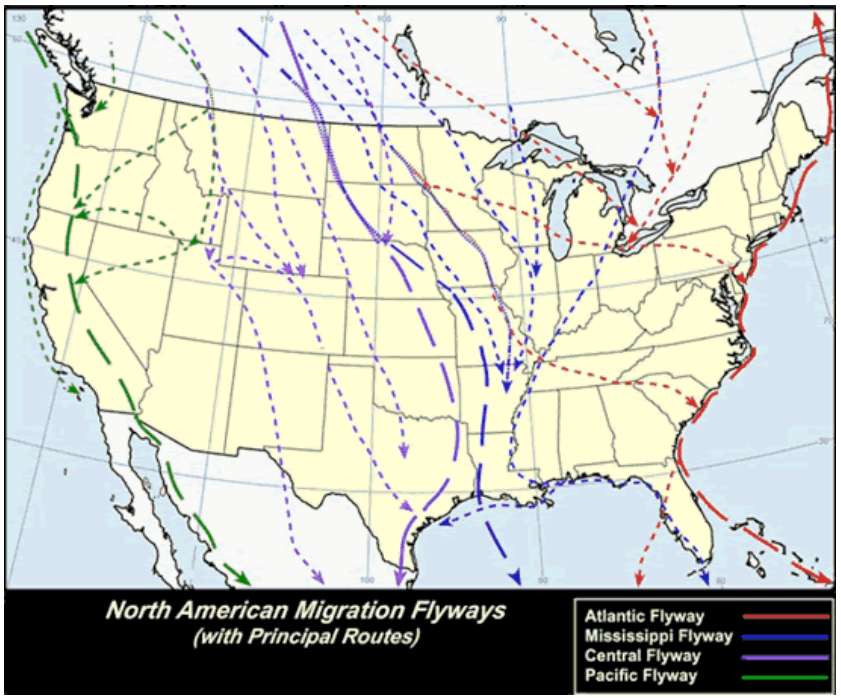
Compressing Kinetic Data From Sensor Networks

Sorelle A. Friedler (Swat '04)
Joint work with David Mount
University of Maryland, College Park

Motivation

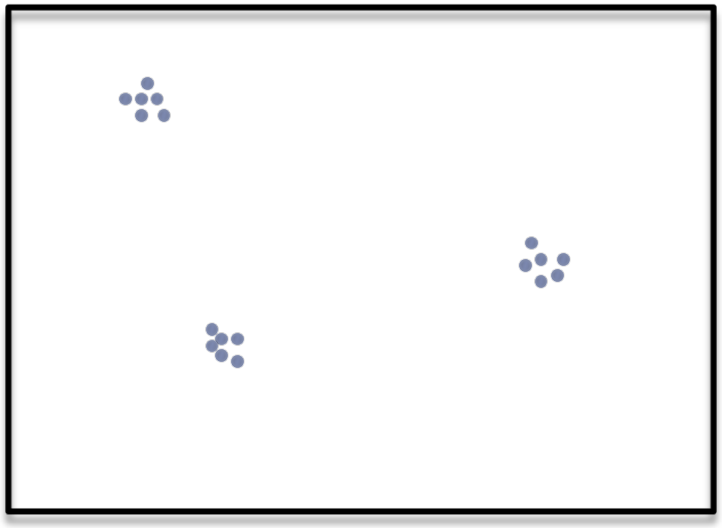

North American Migration Flyways
(with Principal Routes)
Atlantic Flyway
Mississippi Flyway
Central Flyway
Pacific Flyway

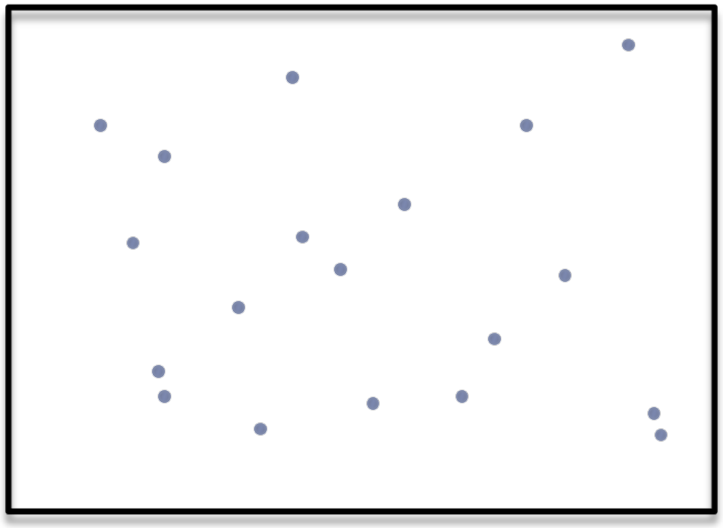
Motivation

- Computer Science
  - Graphics: Image and video segmentation, animation
  - Databases: Maintenance over time
  - Sensor Networks: Data analysis
- Physics
  - Simulations
- Biology
  - Mathematical ecology: Migratory paths
  - HIV strain analysis
- Engineering
  - Traffic patterns and identification

