Geometric Algorithms for Objects in Motion

Dissertation Defense

Sorelle Friedler

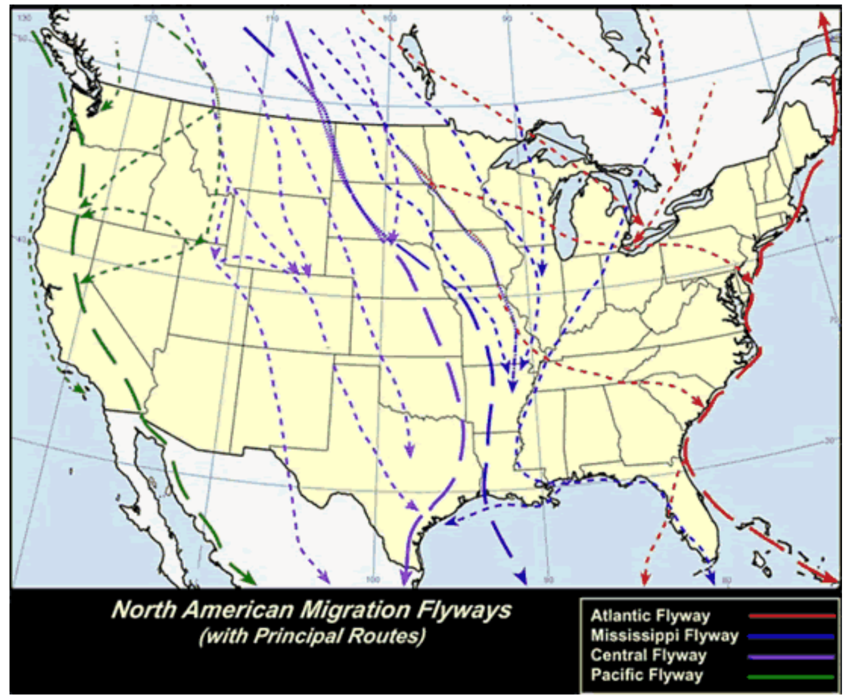
Committee:

David Mount (Chair), William Gasarch

Samir Khuller, Steven Selden, Amitabh Varshney

July 30, 2010

Motivation

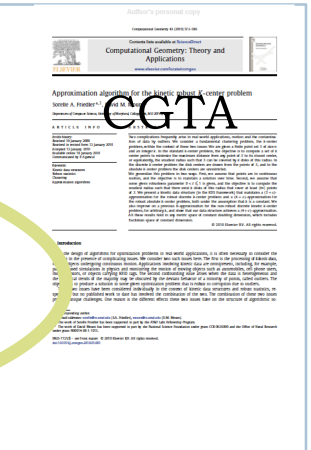

North American Migration Flyways
(with Principal Routes)
Atlantic Flyway
Mississippi Flyway
Central Flyway
Pacific Flyway

Motivation

- Computer Science
  - Graphics: Image / video segmentation and compression
  - Databases: Maintenance over time
  - Sensor Networks: Data analysis
  - Cell phone users: Motion data analysis
    - 4.6 billion subscribers worldwide (in 2009)
    - 4.1 billion text messages per day in the US (in 2009)
- Biology
  - Mathematical ecology: Migratory paths, invasive species
  - Genomic data analysis: HIV strain analysis
- Engineering
  - Traffic patterns and identification

Outline

- Kinetic Robust K-Center

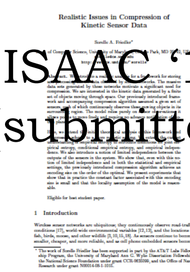


CGTA

- Sensor-Based Framework Kinetic Data Compression

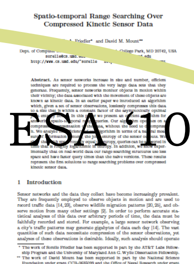
AlgoSensors '09

- Realistic Issues in Kinetic Sensor Data Compression



ISAAC '10 (submitted)

