technische universität dortmund

SFB 876 Verfügbarkeit von Information durch Analyse unter Ressourcenbeschränkung

Coreset and sampling approaches for the analysis of very large data sets

Christian Sohler

European Research Council
Established by the European Commission

Big Data - Examples

Image: Wikipedia

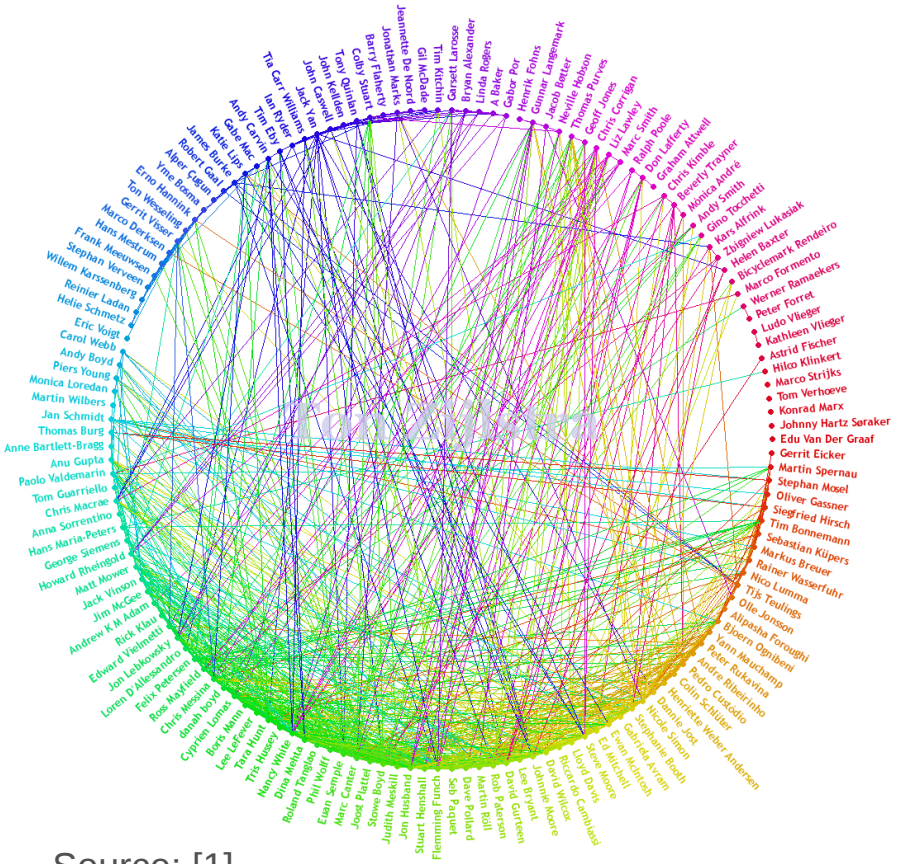
IceCube

- World‘s biggest neutrino observatory
- Understanding of basic concepts in astrophysics
- Supernovas
- Dark matter
- Black holes

Data size

- approx. 10 TeraByte per data set

Very Large Networks



Source: [1]

Social Networks

- Reflect social structures in detail
- Example question: Can we distinguish democratic countries from totalitarian ones by looking at their Facebook structure?

Data size

- GigaByte upto TeraByte (only the graph)
- Data exchange (movies, pictures, etc.) in the Peta-Byte range

Big Data - Examples

Image: Danard Vincente

The World Wide Web

- Giant source of information
- Very fast access through search engines
- Automatic scoring of the quality of web pages

Data size

- 500 ExaByte in 2009
  (Richard Wray, The Guardian)

Big Data Analytics

Image: Roger Smith

Primary Requirements

- Sequential processing
- Sublinear memory requirement
- Sublinear, linear or near linear running time

Secondary Requirements

- Energy efficiency
- Easy to distribute/parallelize
- Limited communication

Outline

Part I: Unstructured Data

- Clustering
- Definition of coresets
- Streaming & distributed algorithms via coresets
- Coreset constructions

Part II: Semi-Structured Data

- Property Testing
- An introductary example
- Local vs. global structure of planar graphs

Cluster analysis

Clustering

- Partition a set of objects into groups such that (ideally)

  (a) Objects in the same group are similar to each other
  (b) Objects in different groups are dissimilar to eachother

Representation of objects

- Objects are represented by real valued vectors / sets of points

k-Means Clustering

Definition of k-Means Clustering

- Input:Set P of n d-dimensional points
- Output: Set C of k points (centers) that minimize

$$cost(P,C) = \sum_{p \in P} \min_{c \in C} ||p-c||^2$$

- The corresponding clustering can be obtained by assigning each point to its nearest center

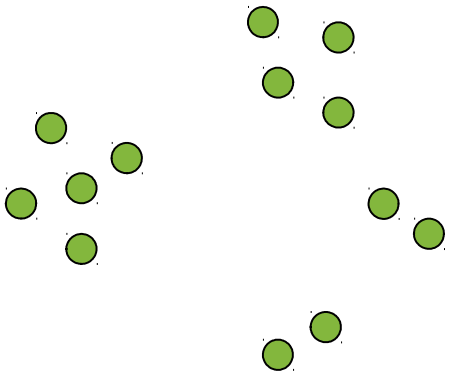
k-Means Clustering

Definition of k-Means Clustering

- Input:Set P of n d-dimensional points
- Output: Set C of k points (centers) that minimize

$$\text{cost}(P,C) = \sum_{p \in P} \min_{c \in C} ||p-c||^2$$

- The corresponding clustering can be obtained by assigning each point to its nearest center



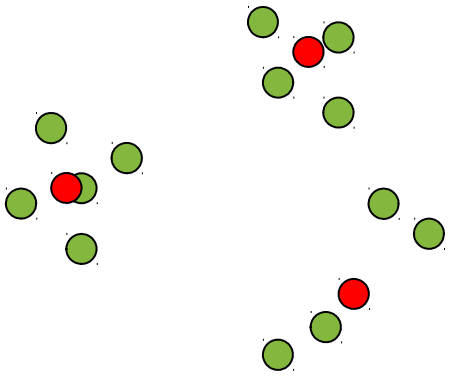
k-Means Clustering

Definition of k-Means Clustering

- Input:Set P of n d-dimensional points
- Output: Set C of k points (centers) that minimize

$$\text{cost}(P,C) = \sum_{p \in P} \min_{c \in C} ||p-c||^2$$

- The corresponding clustering can be obtained by assigning each point to its nearest center



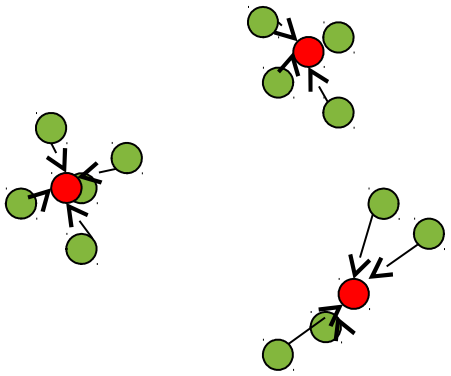
k-Means Clustering

Definition of k-Means Clustering

- Input:Set P of n d-dimensional points
- Output: Set C of k points (centers) that minimize

$$\text{cost}(P,C) = \sum_{p \in P} \min_{c \in C} ||p-c||^2$$

- The corresponding clustering can be obtained by assigning each point to its nearest center



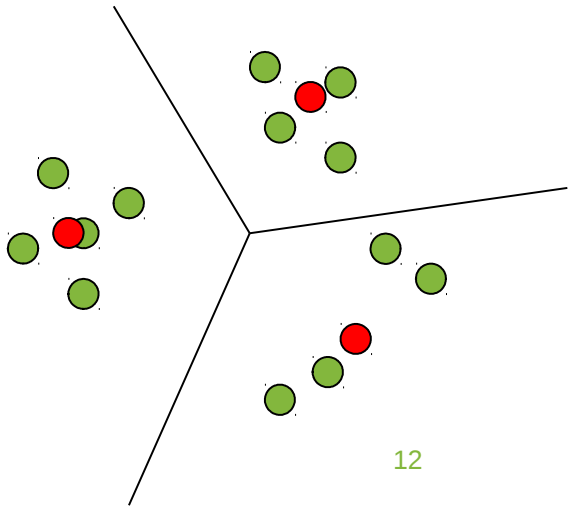
k-Means Clustering

Definition of k-Means Clustering

- Input:Set P of n d-dimensional points
- Output: Set C of k points (centers) that minimize

$$\text{cost}(P,C) = \sum_{p \in P} \min_{c \in C} ||p-c||^2$$

- The corresponding clustering can be obtained by assigning each point to its nearest center



Coresets – the general idea

Coreset

- A coreset is an approximation of the data with respect to an optimization problem such that a „good“ solution on the coreset is almost as good as a „good“ solution on the original point set

- There is no common definition
- Many approaches can be viewed as coreset approaches (sampling, etc.)

Coresets – a definition

Coresets for k-Means Clustering [Har-Peled, Mazumdar, 2004]

A small weighted set S is called ***(k,ε)-coreset*** for P, if for every set C of k centers we have

$$(1-\varepsilon)\,\mathrm{cost}(P,C) \le \mathrm{cost}(S,C) \le (1+\varepsilon)\,\mathrm{cost}(P,C)$$

In the above definition, the weight of a point is interpreted as its multiplicity.

Coresets – a definition

Coresets for k-Means Clustering [Har-Peled, Mazumdar, 2004]

A small weighted set S is called ***(k,ε)-coreset*** for P, if for every set C of k centers we have

$$(1-\varepsilon)\,\mathrm{cost}(P,C) \le \mathrm{cost}(S,C) \le (1+\varepsilon)\,\mathrm{cost}(P,C)$$

In the above definition, the weight of a point is interpreted as its multiplicity.

Coresets – a definition

Coresets for k-Means Clustering [Har-Peled, Mazumdar, 2004]

A small weighted set S is called ***(k,ε)-coreset*** for P, if for every set C of k centers we have

$$(1-\varepsilon)\,\mathrm{cost}(P,C) \le \mathrm{cost}(S,C) \le (1+\varepsilon)\,\mathrm{cost}(P,C)$$

In the above definition, the weight of a point is interpreted as its multiplicity.

