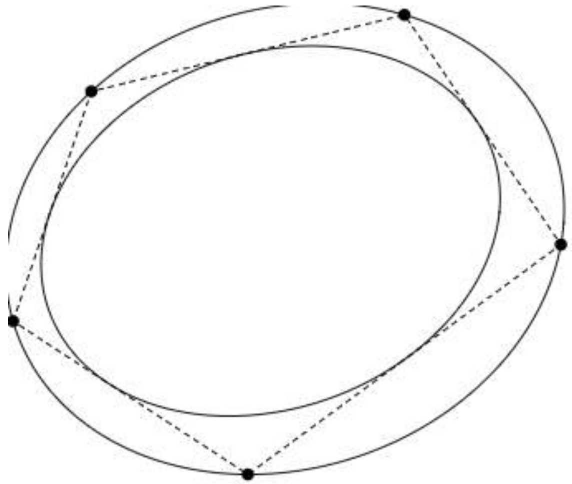
Computational Geometry for Robotics & Big Data



Dan Feldman
Murad Tukan

Robotics & Big Data Lab

אוניברסיטת חיפה
University of Haifa

אוניברסיטת חיפה
University of Haifa

Videos from

The Robotics and Big Data Lab

in the University of Haifa

Big Data

- Volume: huge amount n of data points
- Variety: high dimensional d space
- Velocity: data arrive in real-time

Need to support:

- Streaming (one pass, logarithmic memory)
- Distributed data (on cloud)
- Simple computations (embarrassingly parallel)
- No assumption on order of points

Big Data Computation model

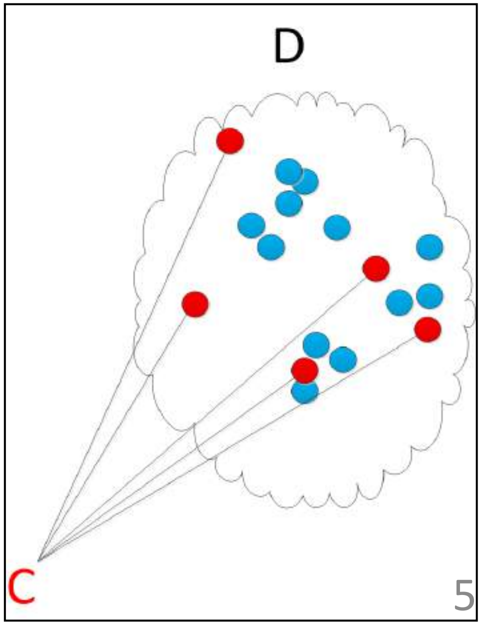
- = Streaming + Parallel computation
- Input: infinite stream of vectors
- $n=$ vectors seen so far
- $\sim\log n$ memory
- *M* processors
- *~log (n)/M* insertion time per point

(Embarrassingly parallel)

Challenge:
Find RIGHT data from Big Data

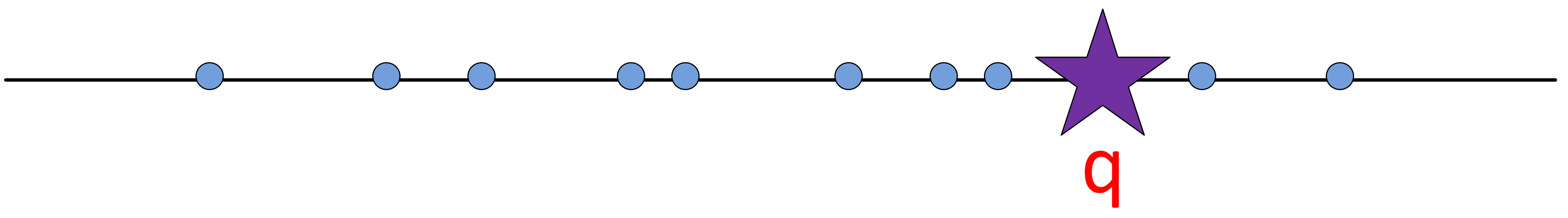
Given data *D* and Algorithm *A* with *A(D)* intractable, can we efficiently reduce *D* to *C* so that *A*(C) fast and *A*(C)~*A*(D)?

Provable guarantees on approximation with respect to the size of C

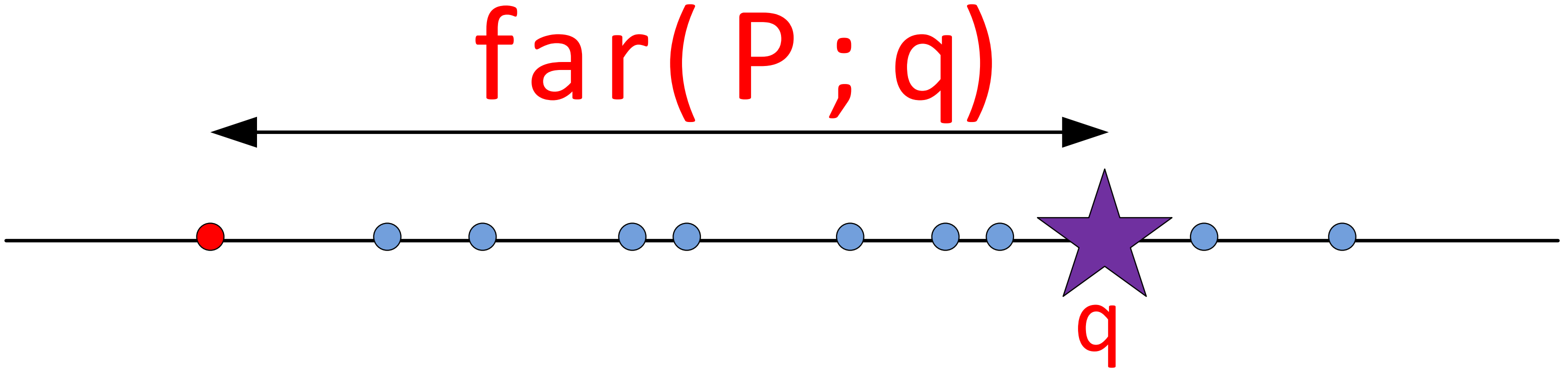


Coreset for Enclosing Balls $P \subseteq \mathbb{R}$

Coreset for Enclosing Balls $P \subseteq \mathbb{R}$



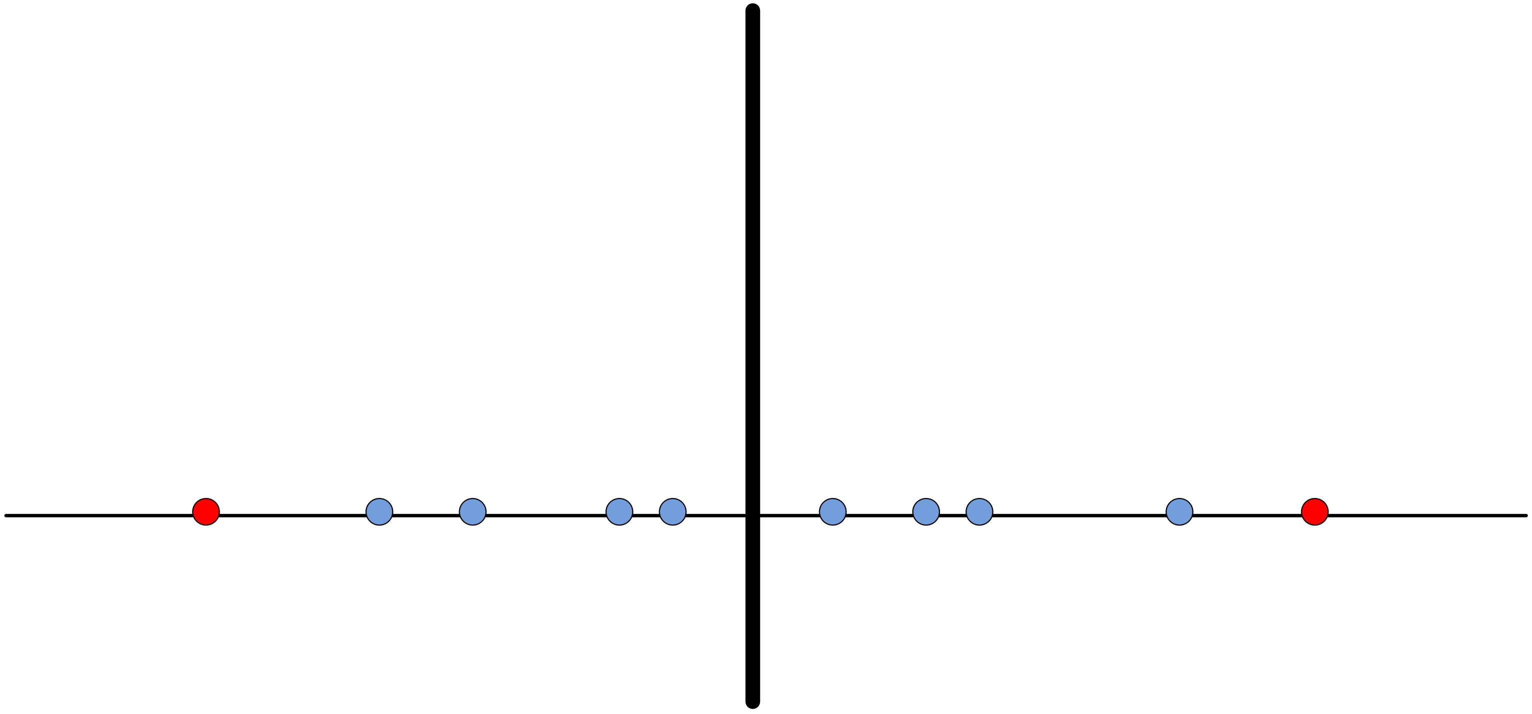
Coreset for Enclosing Balls $P \subseteq \mathbb{R}$



Coreset for Enclosing Balls $P \subseteq \mathbb{R}$

The fathest point from every query q 2 R

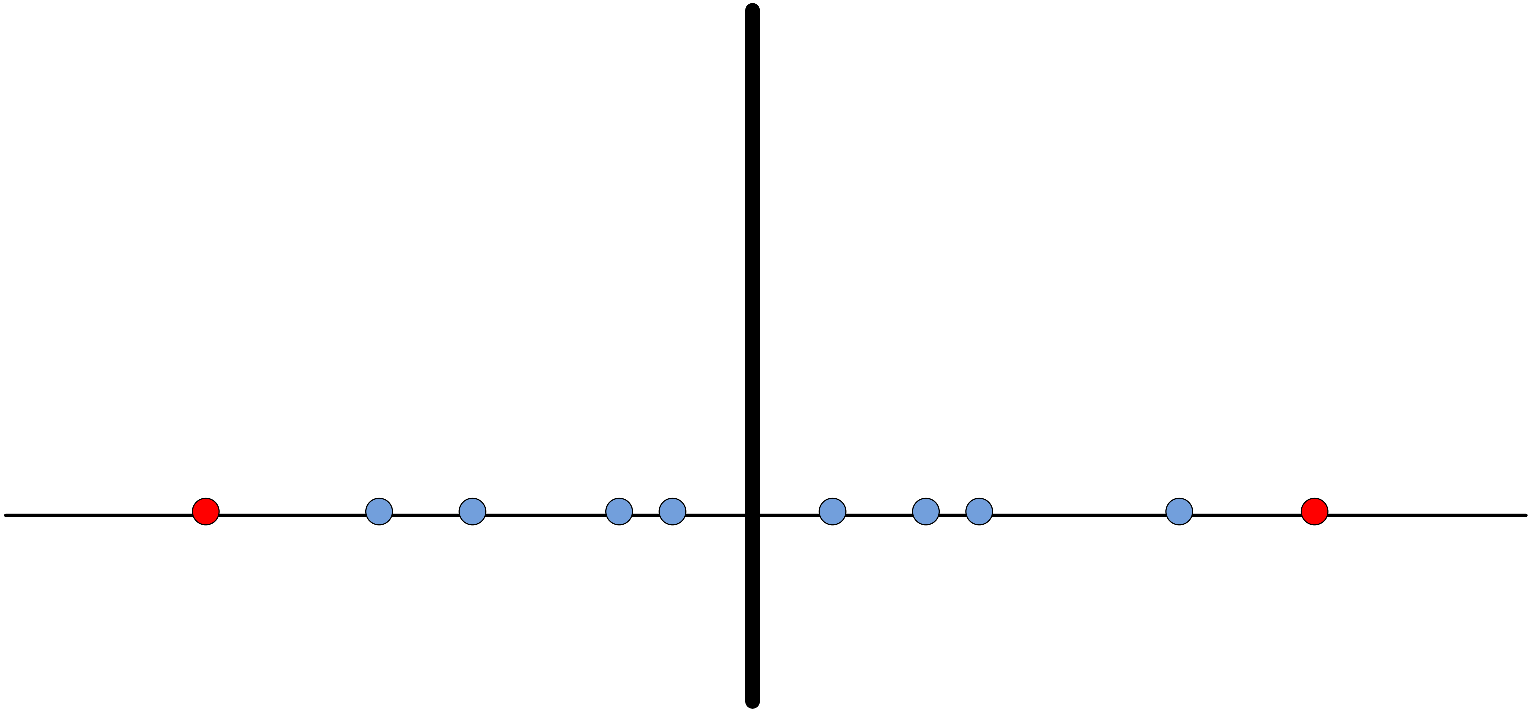
is a red point



Coreset for Enclosing Balls $P \subseteq \mathbb{R}$

The fathest point from every query $q \in R^d$

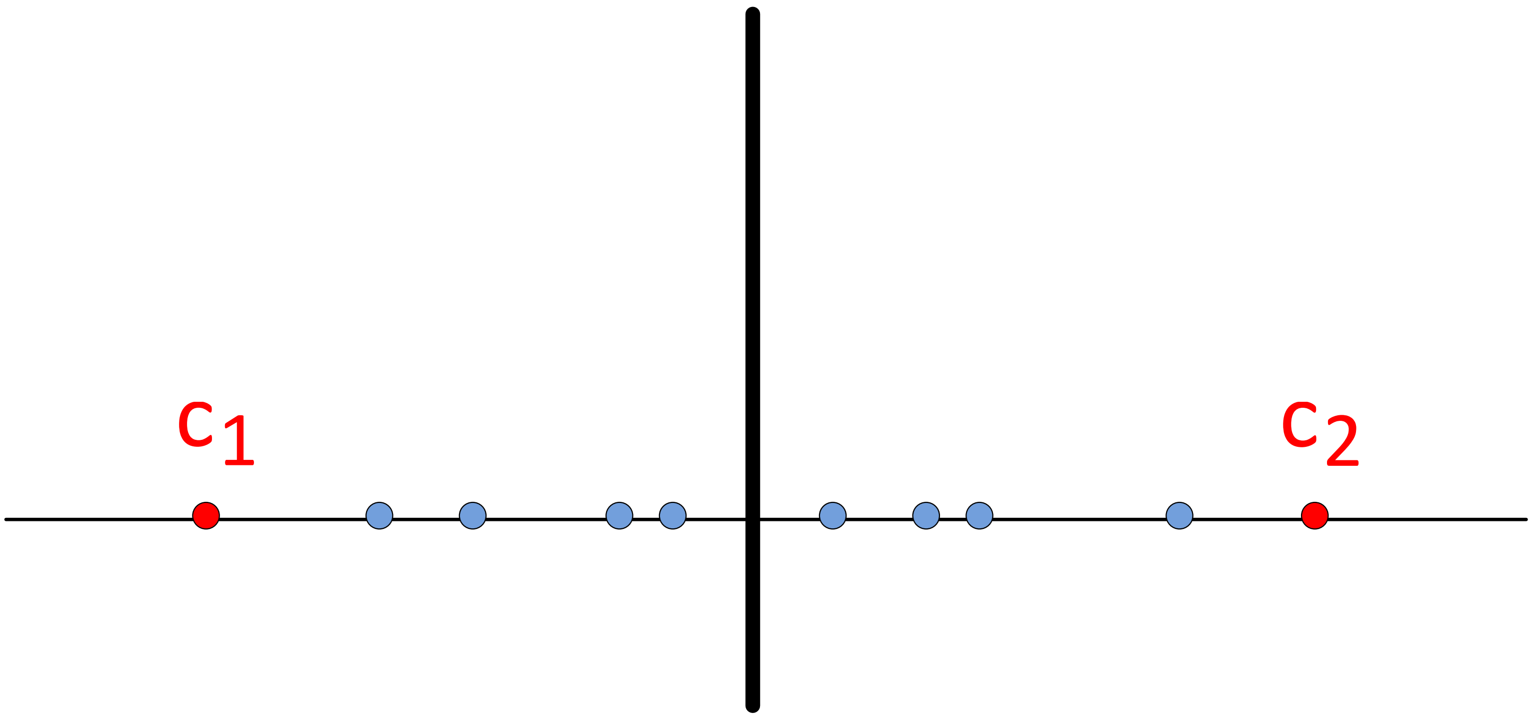
is a red point



Coreset for Enclosing Balls $P \subseteq \mathbb{R}$

The fathest point from every query q 2 R^d

is a red point



C := f c_1; c_2 g is a coreset for P

Simplest coreset definition

Let

- P be a set, called *point set*
- X be a set, called *query set*
- $\mathrm{cost}(P,x)$: maps every query $x \in X$ into a non-negative number

For a given $\epsilon > 0$, the set $C \subseteq P$ is a *core-set* if for every $x \in X$ we have

$\mathrm{cost}(P,x) \sim cost(C,x)$

up to $(1 \pm \epsilon)$ approximation factor

Coreset Techniques

Graph Theory
Sparsifiers
Batson, Speilman, Srivastava, ...

Computational Geometry
Coresets
Har-Peled, Agarwal, Sohler, Chen

Matrix Approximation
Volume Sampling
Clarkson, Mahoney, Drineas ...

Combinatorial Geometry
ε-nets, ε-approximations
Haussler, Welzl, Alon, Matousek, Sharir,...

Statistics
Importance Sampling
Srinivasan, Ripley, , ...

