


Kernel Density Estimate Runtime
Log (Time in Sec)
Log(DataSet Size (x1000))
Naive
DualTree

2-point


Two Point Correlation Radius = 0.0011
Log (Time in Sec)
Log(DataSize x 1000)
Naive
DualTree

MLPACK-db vs Naïve: All Nearest Neighbors

No. of points	MLDB (Dual tree)	Naive
40,000	8 seconds	159 seconds
200,000	43 seconds	3480 seconds
2,000,000	297 seconds	80 hours
10,000,000	29 mins 27 sec	74 days
20,000,000	58mins 48sec	280 days
40,000,000	112m 32 sec	2 years

Recent Results in Astrostatistics

1. **Fast algorithms:** n-point, friends-of-friends, theory
2. **Software:** databases
3. **Statistical methodology:** new ML methods, ML with measurement errors

New ML Methods

- *Dimensionality reduction:* rank-preserving manifold learning (2008), convex /isometric NMF (2009)
- *Density estimation:* submanifold kernel density estimation (2010), convex adaptive kernel estimation (2010)
- *Time series:* functional ICA (2009), kernels for time series models (2010)
- etc

Measurement Errors

- Common approach in nonparametric setting
  - Assume an "additive" error model

    $$X = U + \varepsilon$$

    - U: Noise-free data
    - ε: I.i.d. noise
    - X: Measured, noisy data
  - The PDF of X is the convolution of those of U and ε
  - Use kernel-based approaches with a "deconvolved kernel"

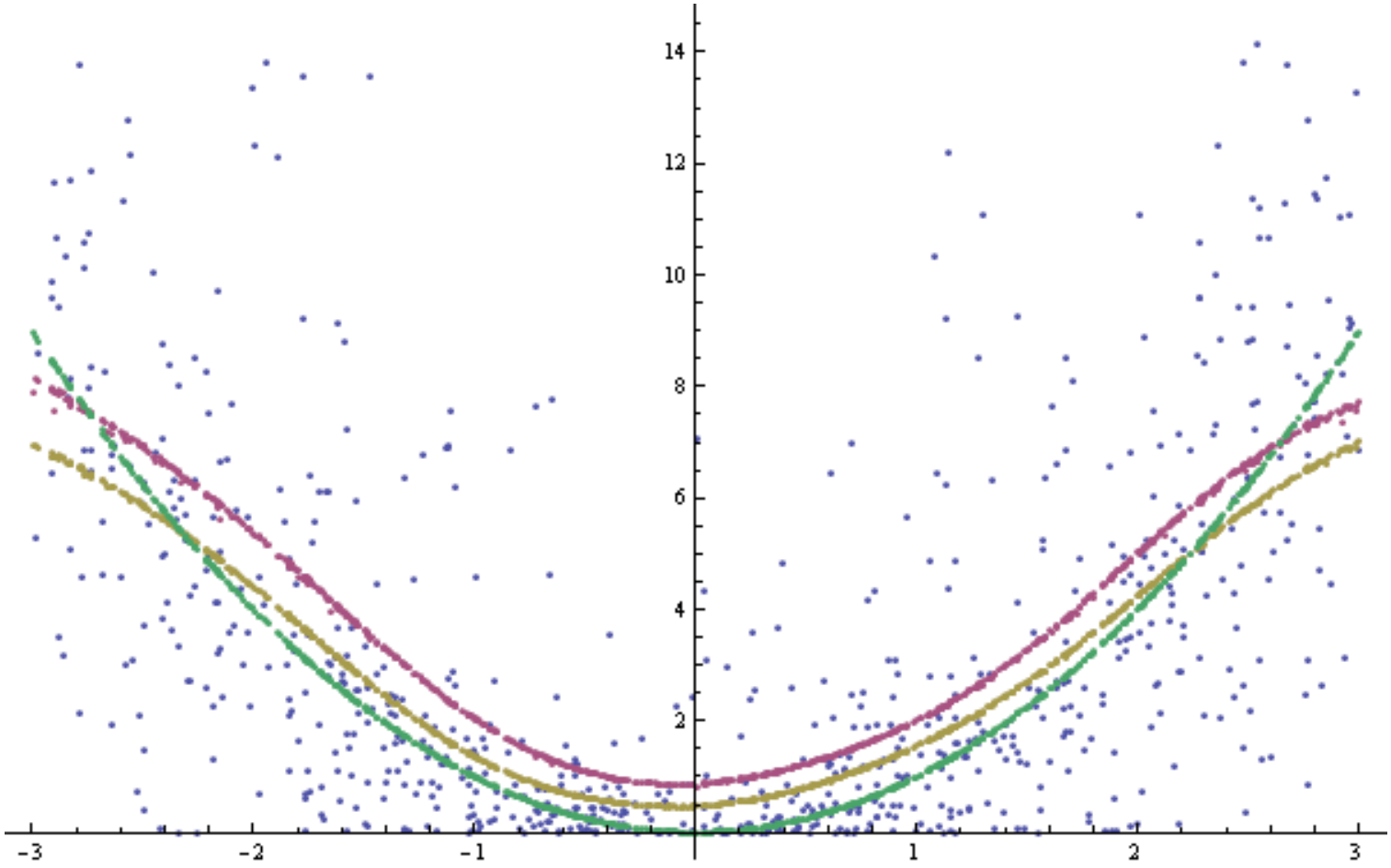
Measurement Errors

- The deconvolution approach is known to be consistent, with good asymptotic rates
- However, it has drawbacks
  - Hard to deal with non-i.i.d. noise
  - The additive model is not necessarily justified for real life applications