Properties of coresets

- Fast answering of cost queries
- Optimal solution on the coreset yields a $(1+O(\varepsilon))$–approximation
- α-approximation on the coreset yields an $\alpha\cdot(1+O(\varepsilon))$–approximation

First Technique: Bounded Movement of Points

Lemma

- Let Opt be the cost of an optimal k-means solution
- If we move a set of points P by an overall l_2^2-distance of $O(\varepsilon^2 \text{ Opt})$ to obtain Q then we have for every set of k centers C:

$$(1-\varepsilon)\ \text{cost}(P,C) \leq \text{cost}(Q,C) \leq (1+\varepsilon)\ \text{cost}(P,C)$$

Main idea [Har-Peled, Mazumdar, 2004]

- Move points such that many are in the same position
- Replace multiple points by weighted points

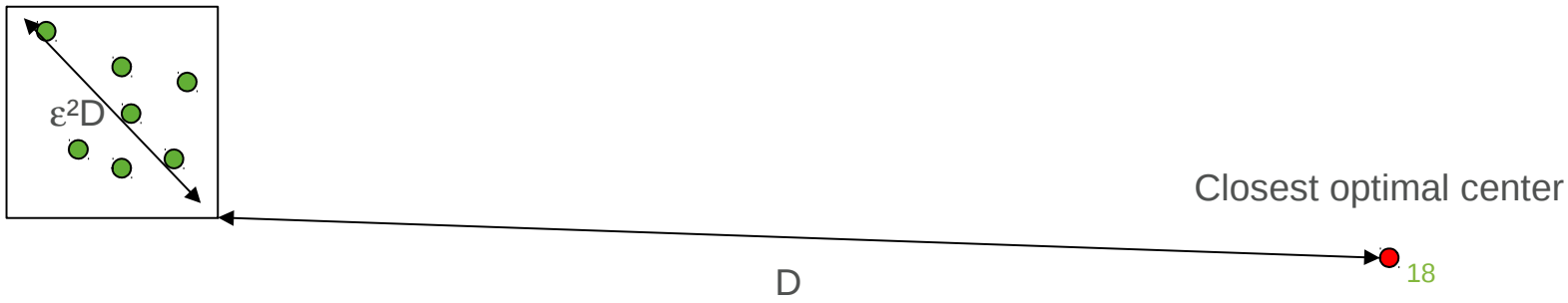
First Technique: Bounded Movement of Points

Theorem [Har-Peled, Mazumdar, 2004]

- There is a coreset of size O(k log n/εd) for the k-means problem.

Proof idea:

- Compute a partition of the space into squares s.t. the diagonal of a square is at most ε^2 times its l_2^2-distance to the nearest center of an optimal solution
- Move all points to the center of the square
- Since every point in a cell with l_2^2-distance D contributes at least D to the optimal solution, we have that the overall movement is ε^2 Opt => Coreset!



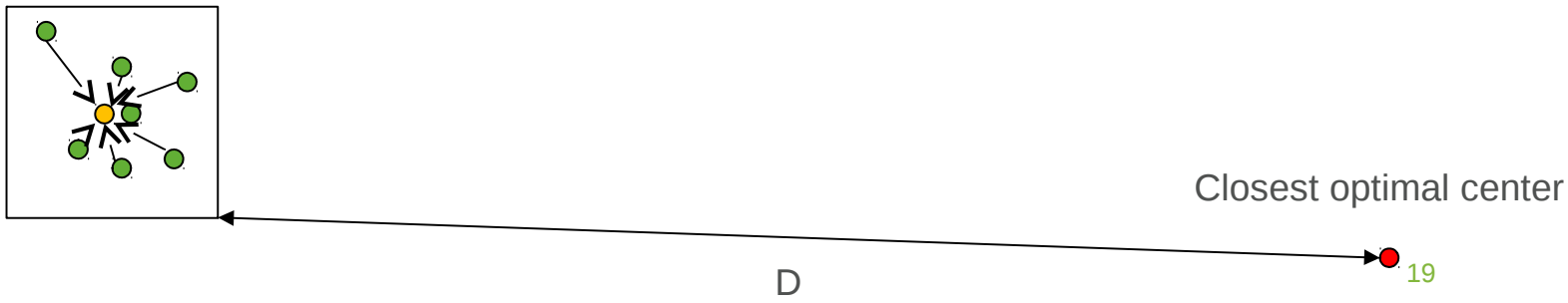
First Technique: Bounded Movement of Points

Theorem [Har-Peled, Mazumdar, 2004]

- There is a coreset of size O(k log n/εd) for the k-means problem.

Proof idea:

- Compute a partition of the space into squares s.t. the diagonal of a square is at most ε^2 times its l_2^2-distance to the nearest center of an optimal solution
- Move all points to the center of the square
- Since every point in a cell with l_2^2-distance D contributes at least D to the optimal solution, we have that the overall movement is ε^2 Opt => Coreset!



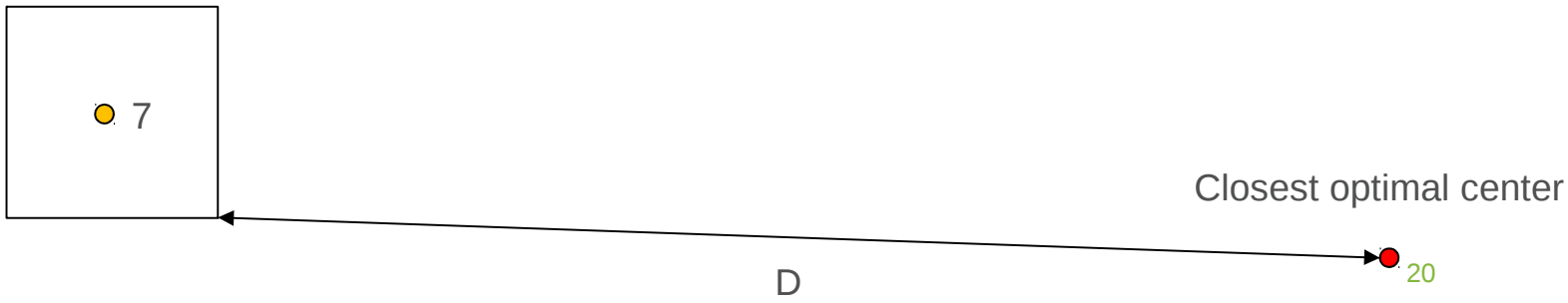
First Technique: Bounded Movement of Points

Theorem [Har-Peled, Mazumdar, 2004]

- There is a coreset of size O(k log n/εd) for the k-means problem.

Proof idea:

- Compute a partition of the space into squares s.t. the diagonal of a square is at most ε^2 times its l_2^2-distance to the nearest center of an optimal solution
- Move all points to the center of the square
- Since every point in a cell with l_2^2-distance D contributes at least D to the optimal solution, we have that the overall movement is ε^2 Opt => Coreset!

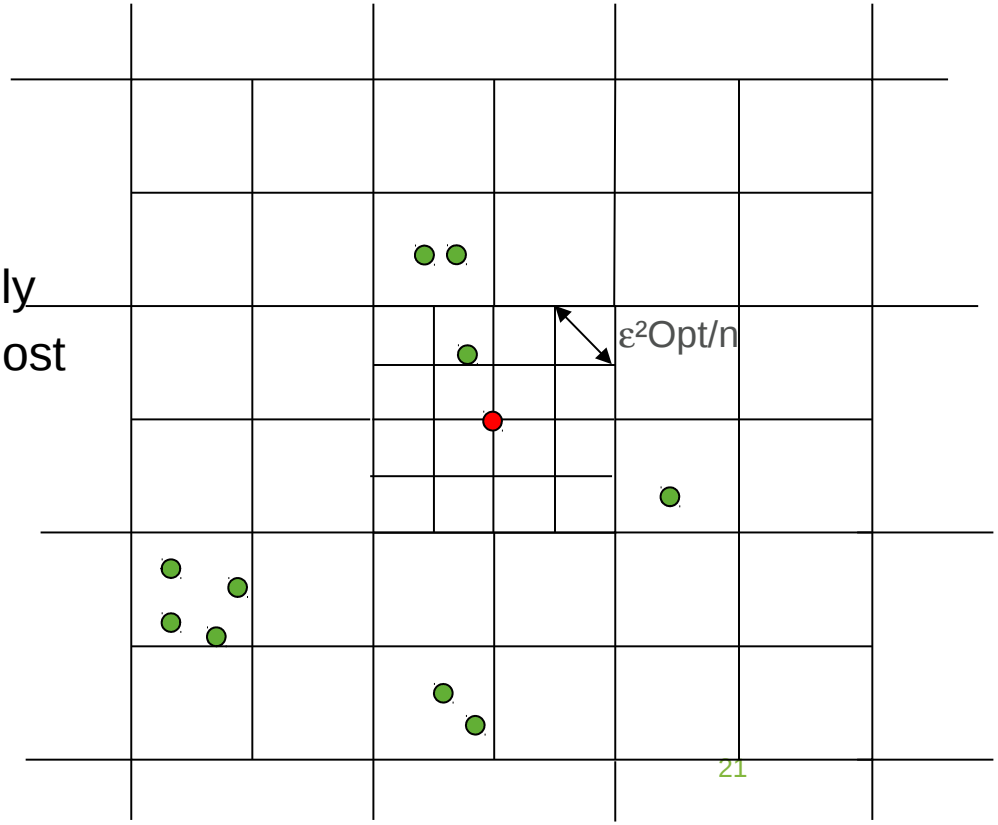


First Technique: Bounded Movement of Points

Theorem [Har-Peled, Mazumdar, 2004]

- There is a coreset of size O(k log n/εd) for the k-means problem.

- Smallest cells have diagonal length $\varepsilon^2 Opt/n$
- Cells size increases exponentially
- Furthest point has distance at most Opt
- => O(k log n/εd) cells



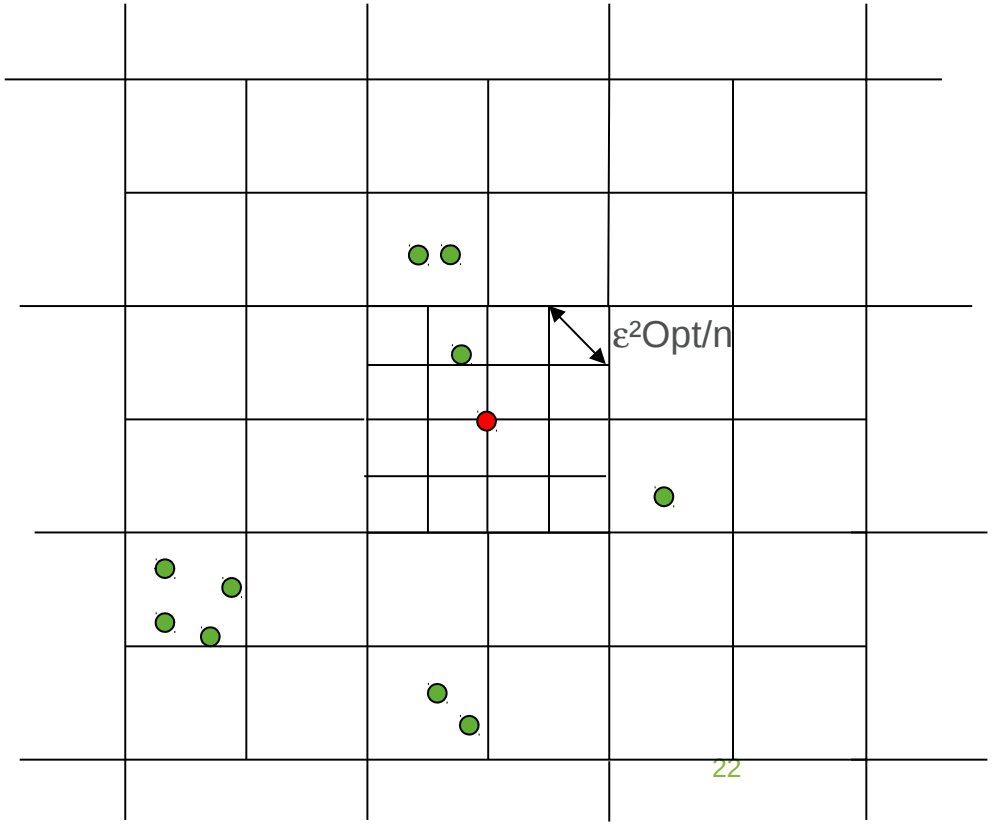
First Technique: Bounded Movement of Points

Theorem [Har-Peled, Mazumdar, 2004]

- There is a coreset of size O(k log n/εd) for the k-means problem.