Motivation

- Develop a framework for kinetic data from sensors
  - No advance object motion knowledge
  - No restrictions on object motion
  - Reasonable assumptions of what a sensor can know
  - Efficiency analysis that is motion sensitive





Motivation

- Kinetic data: data generated by moving objects
- Sensors collect data
- Large amounts of data
- Want to analyze it later
- Don't know what questions we'll want to ask in advance

- Lossless data compression

Entropy

Low Entropy	High Entropy

Entropy

Low Entropy	High Entropy
Point motion is predictable	Point motion is close to random
Representing the motion of the system requires “few” bits	Representing the motion of the system requires “many” bits

Entropy

- Given a random process, entropy measures its
  - uncertainty
  - information content
  - lack of predictability

$- \Sigma_x \, pr(x) \log pr(x)$ for all outcomes x

Example

- A weighted coin that's always heads:

  $-(1 \log 1) = 0$

- A normal coin:

  $-(\frac{1}{2} \log \frac{1}{2} + \frac{1}{2} \log \frac{1}{2}) = 1$

Entropy

- Represents the number of bits to encode a character of a string generated by a random process

- Normalized entropy: for strings of length n, $1/n$ entropy

$- {}^{1}/_{n} \Sigma_{x} \, pr(x) \log pr(x)$ for all events x, characters of the string

Example

- A weighted coin that's always heads: HH

  $-\frac{1}{2}(1 \log 1) = 0$

- A normal coin: HT

  $-\frac{1}{2}(\frac{1}{4} \log \frac{1}{4} + \frac{1}{4} \log \frac{1}{4} + \frac{1}{4} \log \frac{1}{4} + \frac{1}{4} \log \frac{1}{4}) = 1$

Entropy

- Measures the joint information content of two strings generated by random processes
- Joint entropy: The entropy based on joint probabilities of a set of events

  $- \Sigma_{x,y} \, pr(x,y) \log pr(x,y)$ for all events x,y
- Normalized joint entropy: $-1/n \, \Sigma_{x,y} \, pr(x,y) \log pr(x,y)$

WEATHER		NYC	
		Sunny	Rainy
Philly	Sunny	50%	12.5%
	Rainy	12.5%	25%

Example: Joint entropy

$-(\frac{1}{2} \log \frac{1}{2} + \frac{1}{8} \log \frac{1}{8}$

$+ \frac{1}{8} \log \frac{1}{8} + \frac{1}{4} \log \frac{1}{4}) = 1\frac{3}{4}$

Data Compression

- Encoded strings are compressed to a shorter length than the original
  - "Swarthmore College" ➔ "SC", [(S, Swarthmore), (C, College)]
  - "Swarthmore College" ➔ "Swat", [(Swat, Swarthmore College)]

Consider the string generated by a random process…

- Optimal compression algorithms over a string achieve a per bit encoding rate equal to the normalized entropy
- Optimal compression algorithms over a set of strings achieve a per bit encoding rate equal to the normalized joint entropy

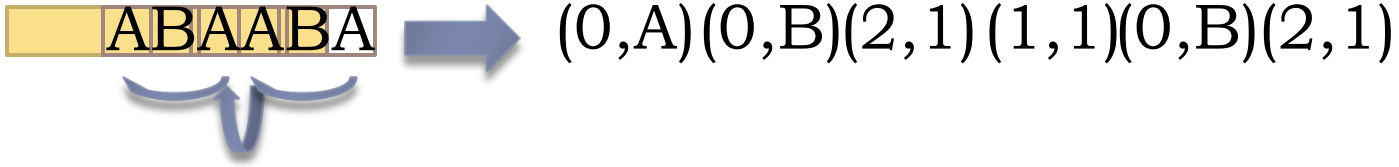
[Shannon48]

