Algorithms for Calculating Statistical Properties on Moving Points

Dissertation Proposal

Sorelle Friedler

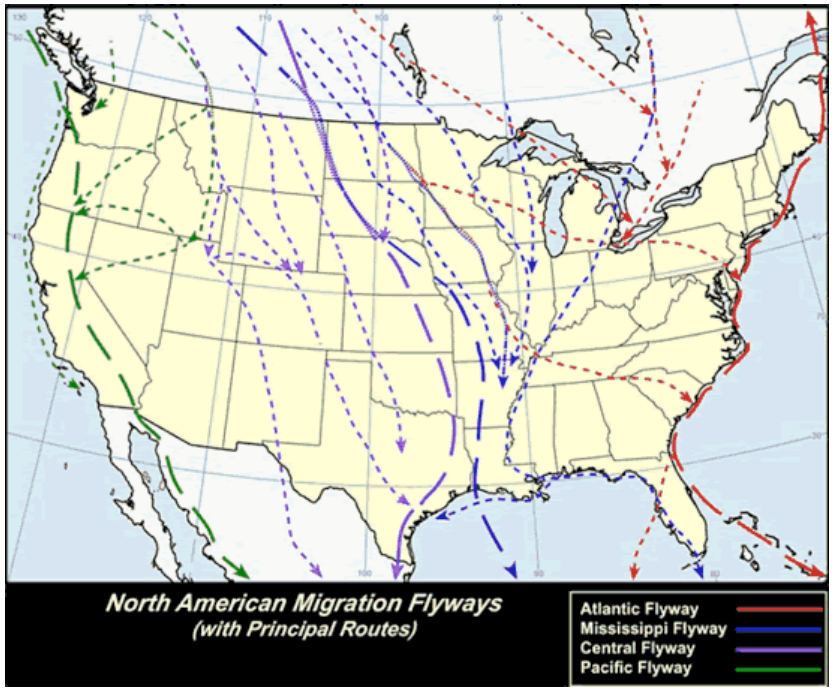
Committee:

David Mount (Chair), William Gasarch

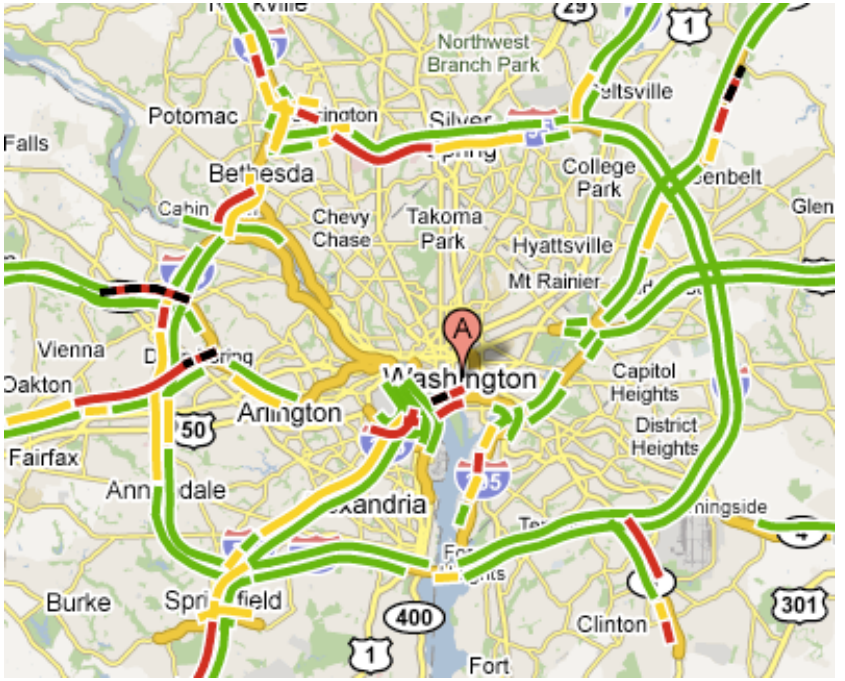
Samir Khuller, Amitabh Varshney

January 14, 2009

Motivation


North American Migration Flyways
(with Principal Routes)
Atlantic Flyway
Mississippi Flyway
Central Flyway
Pacific Flyway

Motivation


Northwest
Branch Park
Potomac
Falls
Bethesda
College
Park
Cabin
Chevy
Chase
Takoma
Park
Hyattsville
Mt Rainier
Glen
Vienna
Capitol
Heights
District
Heights
Arlington
50
Fairfax
Burke
400
1
Clinton
301
Fort

Motivation

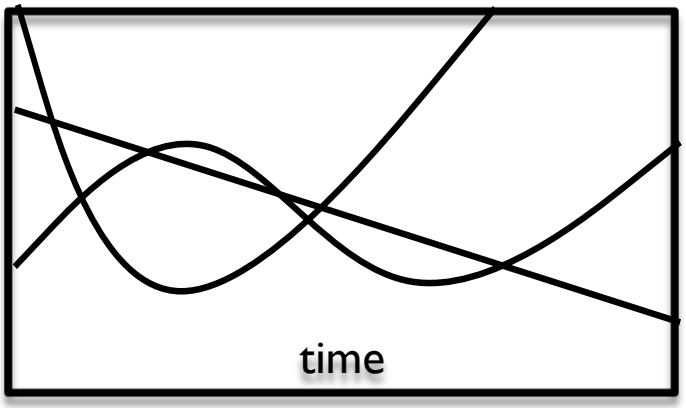
- Computer Science
  - Graphics: Image and video segmentation, animation
  - Databases: Maintenance over time
  - Sensor Networks: Data analysis
- Physics
  - Simulations
- Biology
  - Mathematical ecology: Migratory paths
  - Developmental biology: Neural crest cell migration
- Engineering
  - Traffic patterns and identification

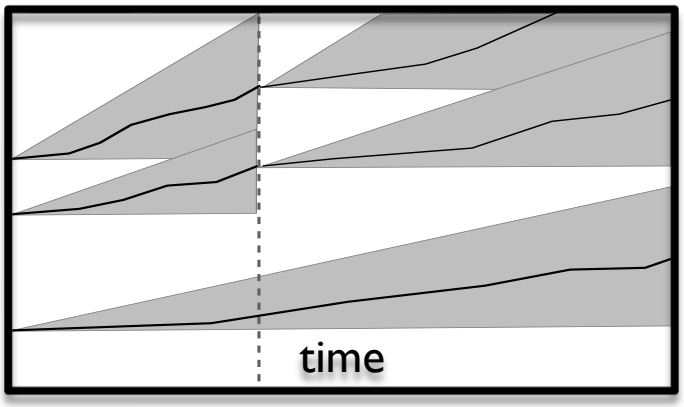
Outline

- Motivation
- Past work
- Kinetic Robust K-Center Problem Algorithm
- Sensor-Based Framework for Kinetic Data
- Proposed Work