Final observation:

- The construction works, if we replace the optimal solution by an O(1)-approximation



First Technique: Bounded Movement of Points

A different construction [Frahling, S., 2005]

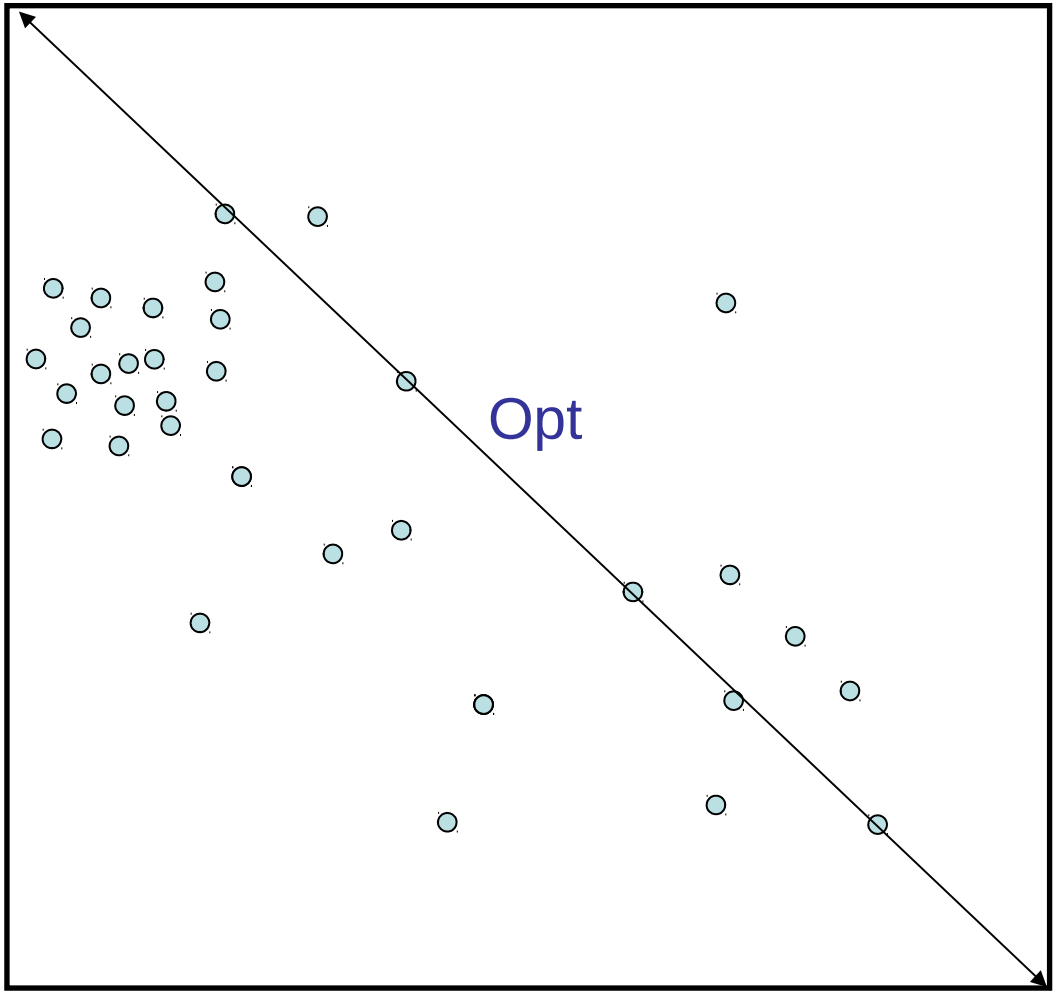
- Idea: Distribute the „movement cost“ equally among all cells
- Movement cost = l_2^2 cell diameter · #points in cell

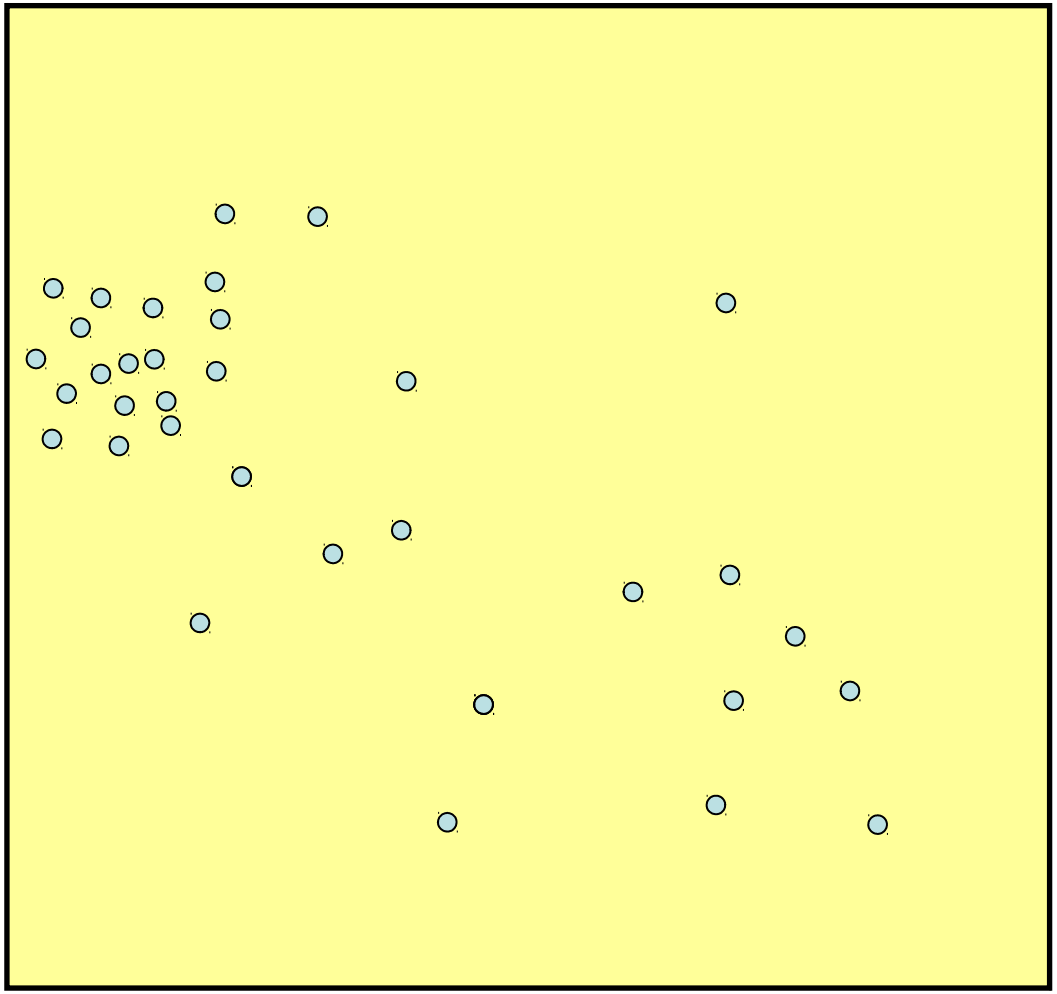
Algorithm

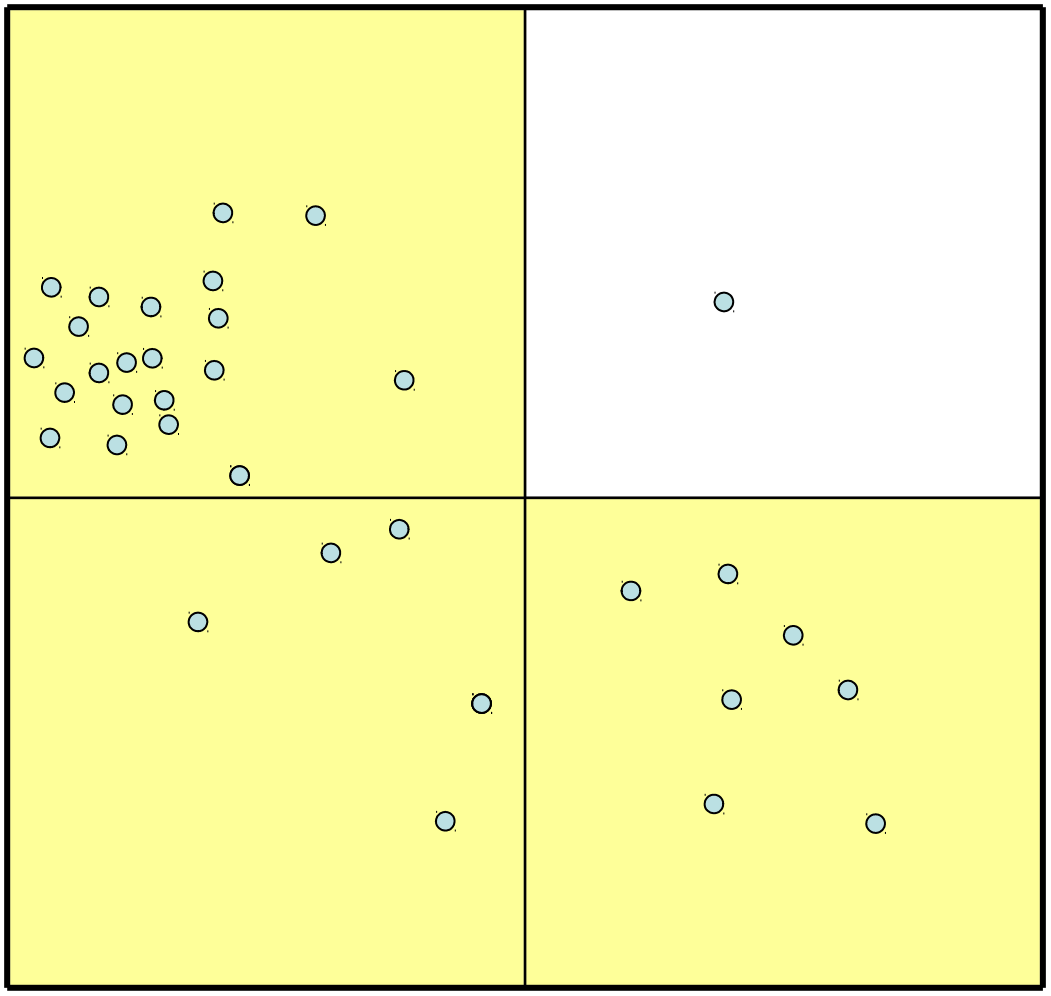
- Guess the cost of an optimal solution, fix appropriate δ
- Call a cell δ-heavy, if l_2^2 cell diameter · #points in cell > δ Opt
- Put a coreset point in each minimal δ-heavy cell
- Move each point to a coreset point in the smallest heavy cell it is contained in