- Spatio-temporal Range Searching

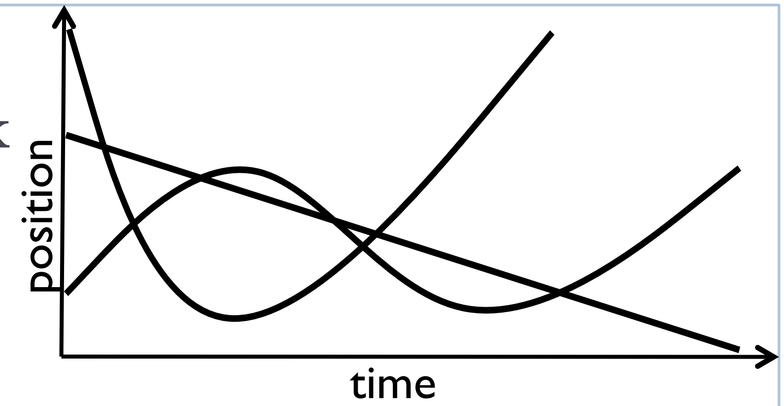


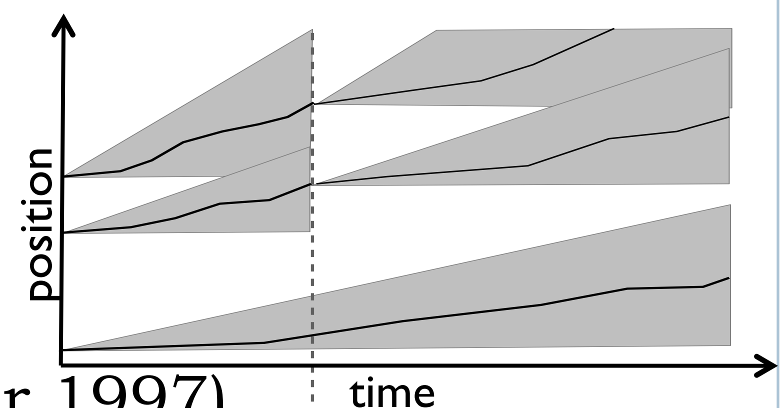
ESA '10

Kinetic Robust K-Center Problem

Existing Frameworks for Kinetic Data

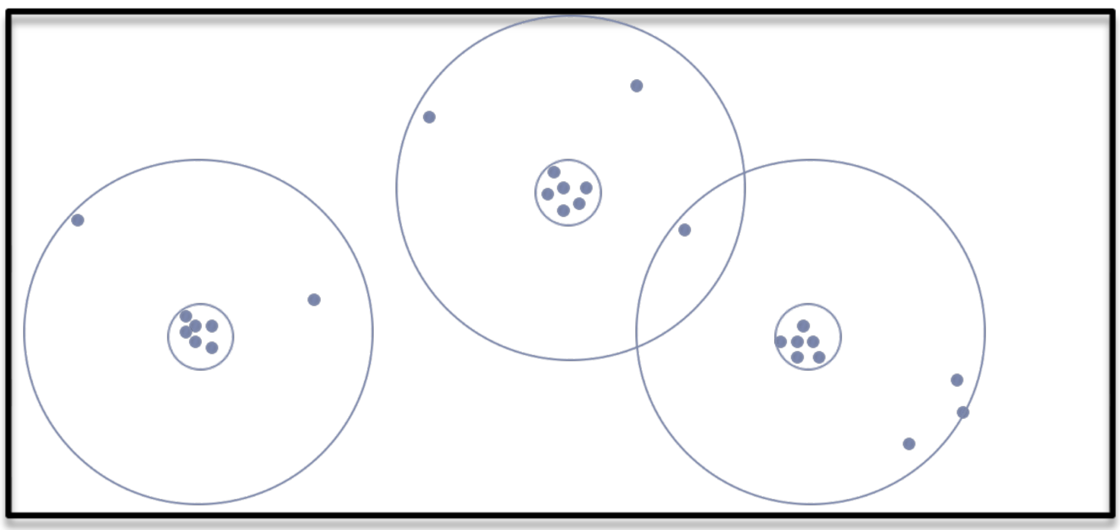
- Atallah (1983)
  - Polynomial motion of degree k
  - Motion known in advance
  - Points lie in R^d
  - Analysis in R^{d+1}
- Kahan (1991)
  - Bounds on point velocity
  - Update function provided
  - Limit queries to function
- Kinetic Data Structures (Basch, Guibas, Hershberger 1997)
  - Points have flight plans (algebraic expressions) that can change





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



Results

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

- $(3+\varepsilon)$-approximation algorithm for discrete *k*-center
  - Improves Gao *et al.* 8-approximation
  - Close to Charikar *et al.* 3-approximation
- $(4+\varepsilon)$-approximation algorithm for absolute *k*-center
  - First absolute solution
- Efficient by kinetic data structure standards

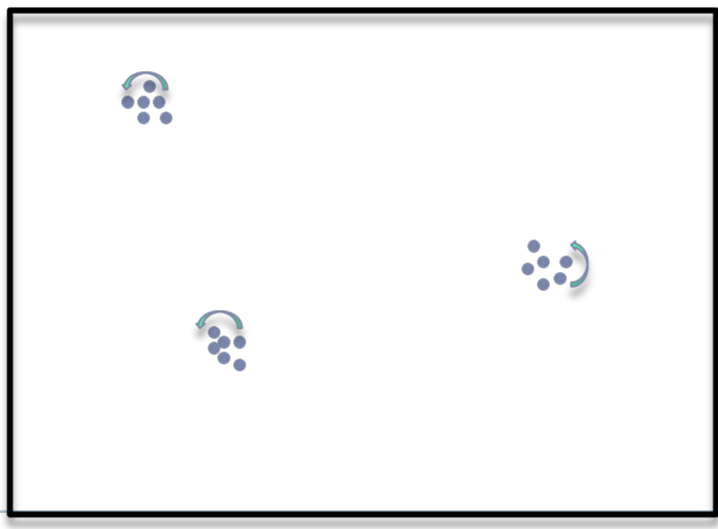
A Sensor-Based Framework for Kinetic Data Compression

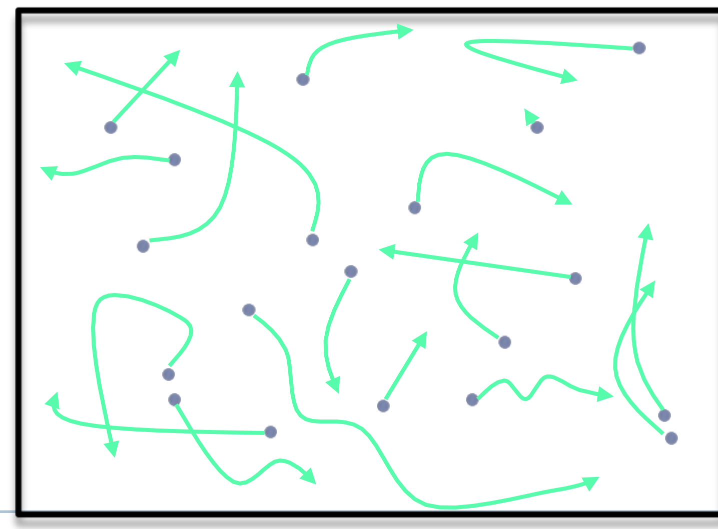
Existing Frameworks for Sensor Data

- Gandhi, Kumar, Suri (2008)
  - Sensors can count objects within their detection region
- Guitton, Skordylis, Trigoni (2007)
  - Sensors can calculate traffic flow (cars/time), occupancy (cars/area)
- Kastrinaki (2003 Survey)
  - Sensors can calculate object speed, change in angle, etc

Motivation

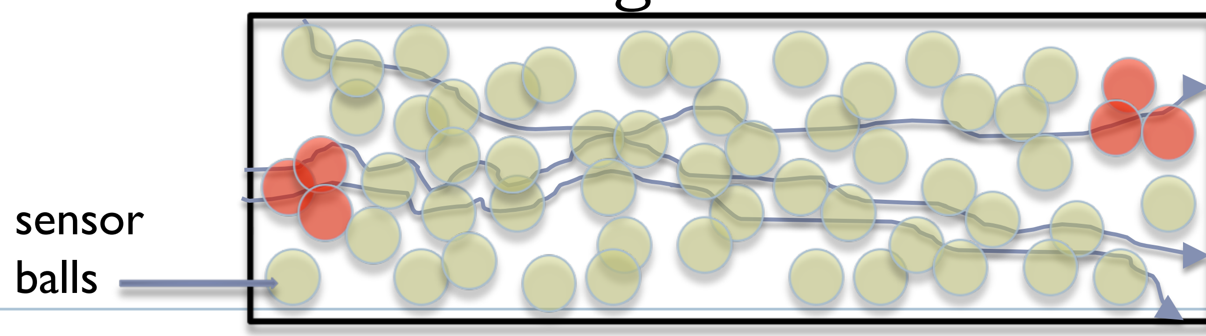
- Develop a framework for kinetic data from sensors
  - No advance object motion knowledge
  - No restrictions on object motion
  - Reasonable assumptions of what a sensor can know
  - Efficiency analysis that is <u>motion sensitive</u>