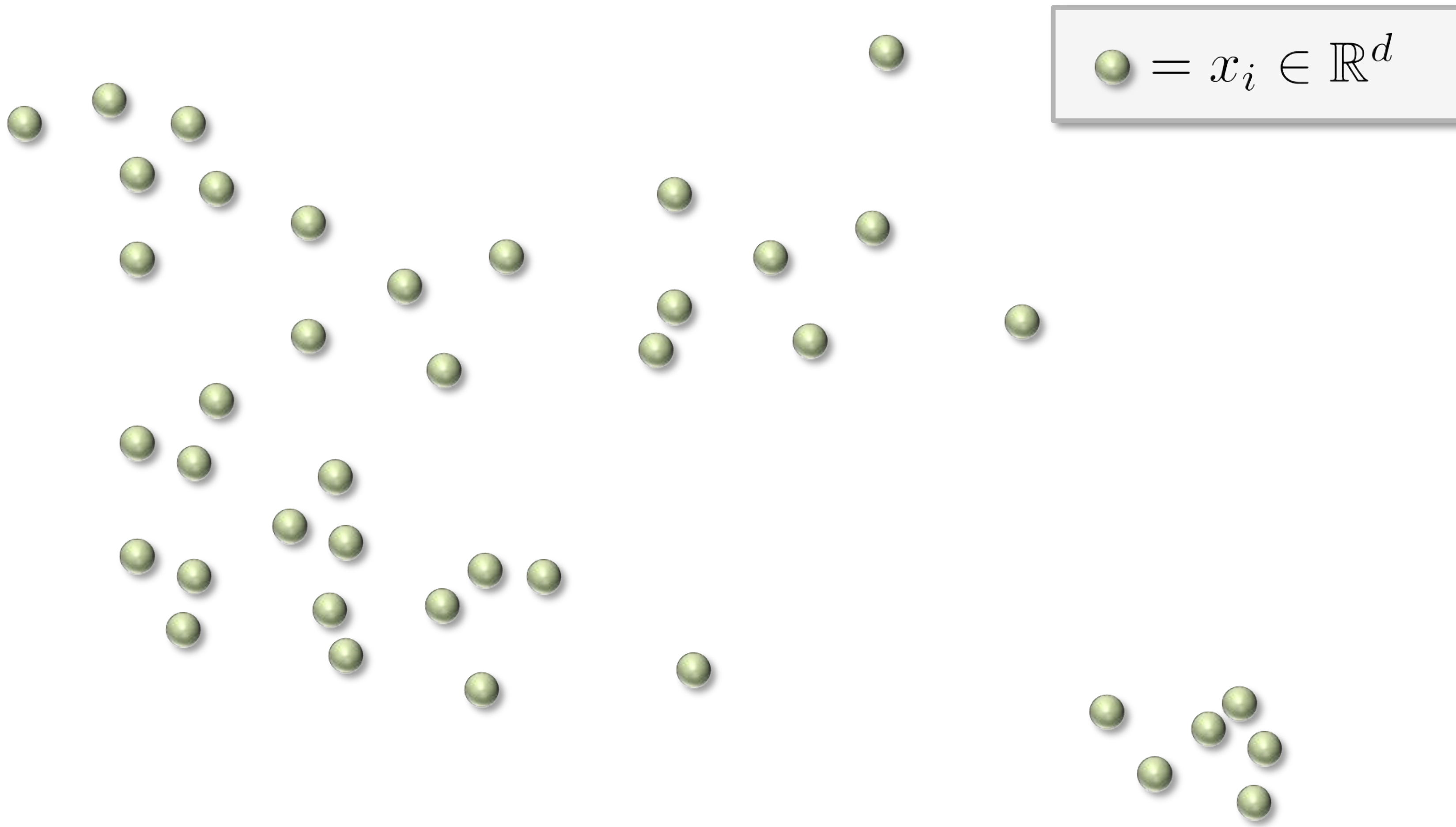
PAC-Learning
ε-sample
Vapnik, Chervonenkis, Valiant, ...

Compressed Sensing
Sketches
Property Testing

Example Coresets

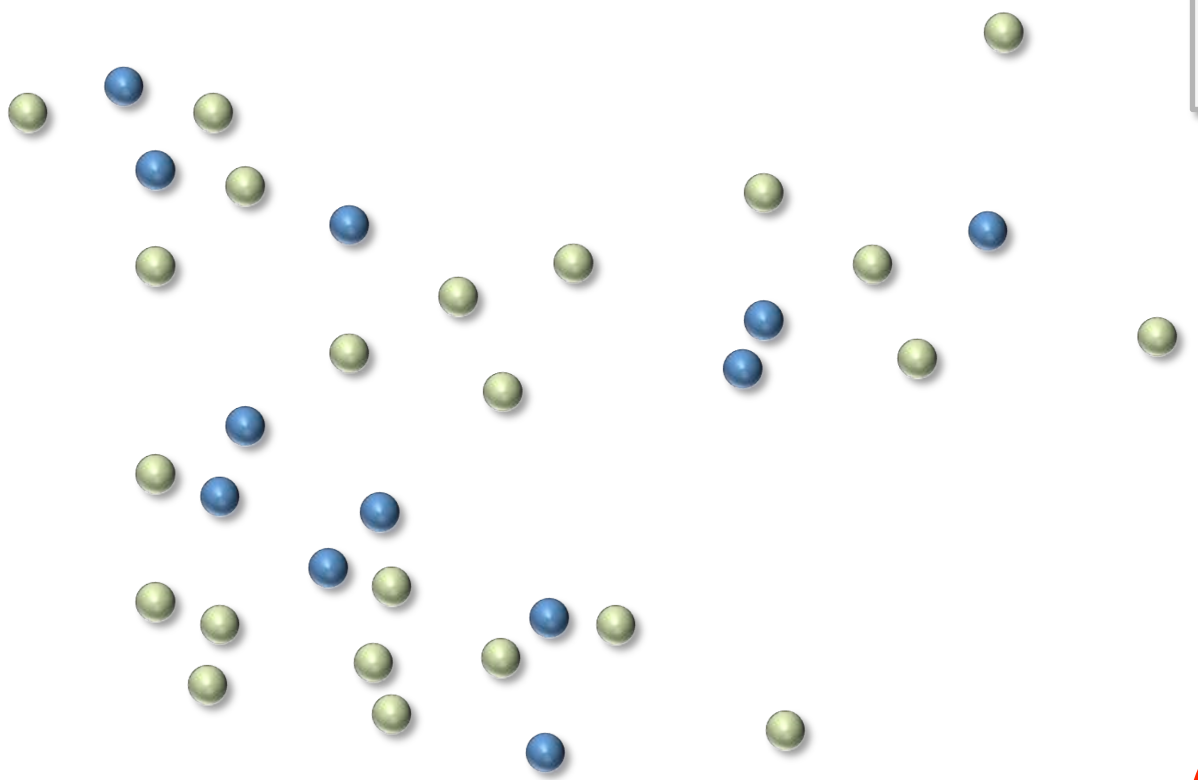
- Graph Summarization [F, Sedat, Rus, ICML’17]
- Mixture of Gaussians [F, Krause, etc JMLR’17]
- LSA/PCA/SVD [F, Rus, and Volkob, NIPS’16]
- k-Means [F, Barger, SDM’16]
- Non-Negative Matrix Factorization [F, Tassa, KDD15]
- Robots Localization [F, Cindy, Rus, ICRA’15]
- Robots Coverage [F, Gil, Rus, ICRA’13]
- Segmentation [F, Rosman, Rus, Volkob, NIPS’14]
- ....
- k-Line Means [F, Fiat, Sharir, FOCS’06]

Naïve Uniform Sampling

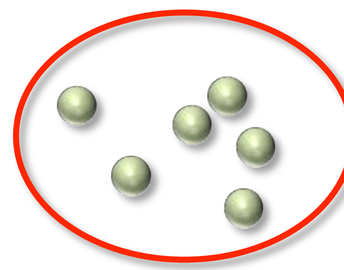


Naïve Uniform Sampling

● $= x_i \in \mathbb{R}^d$



Small cluster
is missed



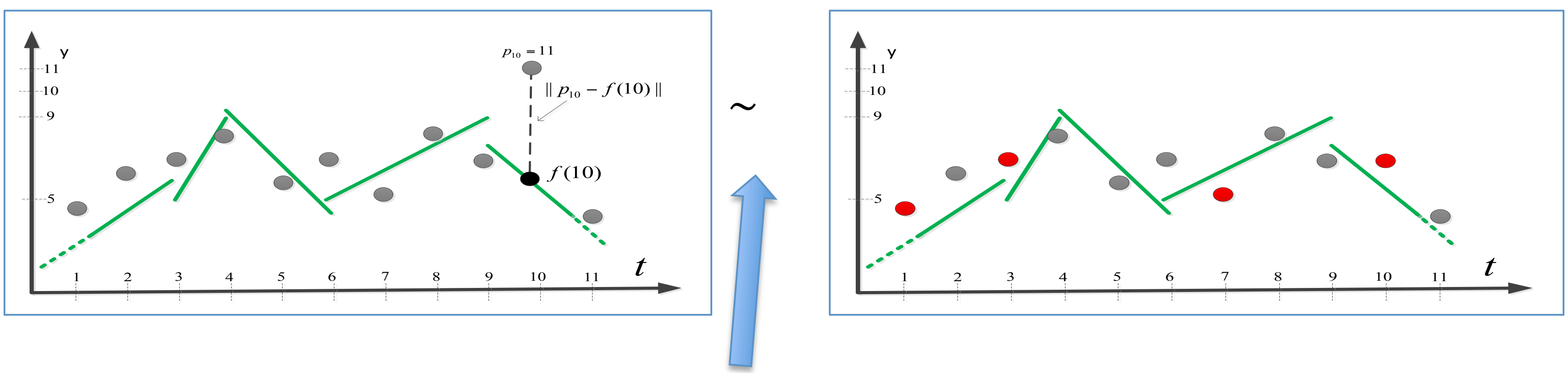
Sample a set *U* of *m* points uniformly

← High variance

From Big Data to Small Data

Suppose that we can compute such a corset C of size $1/\epsilon$ for every set P of n points

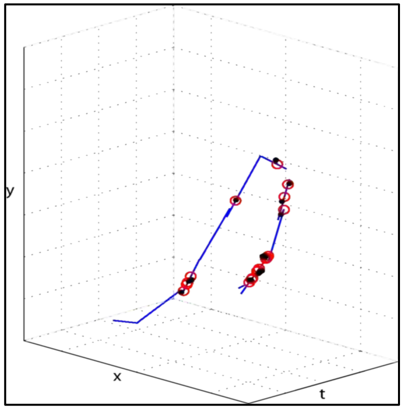
- in time n^3,
- off-line, non-parallel, non-streaming algorithm



$(1 \pm \epsilon)$

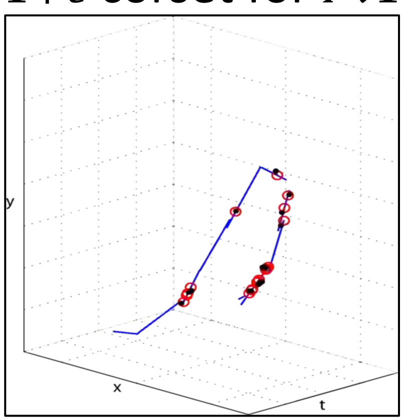
Read the first $2/\epsilon$ streaming points and reduce them into $1/\epsilon$ weighted points in time $(2/\epsilon)^5$

$1+\epsilon$ corset for P_1

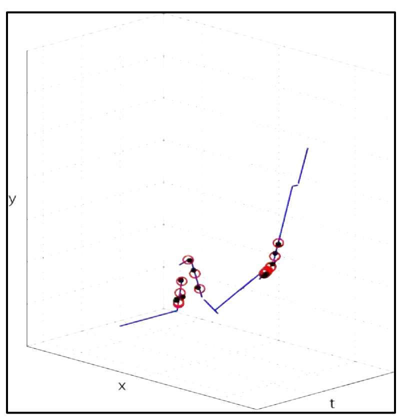


Read the next $2/\epsilon$ streaming point and reduce them into $1/\epsilon$ weighted points in time $(2/\epsilon)^5$

$1+\epsilon$ corset for P_1

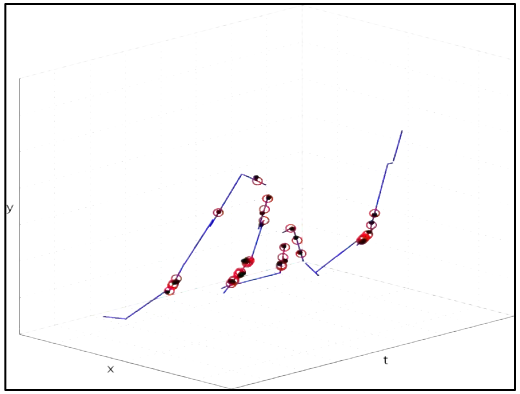


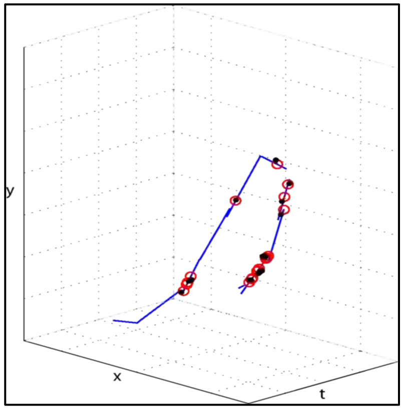
$1+\epsilon$ corset for P_2

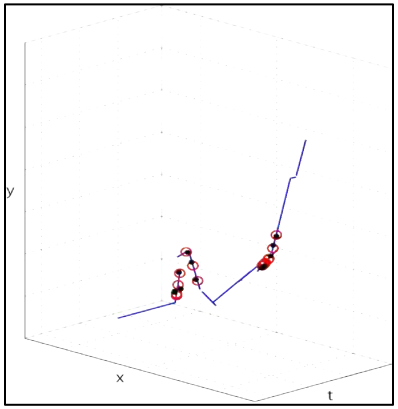


Merge the pair of ϵ-coresets into an ϵ-corset of $2/\epsilon$ weighted points

$1+\epsilon$-corset for $P\downarrow 1 \cup P\downarrow 2$

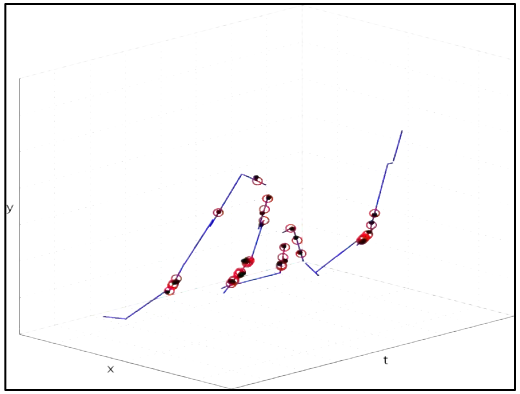


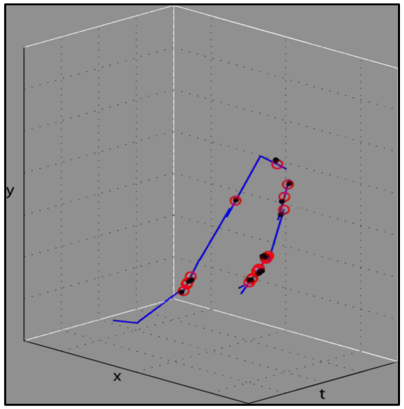


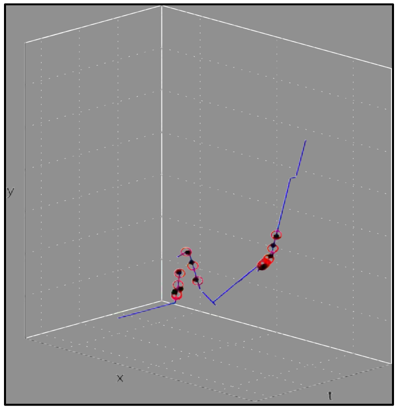


Delete the pair of original coresets from memory

$1+\epsilon$-corset for $P{\downarrow}1 \cup P{\downarrow}2$



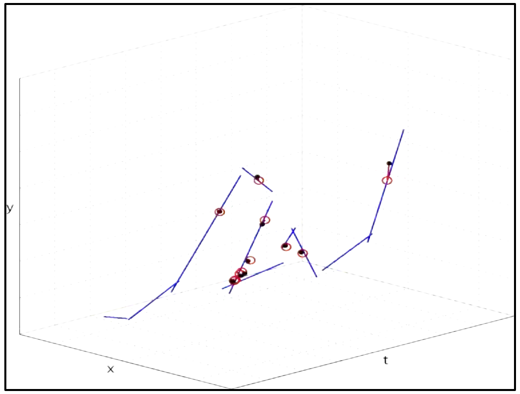


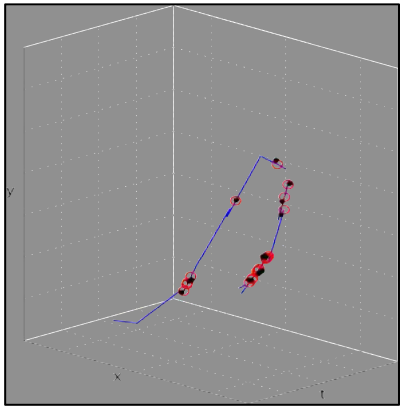


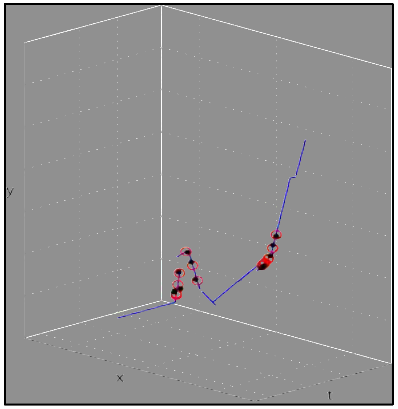
Reduce the $2/\epsilon$ weighted points into $1/\epsilon$ weighted points by constructing their coreset

$1+\epsilon$-corset for

$1+\epsilon$-corset for $P\downarrow 1 \cup P\downarrow 2$



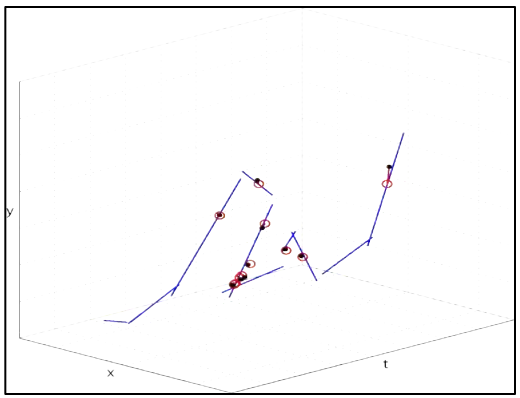




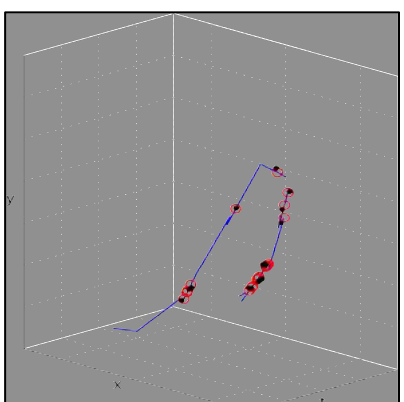
Reduce the $2/\epsilon$ weighted points into $1/\epsilon$ weighted points by constructing their coreset