Data Compression Options

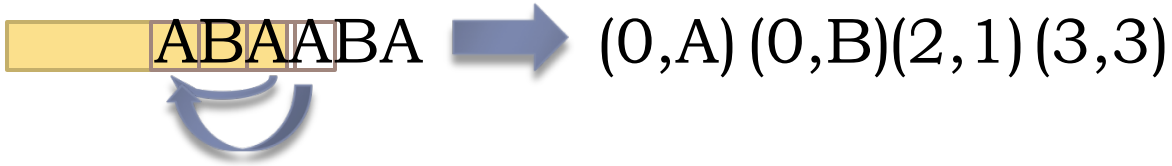
Lossless	Lossy
▸ Data is completely retrievable ▸ Compression bounds are theoretically provable ▸ Sliding-window Lempel-Ziv algorithm 1977 (used by gzip)	▸ Some data may be lost ▸ Can compress the data to fit in the space you have

Lempel-Ziv Sliding-Window Algorithm

- Look back within window to find repeating phrase
- Store pointer to most recent previous occurrence
- Example:
  - window size: 2



  - window size: 3



  - window size: 3

ABAABAABA… → (0,A)(0,B)(2,1)(3,6)

- Shown to be an entropy encoding algorithm [WynerZiv94]

Lempel-Ziv Sliding-Window Algorithm

- Decoding takes an encoded string and translates it back to the original.

- Example:
  - (0,A)(0,B)(2,1)(3,6) → ABAABAABA
  - (0,A)(1,1)(0,B)(1,1)(4,2)(5,2) → AABBAAAB

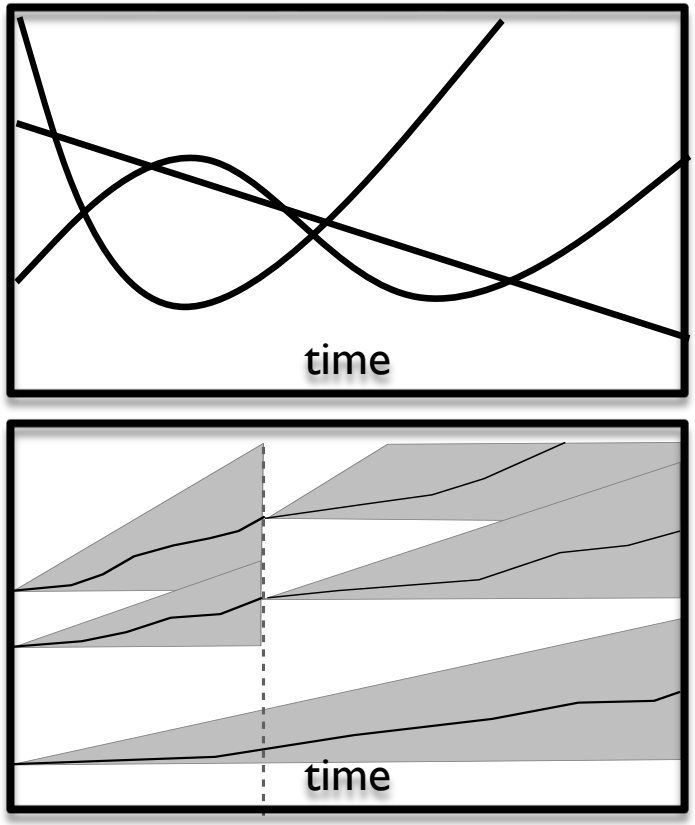
Motivation

- Kinetic data: data generated by moving objects
- Sensors collect data
- Large amounts of data
- Want to analyze it later
- Don't know what questions we'll want to ask in advance

- Lossless data compression

Existing Frameworks for Kinetic Data

- 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



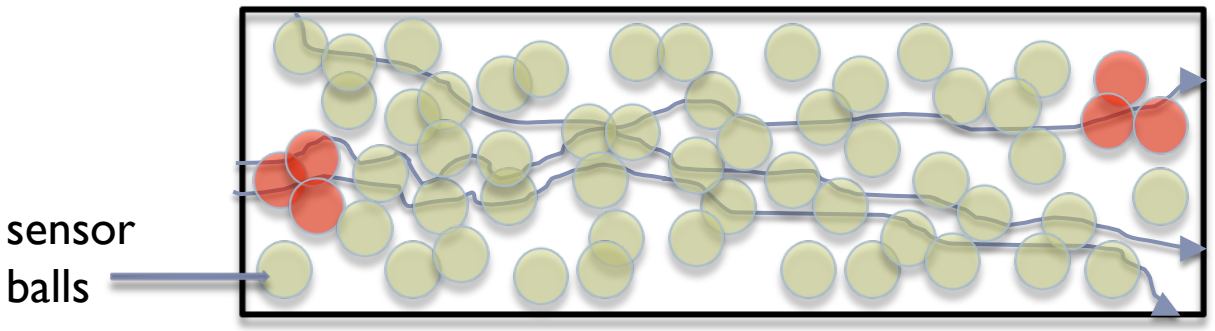
- Kinetic Data Structures (Basch, Guibas, Hershberger 97)
  - Points have flight plans (algebraic expressions) that can change