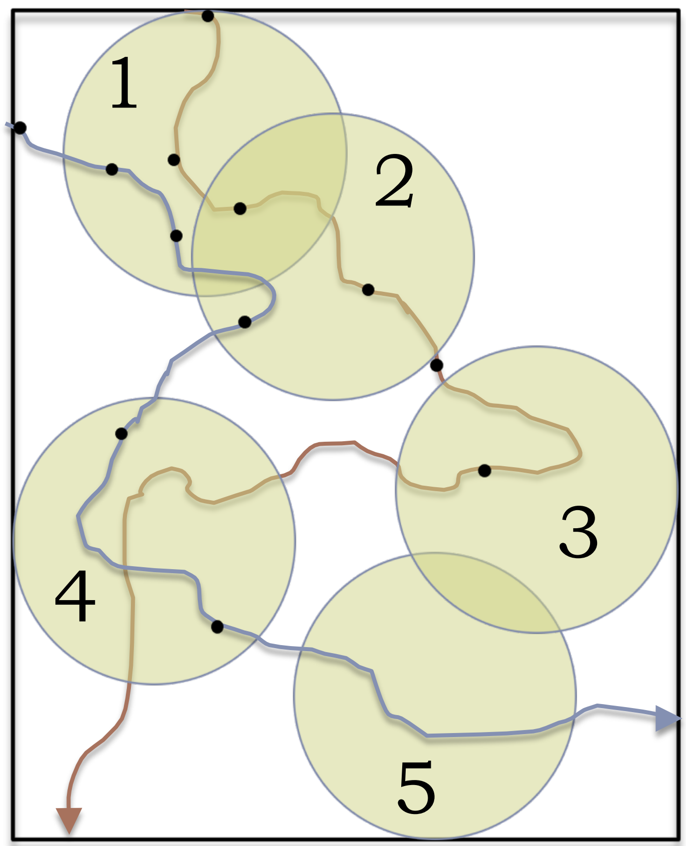
Our Framework

- 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

x_1	x_2	x_3	x_4	x_5
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

Motivation: Data Compression

- <u>Kinetic data</u>: 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

- Next: Lossless data compression
- Later: Retrieval

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:

  $$-(1 \log 1) = 0 \quad \text{vs.} \quad -(½ \log ½ + ½ \log ½) = 1$$

Entropy Generalizations

- 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

- 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

Finding a Data Compression Algorithm

Goal: smallest *lossless encoding* of sensor data

- **Optimal**: encoded (set) length = underlying (joint) entropy

- Idea 1: compress all strings separately
  - Not optimal
- Idea 2: compress all strings together
  - Window size needed for repetition too large to be practical

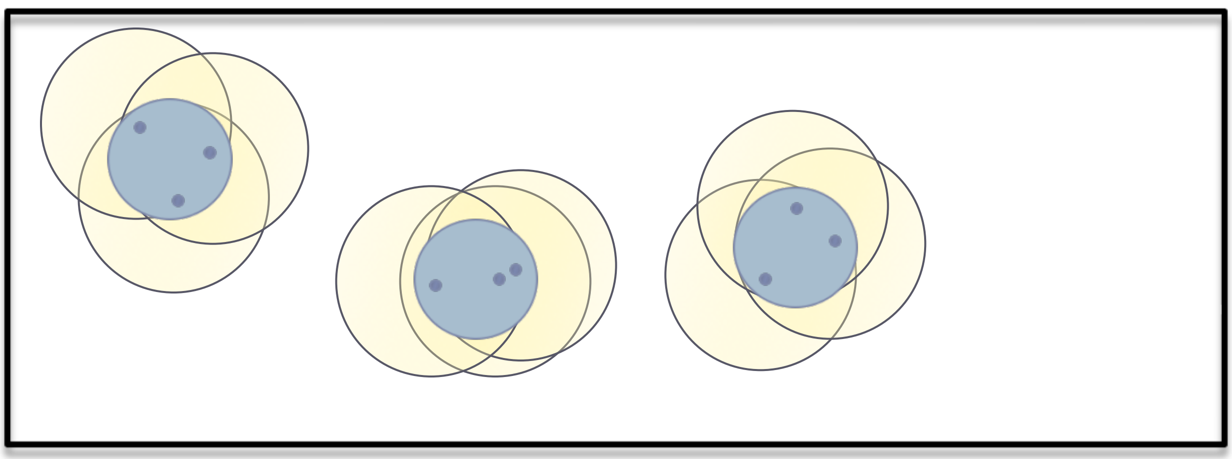
Optimal encoded sensor stream length: $H(\mathbf{X}) = H_k(\mathbf{X})$

- **Key**: compress statistically dependent
  strings together – want groups of k strings

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.