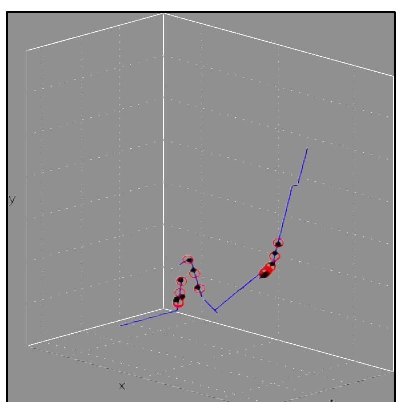
$1+\epsilon$-corset for

$1+\epsilon$-corset for $P_1 \cup P_2$

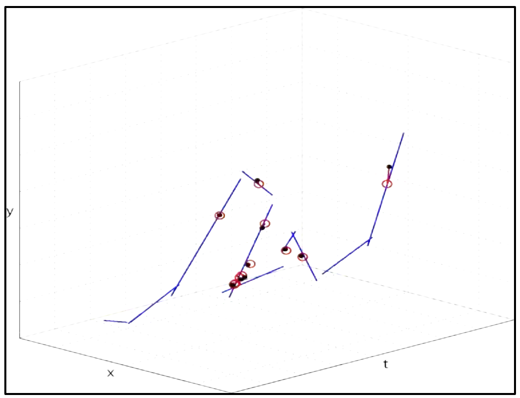


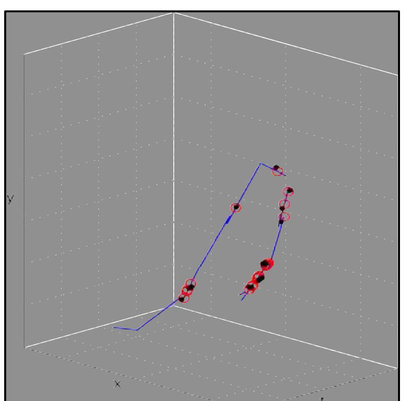
$=(1+\epsilon)^2$ -corset for $P_1 \cup P_2$

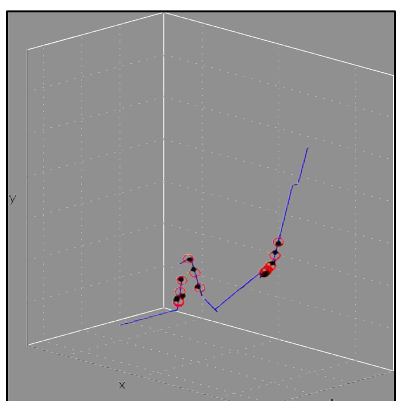




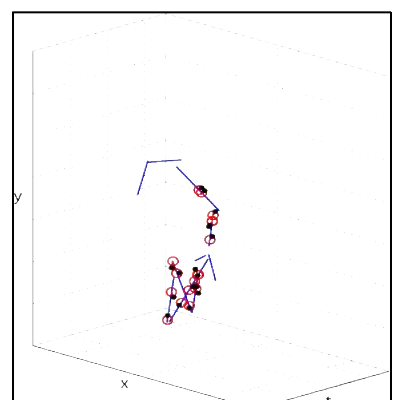
$(1+\epsilon)^{2}$-corset for $P_{1} \cup P_{2}$



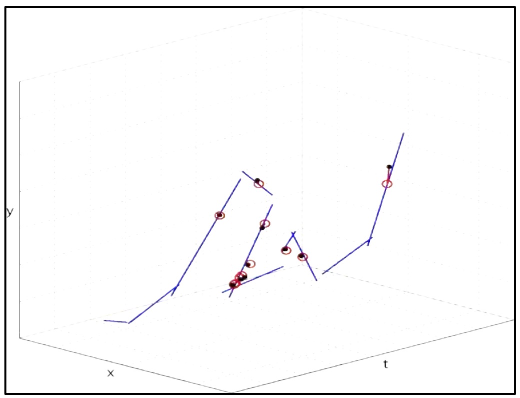


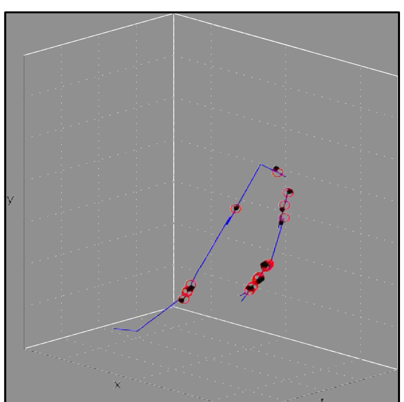


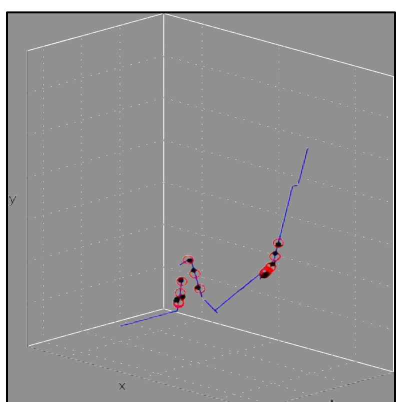
$(1+\epsilon)$-corset for P_{3}



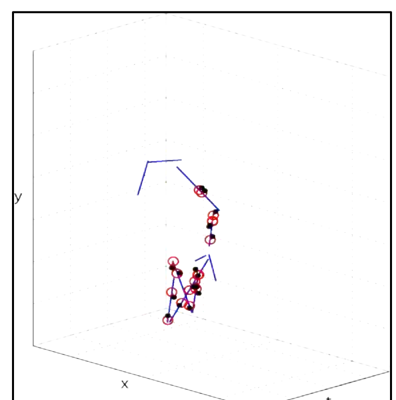
$(1+\epsilon)^2$-corset for $P_1 \cup P_2$



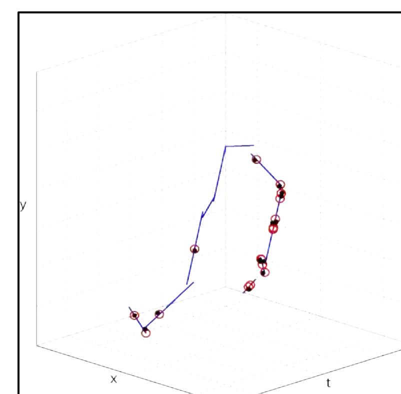




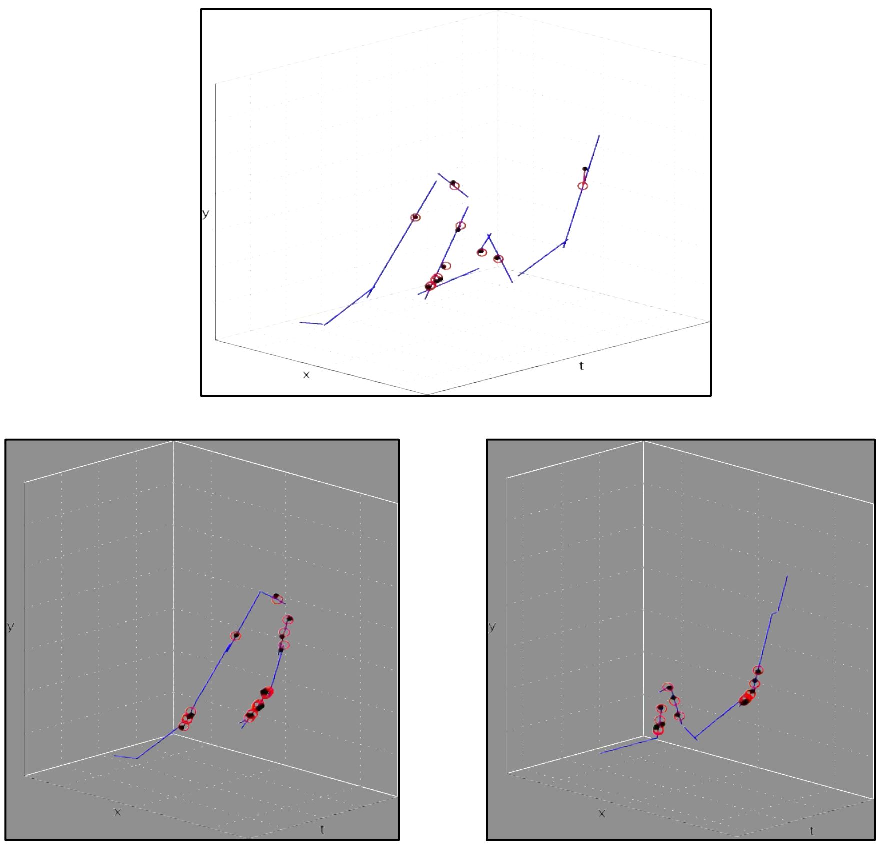
$(1+\epsilon)$-corset for P_3



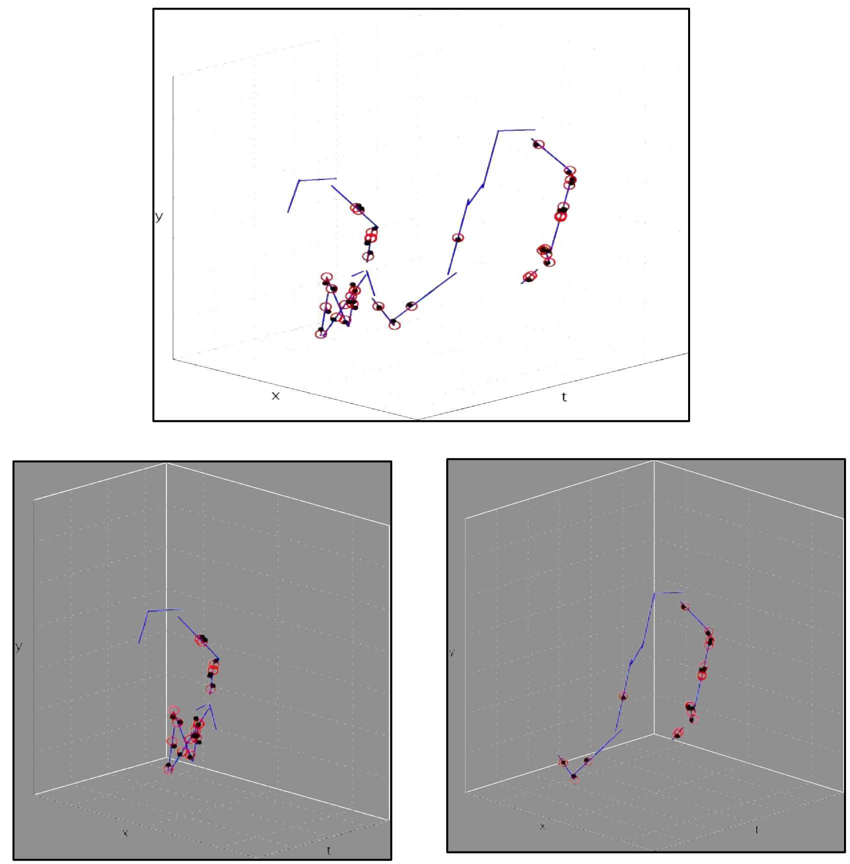
$(1+\epsilon)$-corset for P_4



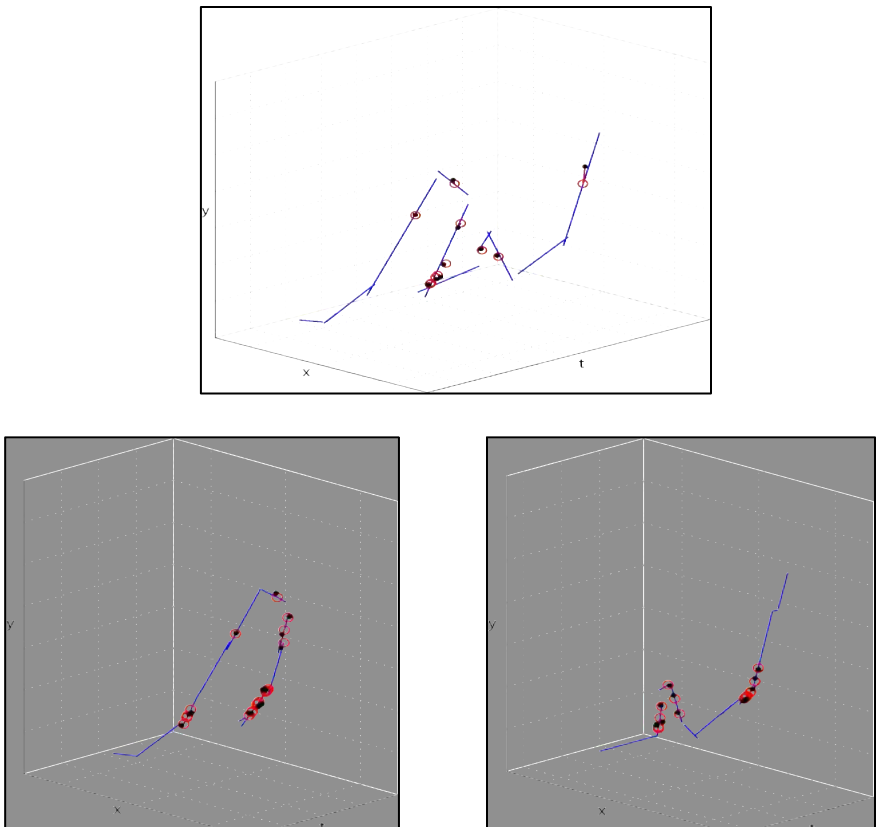
$(1+\epsilon)^2$ -corset for $P_1 \cup P_2$



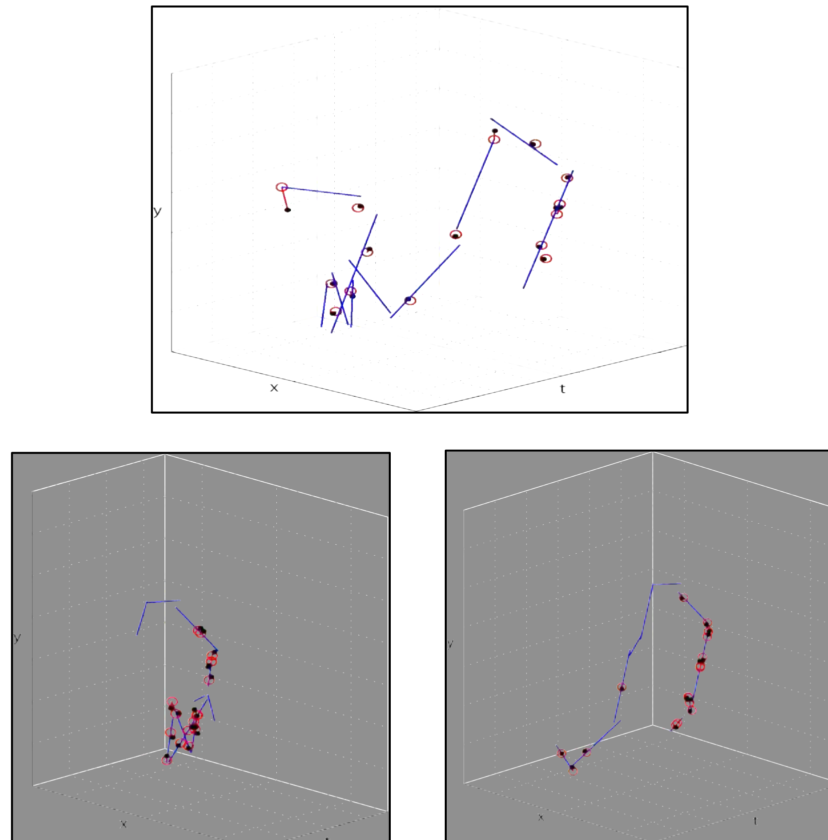
$(1+\epsilon)$-corset for $P_3 \cup P_4$

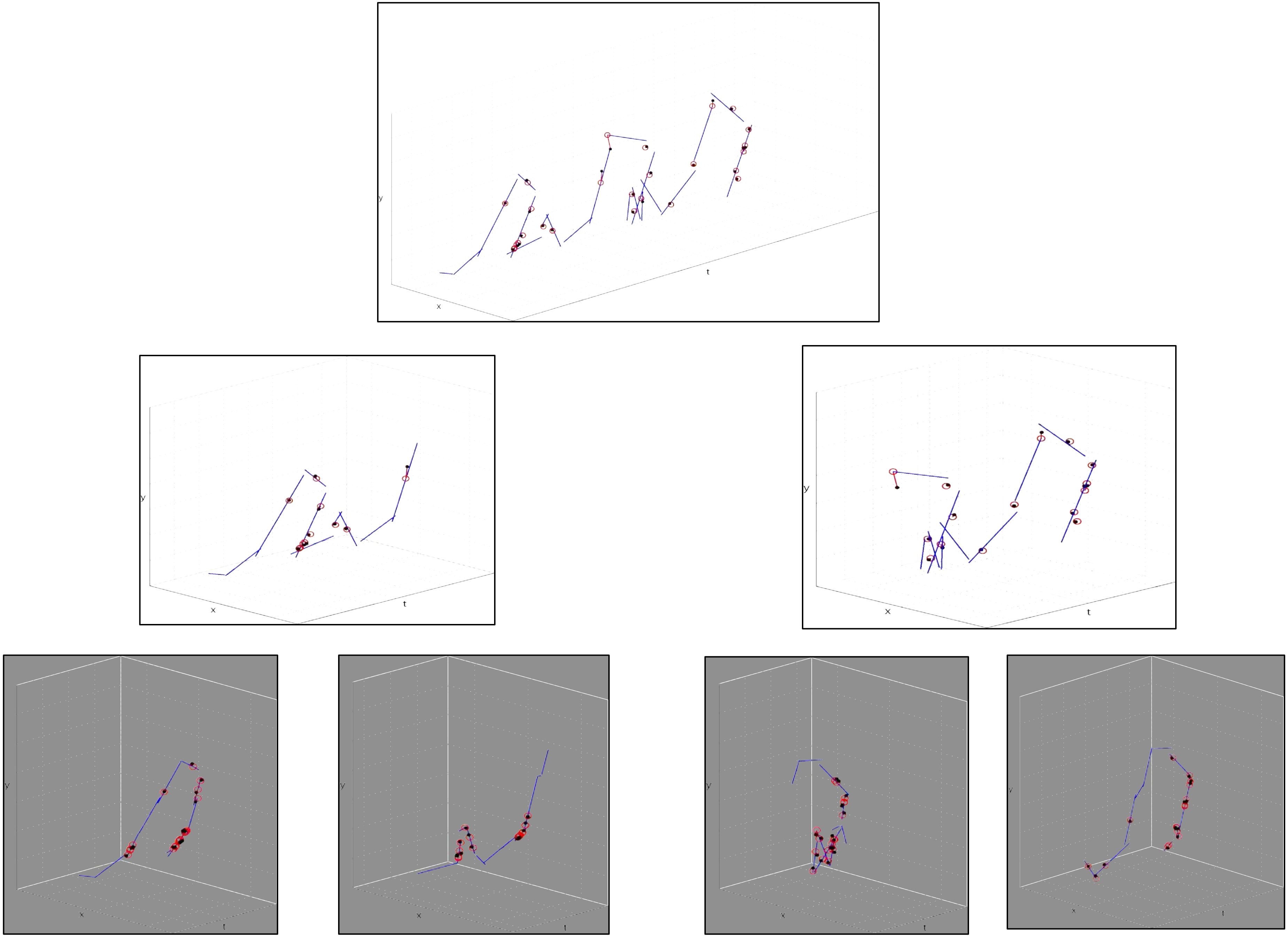


$(1+\epsilon)^2$-corset for $P_1 \cup P_2$



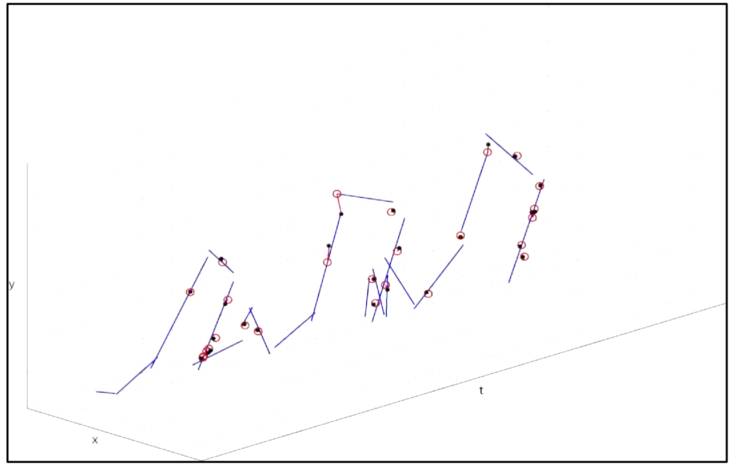
$(1+\epsilon)^2$-corset for $P_3 \cup P_4$