Motion Data Structures

- Atallah (1983)
  - Polynomial motion of degree k
  - Motion known in advance
  - Points lie in $\mathbb{R}^d$
  - Analysis in $\mathbb{R}^{d+1}$
- Kahan (1991)
  - Bounds on point velocity
  - Update function provided
  - Limit queries to function
- Other structures
  - Guibas et al. (1993), Schomer and Thiel (1995, 1996), Gupta et al. (1996)



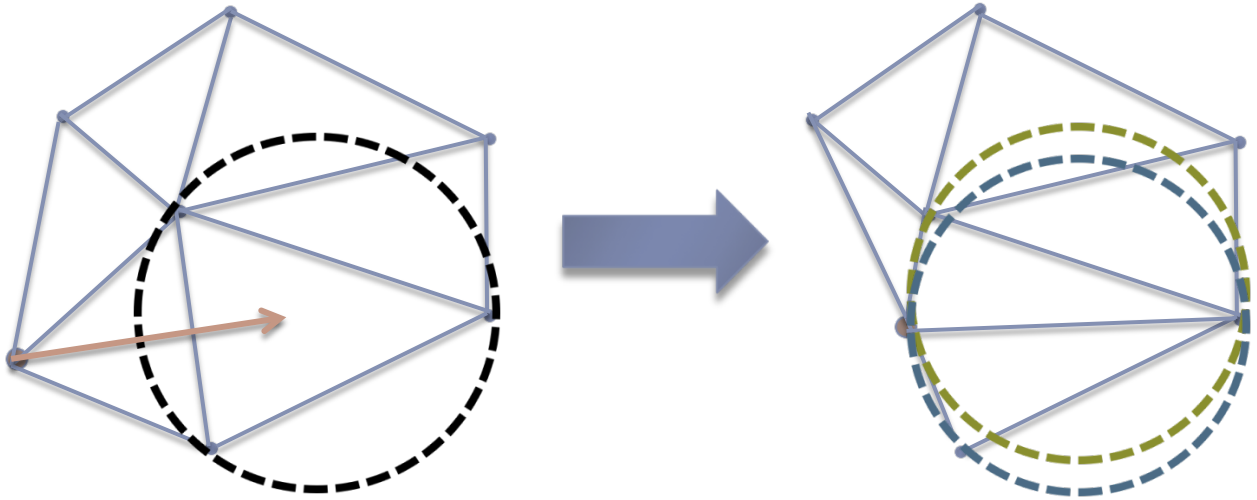


Kinetic Data Structures (KDSs)

- Each point has a flight plan (algebraic expression)
- Flight plans may change
- Computational structure (e.g. Delauney triangulation, lower envelope, etc.) is maintained online
- Certificates guarantee boolean properties
- Certificate failure times are computed and put in a priority queue. Rules are given to update the property on failure.

Kinetic Data Structures Example

- Delauney Triangulation
  - Certificates: empty circle property
  - On certificate failure: edge flip

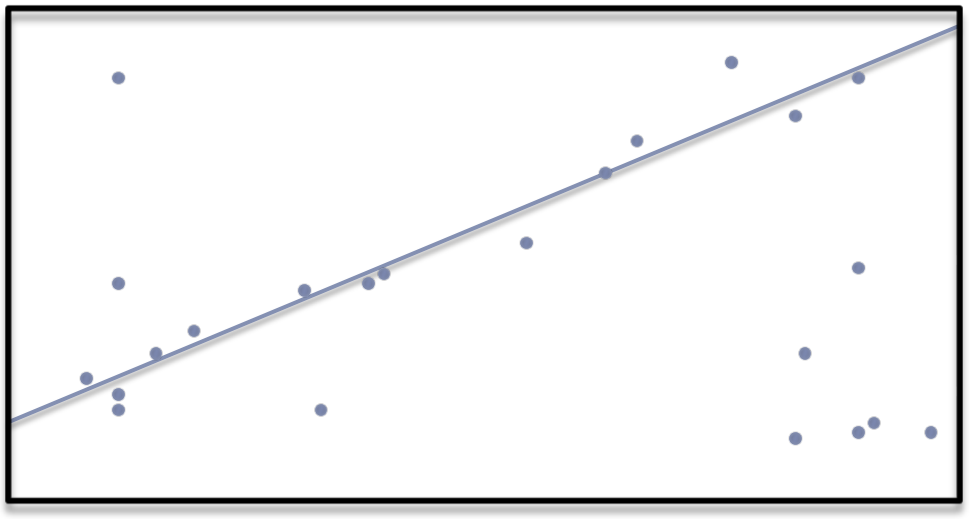


Kinetic Data Structures Analysis

- Responsiveness – certificate update time
- Compactness – total number of certificates
- Locality – certificates per point
- Efficiency – total certificate failures (compared to combinatorial changes)

Robust Statistics

- Describe data trends
- Robust to outliers
- Breakdown point: the minimum fraction (often expressed as a percentage) of outliers that can affect the resulting estimate