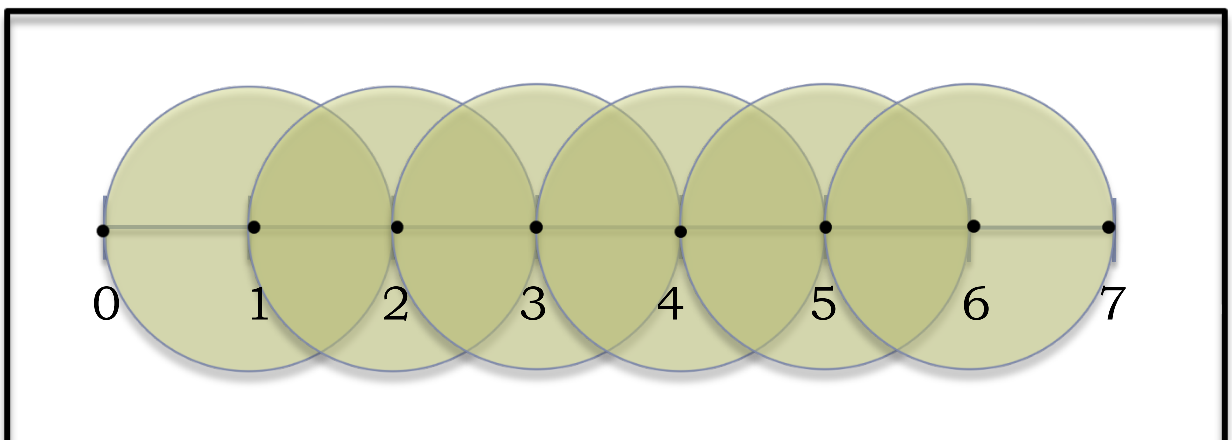
2-clusterable example



Data Compression Algorithm: Partitioning Lemma

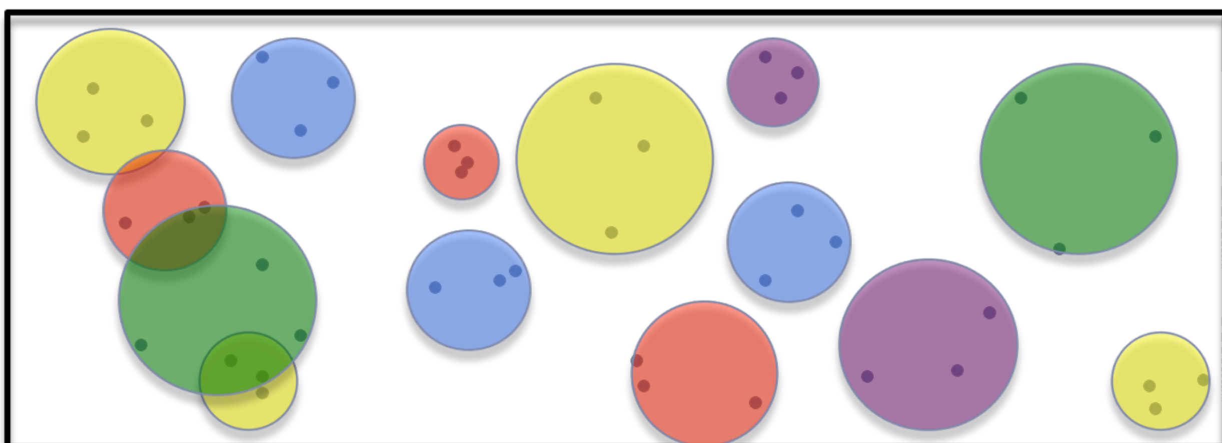
- *k*-clusterable: A point set that can be clustered into subsets of size at most $k+1$ so that if p and q are among each other's k nearest neighbors then they are in the same cluster.

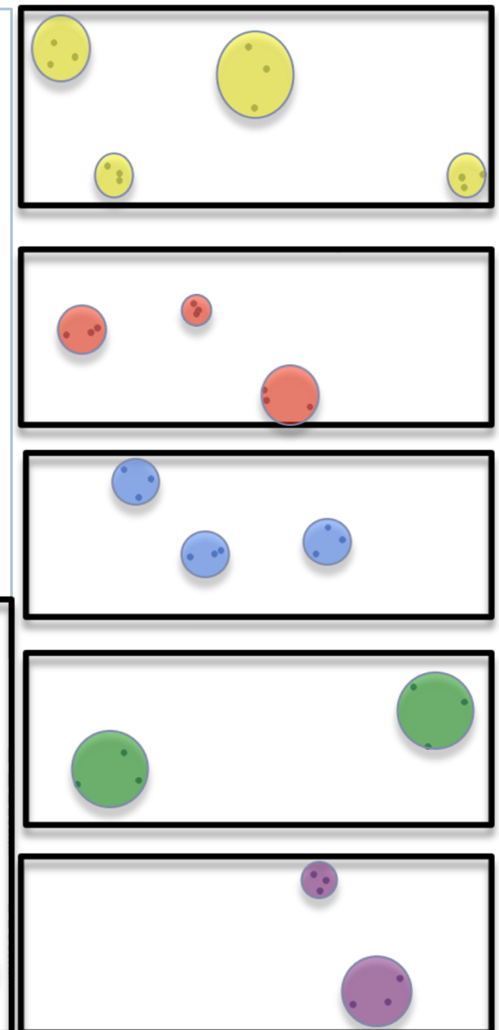
not 2-clusterable example



Data Compression Algorithm: Partitioning Lemma

- <u>Lemma</u>: 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.

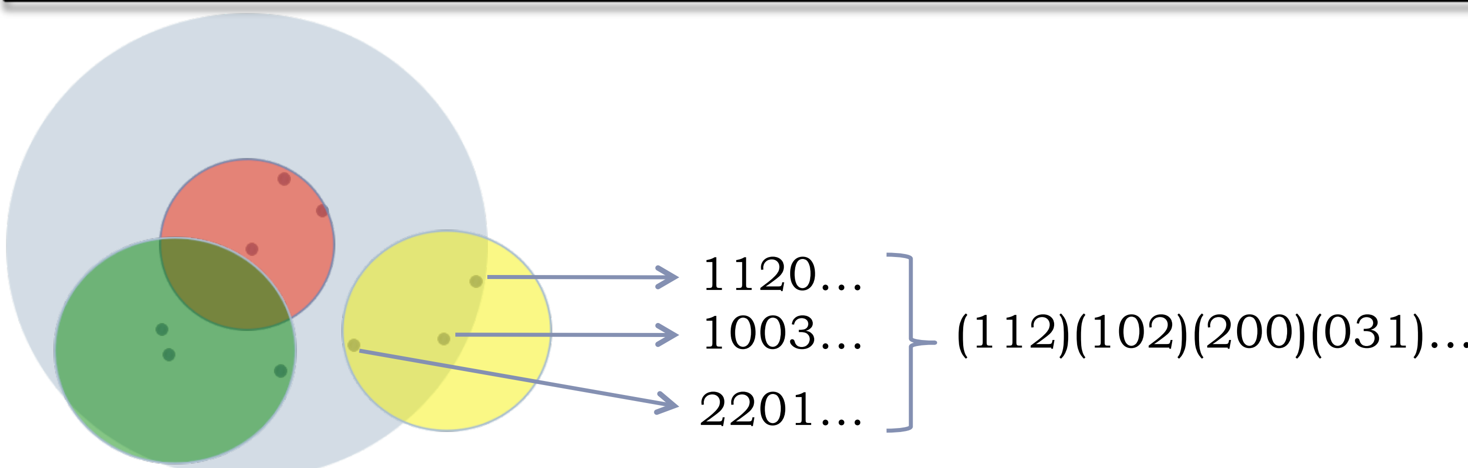




Data Compression Algorithm

- 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
    - c is O(1)

Realistic Issues in Compression of Kinetic Sensor Data

Two Main Issues and Solutions

Entropy	**Independence**
Shannon Entropy - assumes an underlying random process - bounds hold in the limit Empirical Entropy - relies on observed probabilities	Independence - strict independence ignores systemic patterns δ-Independence - allows limited underlying dependence between sensor outputs