Existing Frameworks for Sensor Data

- Minimal sensor assumptions [GandhiKumarSuri08]
  - Sensors can count objects within their detection region
- Traffic detection assumptions [GuittonSkordylisTrigoni07]
  - Sensors can calculate traffic flow (cars/time), occupancy (cars/area)
- Exact state of object knowledge [Kastrinaki03 Survey]
  - Sensors can calculate object speed, change in angle, etc

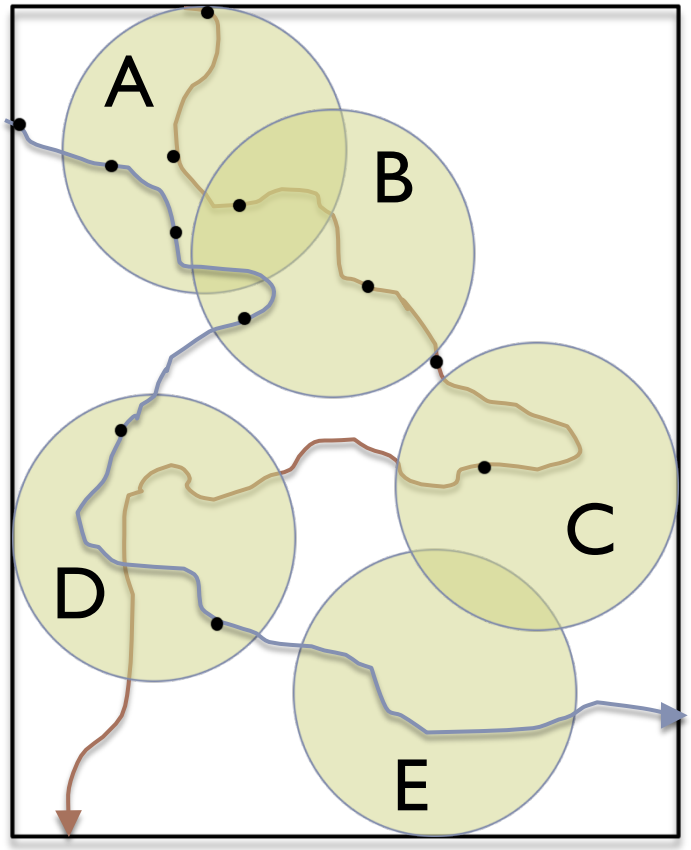
Our Framework

- Detection region around each sensor (stationary sensors)
- Point motion unrestricted
- No advance knowledge about motion
- Each sensor reports the count of points within its region 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

What is Optimal?

- Optimal: the smallest lossless encoding of the sensor data
- Joint entropy chain rule ($\mathbf{X} = \{X_1, X_2, \ldots, X_S\}$):
  - $H(\mathbf{X}) = H(X_1)+H(X_2|X_1)+\ldots+H(X_S|X_1,\ldots,X_{S-1})$
- *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 $H(\mathbf{X}) = H_k(\mathbf{X})$

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 the window size necessary to achieve the needed repetition is too large to be practical