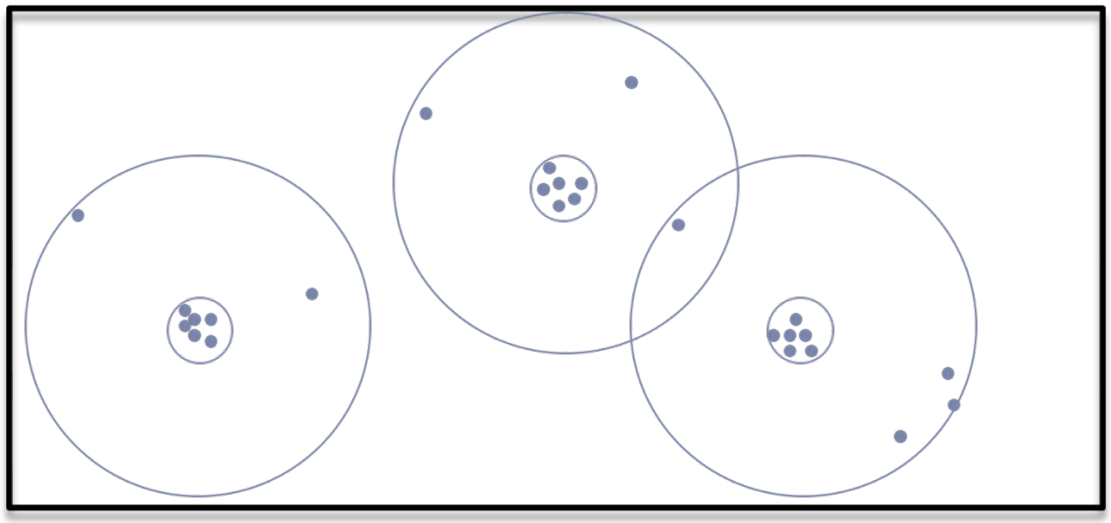


Robust Calculations on Changing Data

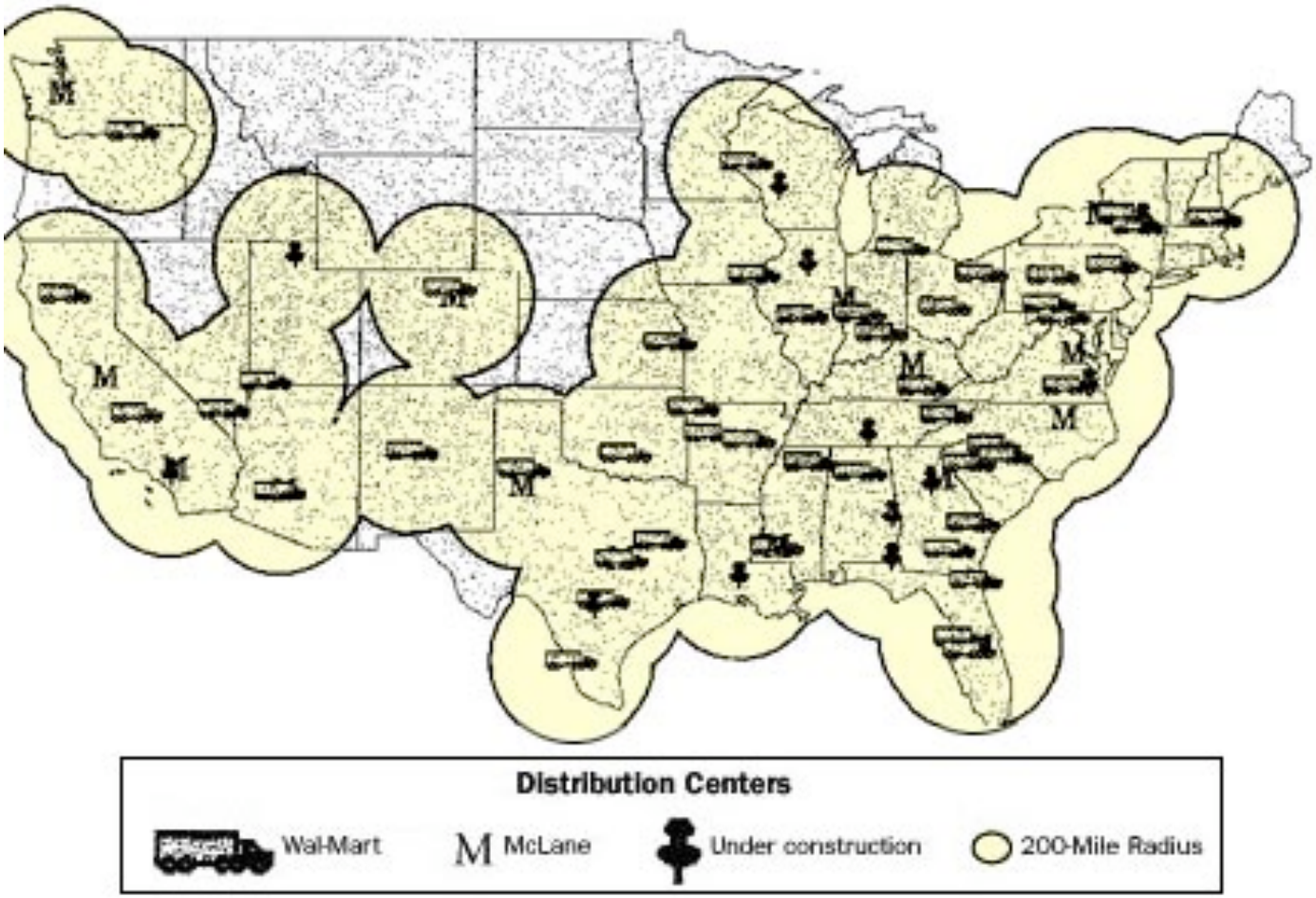
- No previous work in kinetic context
- Approximate function values as they change over time
  (Cormode *et al.* 08, Yi and Zhang 09)
- Regression analyses of data streams
  (Chen *et al.* 02)
- Video segmentation and tracking
  (Mittal and Davis 01)
- Begin by considering robust clustering on KDS…

Preliminary Work #1: Kinetic, Robust, *K*-Center Problem

- *k*-center problem: choose *k* centers that minimize the maximum distance from any point to its closest center
- robust *k*-center problem: allow a fraction (1-*t*) of the points to remain unclustered