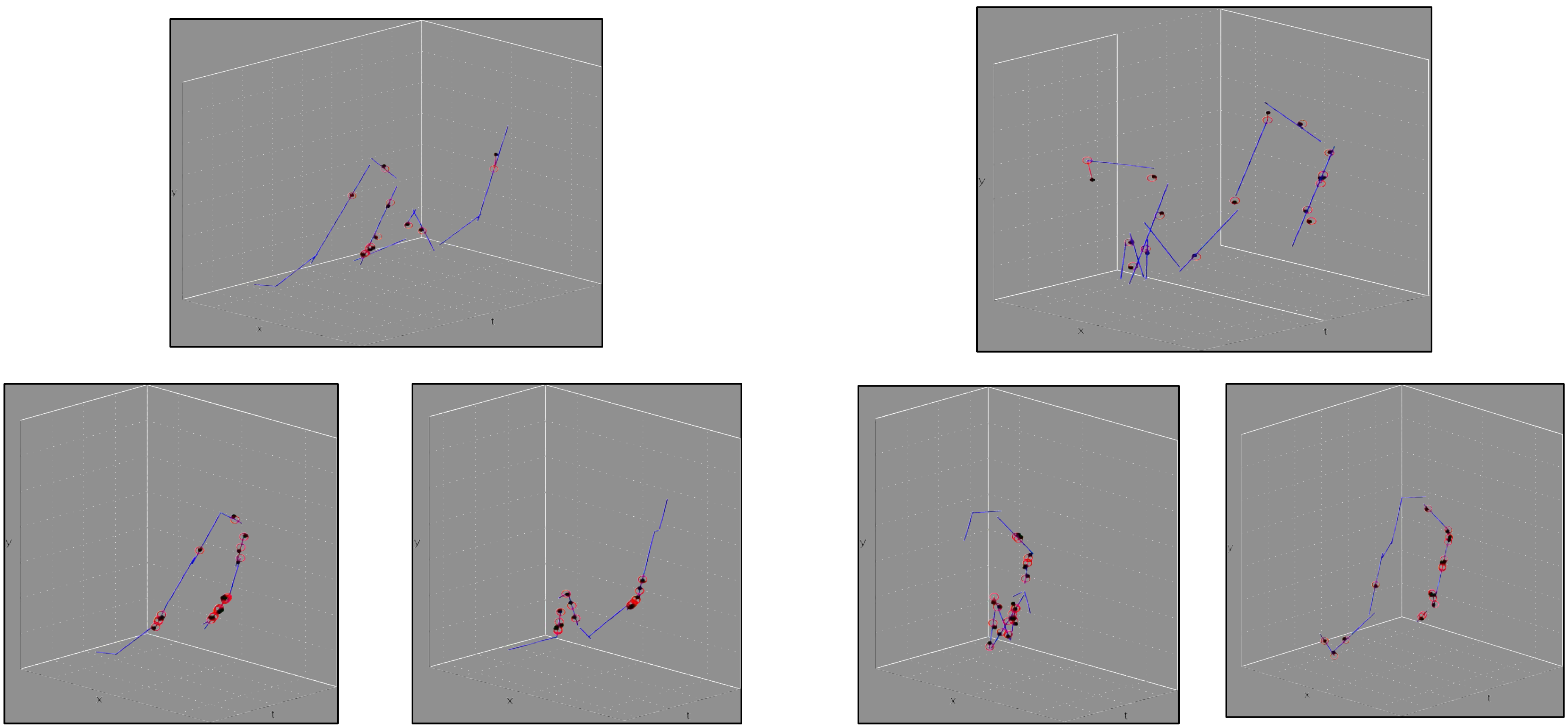



y
x
t

$(1+\epsilon)^2$ -coreset for

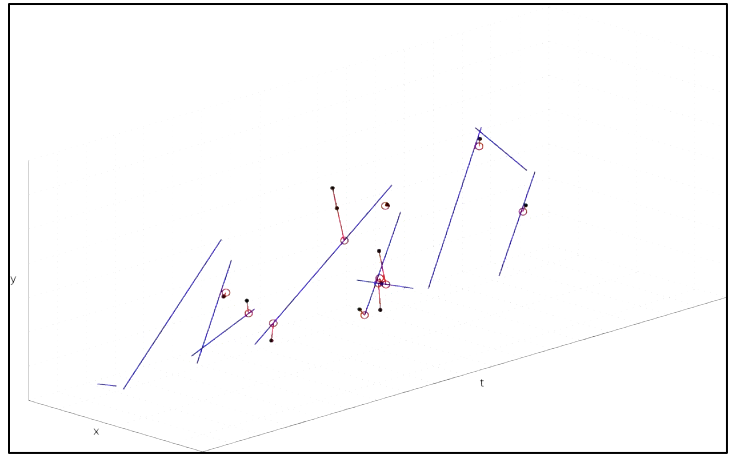
$P_1 \cup P_2 \cup P_3 \cup P_4$

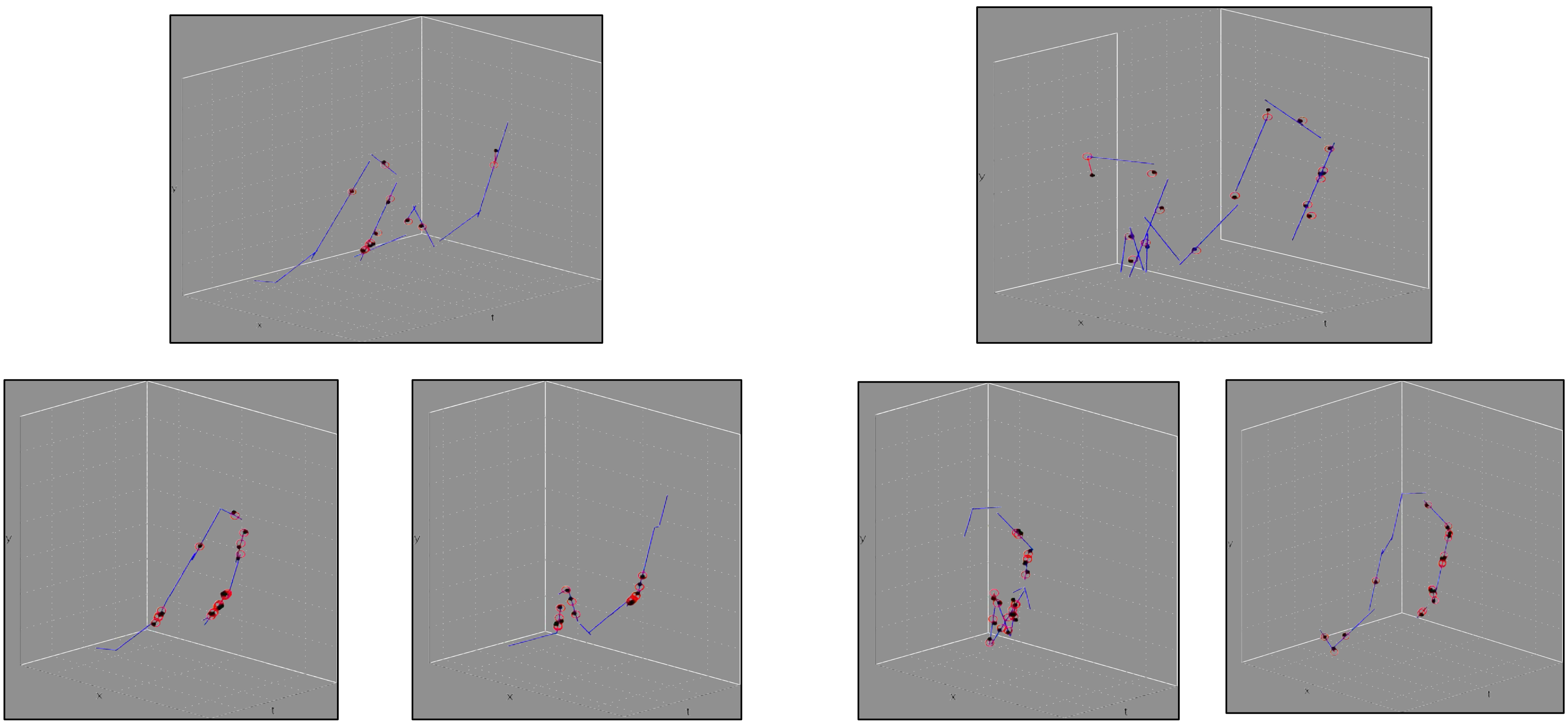


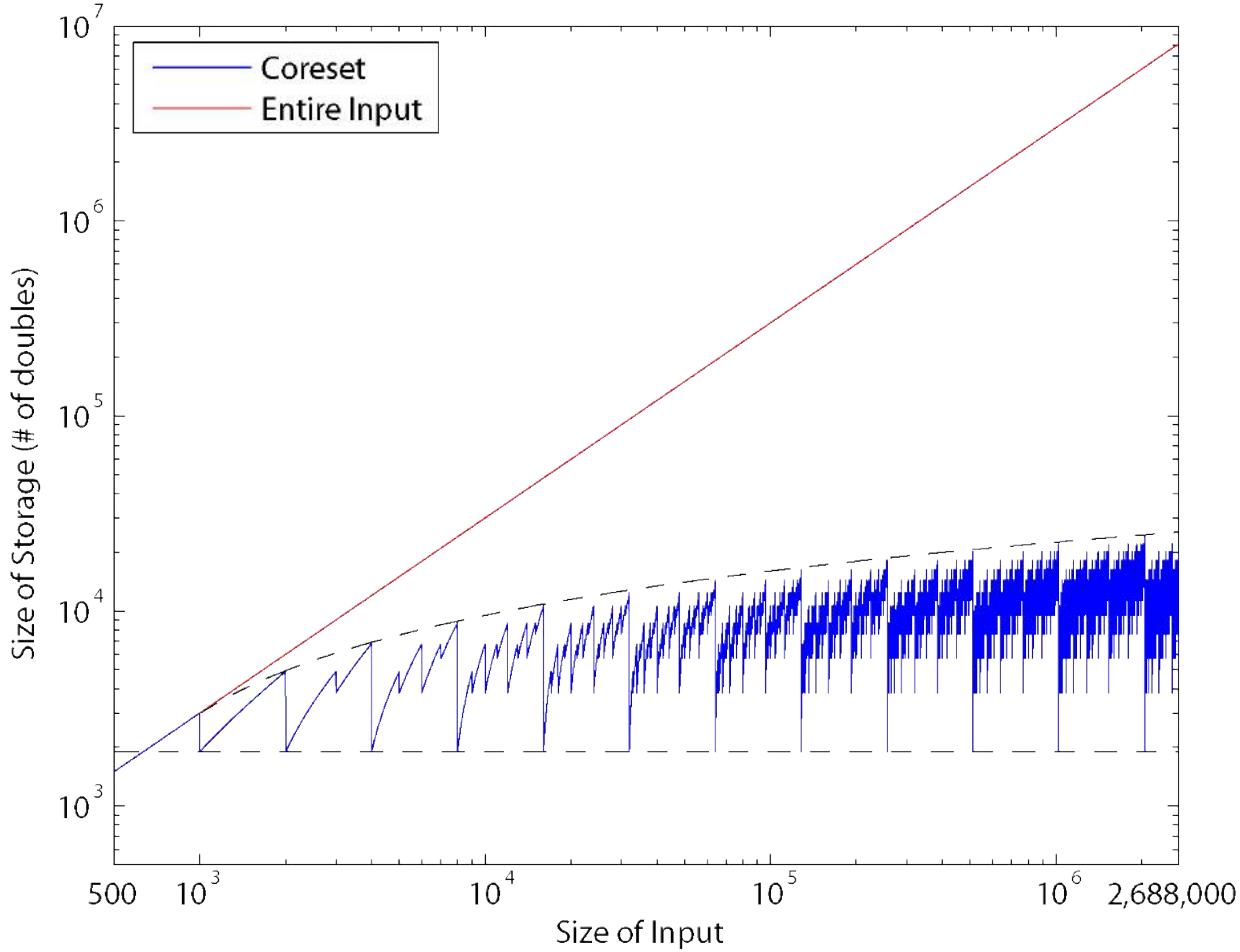


$(1+\epsilon)^3$-coreset for

$P_1 \cup P_2 \cup P_3 \cup P_4$

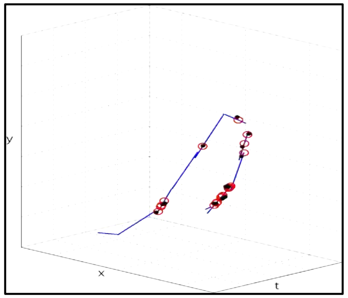


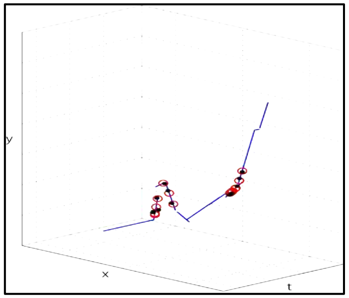


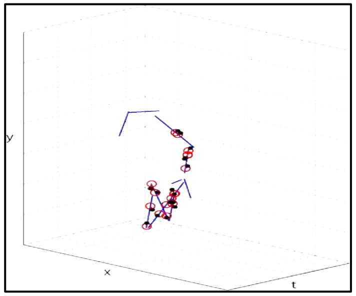


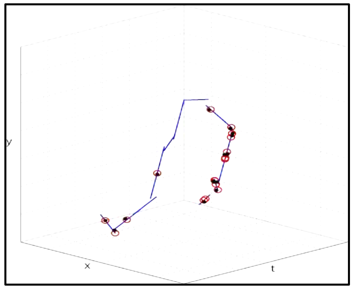
Coreset
Entire Input
Size of Storage (# of doubles)
Size of Input
10^7
10^6
10^5
10^4
10^3
500
10^3
10^4
10^5
10^6
2,688,000