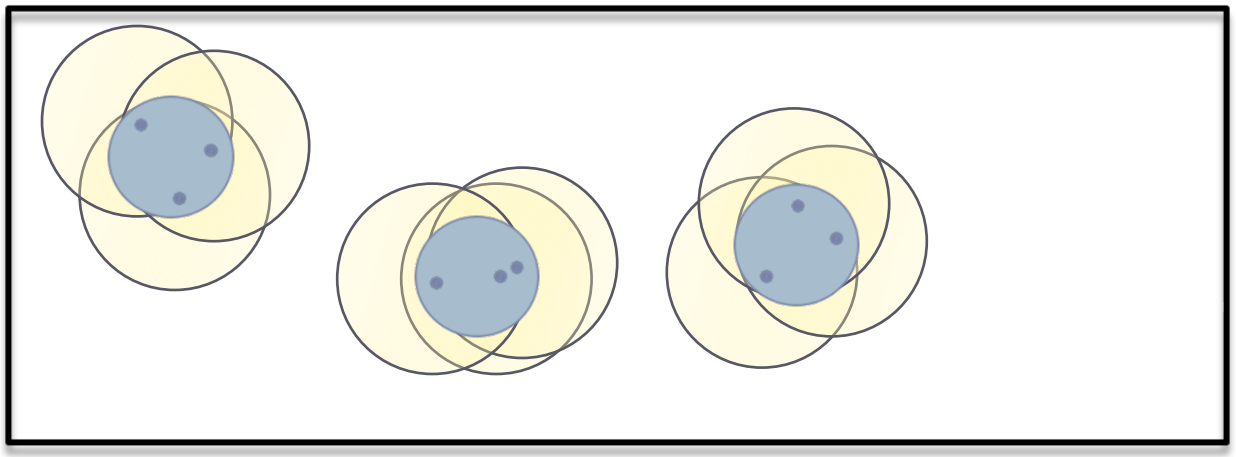
- Key: Need statistically dependent streams to be compressed together

- Since $H(\mathbf{X})=H_k(\mathbf{X})$ 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 among each other's k nearest neighbors then they are in the same cluster.