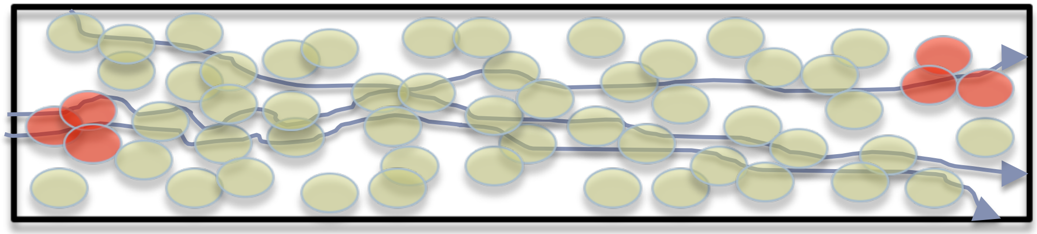
Analysis of Compression Algorithm

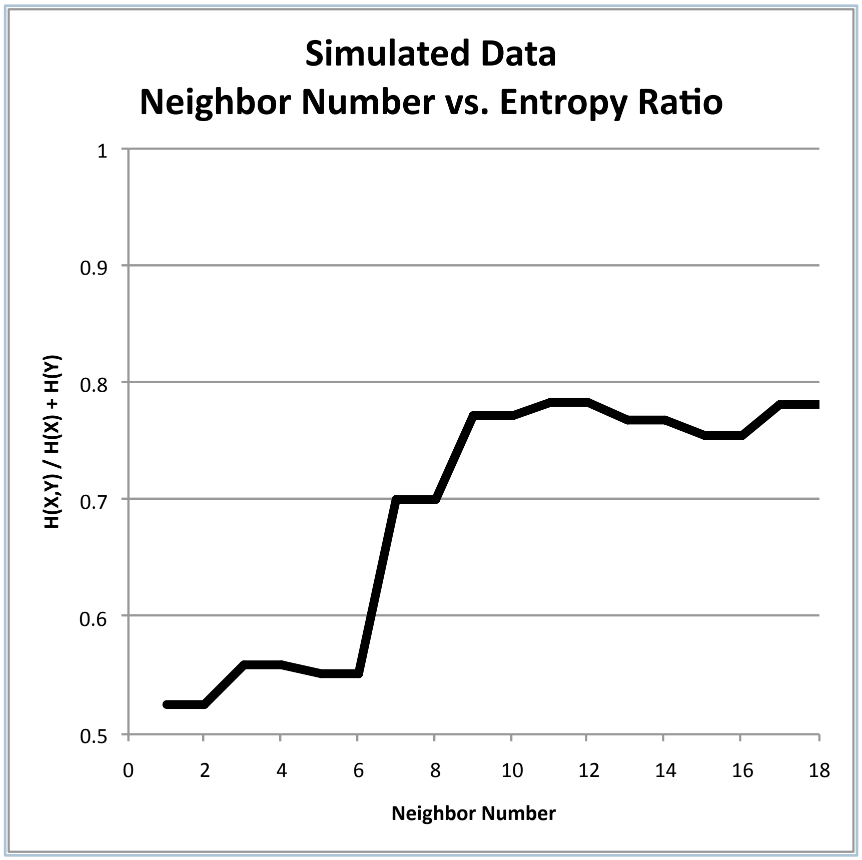
Statistical and Empirical Settings

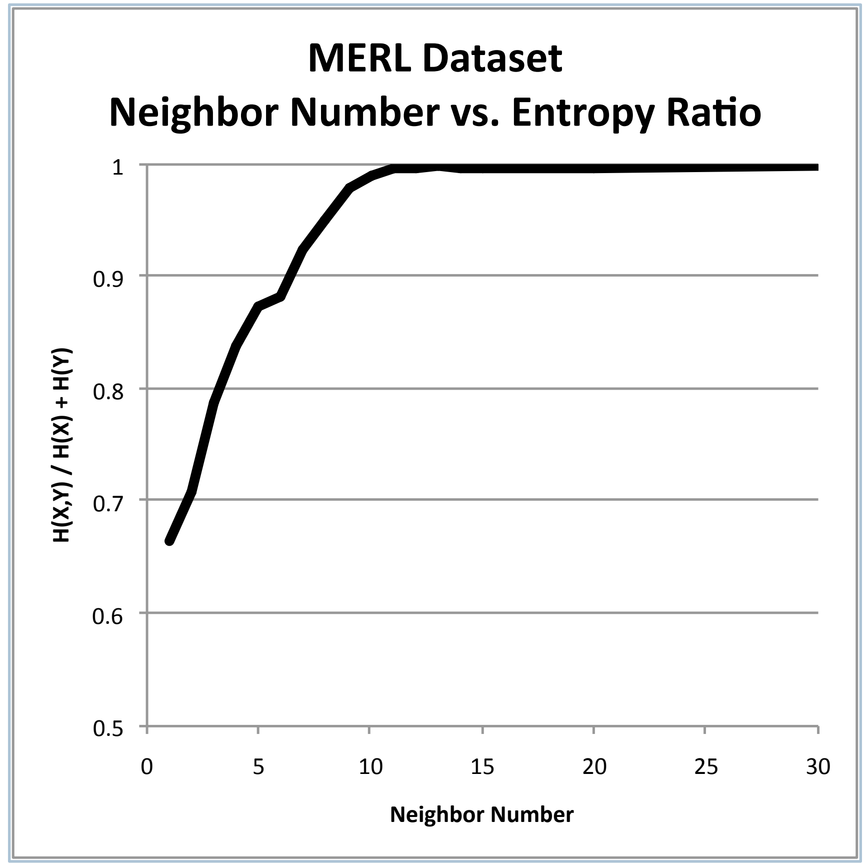
Strict Independence	δ-Independence
$O(S_{opt}(\mathbf{X}))$	$O(\max\{\delta T, S_{opt}(\mathbf{X}, \delta)\})$

- $\mathbf{X}$: set of sensor system observations
- T: length of observed time period
- δ : independence parameter
- $S_{opt}(\mathbf{X})$: optimal encoding space size for $\mathbf{X}$

Locality Experiments







Spatio-temporal Range Searching over Compressed Kinetic Sensor Data

Motivation: Data Retrieval

- 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
- Done: Lossless data compression
- Next: Retrieval without decompressing data

Range Searching: Our Problem

Compress and preprocess the data so as to perform…

- <u>Temporal range query</u>: Given a time interval, return an aggregation of the counts over that time interval.

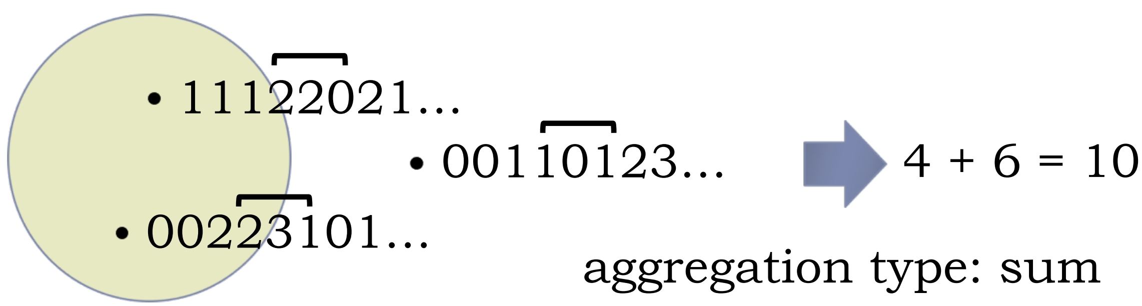
  aggregation type: sum

  t: 1 2 3 4 5 6 7 8 9 10 11

  X: 0,0,4,4,5,4,3,3,1, 1, 0

  17

- <u>Spatio-temporal range query</u>: Given a time interval and spherical spatial region, return an aggregation of the counts over that time interval and within that region.