Robust *k*-Center Example



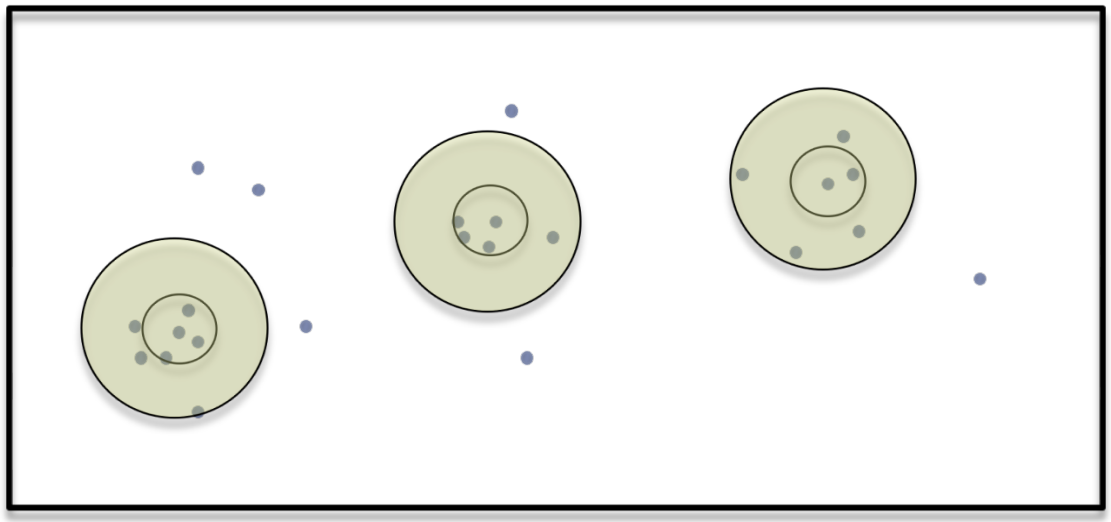
Past Work on *k*-Center

- NP-Hard, Kariv and Hakimi 1979
- 2-approximation, Gonzalez 1985
  - Farthest point algorithm
- 3-approximation, Charikar *et al.* 2001
  - Robust k-center
- 8-approximation, Gao *et al.* 2006
  - Kinetic k-center

Expanded Greedy Algorithm

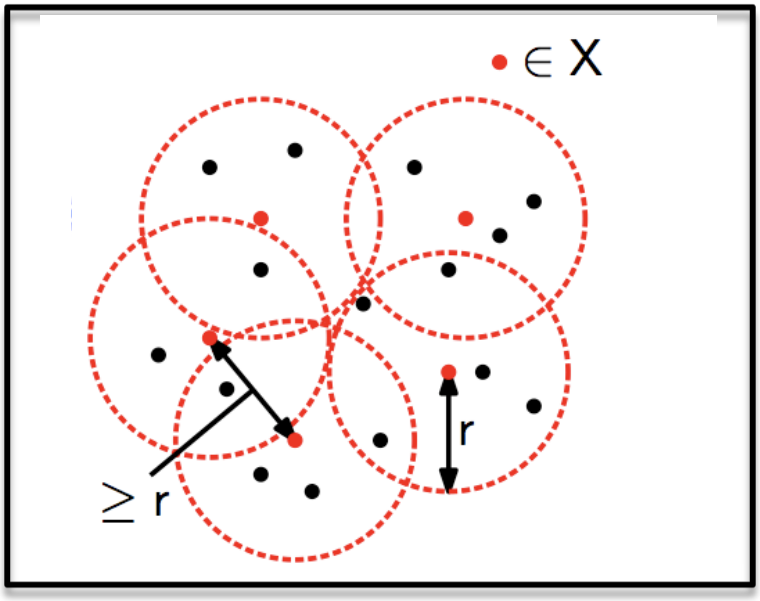
Charikar, Khuller, Mount, Narasimhan 2001

- Choose center that covers the most uncovered points within radius *r*
- Mark all points within *3r* as covered
- Repeat until *k* centers are chosen
- Repeat for all values of *r*, choose minimum that covers *tn*



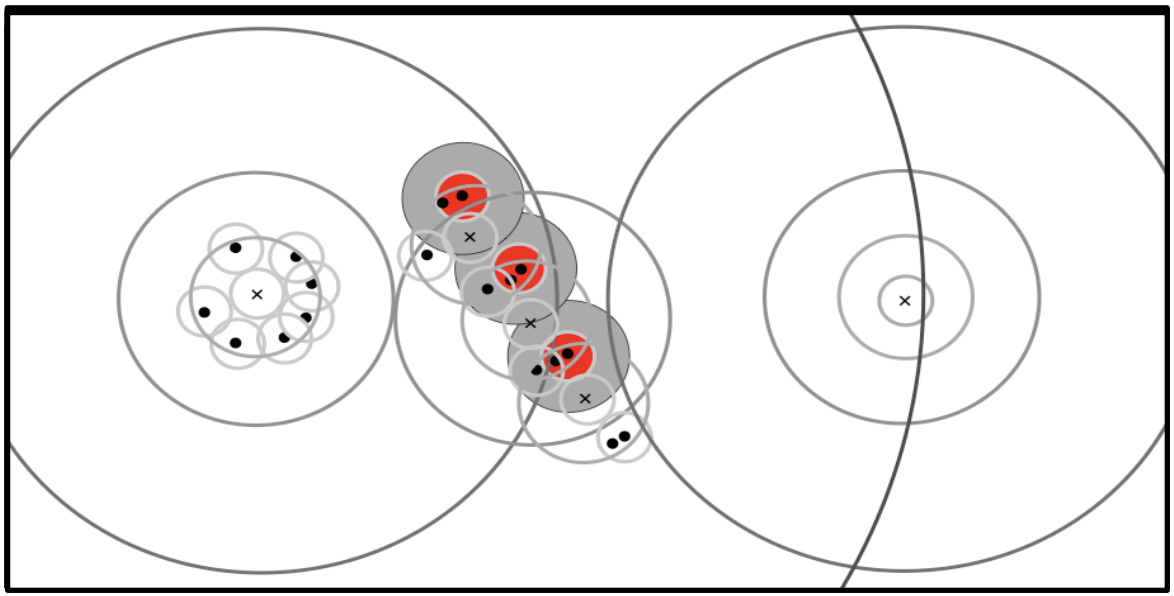
Deformable Spanner
Gao, Guibas, Nguyen 2004

- A sequence of subsets, $S = S_0 \supseteq S_1 \supseteq \ldots \supseteq S_{[\log \alpha]}$
- Each node in S_{i-1} is within 2^i of some node in S_i
- Nodes in S_i are chosen from S_{i-1}
- Neighbors in S^i are within $c2^i$ of each other



Our Initial Kinetic Algorithm

- Run expanded greedy on each level S^i of the deformable spanner for $r=2^i$
- Output the solution from the level that has the minimum radius of the solutions that cover enough points



Technical Issues

Some assumptions made by the expanded greedy algorithm are not valid in this context:

Assumption	Fix
All important radii are considered	Create multiple spanners with varying base distances
All points of the input are candidate centers	Consider points from lower levels
Greedy and expanded disk counts are known exactly	Maintain fuzzy disk counts through range sketching