Parallel Computation

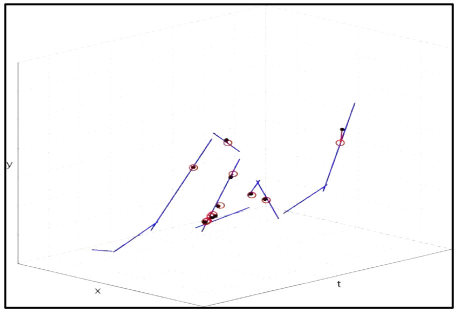


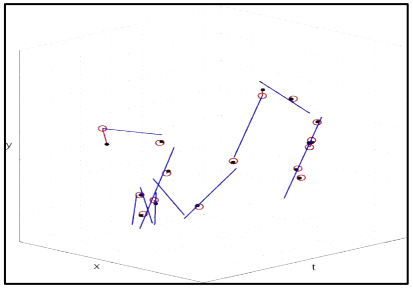


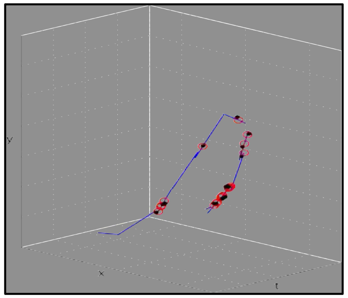


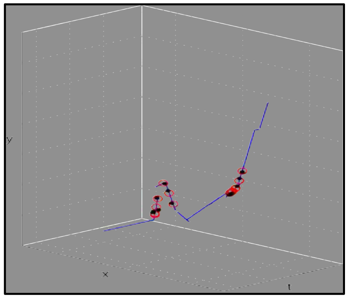


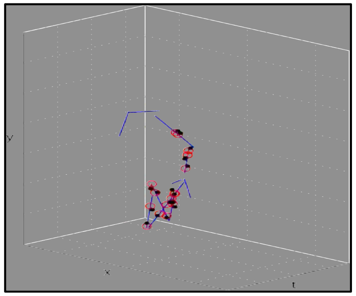
Parallel Computation

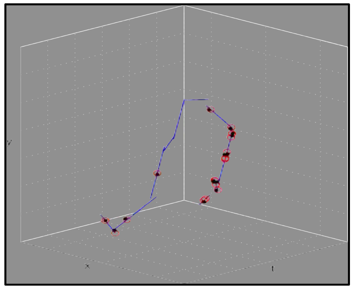






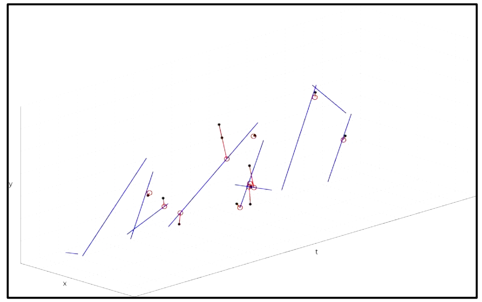


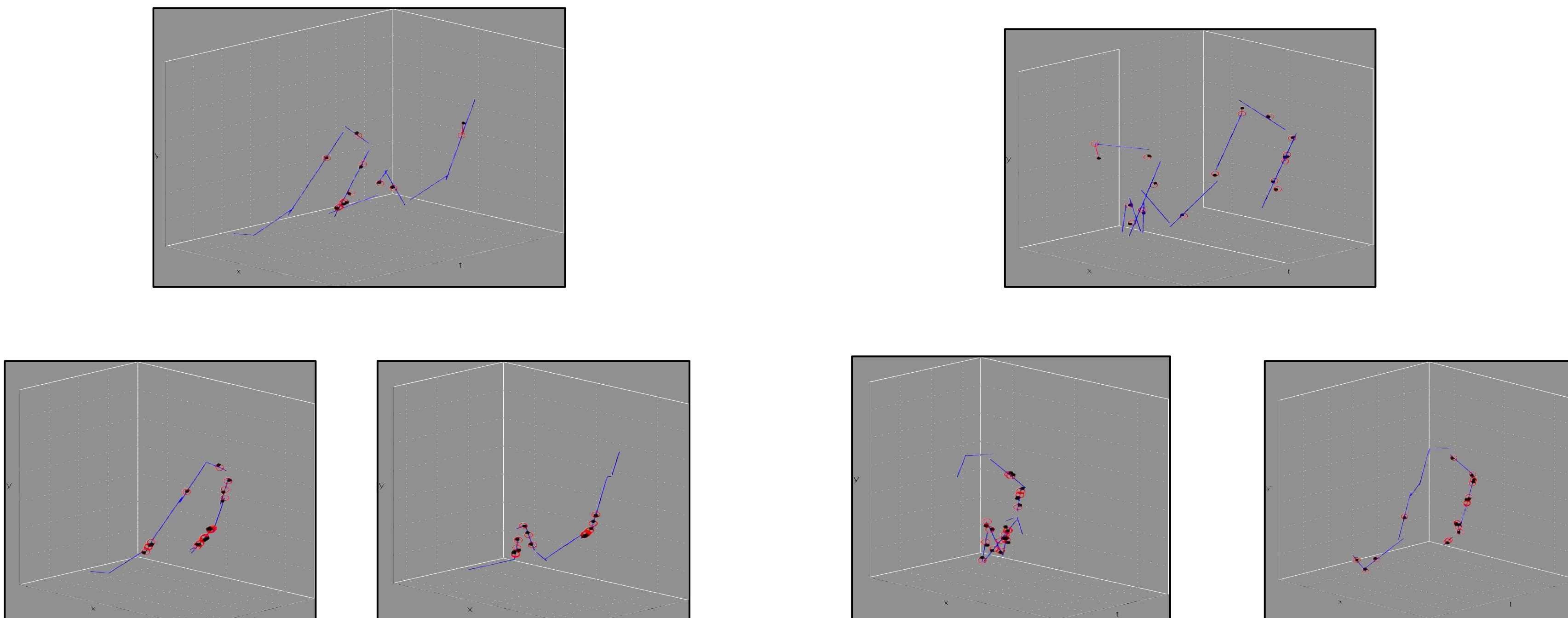




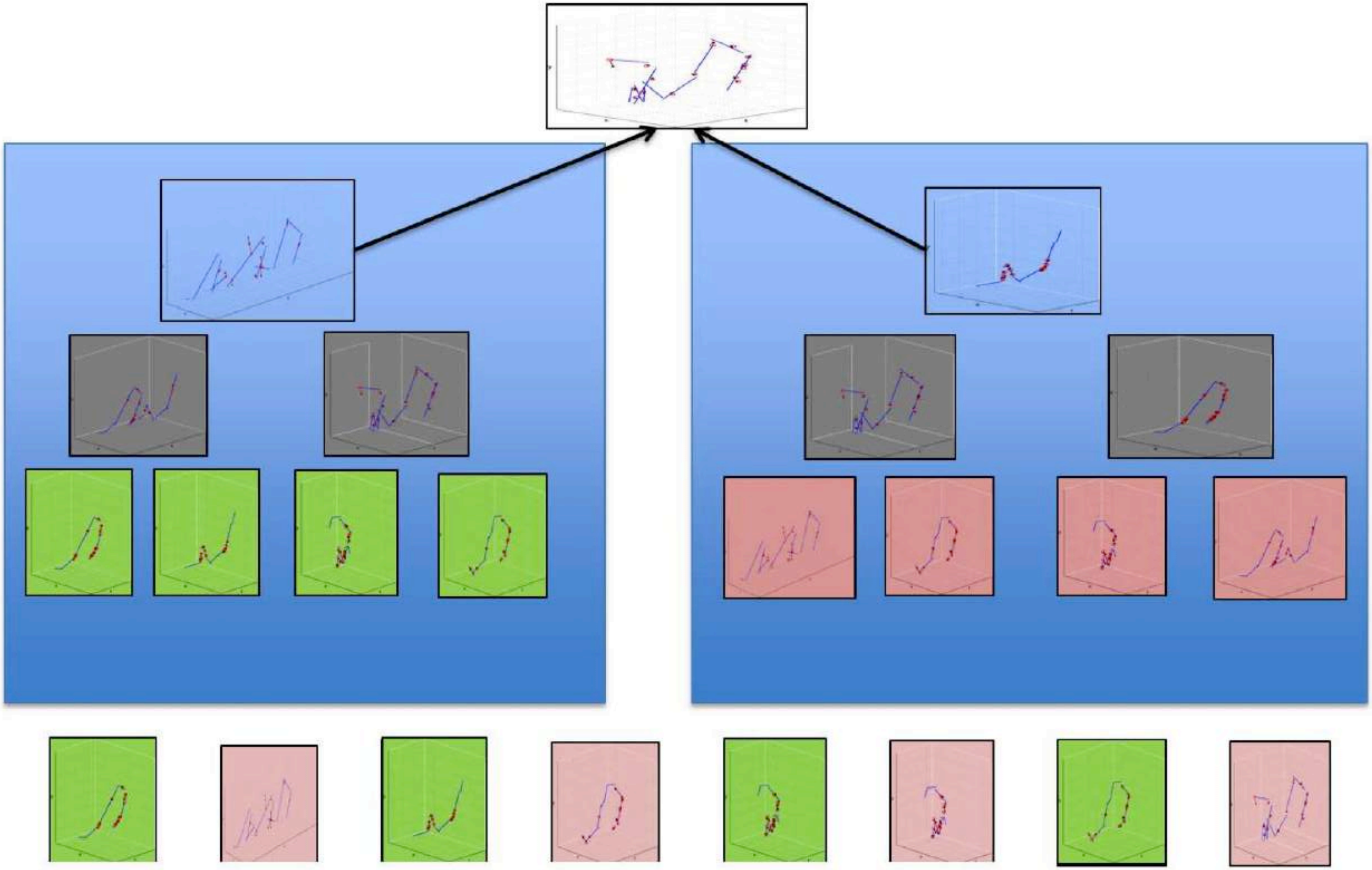
Parallel Computation

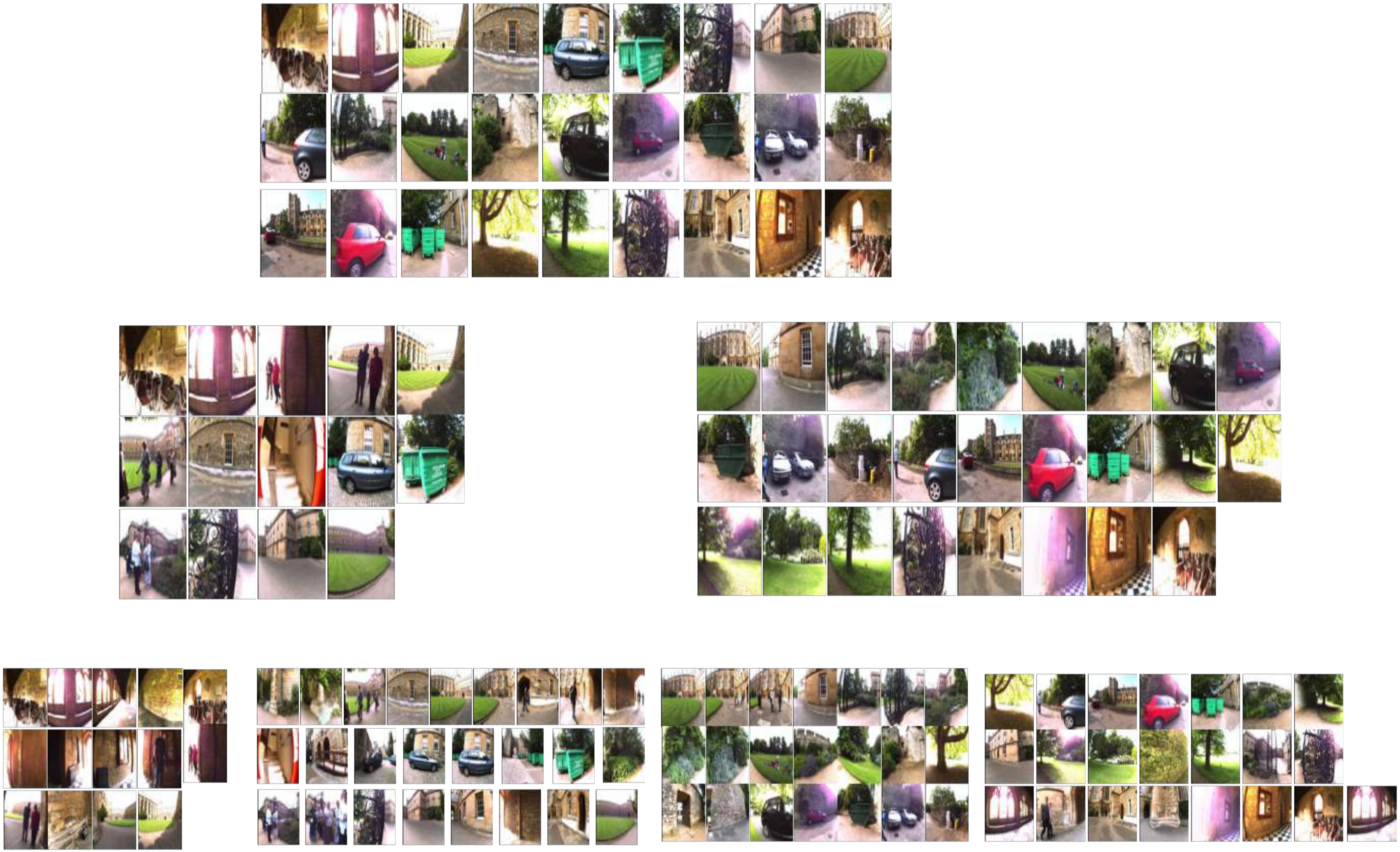
Run off-line algorithm on corset using single computer





Parallel+ Streaming Computation





ICRA’14 (With Rus, Paul and Newman)

In this talk

- Coresets for convex optimization:
  a generic framework for learning kernel
- E.g: Logistic regression, dimensional reduction (SVD) with outliers,
  L_p subspace embedding
- Main tool:
  generic-SVD via coreset for John Ellipsoid
- Relation to obstacle detection
  and path planning

Related Work

- Clarkson (SODA'2005)
  - Approximation for L_1 regression using weak coreset (only for off-line optimization)
- A. Dasgupta, P.Drineas, B. Harb, R. Kumar, M. Mahoney (SODA'2008)

  Weak coreset for L_p regression
- LaValle & Kuffmer, RRT trees (1998)

  Heuristics for path planning using sampling

Theorem [Feldman, Langberg, STOC'11]

[V. Braverman, F., Lang, Submitted]

Suppose that

cost$(P,x) := \sum_{p \in P} w(p) k(p,x)$

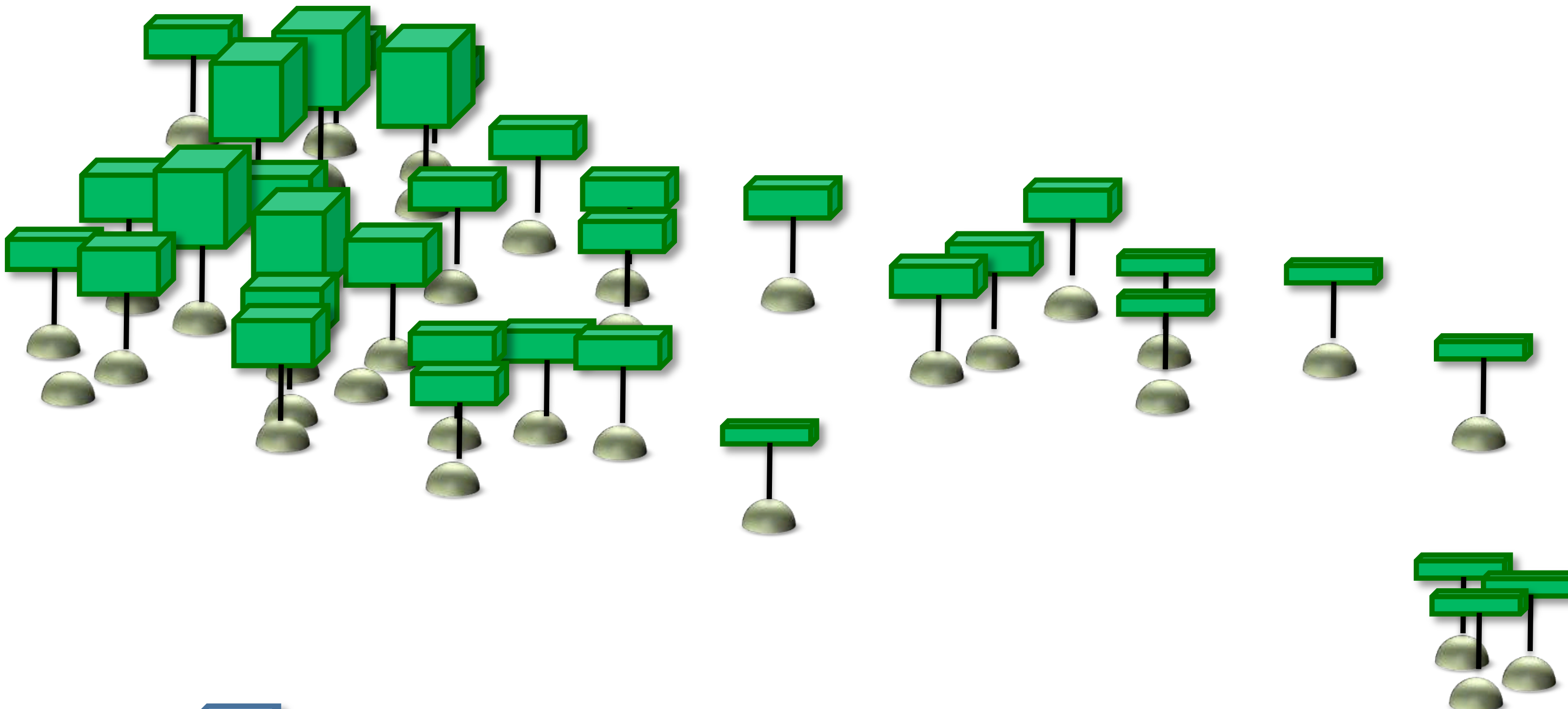
where $k : P \times X \to [0, \infty)$.

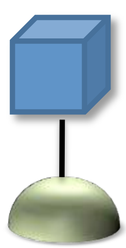
A sample $C \subseteq P$ from the distribution

$\text{sensitivity}(p) = \max_{x \in X} k(p,x) / \sum_{p'} k(p',x)$

is a coreset if $|C| \sim \text{dimension of } X / \epsilon^2 \cdot \sum_{p} \text{sensitibity}(p)$

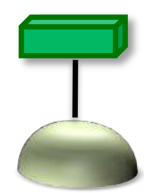
Importance Weights





$Sensitivity(p)$

Sampling distribution



$1/$
$Sensitivity(p)$

Weights

Sensitivity for convex optimization

- We want to minimize/estimate

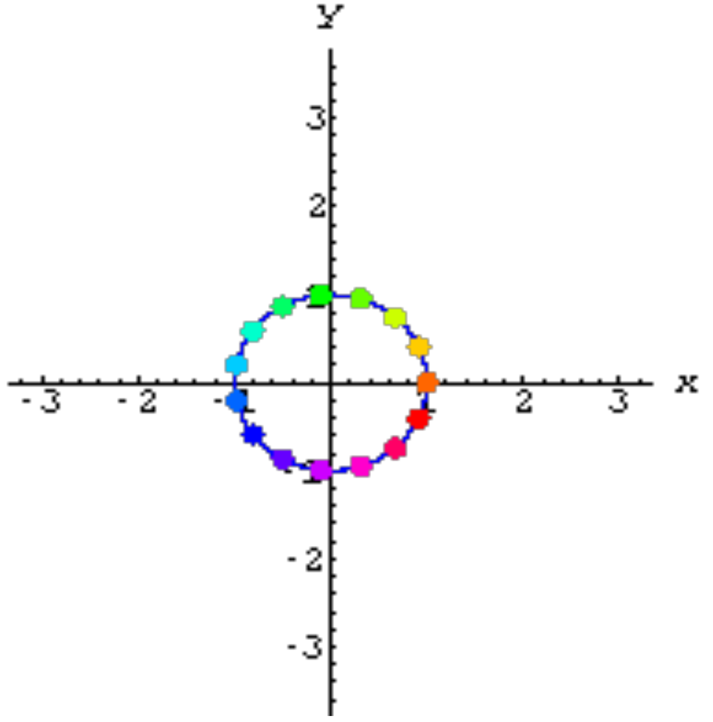
  $f(x) \sim cost(P,x) = \sum_{p \in P} k(p,x)$

  over $x \in X = \mathbb{R}^d$,

  where f is convex

Query space as a convex shape

- Example: $k(p,x) = |px|^2$
  $f(x) = ||Px||^2$,



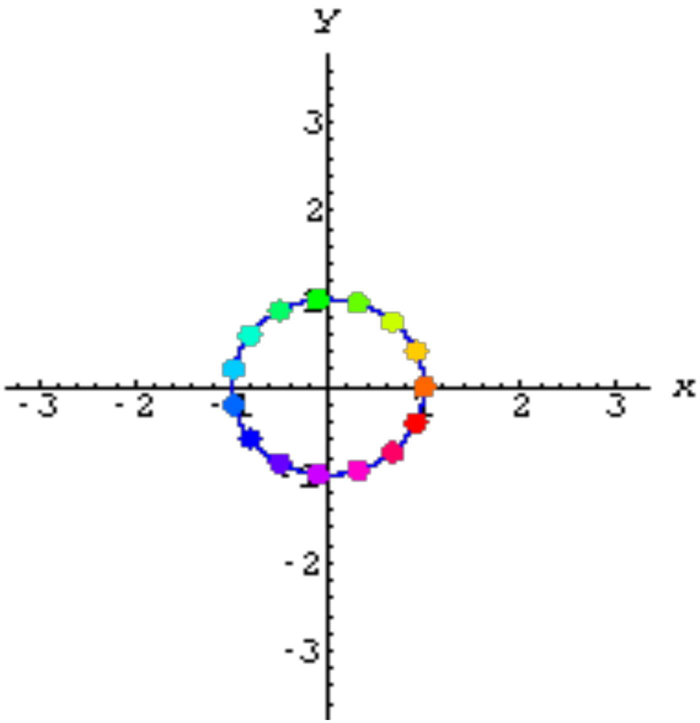
Every unit vector x is mapped to $x \cdot f(x)$

Gif by Todd Will

Query space as a convex shape

- Example: $k(p,x)=|px|^2$

$f(x)=||Px||^2$,



Every unit vector x is mapped to $x\cdot f(x)$

The result is the Ellipsoid

$X_f = x\in\mathbb{R}^d\ f(x)\leq 1$

$=\{x\in\mathbb{R}^d\ |\ ||DV^T x||\leq 1\}$

where $P=UDV^T$ is the SVD of A, and we have an exact "coreset"