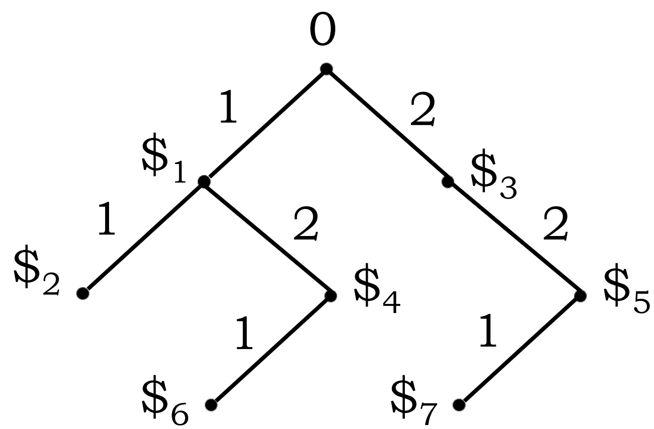


Lempel-Ziv Dictionary Compression [LZ78]

11121222121221

1 11 2 12 22 121 221

$\$_1$ $\$_2$ $\$_3$ $\$_4$ $\$_5$ $\$_6$ $\$_7$



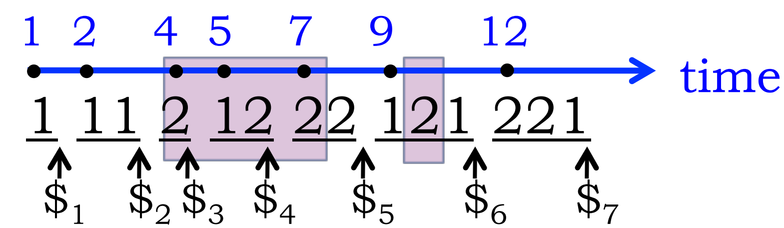
Create a trie while scanning through a string. The compressed string contains pointers to this dictionary.

(LZ78 is an optimal entropy encoding algorithm.)

Temporal Range Searching

- Create trie with accompanying pointers
- Annotate trie with aggregate values and word start times
- Given a temporal range $[t_0, t_1]$ find the anchor points $\$^0$ and $\$^1$ such that $\$^0 \leq t_0$ and $\$^1 \geq t_1$ (binary search)
- Use stored prefixes, words, and subtraction of prefixes to find aggregates

Encoded String



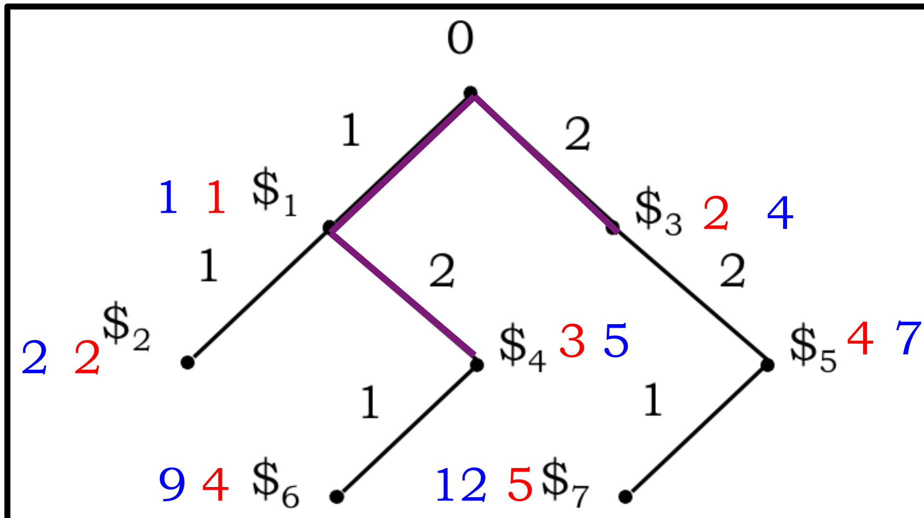
Query Examples

overlapping query: [4,7]

2 + 3 + 2 = 7

internal query: [10,10]

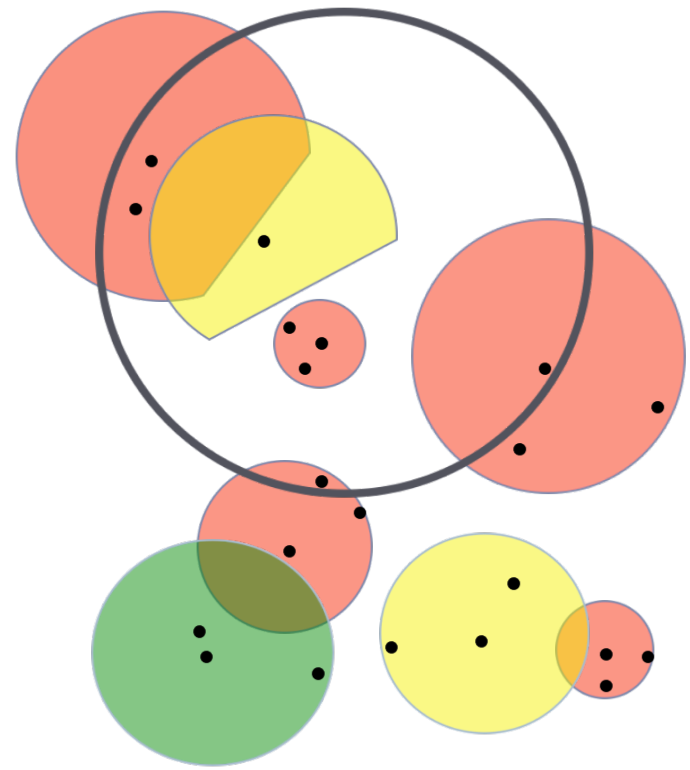
3 - 1 = 2



aggregate values word start times

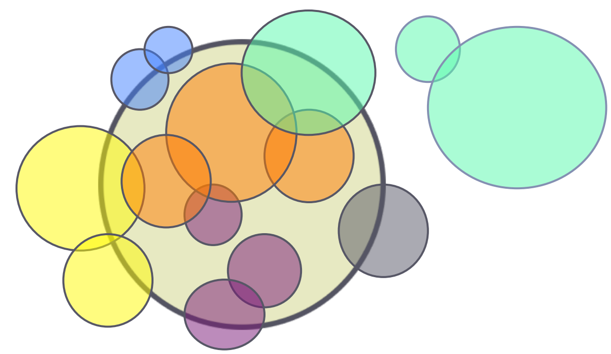
Sensor Clumps

- Recall: The sensors are partitioned, clustered, and compressed
- *Set of clumps*: A finite set of balls with a packing property limiting the number of intersections of any ball with a clump.
- Lemma: In a single partition, the nearest neighbor balls form a set of clumps that contain the sensor clusters