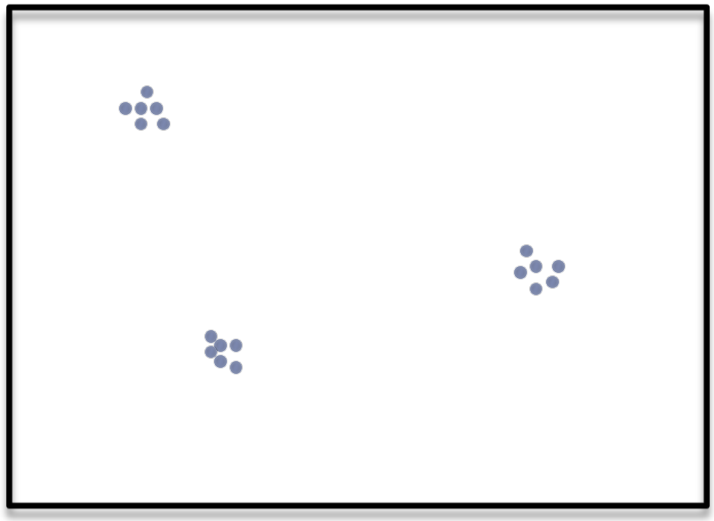
Results

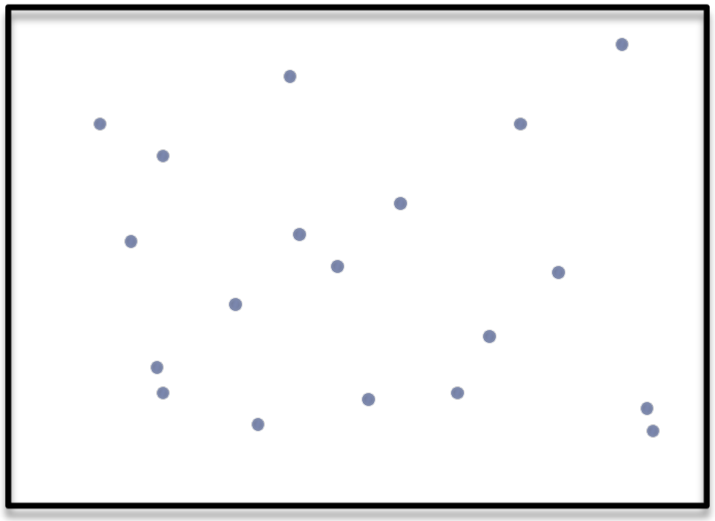
- Discrete: centers taken from input
- Absolute: centers any point in space

- (3+ ε)-approximation algorithm for discrete *k*-center
  - Improves Gao *et al.* 8-approximation
  - Close to Charikar *et al.* 3-approximation
- (4+ ε)-approximation algorithm for absolute *k*-center
  - First absolute solution
- Locality $O(\log \alpha / \varepsilon^{d})$, Compactness $O(n/ \varepsilon^{d+1})$, Responsiveness $O((\log \alpha \log n)/ \varepsilon^{d})$, Efficiency $O(n^2 \log \alpha / \varepsilon)$

Preliminary Work #2: Sensor-Based Framework for Kinetic Data

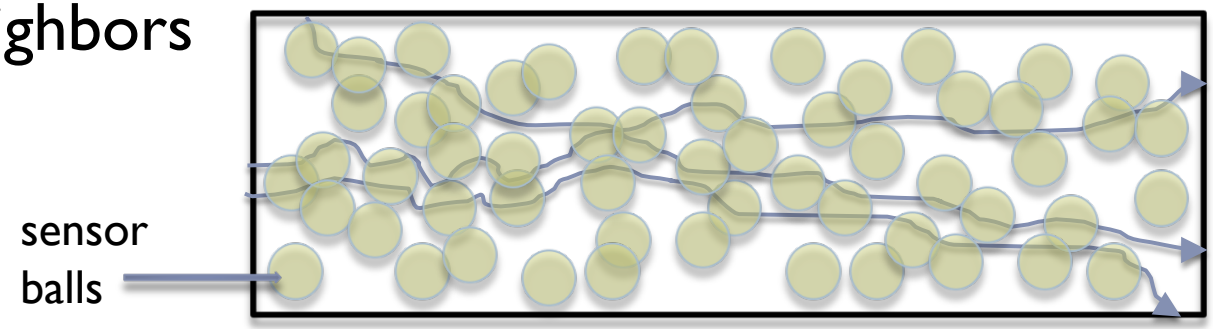
- KDS Limitations
  - Requires advance flight plan knowledge
  - Assumes points move with algebraic motion
  - Efficiency analysis is not motion sensitive





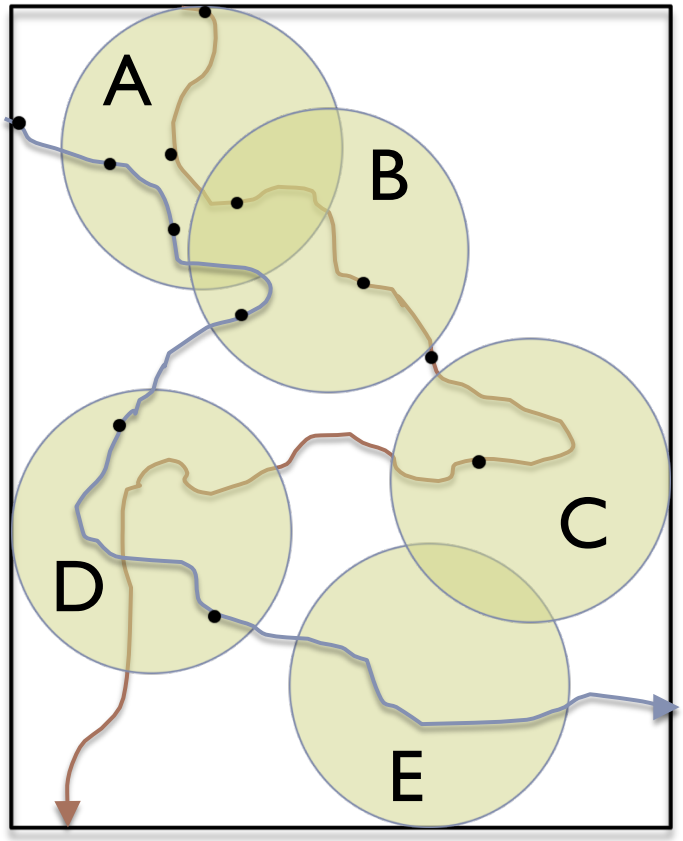
Our Framework

- Densely deployed sensor network (stationary sensors)
- Radius λ defines detection ball around each sensor
- Point motion unrestricted
- No advance knowledge about motion
- Each sensor reports the count of points within λ at each synchronized time step
- *k*-local: Sensor outputs statistically only dependent on *k* nearest neighbors