$||Px||=||UDV^T x||=||DV^T x||$

Gif by Todd Will

From Sensitivity Lens

$$k(p,x)/f(x) = |px|^2 / ||Px||^2 = |px/||Px|| |^2 = |uDV^T x/||UDV^T x|| |^2$$

$$= |uDV^T x/||DV^T x|| |^2 = |u \cdot DV^T x/||DV^T x|| |^2 \leq ||u||^2$$

$$\sum_{i=1}^{n} ||u_i||^2 = ||U||_F^2 = d$$

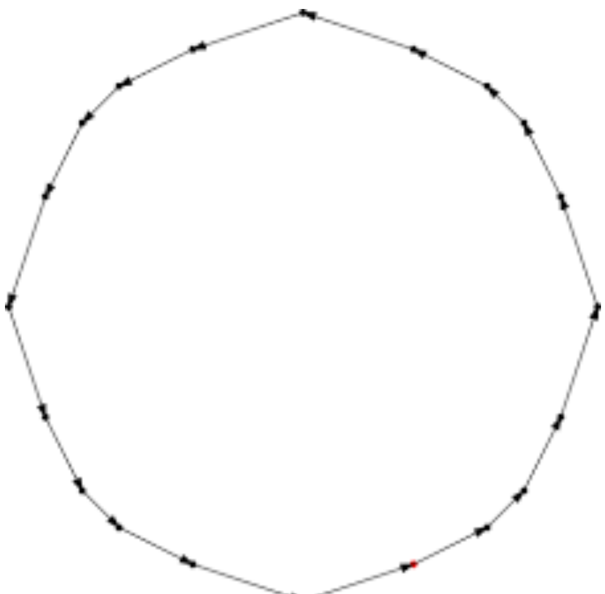
The general case

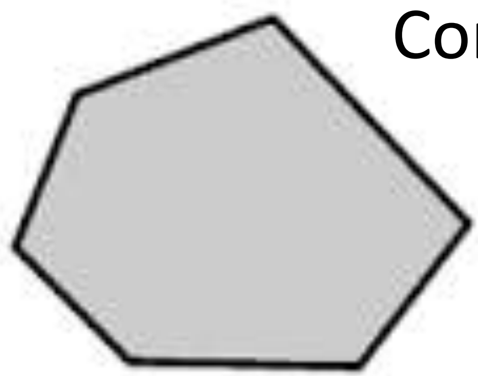
- Example: $k(p,x)=|px|^{}$

$f(x)=||Px||_1$

- Every unit vector x is mapped to $x \cdot f(x)$
- The result is a convex shape

$$X_f = x \in \mathbb{R}^d \, f(x) \leq 1$$
$$=\{x \in \mathbb{R}^d \mid ||Ax||_1 \leq 1\}$$



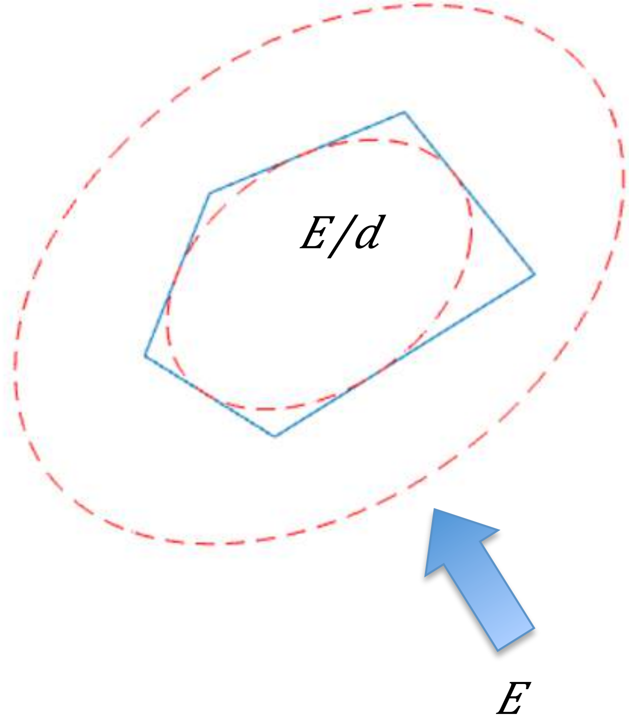


Complexity > $n^d > n$

Theorem (John's Ellipsoid)

- Every convex body contains an ellipsoid E/d such that E contains it.

- For a $E\in\mathbb{R}^{d\times d}$ and every $x\in\mathbb{R}^{d}$:



$$f(x)\sim ||Ex|| = ||DV^{T}x||$$

- We define $P=UDV^{T}$ as the *f*-SVD of *P*
- Bugs: (i) only d-approximation

  (ii) not subset of input point set P

From Sensitivity Lens

$$k(p,x)/f(x) = |px| / \|Px\|_1 = |px| / \|UDV^T x\|_1 \approx |uDV^T x| / \|DV^T x\|_2 \leq \|u\|_1$$

$$\sum_{i=1}^{n} \|u_i\|_1 = ?$$

Sensitivity for convex optimization

- We want to minimize/answer

  $f(x) \sim \sum_{p \in P} k(p,x)$

- $k(p,x) \sim g(|px|)$
- $a \cdot k(p,x) \sim k(p, a \cdot x)$
- Otherwise, we use level sets for X_f

Main Theorem [F, Tukan]

The sensitivity of a point $p \in P$ is at most

$$\max_{x} k(p,x)/f(x) \le \sum_{i=1}^{d} k(p, E^{-1} e_i)$$

and the total sensitivity (~size of coreset):

$$\sum_{p \in P} s(p) \in d^{O(1)}$$

Proof Sketch - sensitivity

$$k(p,x)/f(x) \sim k(p,x)/\|Ex\| \sim k(p, x/\|Ex\|) = k(uE, E^{-1} y)$$

$$\sim g(|uy|) \leq g(|u|_2) \leq g(|u|_1)$$

$$= g\left(\sum_{i=1}^{d} |ue_i|\right) \sim \sum_{i=1}^{d} g(|ue_i|)$$

$$\sim \sum_{i=1}^{d} k(uE, E^{-1} e_i) = \sum_{i=1}^{d} k(p, E^{-1} e_i)$$

Proof Sketch – total sensitivity

$$\sum_{p\in P} \sum_{i=1}^{d} k(p, E^{-1} e_i) = \sum_{i=1}^{d} \sum_{p\in P} k(p, E^{-1} e_i)$$

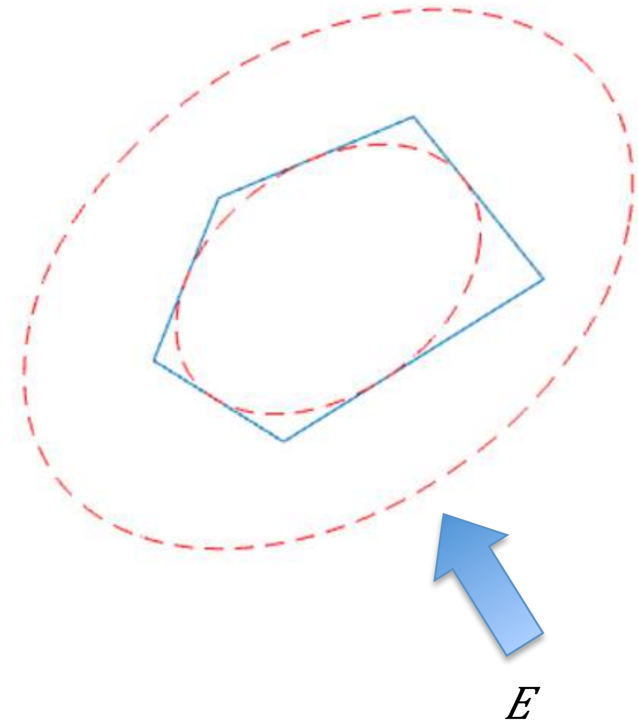
$$= \sum_{i=1}^{d} f(E^{-1} e_i) \sim \sum_{i=1}^{d} ||E \cdot E^{-1} e_i|| \sim$$

$$\sum_{i=1}^{d} ||e_i|| = d$$

How do we compute the ellipsoid E?

$X\downarrow f = x \in \mathbb{R}\uparrow d \; f(x) \leq 1$

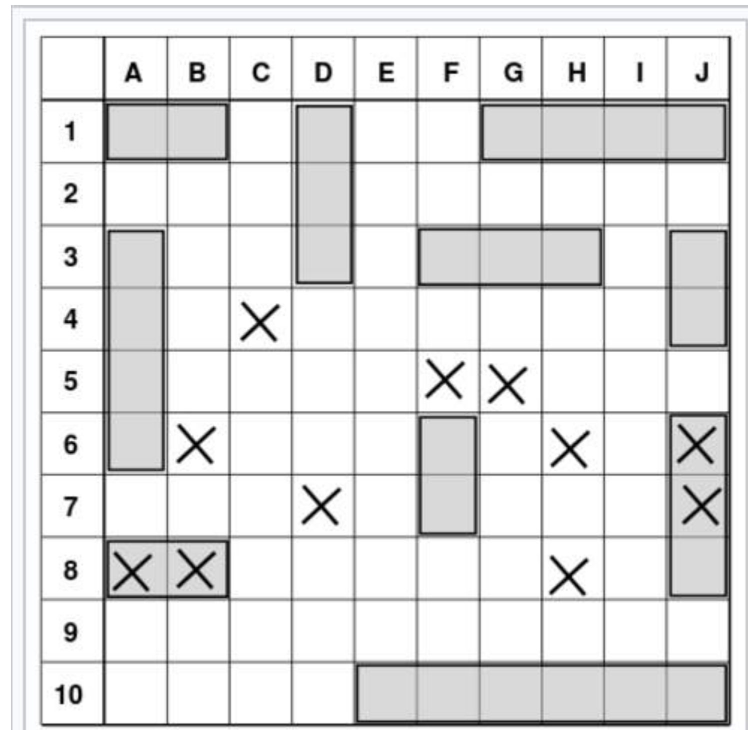
$\sim ||Ex|| = ||DV\uparrow T \; x||$



Only using oracle membership.

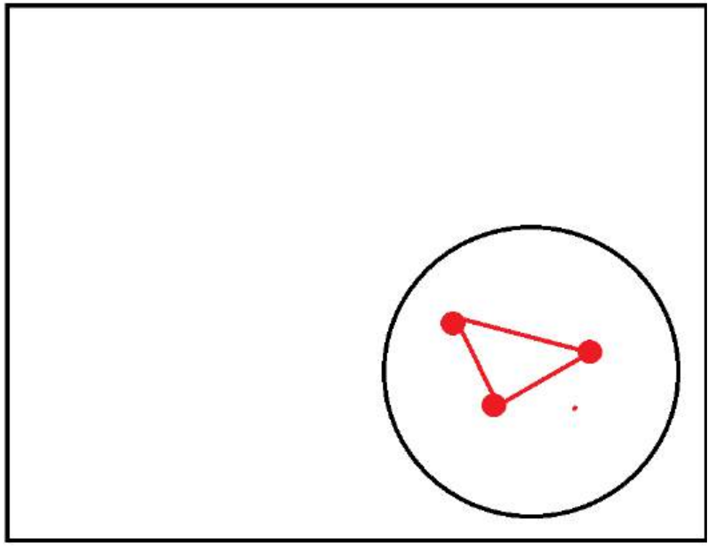
Path Planning in the Dark

- In control space we know start & destination configurations
- Can only ask boolean queries regarding feasible positions
- As in Battleships (game)

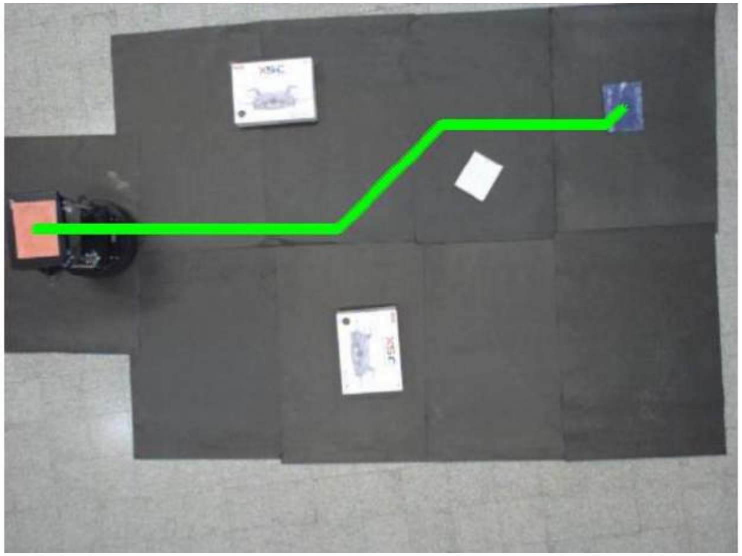


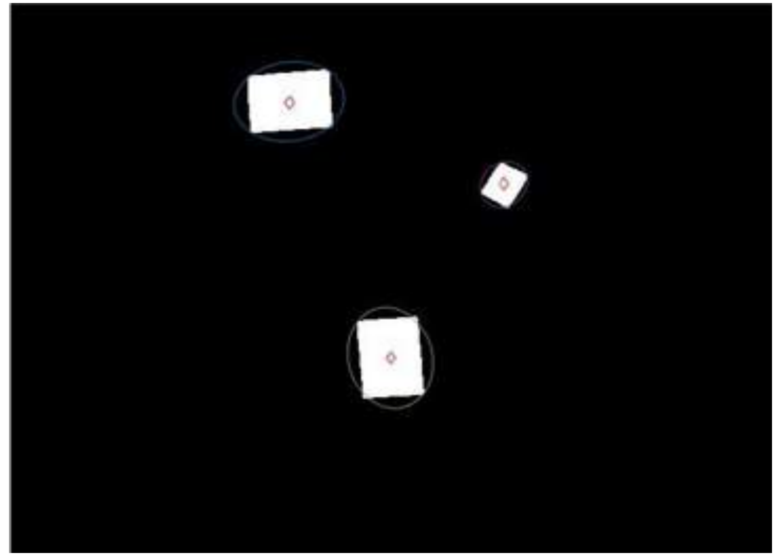
Path Planning in the Dark

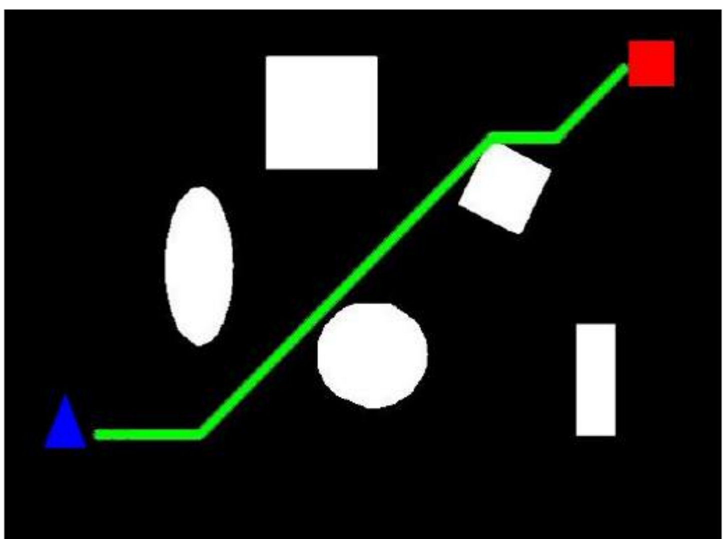
- We want minimum number of queries for maximum approximation error
- Existing algorithms have no guarantee for optimality
- Approximation by convex polygons