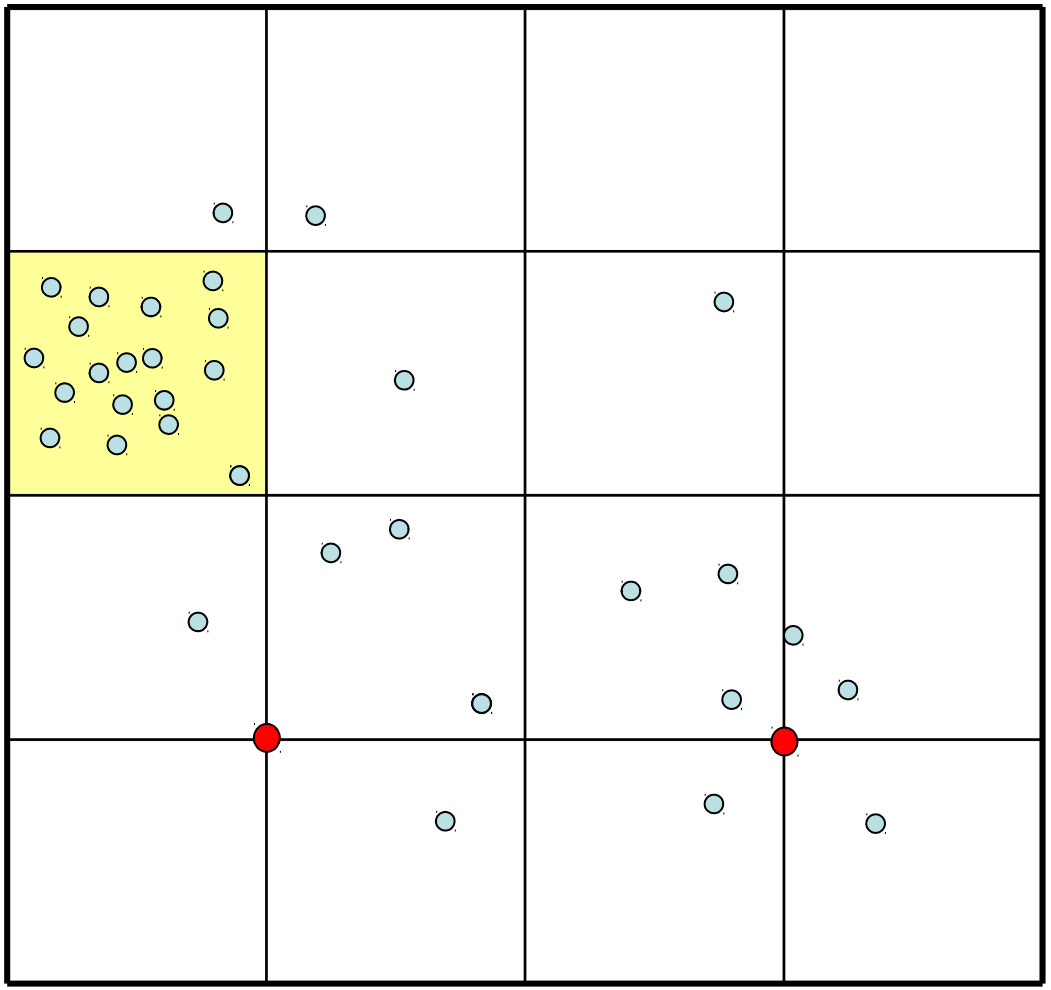
Advantage

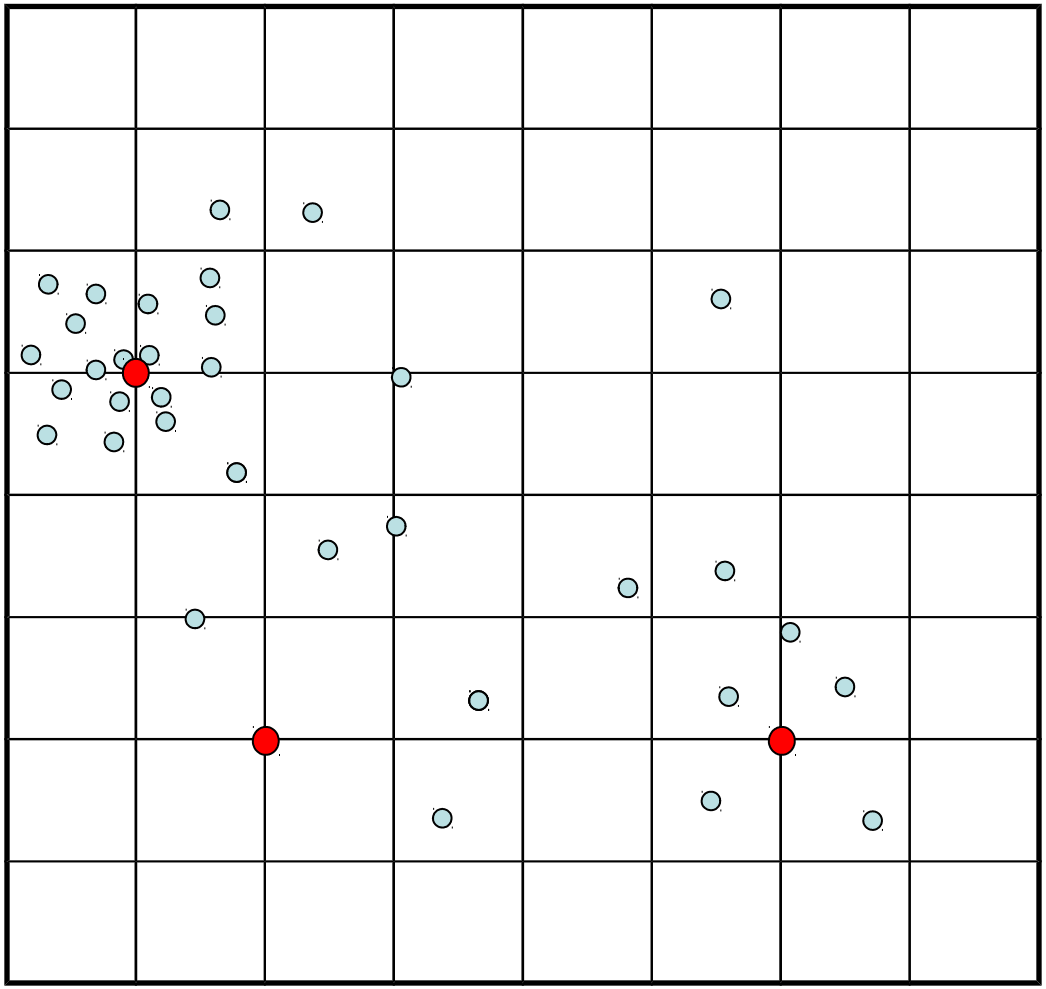
- We do not need to compute an O(1) approximation
- Gives first algorithm for insertion/deletion streams