Data Collection

Data based on underlying geometric motion



Sensor data streams

A	B	C	D	E
1	0	0	0	0
2	0	0	0	0
2	1	0	0	0
0	2	0	0	0
0	0	0	1	0
0	0	1	1	0

time

Entropy

Consider the string generated by a random process…

- Entropy: The information content of a string or a measurement of the predictability of the random process

  $$- \Sigma_x \, pr(x) \log pr(x)$$

  - Example: A weighted coin that's always heads vs. a normal coin:

    $$-(1 \log 1) = 0 \quad \text{vs.} \quad -(\tfrac{1}{2} \log \tfrac{1}{2} + \tfrac{1}{2} \log \tfrac{1}{2}) = 1$$

- Joint entropy: The entropy for joint probabilities of a set of events occurring
- Normalized entropy: bits to encode each character, entropy/n for strings of length n

Data Compression

- Encoded strings are compressed to a shorter length than the original
- Optimal compression algorithms encode strings to a length equal to the entropy of their underlying process
- Optimal compression algorithms over a set of strings return an encoding the length of the joint entropy

Lempel-Ziv Sliding-Window Algorithm

- Look back within window to find repeating phrase
- Store pointer to most recent previous occurrence
- Example:

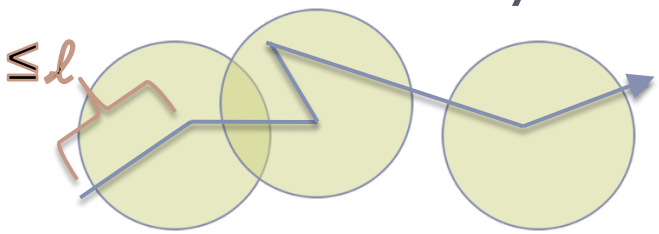
  AABAAB → (0,A)(1,1)(0,B)(3,3)

- Shown to be an entropy encoding algorithm

  (Wyner, Ziv 94)

Encoding Efficiency Comparison to KDS

- Does the discrete nature of the framework unreasonably increase the storage size?
- Compare to KDS with these assumptions:
  - Objects move for some long time *T*
  - Objects move at some constant velocity in $[v_{min}, v_{max}]$
  - Objects change velocity occasionally so that the maximum length of any path line segment is at most ℓ
  - Object coordinates are each represented with at most *b* bits
  - Sensors are not too densely clustered, but are still *k*-local



Preliminaries

- B_{KDS}: KDS encoding size per time step
- n moving objects, S sensors

- Joint entropy chain rule ($X = \{X_1, X_2, \ldots, X_S\}$):
  - $H(X) = H(X_1)+H(X_2|X_1)+\ldots+H(X_S|X_1,\ldots,X_{S-1})$

- *k*-local entropy (H_k): normalized joint entropy of a set of streams that are only dependent on up to *k* streams from their *k* nearest neighbors

- Optimal compression of sensor streams is $H(X) = H_k(X)$

Comparison Theorem

- Theorem: $H_k(n, v_{\min}, \ell) = O(B_{\text{KDS}}(n, v_{\min}, \ell))$

Proof sketch:

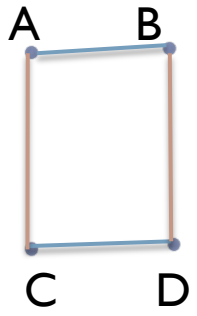
- Lower bound: $B_{\text{KDS}}(n, v_{\min}, \ell) \geq \frac{4n \cdot v_{\min} \cdot b}{\ell}$
  - For each point and each velocity change, *4b* bits are needed
  - For each point, the total number of segments per time step is v_{min}/ℓ
- Upper bound: $H_k(n, v_{\min}, \ell) = O(B_{\text{KDS}}(n, v_{\min}, \ell))$
  - For a subsegment $\ell' < \ell$, all sensor regions that detect the point are in each other's k nearest neighbors (by sparseness)
  - The joint entropy of these sensors' streams is at most 4b (2^{4b} possible lines)

Data Compression Algorithm

- The optimal bound is the joint entropy of the set of streams
- Compressing each separately doesn't reach this bound
- Compressing all together reaches bound, but only in the limit, so it's not practical
- Since $H(X)=H_k(X)$ we want groups of roughly k streams

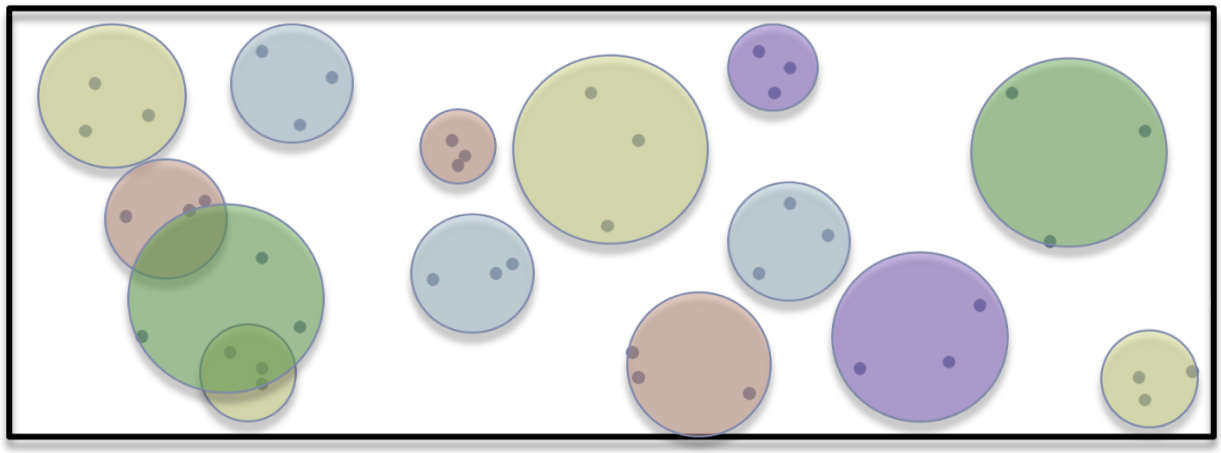
Data Compression Algorithm: Partitioning Lemma