Range Searching Among Clumps

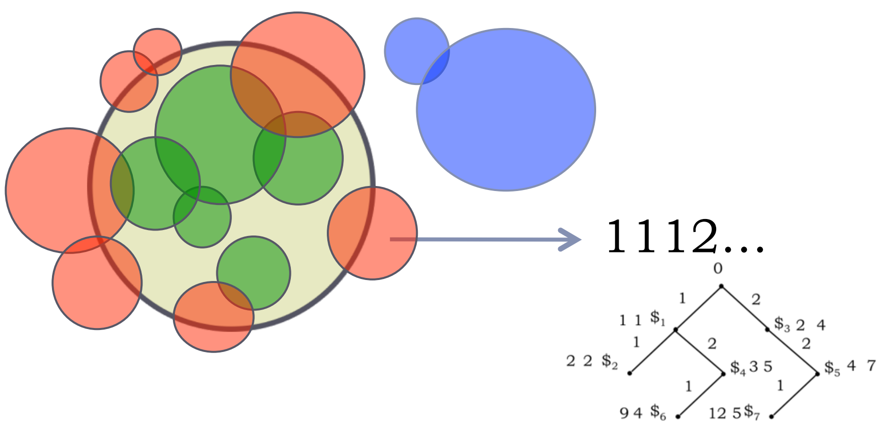
- Range Searching Among Clumps: Given any query range $\mathcal{R}$ and using a quadtree variant, we can report
  - a subset of clump subsets that form a disjoint cover of the clumps within $\mathcal{R}$
  - the subset of clumps that $\mathcal{R}$ intersects



- Lemma: A quadtree variant based data structure can answer range searching queries among clumps.

Spatio-temporal Range Searching

- <u>Main Theorem</u>: By adding an auxiliary data structure to answer temporal range queries to each node in the range searching among clumps solution we can answer spatio-temporal range queries.



- One range searching among clumps structure for each partition
- One temporal range structure for each clump and each internal quadtree node
- Get temporal sums for each clump and overlapped sensor
- Sum over all partitions

Results

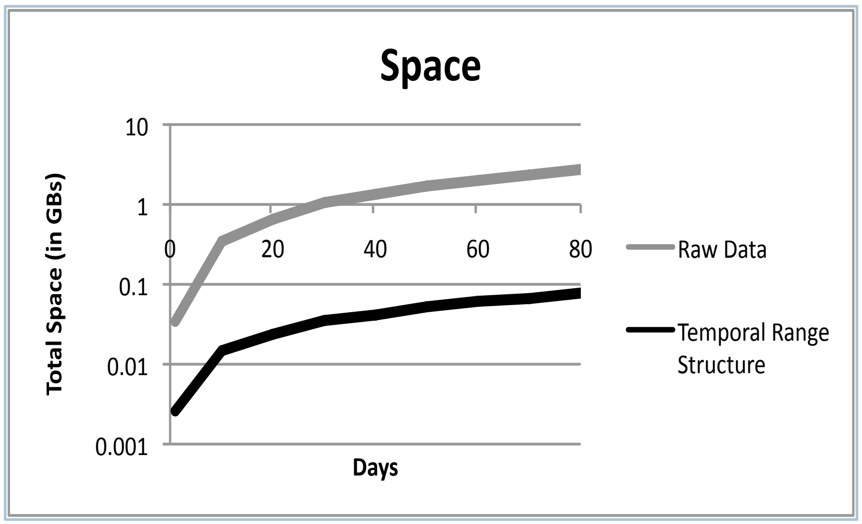
Bounds for Range Searching

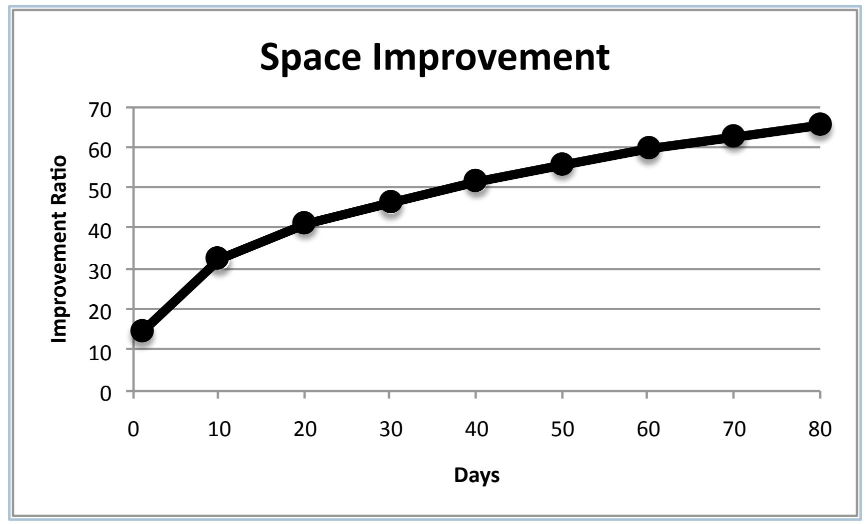
	Temporal	Spatio-temporal
Preprocessing time	$O(\mathrm{Enc}(X))$	$O(\mathrm{Enc}(\mathbf{X}))$
Query time	$O(\log T)$	$O(((1/\varepsilon^{d-1}) + \log S) \log T)$
Space	$O(\mathrm{Enc}(X))$	$O(\mathrm{Enc}(\mathbf{X}) \log S)$

- X: The set of sensor system observations
- Enc(X): The encoded size (in bits) of the compressed data
- T: The total time over which data was collected
- S: The total number of sensors
- d: The dimension of the sensor space
- ε: An error parameter (for approximate range searching)