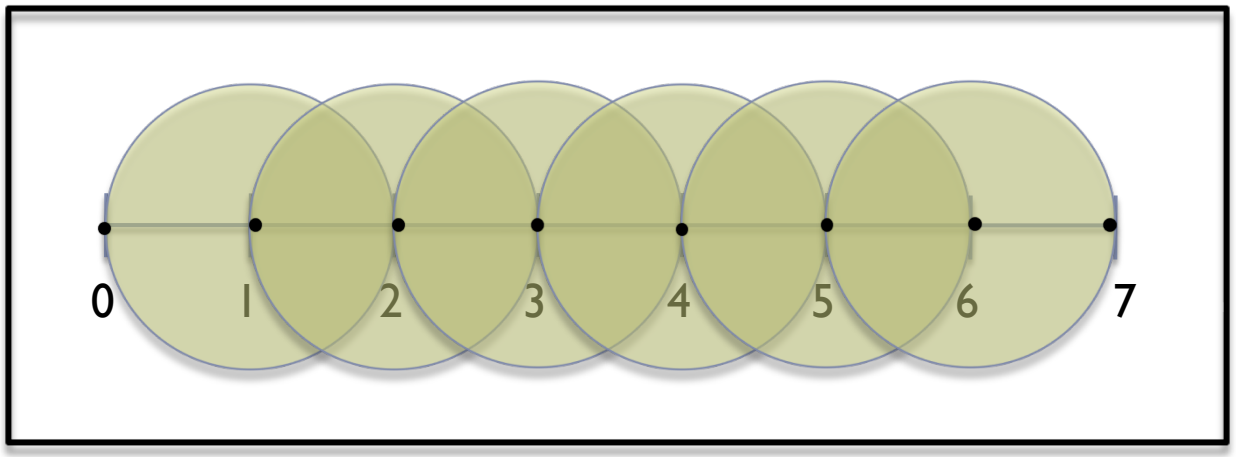
2-clusterable example



Data Compression Algorithm: Partitioning Lemma

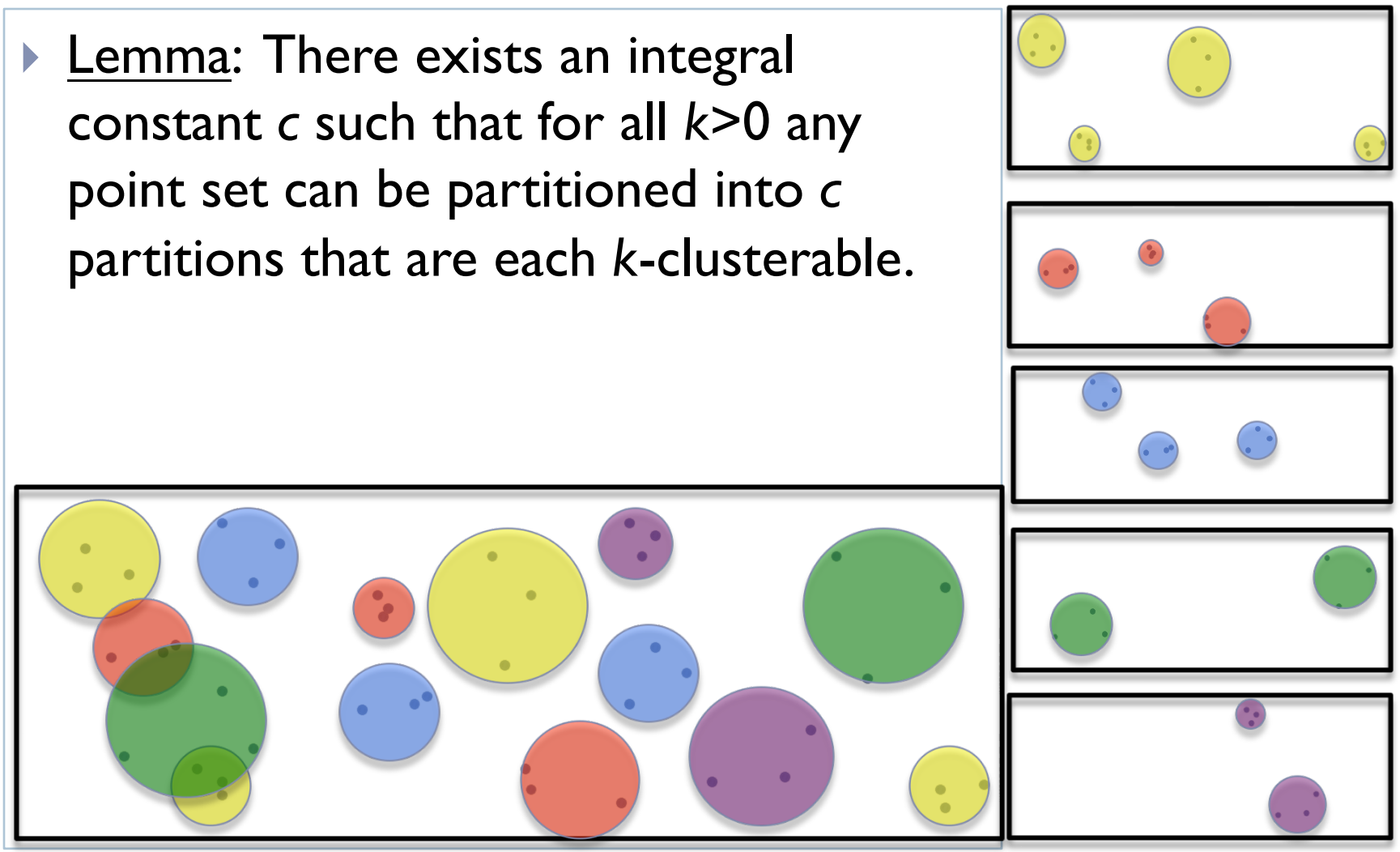
- *k*-clusterable: A point set that can be clustered into subsets of size at most *k+1* so that if *p* and *q* are among each other's k nearest neighbors then they are in the same cluster.