- *k*-clusterable: A point set that can be clustered into subsets of size at most $k+1$ so that if p and q are within each other's k nearest neighbors then they are in the same cluster.
  - In a graph with edges between two mutually close points, all points in a connected component must be in the same cluster.
- Example:
  - 1-clusterable (cluster size is 2)
    - $NN_1(A)=\{B\}$, $NN_1(B)=\{A\}$, $NN_1(C)=\{D\}$, $NN_1(D)=\{C\}$
  - Not 2-clusterable (no size 3 connected component)
    - $NN_2(A)=\{B,C\}$, $NN_2(B)=\{A,D\}$,
    - $NN_2(C)=\{A,D\}$, $NN_2(D)=\{B,C\}$,



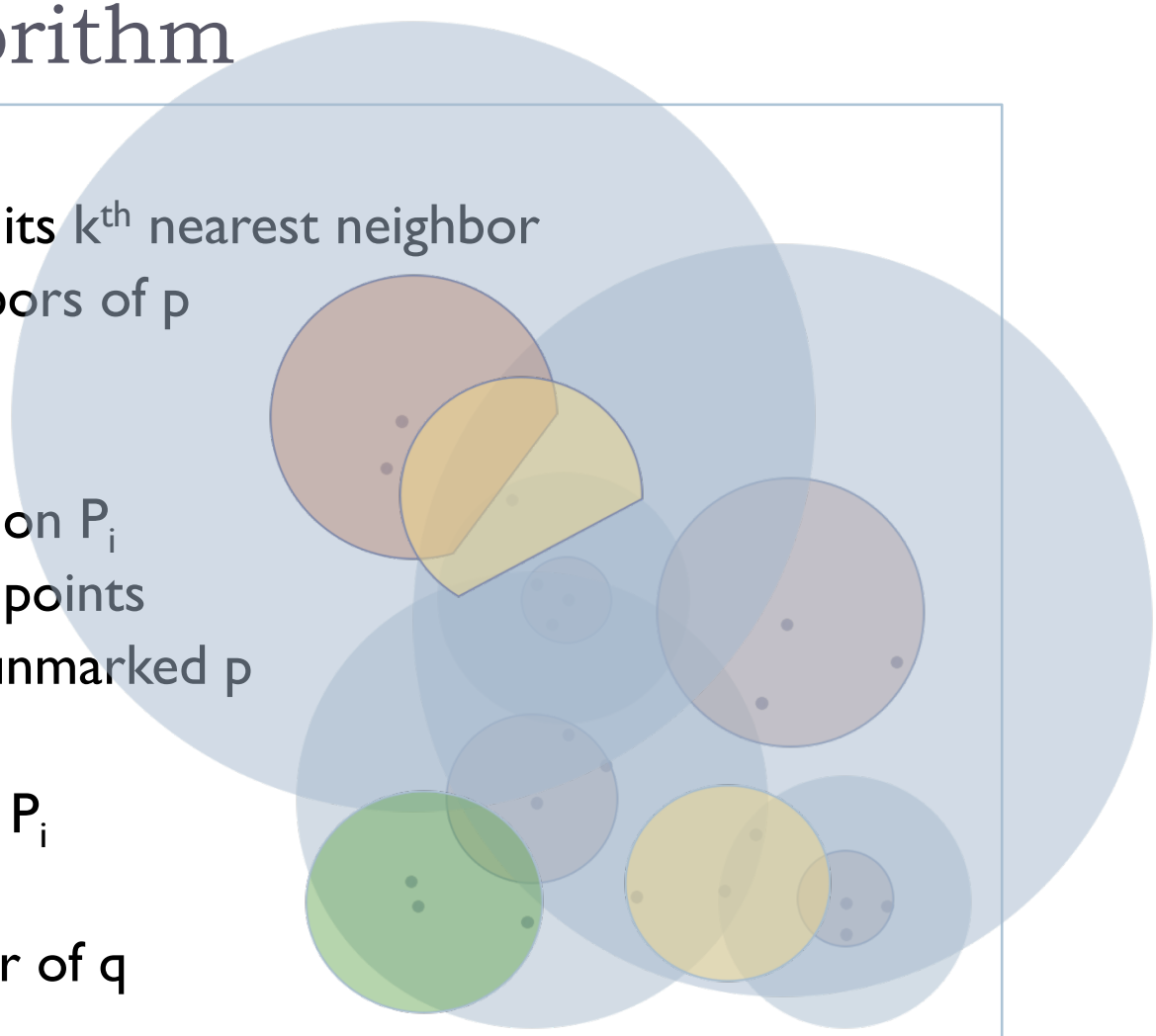
Data Compression Algorithm: Partitioning Lemma

- Lemma: There exists an integral constant *c* such that for all *k*>0 any point set can be partitioned into *c* partitions that are each *k*-clusterable.



Partitioning Algorithm

for all points find
  $r_k(p)$ = distance from p to its k^{th} nearest neighbor
  $NN_k(p)$ = k nearest neighbors of p
while P is nonempty
  unmark all points in P
  create a new empty partition P_i
  while there are unmarked points
    r = minimum $r_k(p)$ for unmarked p
    q = point with $r_k(q) = r$
    add q and its $NN_k(q)$ to P_i
      and remove from P
    mark all points within 3r of q
return $\{P_1, P_2, \ldots, P_c\}$



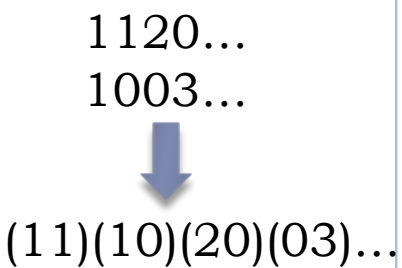
Partitioning Lemma Proof Sketch

- By nature of marking and order of clustering, all partitions are *k*-clusterable
  - No points in the partition are within 2r of a radius r cluster
  - Increasing $r_k(p)$ choices ensure non-mutual $NN_k(p)$ relations are separated into different clusters
- There are *c* partitions
  - In each round every point is either marked or removed from P
  - A point *p* is marked only by points within 12 $\min(\{r_k(p)\})$
  - Points that mark *p* are separated by distance $\min(\{r_k(p)\})$
  - Packing argument bounds the number of times a point can be marked to $c = O(1 + 12^{O(1)}) = O(1)$