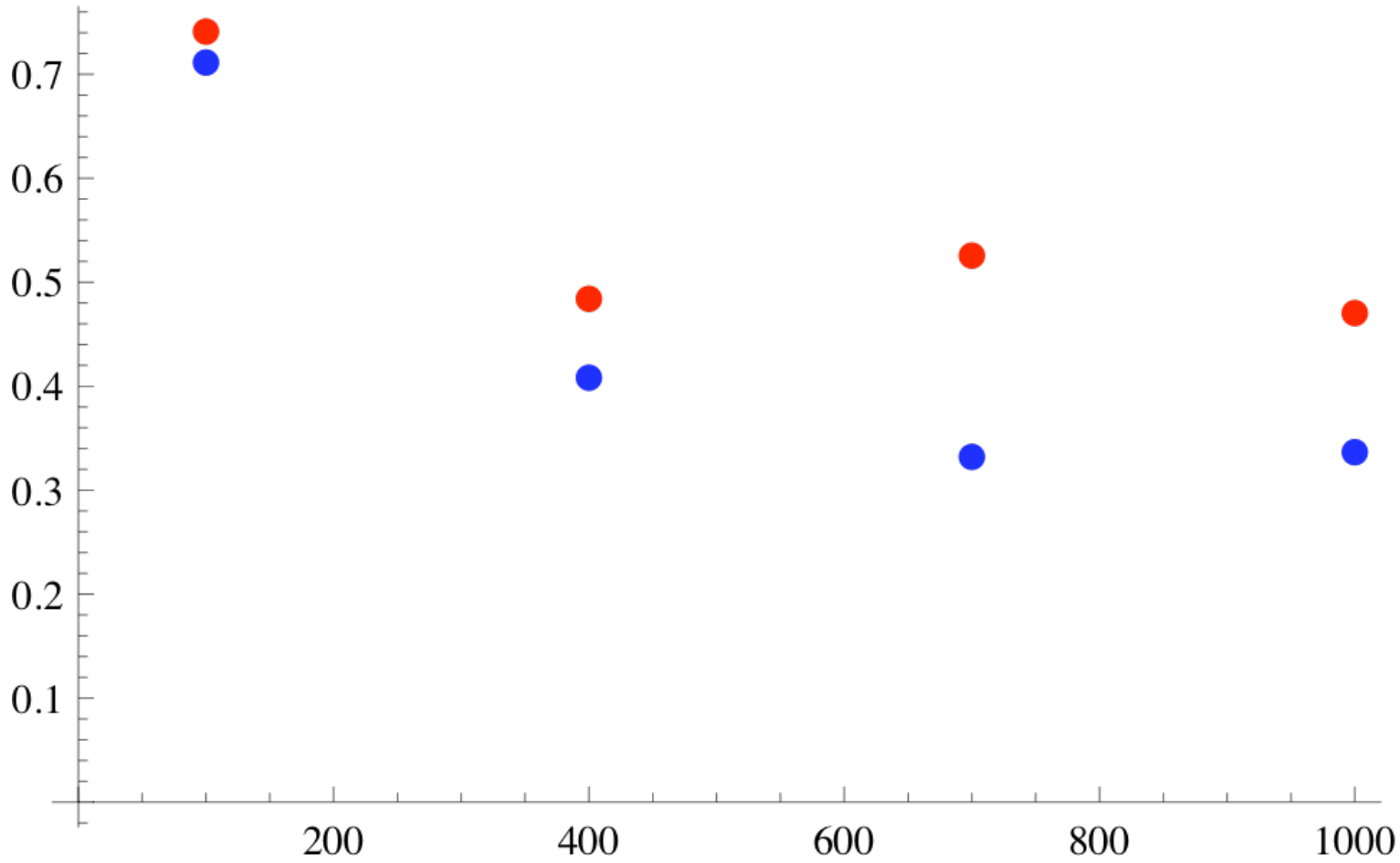
Measurement Errors: A new approach

- Two distinctive features
  - A generative model of noisy data, instead of additive noise
  - Treating each measurement as a PDF
- Advantages:
  - Natural way to deal with non-i.i.d. noise
  - Alternative noise model is a new framework for measurement errors, could possibly model real life data more accurately

Measurement Errors: Preliminary Tests



Measurement Errors: Decreased Error



Measurement Errors: A new approach

- Preliminary results, observations:
  - When evaluated on small synthetic data sets, the method outperforms regular Nadaraya-Watson regression (bandwidth is taken to zero with the optimal rate for NWR).
  - Initial explorations suggest a consistent estimator may be provably non-existent in the new framework. The ultimate aim would be to obtain the best asymptotic accuracy possible.
  - The new framework opens the door to other approaches.

The end

- You can now use many of the state-of-the-art data analysis methods on large datasets
- Talk to me… I have a lot of problems (ahem) but always want more!

Alexander Gray agray@cc.gatech.edu
www.fast-lab.org