not 2-clusterable example



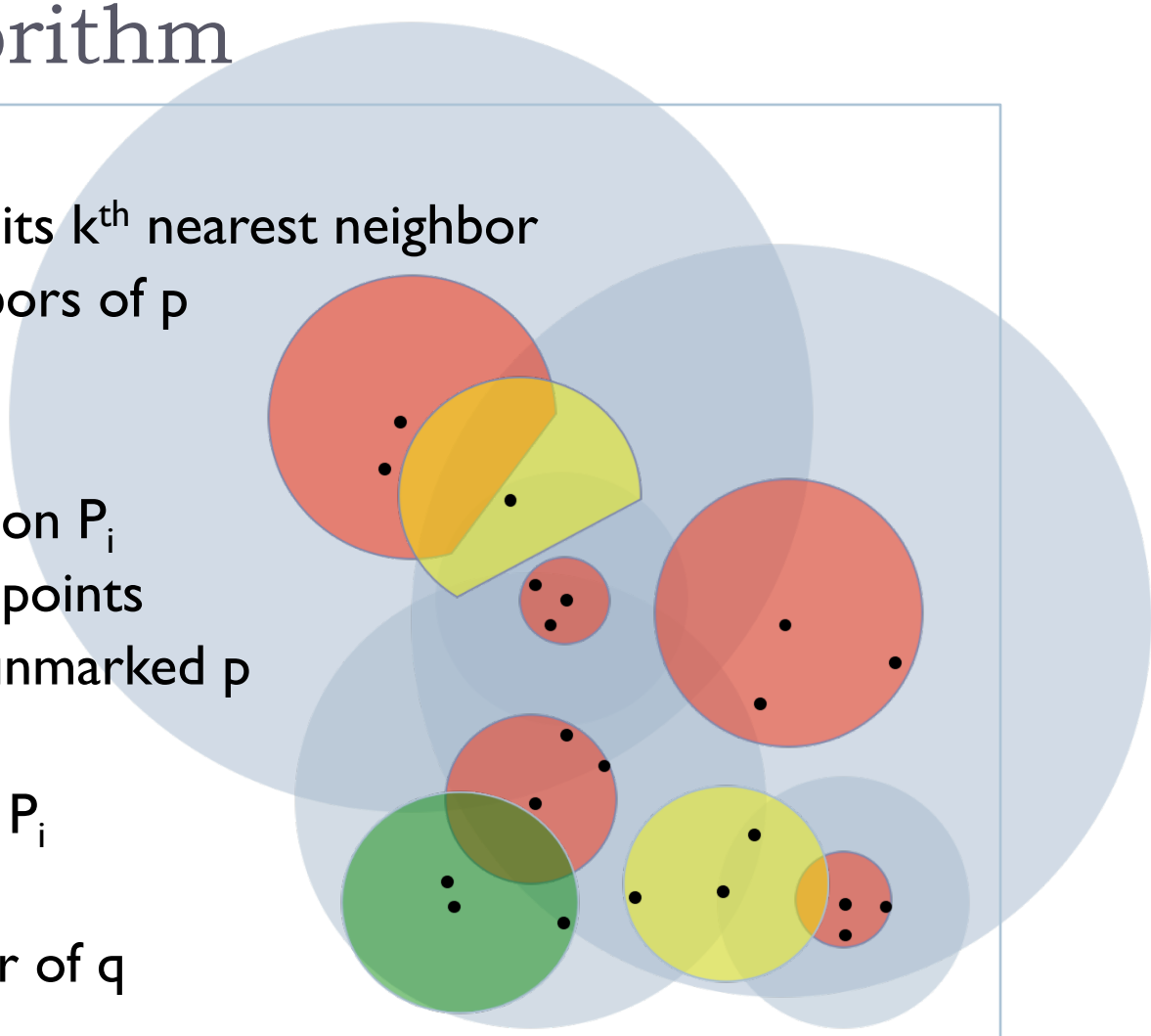
Data Compression Algorithm: Partitioning Lemma

- 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.



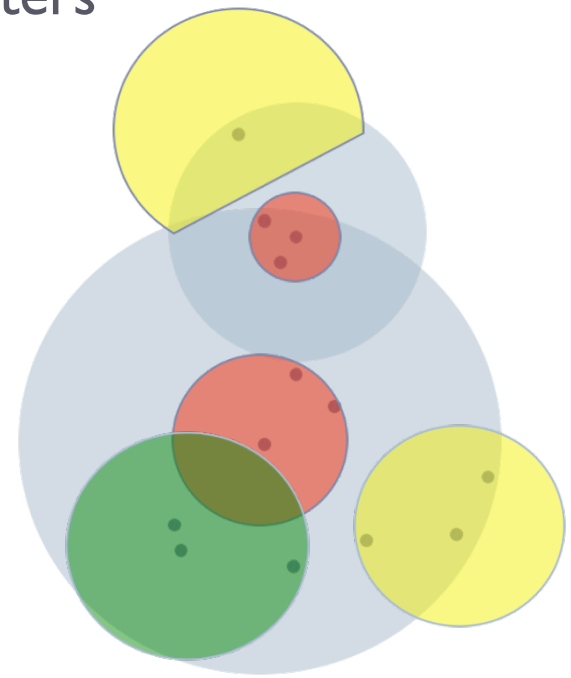
Partitioning Algorithm

for all points in P find
  $r_k(p)$ = distance from p to its k^{th} nearest neighbor
  $NN_k(p)$ = k nearest neighbors of p
while P (points) 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 $\{P_1, P_2, \ldots, P_c\}$