First range searching bounds <u>over compressed data</u>

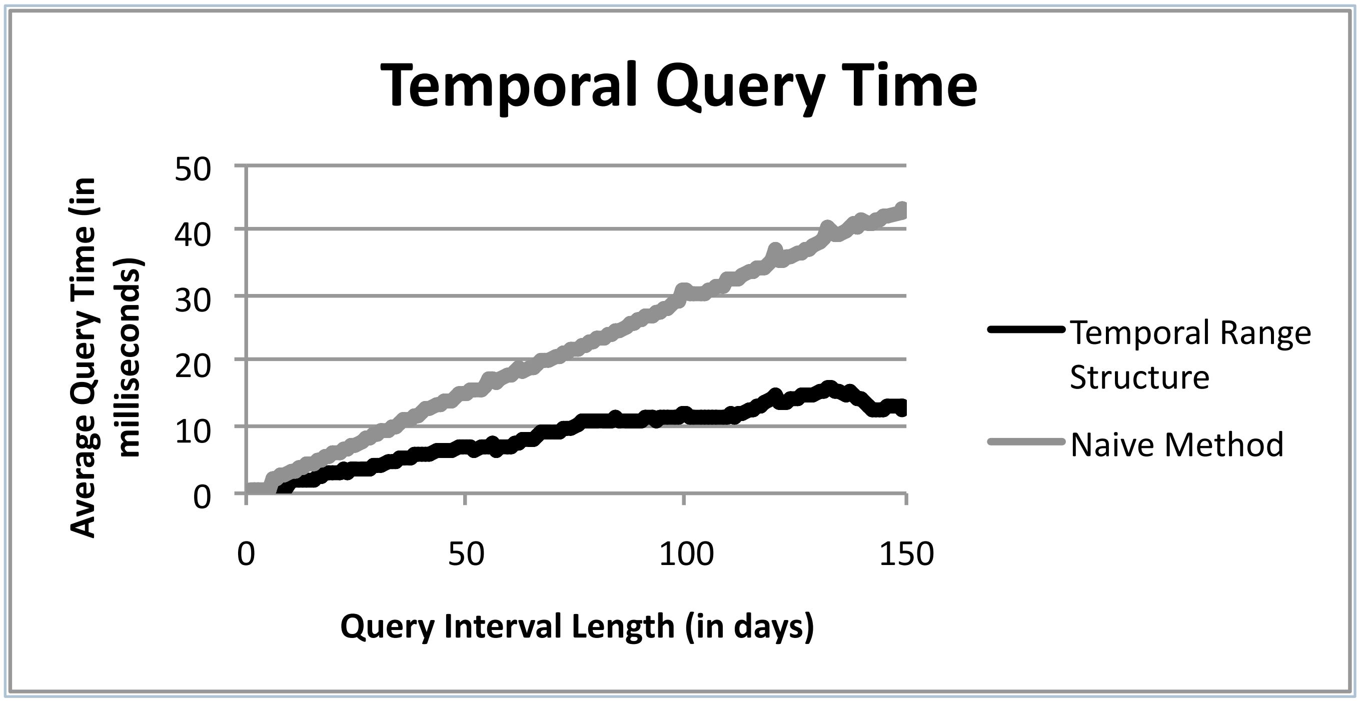
Experimental Results: Space





C. R. Wren, Y. A. Ivanov, D. Leigh, and J. Westbues.
The MERL motion detector dataset: 2007 workshop on massive datasets.
Technical Report TR 2007-069,
Mitsubishi Electronic Research Laboratories, Cambridge, MA, USA, August 2007.

Experimental Results: Time



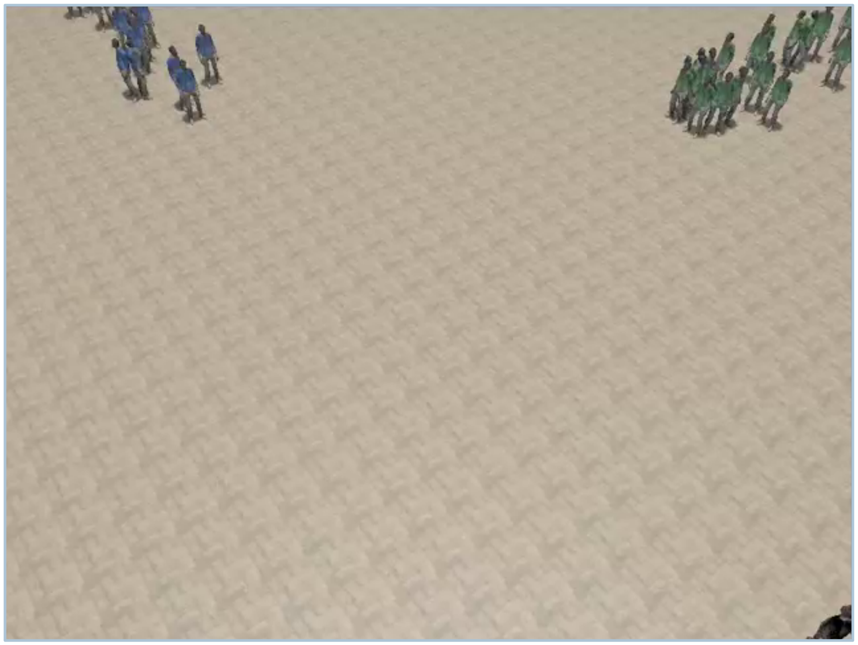
C. R. Wren, Y. A. Ivanov, D. Leigh, and J. Westbues.
The MERL motion detector dataset: 2007 workshop on massive datasets.
Technical Report TR 2007-069,
Mitsubishi Electronic Research Laboratories, Cambridge, MA, USA, August 2007.

Conclusions and Open Problems

Results

- Robust clustering within the KDS model
- Framework for kinetic sensor data
  - No assumptions about object motion or advance knowledge
- Lossless compression algorithm that takes space O(optimal)
- Realistic analysis to consider empirical entropy and a limited notion of independence
- Spatio-temporal range searching over compressed kinetic sensor data
- Experimental analyses of locality, space, and query time

Spatio-temporal k-Center Problem



simulation by the UNC collision avoidance team

- $\mathbf{X} = \{X_1, \ldots, X_S\}$
- $X_i = X_{i1},\ldots,X_{ij}, \ldots , X_{iT}$
- Assign counts to k clusters $C_{ij1}, \ldots, C_{ijk}$ such that for all sensors and times i,j
  - $\Sigma_l \, C_{ijl} = X_{ij}$
- Minimize the maximum $H_\tau(\mathbf{X})$ over all $C_l = \{C_{ijl}\}_j$

Future Work: Understanding Motion

Practical	Theoretical
Relies on reasonable assumptions about motion, data, observations, etc. Reasonable to code and develop algorithms	Algorithm analyses that are motion-sensitive Allowance for both exact theoretical descriptions of motion and observations

Thank you!

Questions?