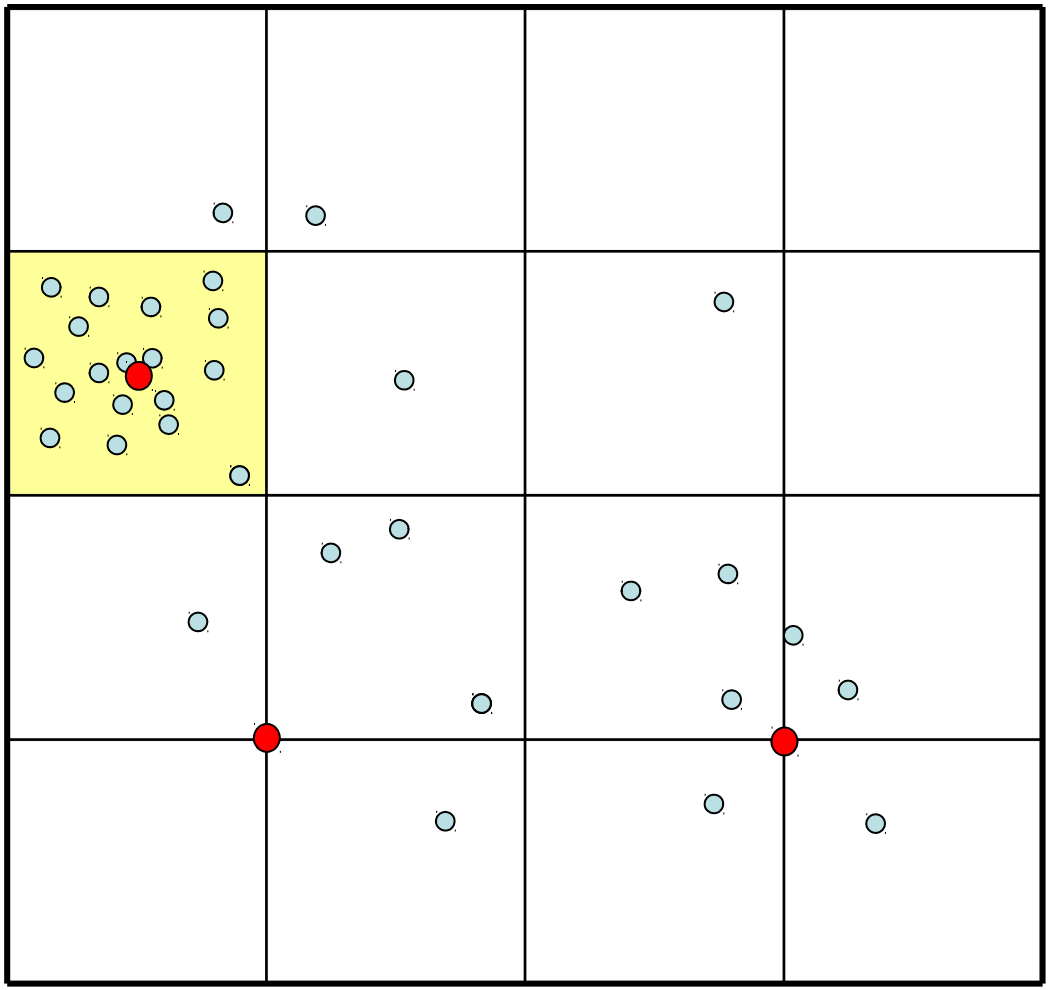

Opt

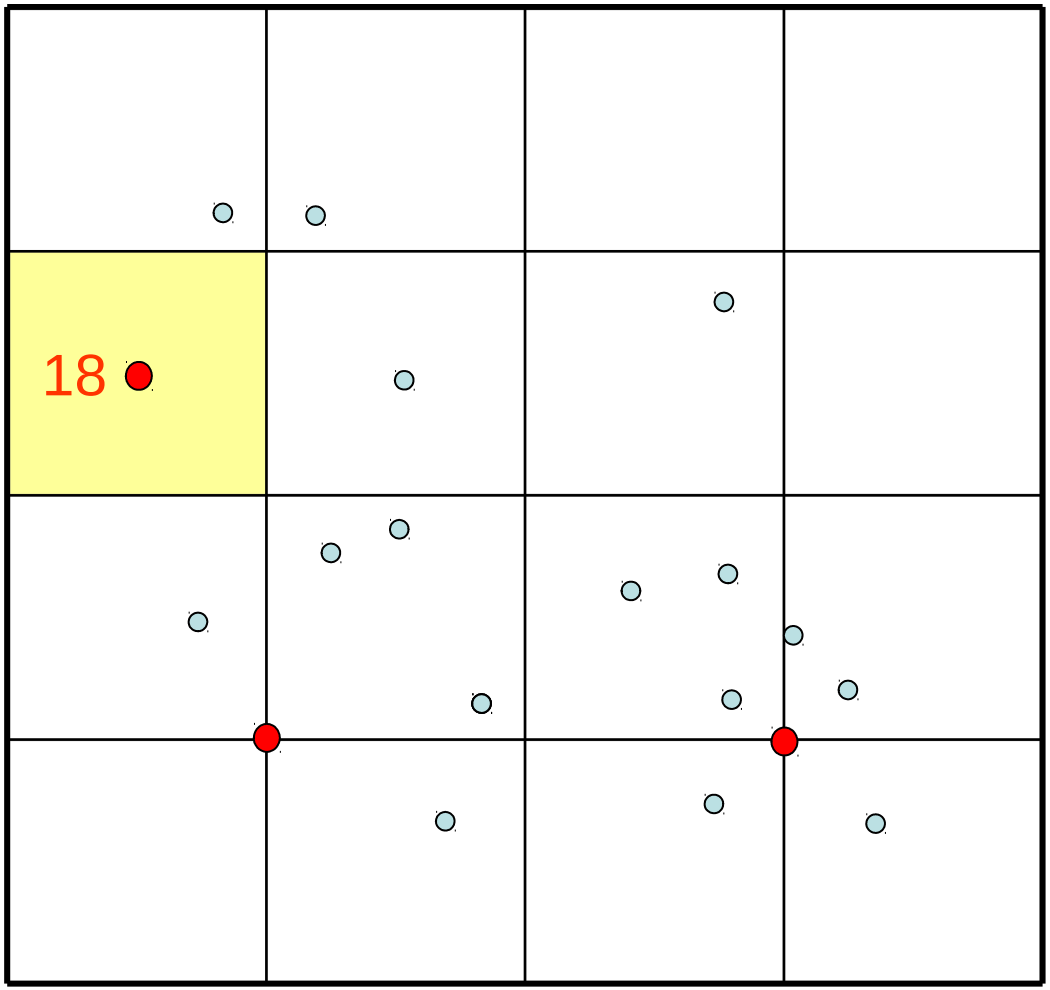


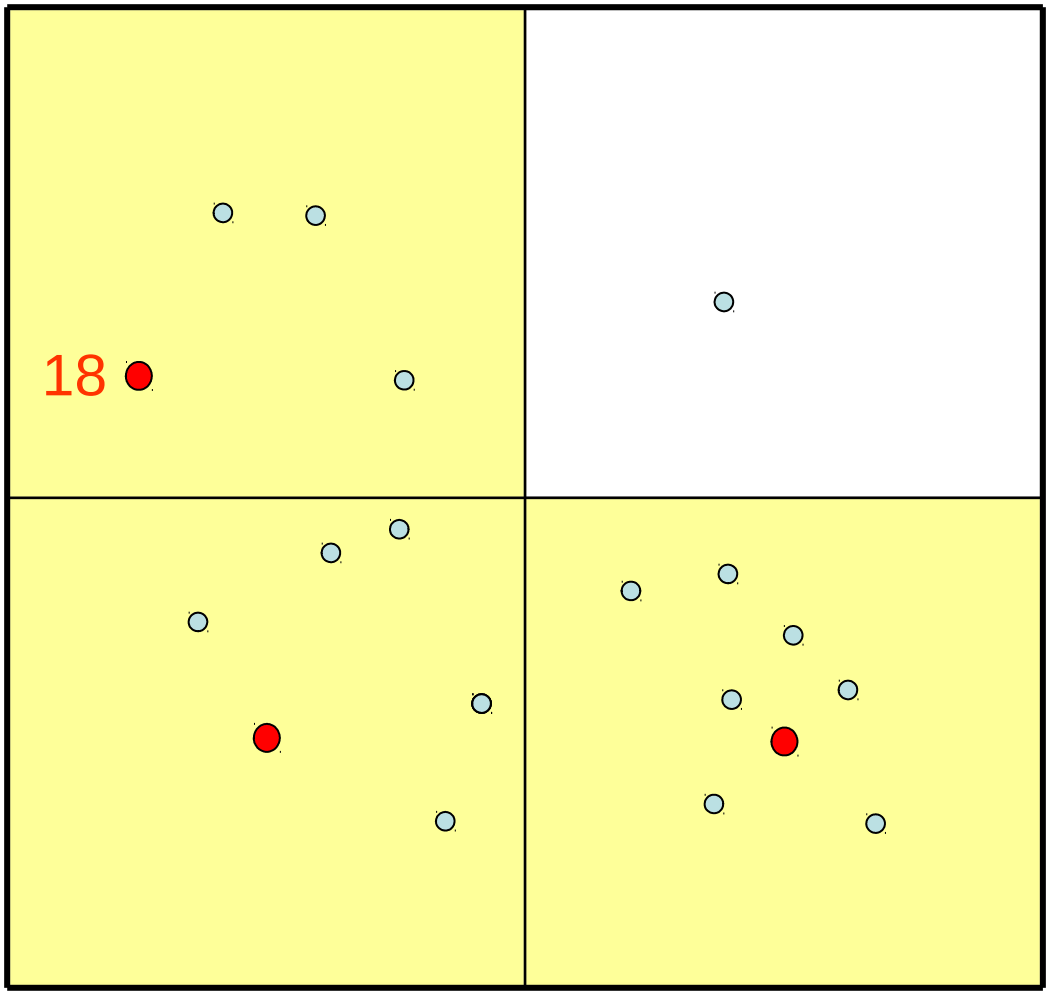


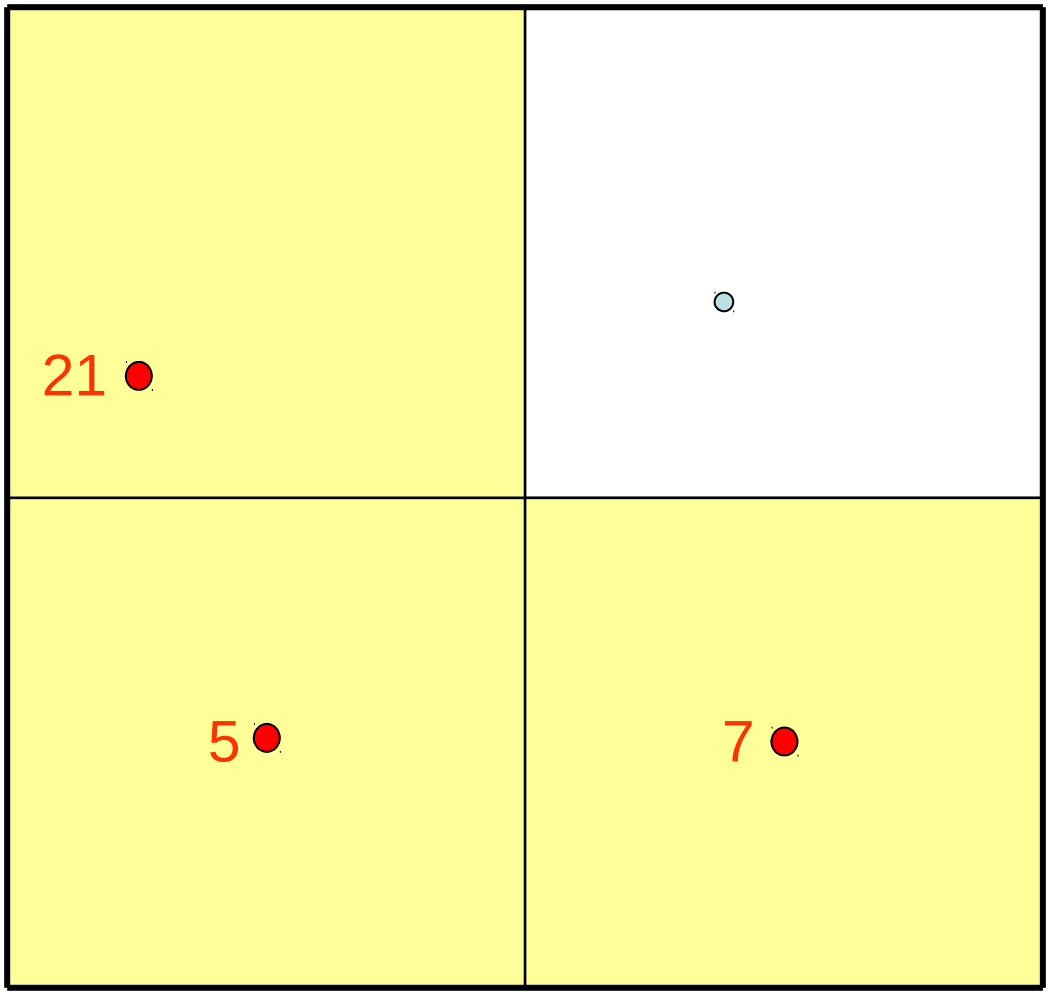


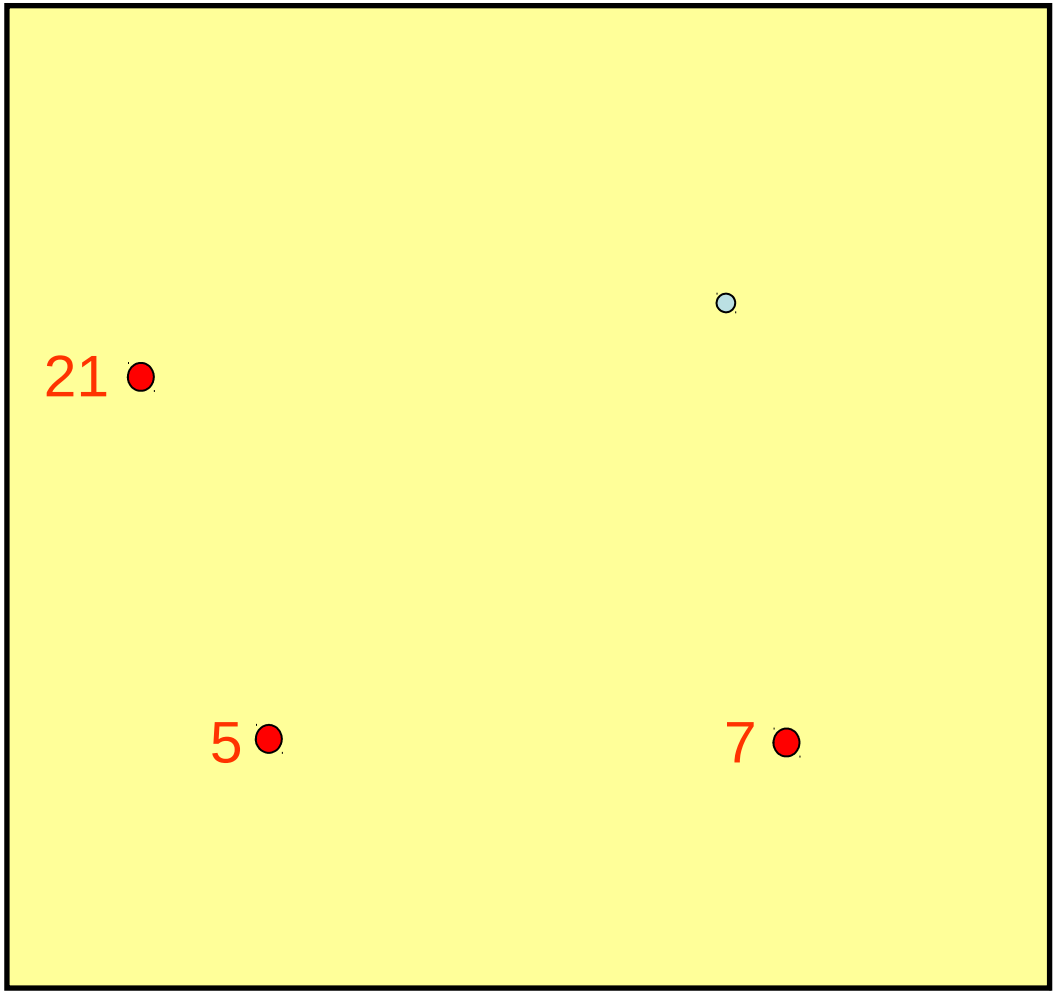


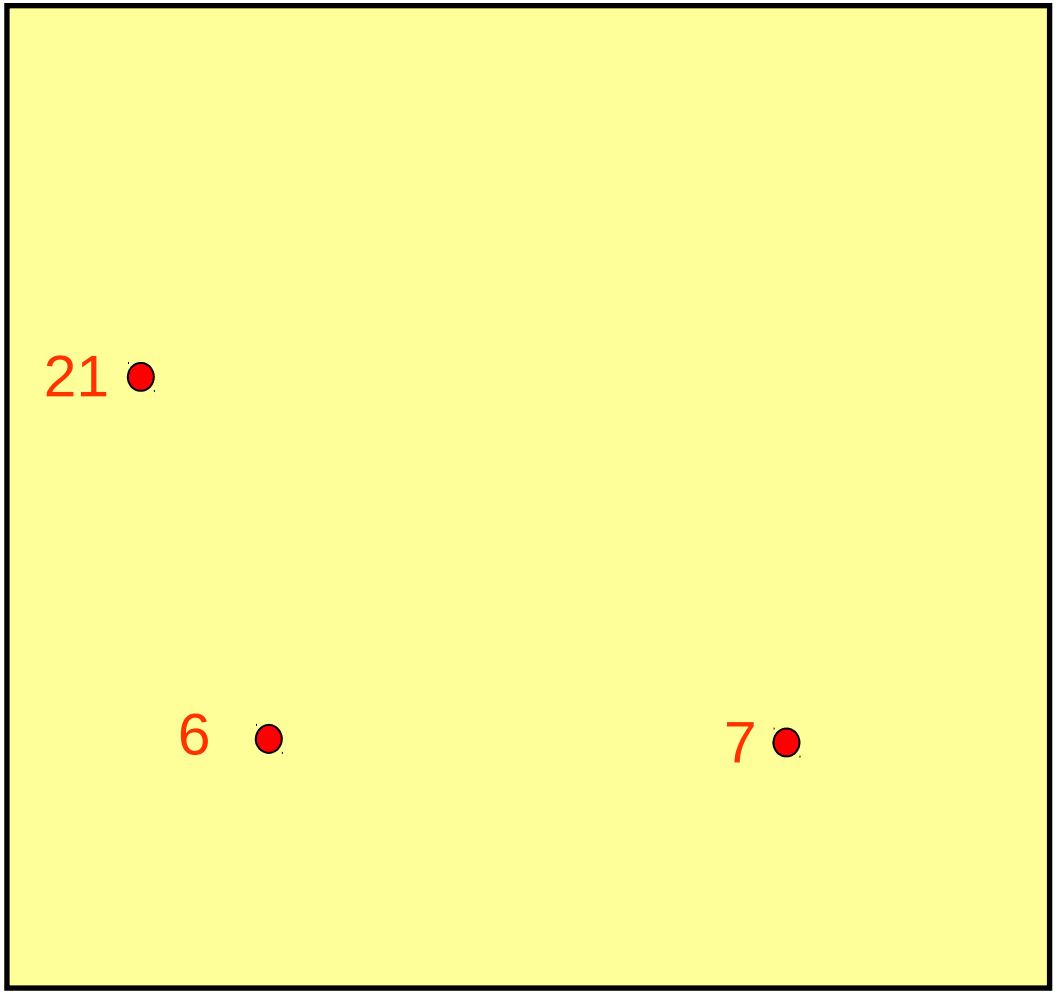


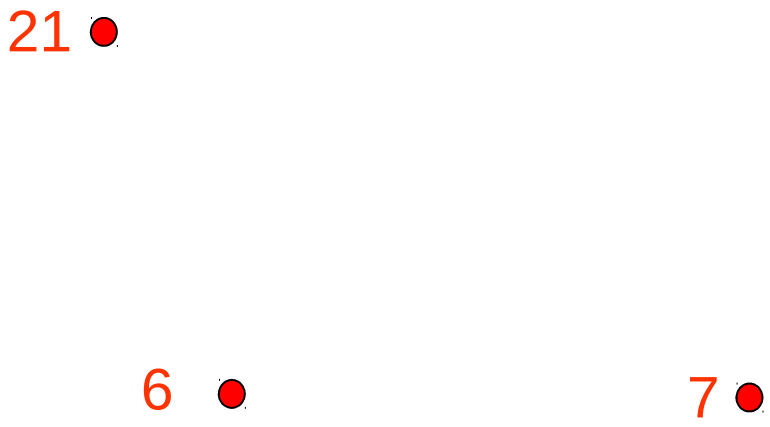

18


18


21
5
7


21
5
7


21
6
7


21
6
7

First Technique: Bounded Movement of Points

Theorem [Frahling, S., 2005]

- The construction just presented gives a coreset for the k-means problem of size $O(k \log n/\varepsilon d)$.

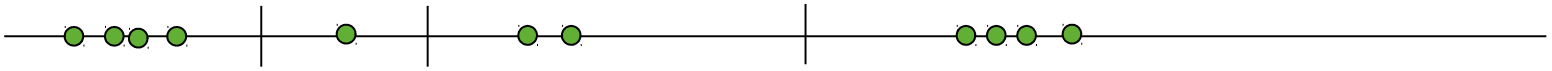
First Technique: Bounded Movement of Points

Theorem [Har-Peled, Kushal, 2005]

- There is a coreset of size $O(k^3/\varepsilon d+1)$ for the k-means problem.

1D Coreset:

- Partition the points into $O(k^2/\varepsilon^2)$ intervals such that for each interval the 1-means cost is bounded by $O(\varepsilon^2 \, Opt/k^2)$



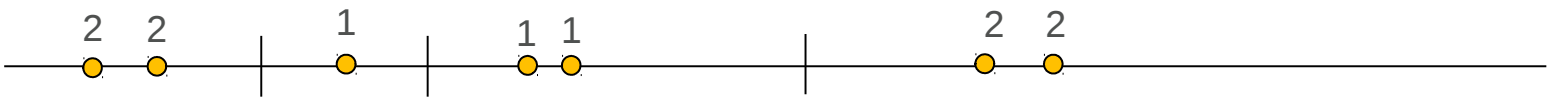
First Technique: Bounded Movement of Points

Theorem [Har-Peled, Kushal, 2005]

- There is a coreset of size $O(k^3/\varepsilon d+1)$ for the k-means problem.

1D Coreset:

- Partition the points into $O(k^2/\varepsilon^2)$ intervals such that for each interval the 1-means cost is bounded by $O(\varepsilon^2\ Opt/k^2)$
- Replace all points Q in each interval by at most two points S, s.t.



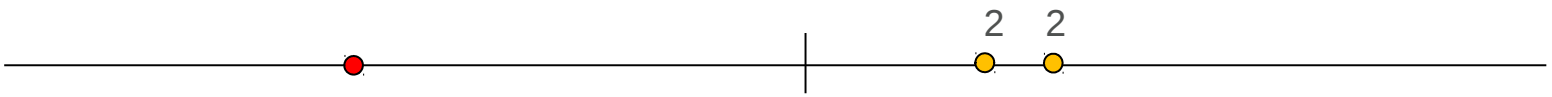
First Technique: Bounded Movement of Points

Theorem [Har-Peled, Kushal, 2005]

- There is a coreset of size $O(k^3/\varepsilon d+1)$ for the k-means problem.