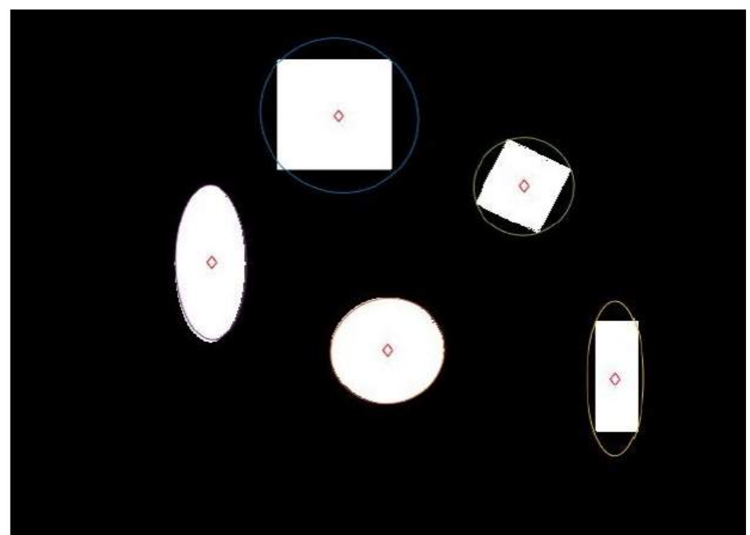


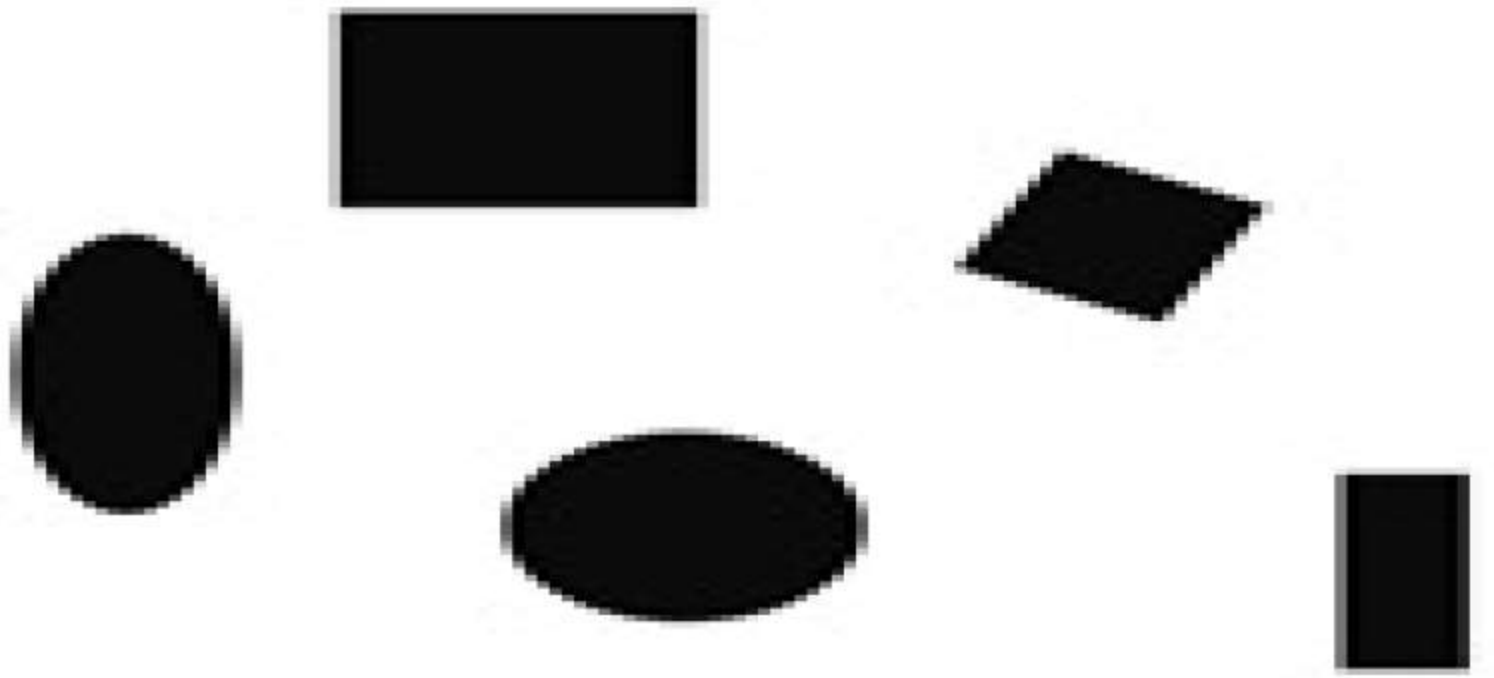
Path Planning



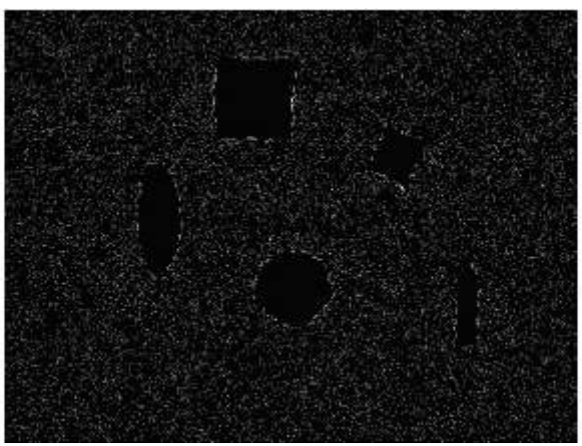




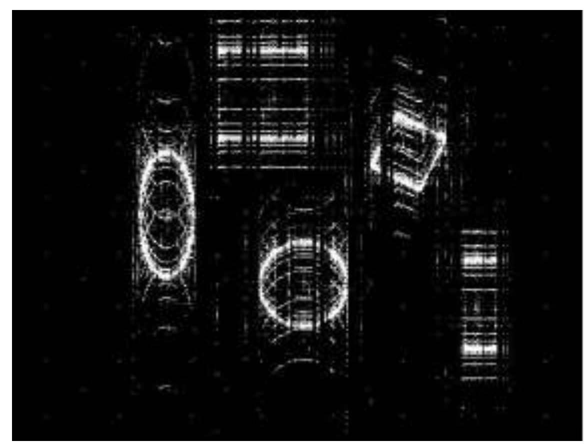


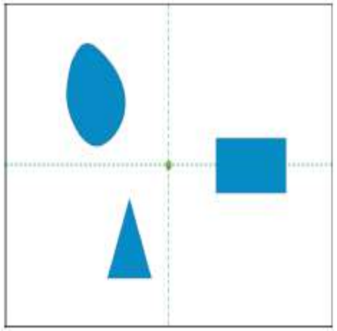


RRT

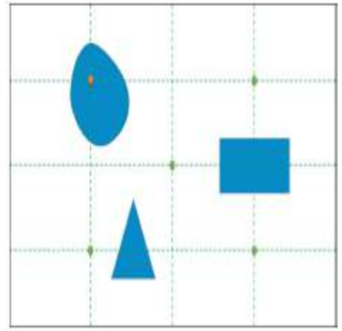


Our Algorithm

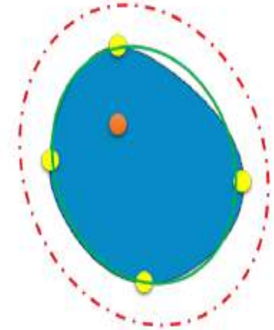




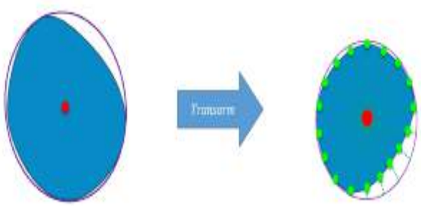
(a) Epsilon grid sampling; First iteration



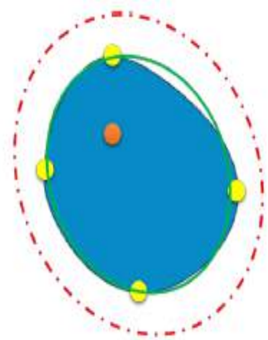
(b) Epsilon grid sampling; Second iteration



(c) d^{2d} approximation to John Ellipsoid



(d) Applying "Epsilon Star" on the transform space



(e) $1+\epsilon$ approximation to the real convex bodies

Open Problems

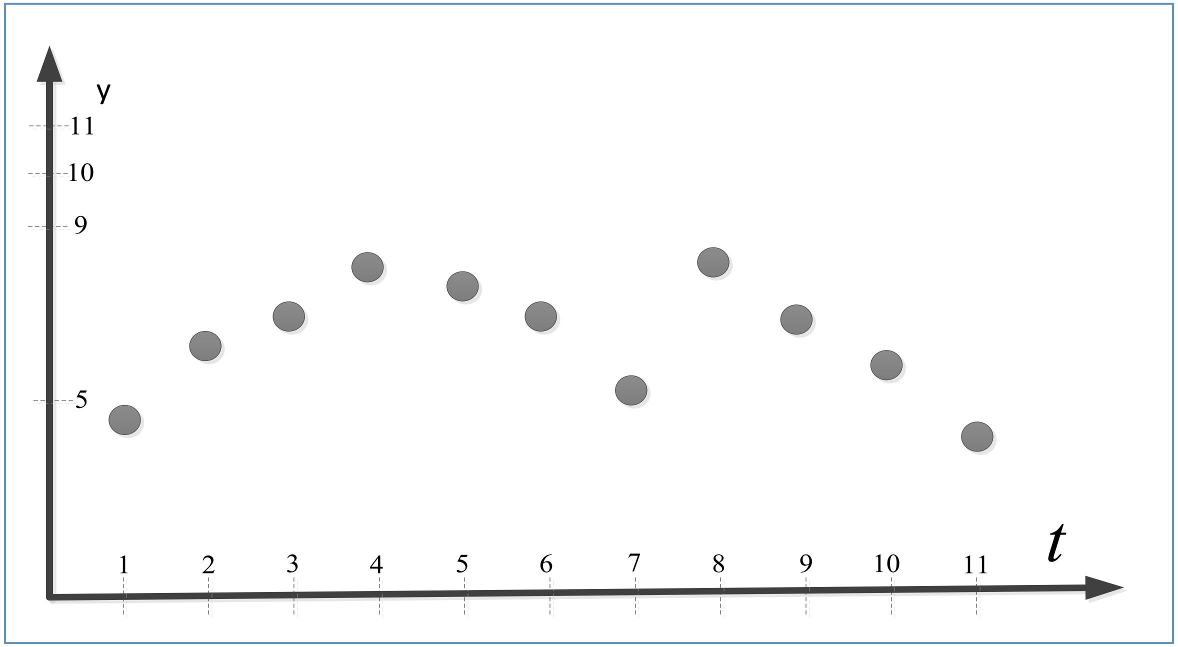
- More Coresets
  - Deep learning, Decision trees, Sparse data

  3D Navigation and Mapping, Robotics
- Private Coresets, [STOC'11, with Fiat et al.]
- Homomorphic Encryption Coresets
- Generic software library
  - Coresets on Demand on the cloud
- Sensor Fusion (GPS+Video+Audio+Text+..)

Thank you !

k—Segment Queries

Input: *d*-dimensional signal *P* over time



Coreset for k-means

[Feldman, Sohler, Monemizadeh, SoCG'07]

Coreset for k-means can be computed by choosing points from the distribution:

$$\text{sensitivity}(p) = dist(p,q^{*})/\sum_{p'} dist(p',q^{*}) + 1/n_{p}$$

q^{*} = k-means of P

n_{p} = number of points in the cluster of p

$|C| = k \cdot d/\epsilon^{2}$

Coreset for k-means

[Feldman, Sohler, Monemizadeh, SoCG'07]

Coreset for k-means can be computed by choosing points from the distribution:

$$\text{sensitivity}(p) = \frac{dist(p, q^*)}{\sum_{p'} dist(p', q^*)} + \frac{1}{n_p}$$

q^* = k-means of P Or approximation [SoCg07, Feldma, Sharir, Fiat]

n_p = number of points in the cluster of p

$|C| = k \cdot d/\epsilon^2$

Coreset for k-means

[Feldman, Sohler, Monemizadeh, SoCG'07]

Coreset for k-means can be computed by choosing points from the distribution:

$$\text{sensitivity}(p) = \frac{dist(p,q^*)}{\sum_{p'} dist(p',q^*)} + \frac{1}{n_p}$$

q^* = k-means of P Or approximation [SoCg07, Feldma, Sharir, Fiat]

n_p = number of points in the cluster of p

$|C| = \cancel{k \cdot d/\epsilon^2}$ $k \cdot (k/\varepsilon)/\epsilon^2$ [SODA'13, Feldman, Schmidt, ..]

Coreset for k-means

[Feldman, Sohler, Monemizadeh, SoCG'07]

Coreset for k-means can be computed by choosing points from the distribution:

$$\mathrm{sensitivity}(p) = dist(p,q^*)/\sum_{p'} dist(p',q^*) + 1/n_p$$

q^* = k-means of P

n_p = number of points in the cluster of p

$|C| = k \cdot d/\epsilon^2$

Coreset for k-means

[Feldman, Sohler, Monemizadeh, SoCG'07]

Coreset for k-means can be computed by choosing points from the distribution:

$$\text{sensitivity}(p) = dist(p,q^*)/\sum_{p'} dist(p',q^*) + 1/n_p$$

q^* = k-means of P Or approximation [SoCg07, Feldma, Sharir, Fiat]

n_p = number of points in the cluster of p

$|C| = k \cdot d/\epsilon^2$

Coreset for k-means

[Feldman, Sohler, Monemizadeh, SoCG'07]

Coreset for k-means can be computed by choosing points from the distribution:

$$\text{sensitivity}(p) = \frac{dist(p, q^*)}{\sum_{p'} dist(p', q^*)} + \frac{1}{n_p}$$

q^* = k-means of P Or approximation [SoCg07, Feldma, Sharir, Fiat]

n_p = number of points in the cluster of p

$|C| = k \cdot d / \epsilon^2$

Coreset for k-means

[Feldman, Sohler, Monemizadeh, SoCG'07]

Coreset for k-means can be computed by choosing points from the distribution:

$$\text{sensitivity}(p) = \frac{dist(p,q^*)}{\sum_{p'} dist(p',q^*)} + \frac{1}{n_p}$$

q^* = k-means of P Or approximation [SoCg07, Feldma, Sharir, Fiat]

n_p = number of points in the cluster of p

$|C| = \cancel{k \cdot d/\epsilon^2}$ $k \cdot (k/\varepsilon)/\epsilon^2$ [SODA'13, Feldman, Schmidt, ..]

The chicken-and-egg problem

1. We need approximation to compute the coreset
2. We compute coreset to get a fast approximation to a problem

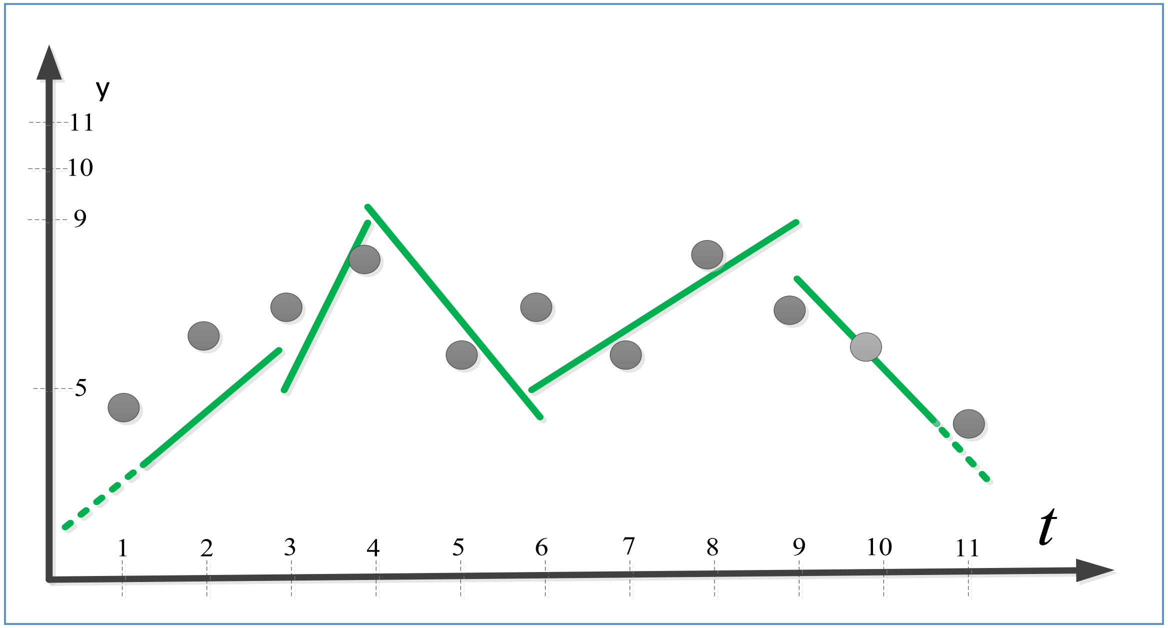
Lee-ways:

I. Bi-criteria approximation
II. Heuristics
III. polynomial time reduced to linear time by the merge-reduce tree

k–Segment Queries

Input: *d*-dimensional signal *P* over time

Query: *k* segments over time



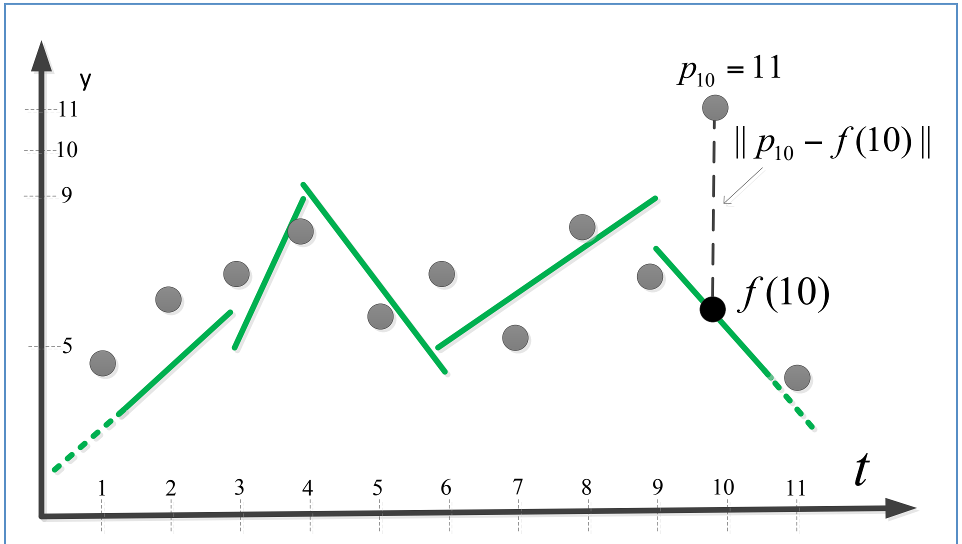
k-Piecewise linear function *f* over *t*

k–Segment Queries

Input: *d*-dimensional signal *P* over time

Query: *k* segments over time

Output: Sum of squared distances from *P*

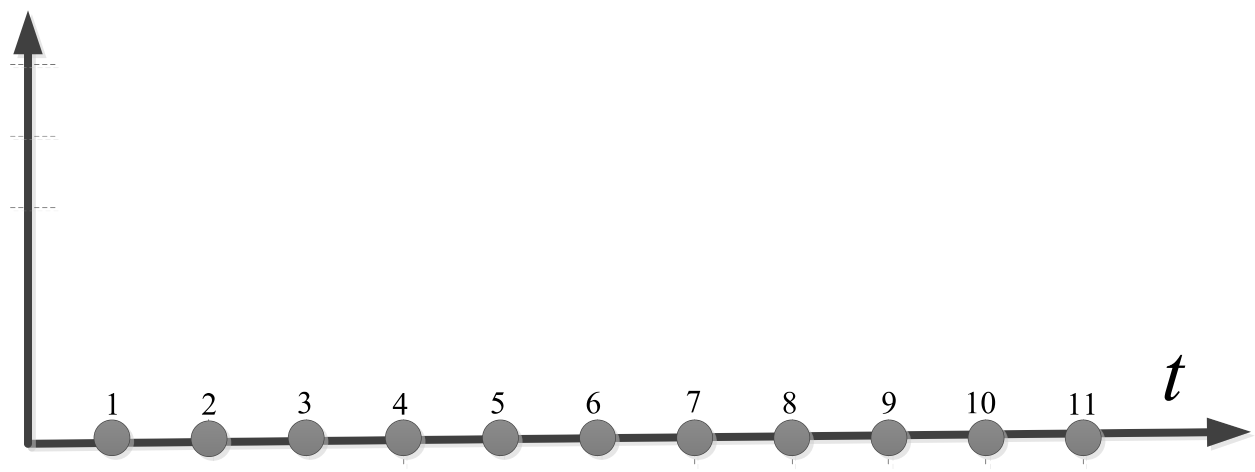


$$\mathrm{cost}(P,f) := \sum_t \| f(t) - p_t \|^2$$

Observation:

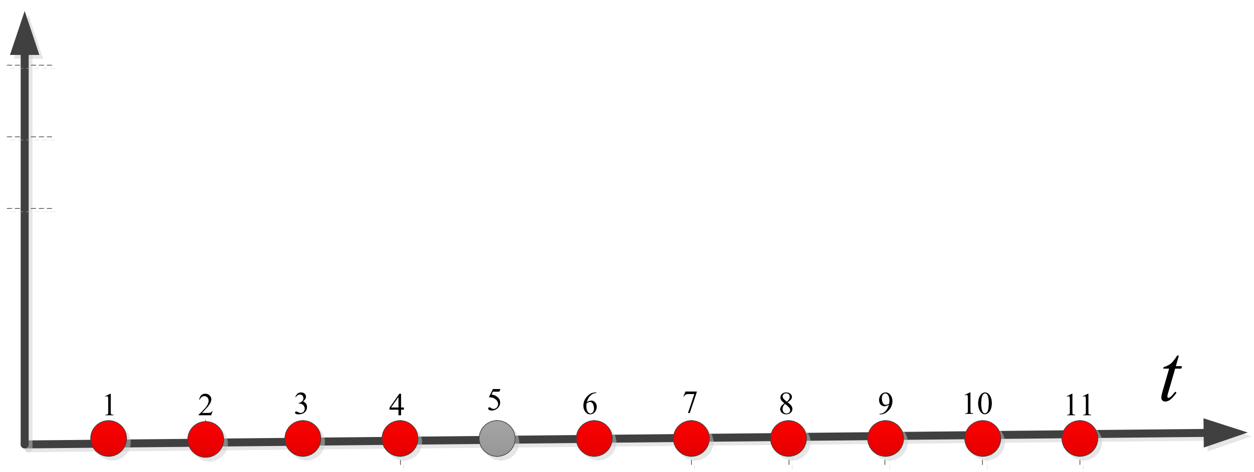
No small coreset $C \subset P$ exists for k-segment queries