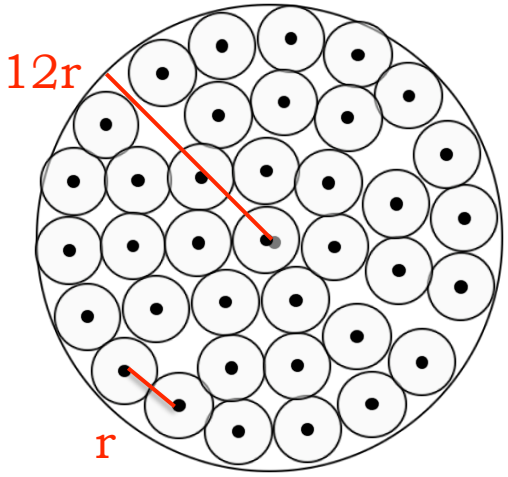
Partitioning Lemma Proof Sketch

- By nature of marking and order of clustering, all partitions are *k*-clusterable
  - Increasing $r_k(p)$ choices ensure non-mutual $NN_k(p)$ relations are separated into different clusters



Partitioning Lemma Proof Sketch

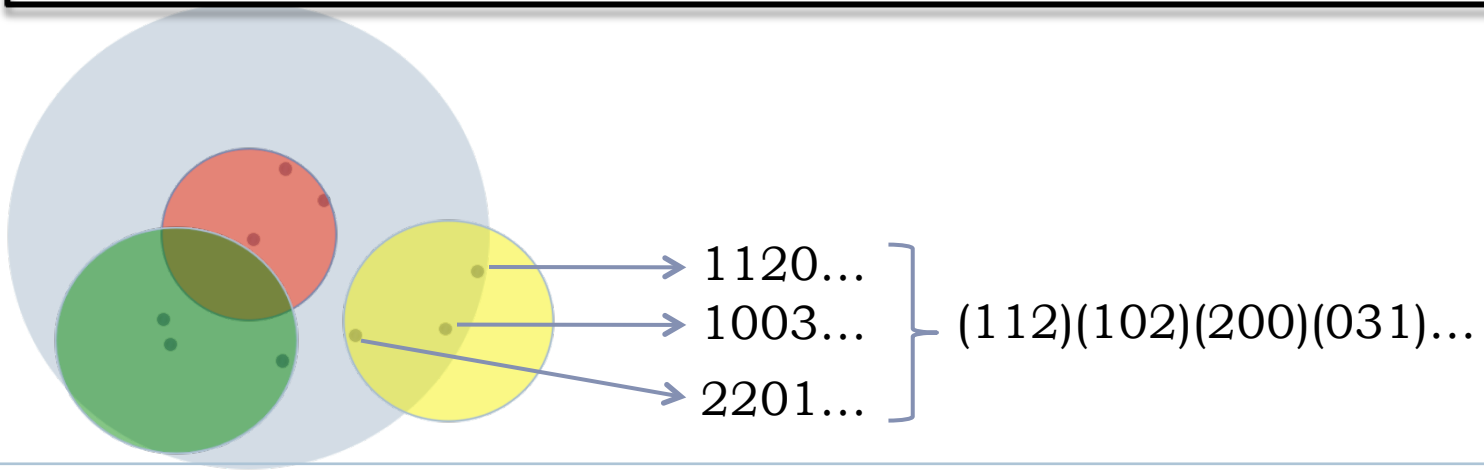
- There are *c* partitions
  - In each round every point is either marked or removed from P
  - By bounding the outer radius that can mark a point and the separation distance between such points, we can use a packing argument to bound the number of times a point can be marked to $c = O(1 + 12^{O(1)}) = O(1)$.



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 and compress it
return the union of the compressed streams
```



Data Compression Algorithm

- Proof Sketch:
  - The joint entropy of the streams is the optimal length
    - Recall: $H(\mathbf{X}) = H_k(\mathbf{X})$
  - Sensor outputs are *k*-local, so each compressed partition is the optimal length:

    statistically dependent streams are compressed together
  - There are *c* partitions, so the total length is *c* times optimal
    - Recall: c is O(1)

Summary of Results

- Framework for kinetic sensor data
  - No assumptions about object motion or advance knowledge
  - Motion sensitive analysis
  - Relies on minimal sensor abilities
- Lossless compression algorithm that compresses the data to $c\ H(X)$, which is $O(\text{optimal})$
  - Assumes the sensor outputs are only dependent on their k nearest neighbors
  - Assumes the sensor outputs can be modeled by an underlying random process

Recent and Future Work

- Extend analysis of compression algorithm to consider empirical entropy (no underlying random process) ✓
- Retrieval without decompressing the data ✓
  - Range searching
  - E.g. given a time period and spatial range, what is the aggregated count?
- Statistical analysis without decompressing the data
- Lossy compression
- Experimental evaluation
- Application in non-sensor contexts

Thank you!

Questions?