1D Coreset:

- Partition the points into $O(k^2/\varepsilon^2)$ intervals such that for each interval the 1-means cost is bounded by $O(\varepsilon^2 \, Opt/k^2)$
- Replace all points Q in each interval by at most two points S, s.t. if the nearest center is not contained in the interval, then
  $cost(Q,c) = cost(S,c)$



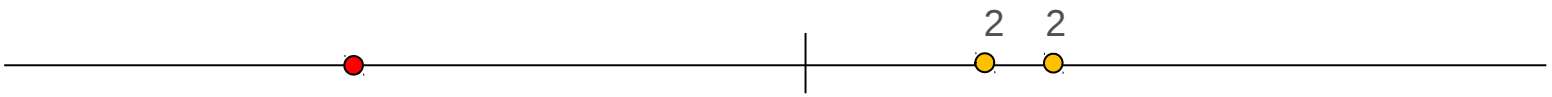
First Technique: Bounded Movement of Points

Theorem [Har-Peled, Kushal, 2005]

- There is a coreset of size $O(k^3/\varepsilon d+1)$ for the k-means problem.

1D Coreset:

- Partition the points into $O(k^2/\varepsilon^2)$ intervals such that for each interval the 1-means cost is bounded by $O(\varepsilon^2 \, Opt/k^2)$
- Replace all points Q in each interval by at most two points S, s.t. if the nearest center is not contained in the interval, then
  $$cost(Q,c) = cost(S,c)$$



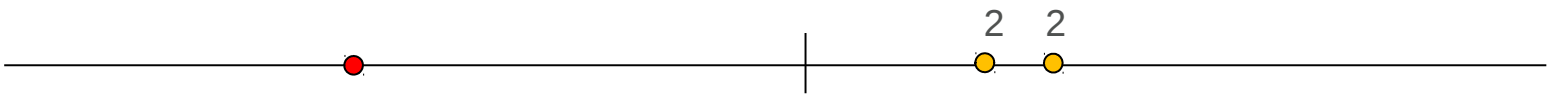
First Technique: Bounded Movement of Points

Theorem [Har-Peled, Kushal, 2005]

- There is a coreset of size $O(k^3/\varepsilon d+1)$ for the k-means problem.

1D Coreset:

- Partition the points into $O(k^2/\varepsilon^2)$ intervals such that for each interval the 1-means cost is bounded by $O(\varepsilon^2 \, Opt/k^2)$
- Replace all points Q in each interval by at most two points S, s.t. if the nearest center is not contained in the interval, then
  $$cost(Q,c) = cost(S,c)$$
- Error only within the intervals that contain a center



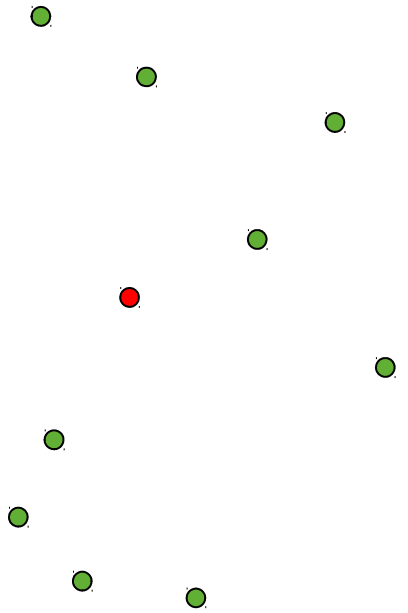
First Technique: Bounded Movement of Points

Theorem [Har-Peled, Kushal, 2005]

- There is a coreset of size $O(k^3/\varepsilon d+1)$ for the k-means problem.

Coresets for higher dimensions:

- Move points to $O(1/\varepsilon d-1)$ lines through each center of O(1)-approximation (overall l_2^2-movement is ε^2 Opt)
- Compute a coreset for each line



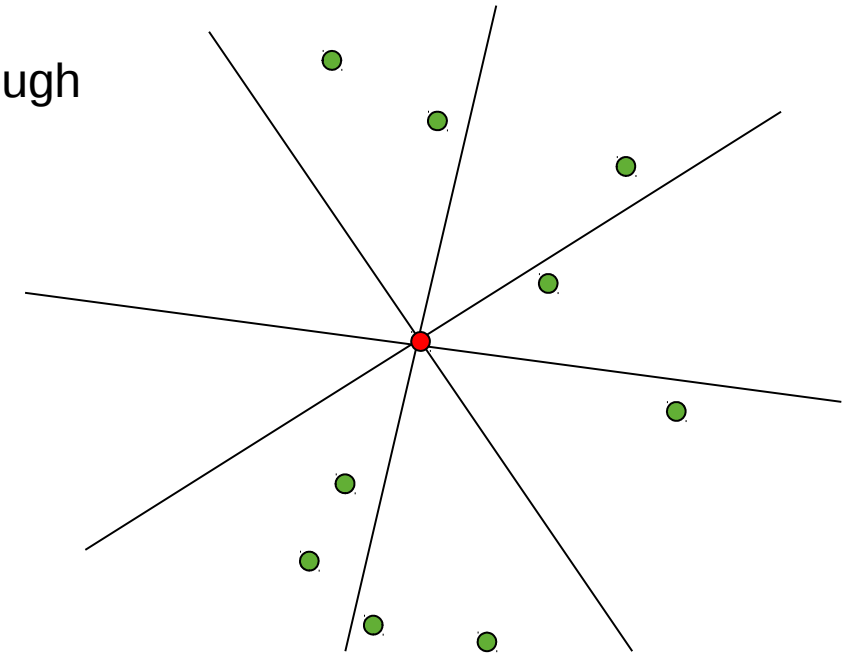
First Technique: Bounded Movement of Points

Theorem [Har-Peled, Kushal, 2005]

- There is a coreset of size $O(k^3/\varepsilon d+1)$ for the k-means problem.

Coresets for higher dimensions:

- Move points to $O(1/\varepsilon d-1)$ lines through each center of O(1)-approximation (overall l_2^2-movement is ε^2 Opt)
- Compute a coreset for each line



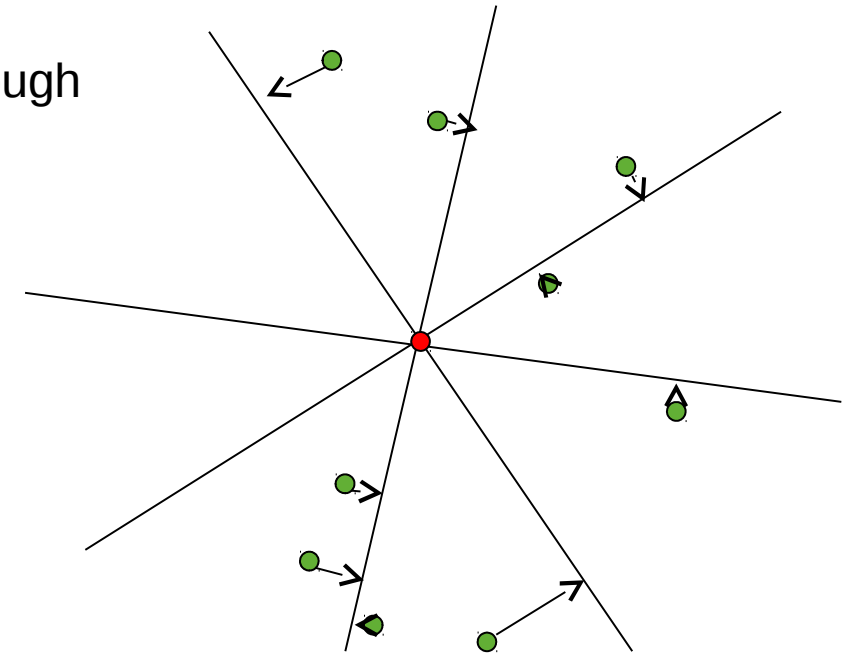
First Technique: Bounded Movement of Points

Theorem [Har-Peled, Kushal, 2005]

- There is a coreset of size $O(k^3/\varepsilon d+1)$ for the k-means problem.

Coresets for higher dimensions:

- Move points to $O(1/\varepsilon d-1)$ lines through each center of O(1)-approximation (overall l_2^2-movement is ε^2 Opt)
- Compute a coreset for each line using the 1D construction



Second Technique: Sampling

Lemma

Let $0<\varepsilon<1$. Let P be a point set in the unit ball. Let C be a fixed set of k centers inside the unit ball. Let S be a sample set taken from P uniformly at random. Then

$$\Pr[\ |\text{cost}(P,C) - |P|/|S|\ \text{cost}(S,C)| > \varepsilon\, n] \ < 2\exp(-\ \varepsilon^2\, |S| / 12\ )$$

Proof:

- Apply Hoeffding‘s bound

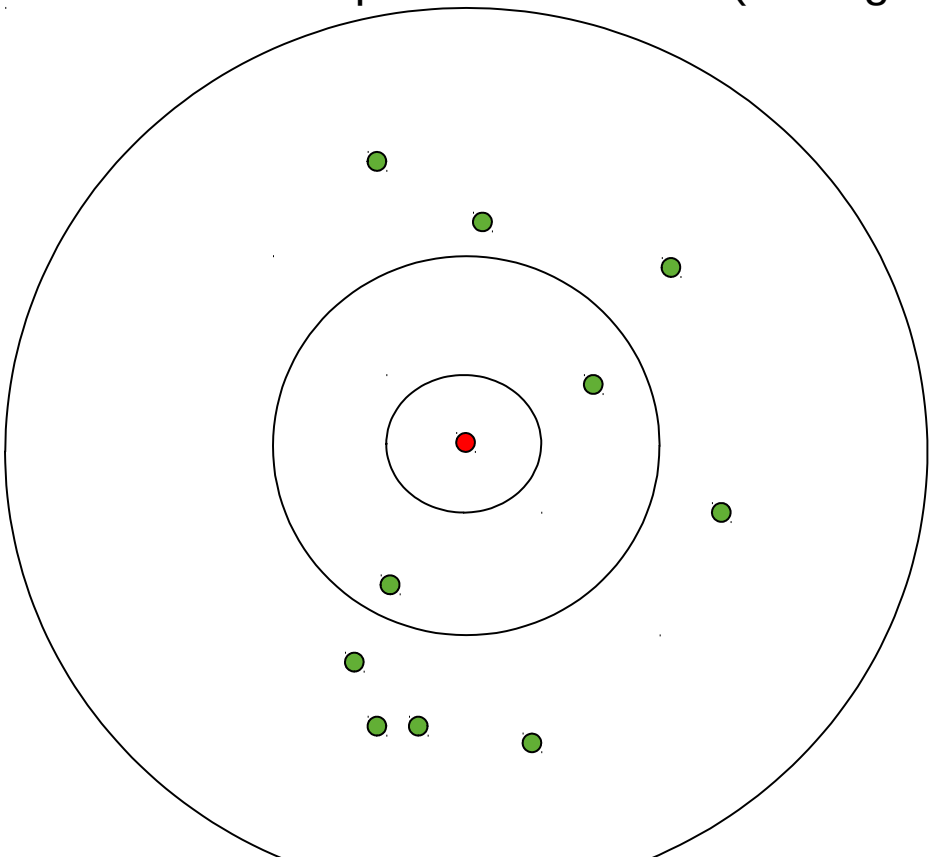
Second Technique: Sampling

Theorem [Chen, 2009]

Let $0<\varepsilon<1$. There is an (ε,k)-coreset for the k-means problem of size $\tilde{O}(dk^2 \log n/\varepsilon^2)$.