Input P: n points on the x-axis



Input P: n points on the x-axis

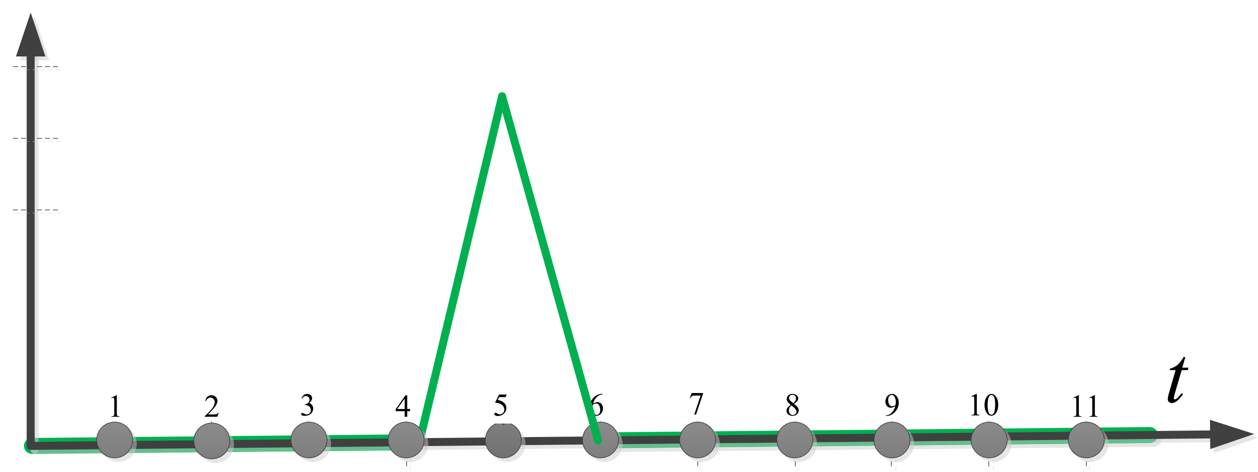
Coreset C: all points except one



Input P: n points on the x-axis

Coreset C: all points except one

Query f: covers all except this one



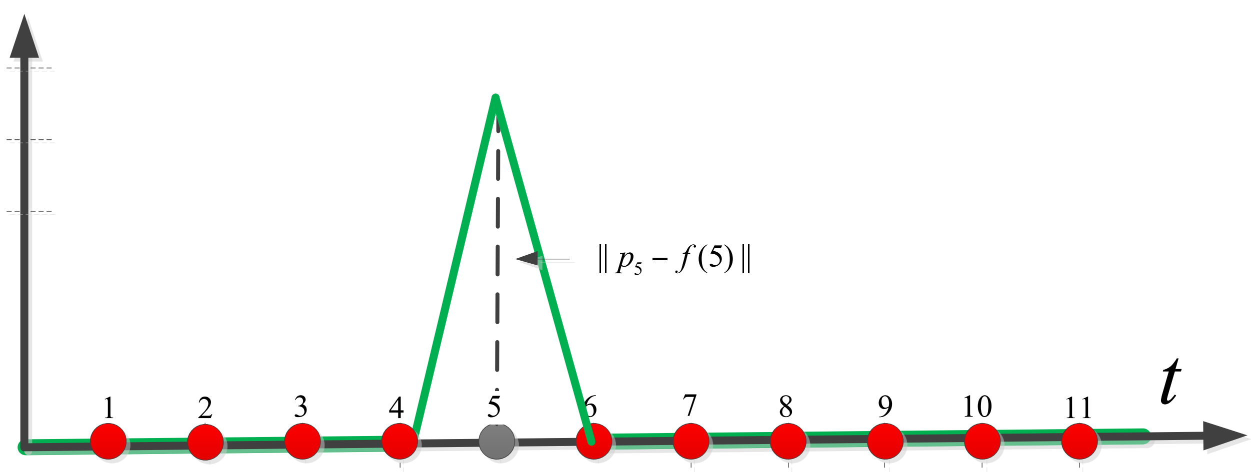
Input P: *n* points on the *x*-axis

Coreset *C*: all points except one

Query *f*: covers all except this one

$\mathrm{Cost}(P,f) > 0$

$\mathrm{Cost}(C,f) = 0$



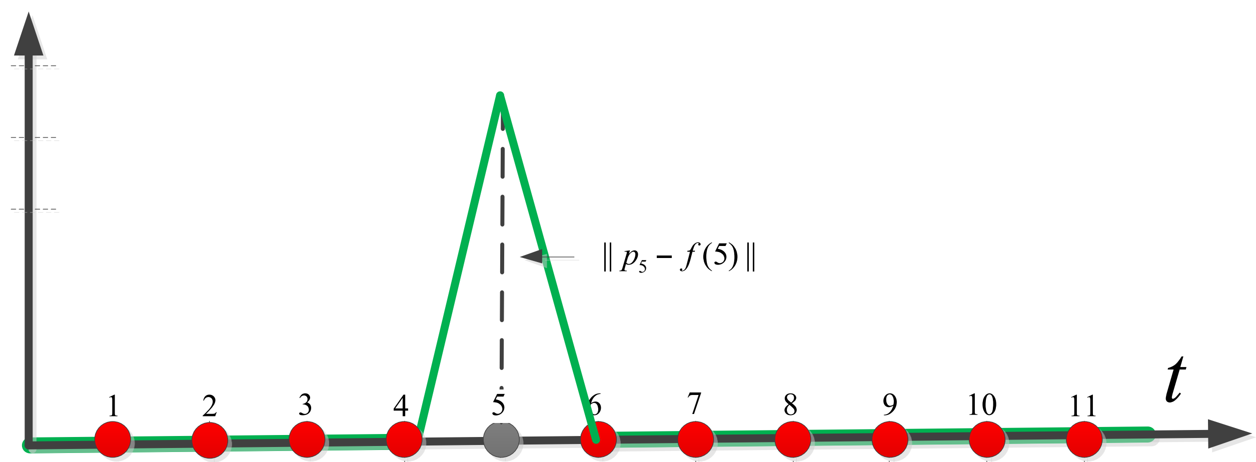
Input P: *n* points on the *x*-axis

Coreset *C*: all points except one

Query *f*: covers all except this one

$$\frac{\text{Cost}(P,f) > 0}{\text{Cost}(C,f) = 0}$$

⇒ Unbounded factor approximation



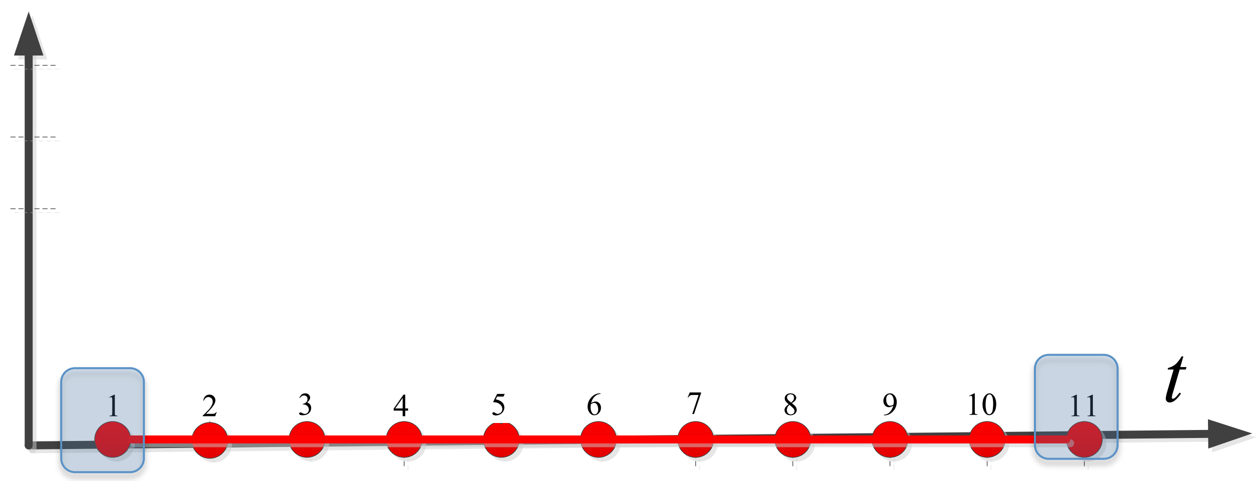
For every point *p*:

Sensitivity(*p*) = $\max_{q \in Q} dist(p,q) / \sum_{p'} dist(p',q) = 1$

Total sensitivities: *n*

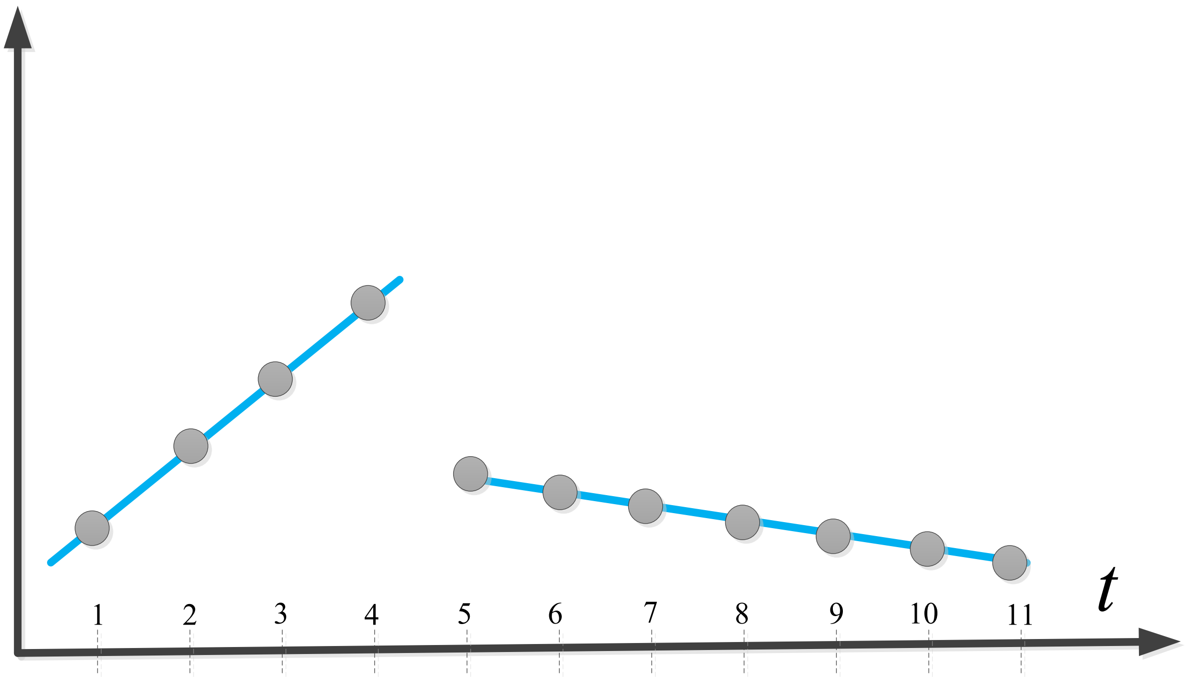
Observation:

Points on a segment can be stored by the two indexes of their end-points



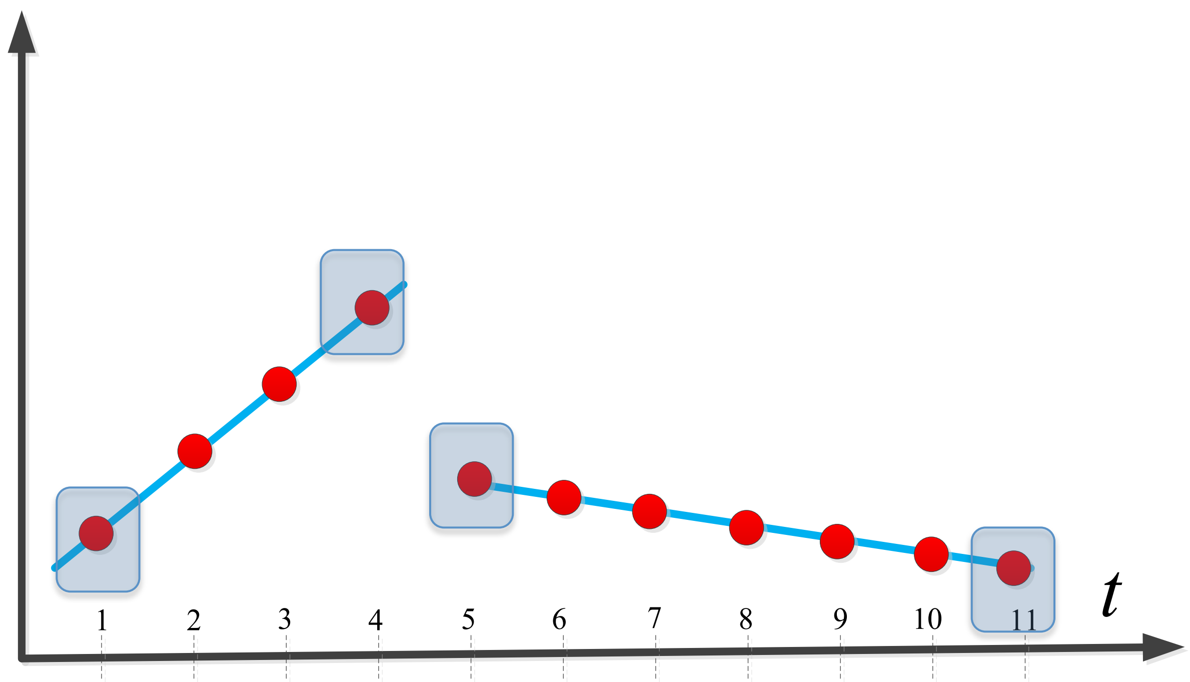
Observation:

Points on a segment can be stored by the two indexes of their end-points and the slope of the segment



Observation:

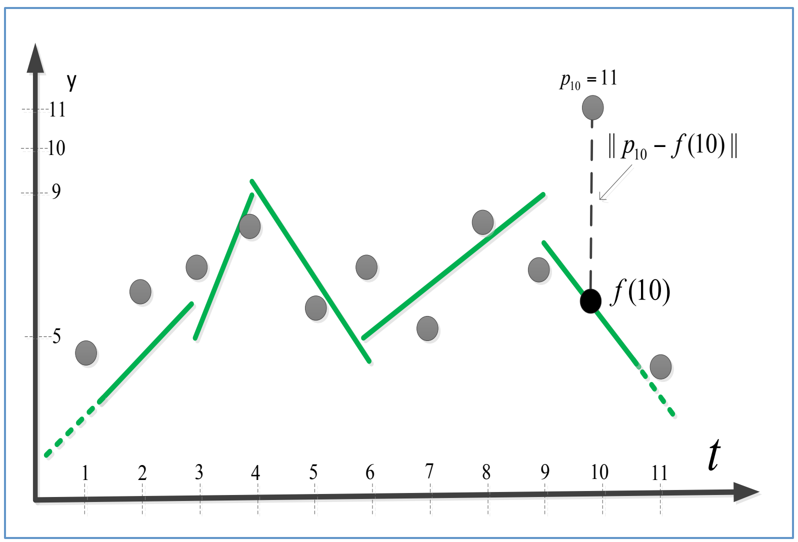
Points on a segment can be stored by the two indexes of their end-points and the slope of the segment



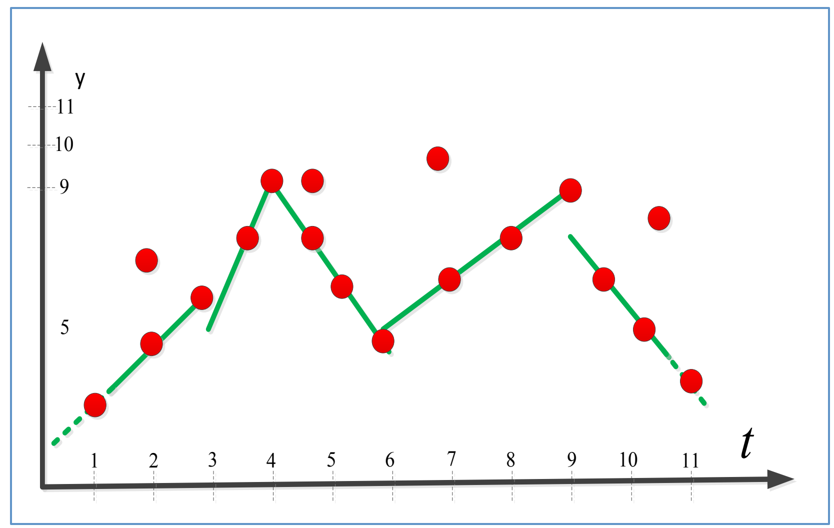
Definition: Coreset

A weighted set $C \subset P$ such that
for every *k*-segment *f* :

$$\text{cost}(P,f) \sim \text{costw}(C,f)$$



$\sim$



$\sum_{t} \|f(t)-p_t\|$

$\sum_{p_t \in C} w(p_t)\cdot \|f(t)-p_t\|$

Surprising Applications

1. (1-epsilon) approximations:
   Heuristics work better on coresets

2. Running constant factor on epsilon-coresets helps

3. Coreset for one problem is good for a lot of unrelated problems

4. Coreset for O(1) points

Implementation

- The worst case and sloppy (constant) analysis is not so relevant

- In Thoery:
  a random sample of size $1/\epsilon$ yields $(1+\epsilon)$ approximation with probability at least $1-\delta$.
  In Practice:
  Sample s points, output the approximation ϵ and its distribution

- Never implement the algorithm as explained in the paper.

Coreset for k-means

[Feldman, Sohler, Monemizadeh, SoCG'07]

Coreset for k-means can be computed by choosing points from the distribution:

$$\text{sensitivity}(p) = dist(p,q^{*})/\sum_{p'} dist(p',q^{*}) + 1/n_p$$

q^{*} = k-means of P

n_p = number of points in the cluster of p

$|C| = k \cdot d/\epsilon^{2}$

Coreset for k-means

[Feldman, Sohler, Monemizadeh, SoCG'07]

Coreset for k-means can be computed by choosing points from the distribution:

$$\text{sensitivity}(p) = dist(p,q^*)/\sum_{p'} dist(p',q^*) + 1/n_p$$

q^* = k-means of P Or approximation [SoCg07, Feldma, Sharir, Fiat]

n_p = number of points in the cluster of p

$|C| = k \cdot d/\epsilon^2$

Coreset for k-means

[Feldman, Sohler, Monemizadeh, SoCG'07]

Coreset for k-means can be computed by choosing points from the distribution:

$$\text{sensitivity}(p) = dist(p,q^*)/\sum_{p'} dist(p',q^*) + 1/n_p$$

q^* = k-means of P Or approximation [SoCg07, Feldma, Sharir, Fiat]

n_p = number of points in the cluster of p

$|C| = k \cdot d/\epsilon^2$

Coreset for k-means

[Feldman, Sohler, Monemizadeh, SoCG'07]

Coreset for k-means can be computed by choosing points from the distribution:

$$\text{sensitivity}(p) = \frac{dist(p,q^*)}{\sum_{p'} dist(p',q^*)} + \frac{1}{n_p}$$

q^* = k-means of P Or approximation [SoCg07, Feldma, Sharir, Fiat]

n_p = number of points in the cluster of p

$|C| = \cancel{k\cdot d/\epsilon^2}$ $k\cdot (k/\varepsilon)/\epsilon^2$ [SODA'13, Feldman, Schmidt, ..]