Data Compression Algorithm

- Partition and cluster the sensors, then compress

```
for each partition P_i
    for each cluster in P_i
        combine the cluster's streams into
        one with longer characters
return the union of the compressed streams
```



- Proof Sketch:
  - The joint entropy of the streams is the optimal length
  - Sensor outputs are *k*-local, so each compressed partition is the optimal length
  - There are *c* partitions, so the total length is *c*•optimal

Proposed Work

- Goal: Answer statistical questions about kinetic data with provable efficiency

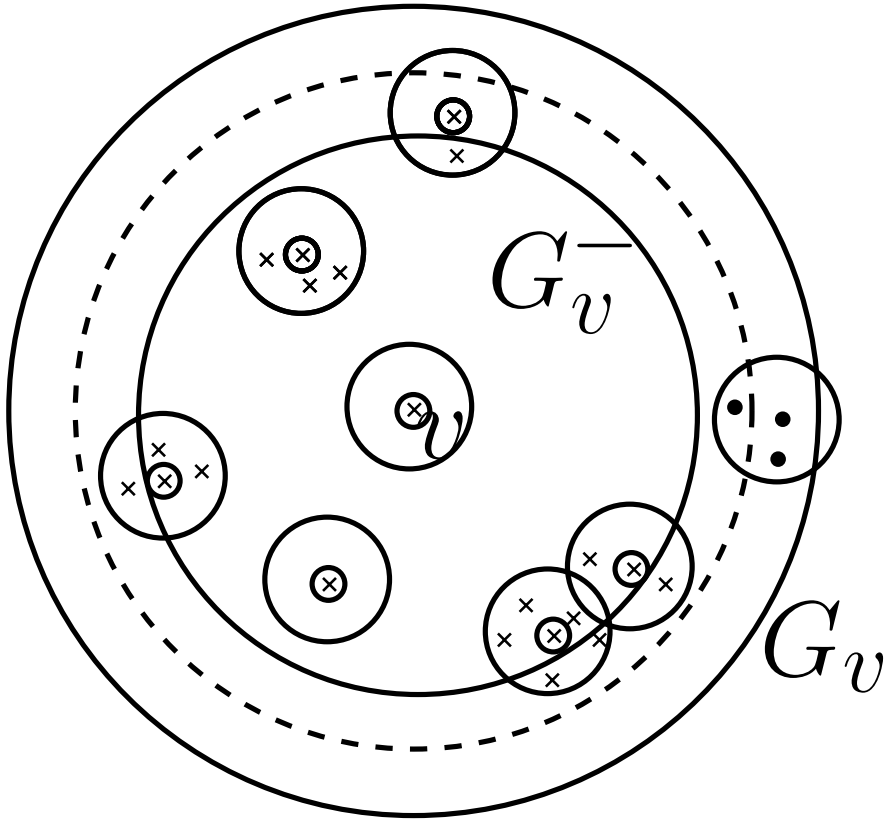
- Short term:
  - retrieval on sensor-based framework
    - e.g. counts for all k nearest neighbors of a sensor
  - *k*-center problem on sensor-based framework

- Long term:
  - other statistical problems on sensor-based framework
  - lossy compression
  - experimental evaluation
  - other frameworks

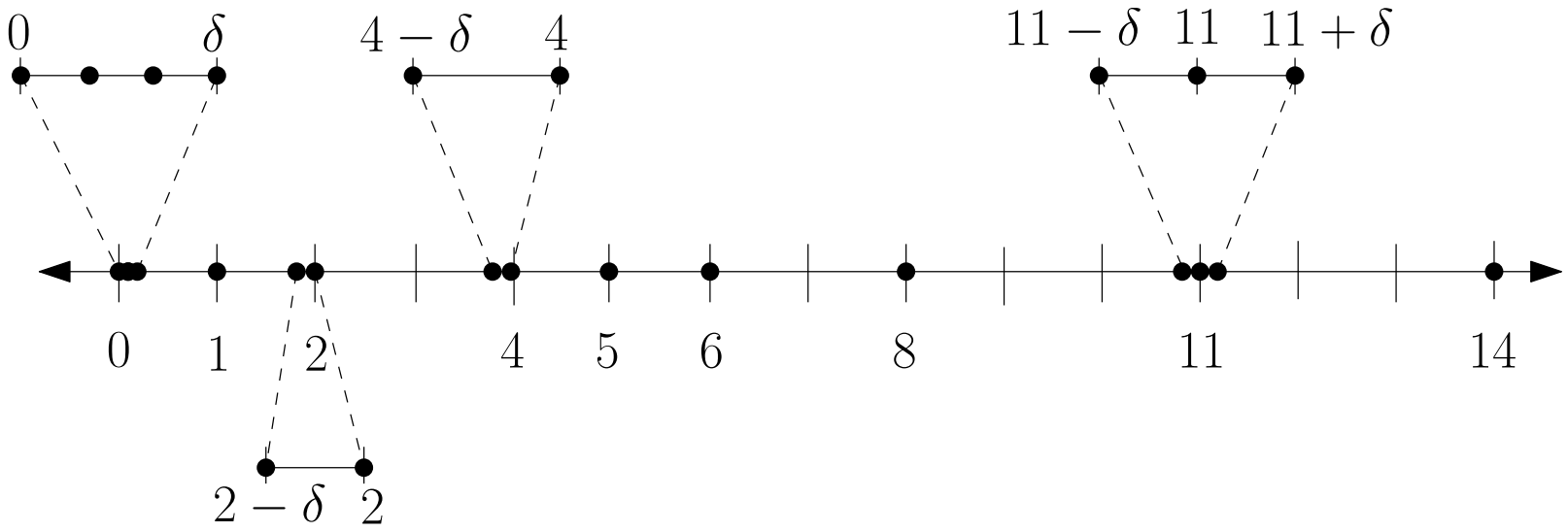
Thank you!

Questions?

Range Sketching



(3-eps) Tight Example, k=2, tn=11



Partitioning Lemma Proof Figure