Algorithm:

- Partition space around O(1)-approximation into exponentially increasing rings
- From each ring take a uniform sample



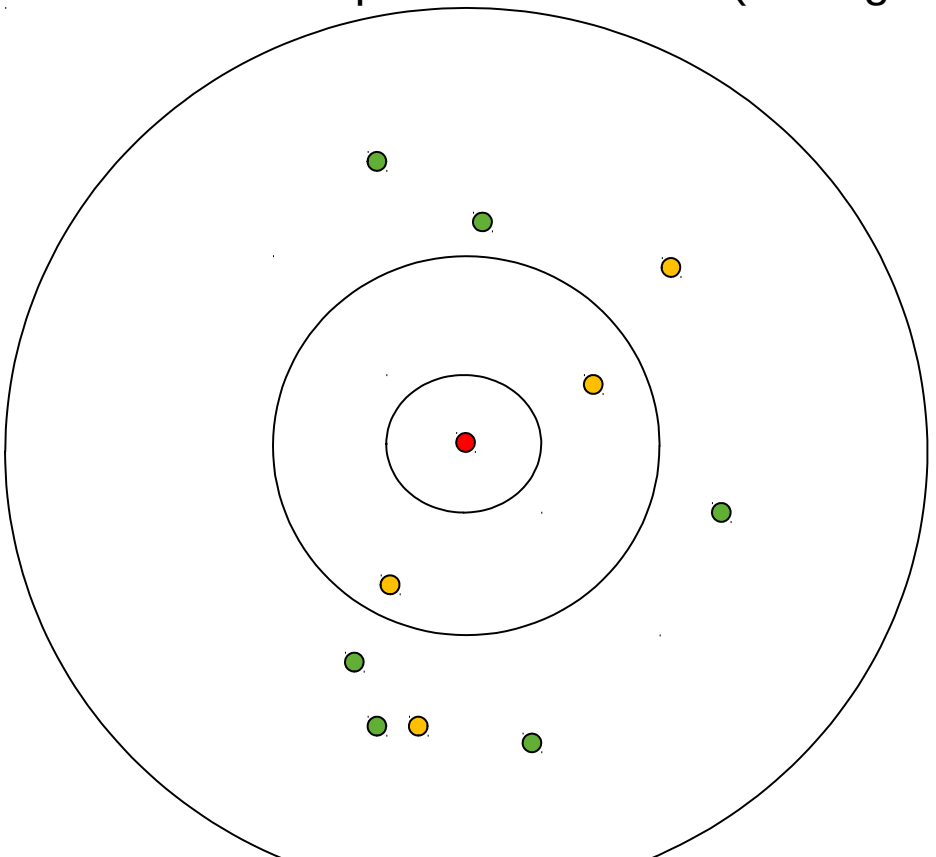
Second Technique: Sampling

Theorem [Chen, 2009]

Let $0<\varepsilon<1$. There is an (ε,k)-coreset for the k-means problem of size $\tilde{O}(dk^2 \log n/\varepsilon^2)$.

Algorithm:

- Partition space around O(1)-approximation into exponentially increasing rings
- From each ring take a uniform sample
- Weight the sample appropriately



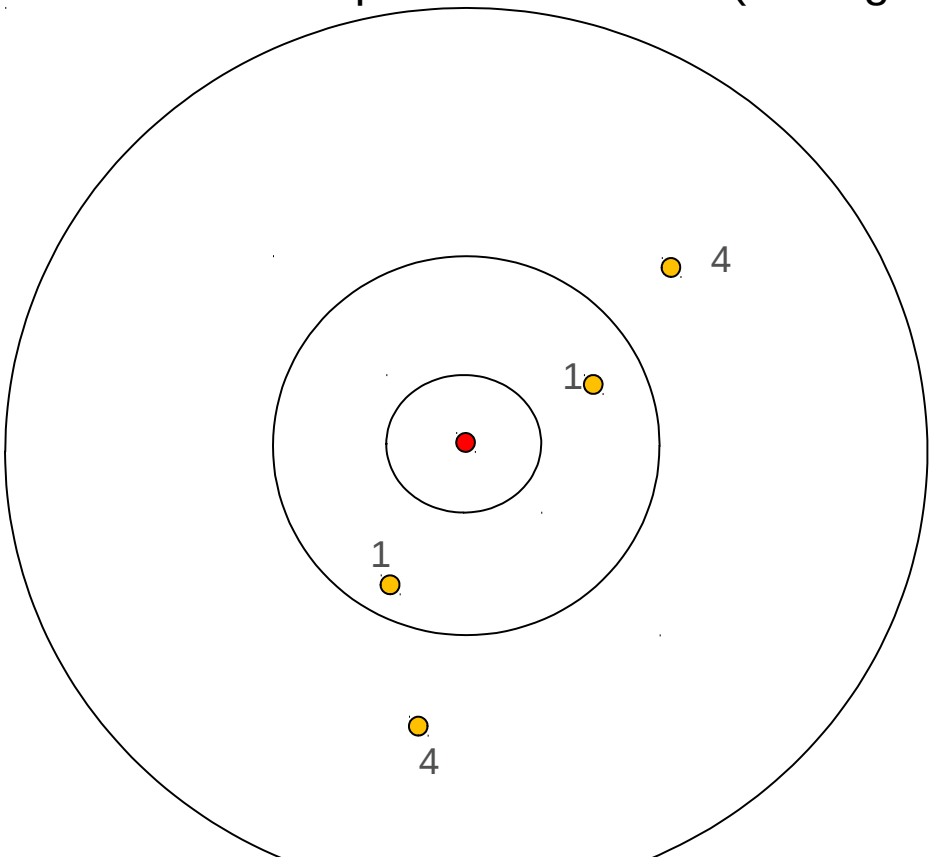
Second Technique: Sampling

Theorem [Chen, 2009]

Let $0<\varepsilon<1$. There is an (ε,k)-coreset for the k-means problem of size $\tilde{O}(dk^2 \log n/\varepsilon^2)$.

Algorithm:

- Partition space around O(1)-approximation into exponentially increasing rings
- From each ring take a uniform sample
- Weight the sample appropriately



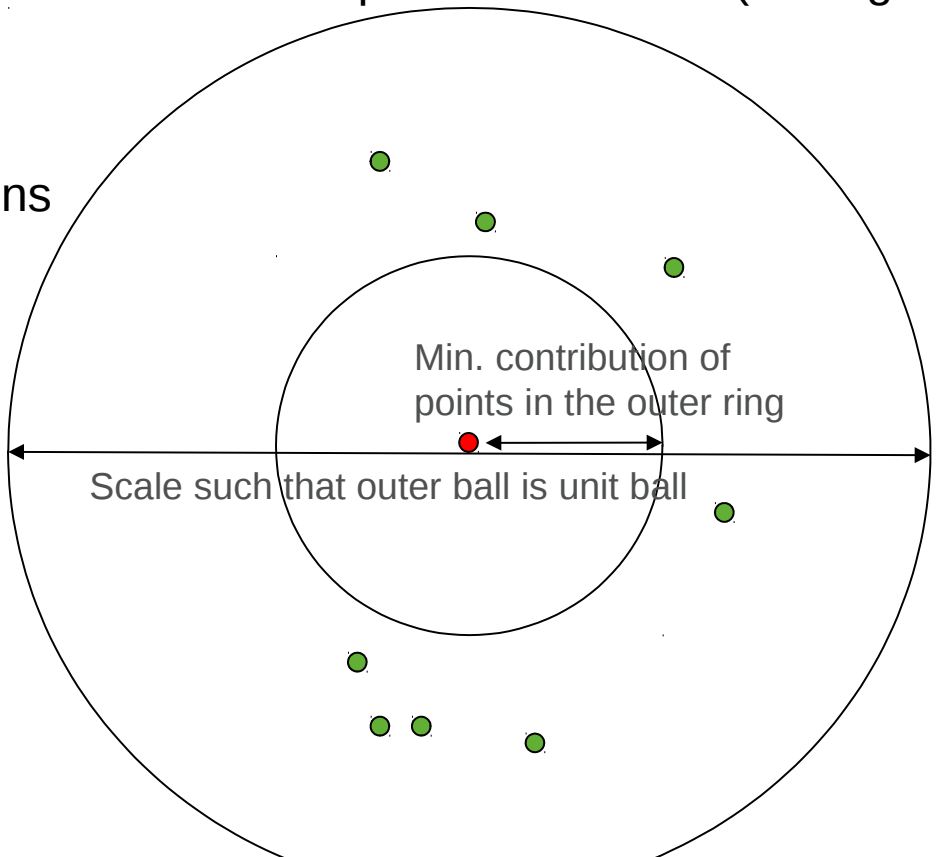
Second Technique: Sampling

Theorem [Chen, 2009]

Let $0<\varepsilon<1$. There is an (ε,k)-coreset for the k-means problem of size $\tilde{O}(dk^2 \log n/\varepsilon^2)$.

Proof:

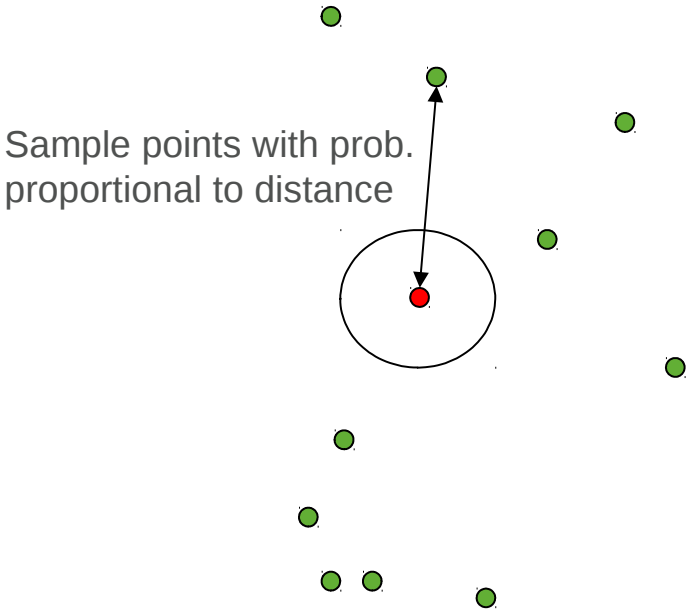
- Define a net of candidate solutions and apply sampling lemma to all solutions from the net
- Charge sampling error to the contribution of the points inside the ring



Technique 2.5: Importance Sampling

Idea [Feldman, Monemizadeh, S., 2005]

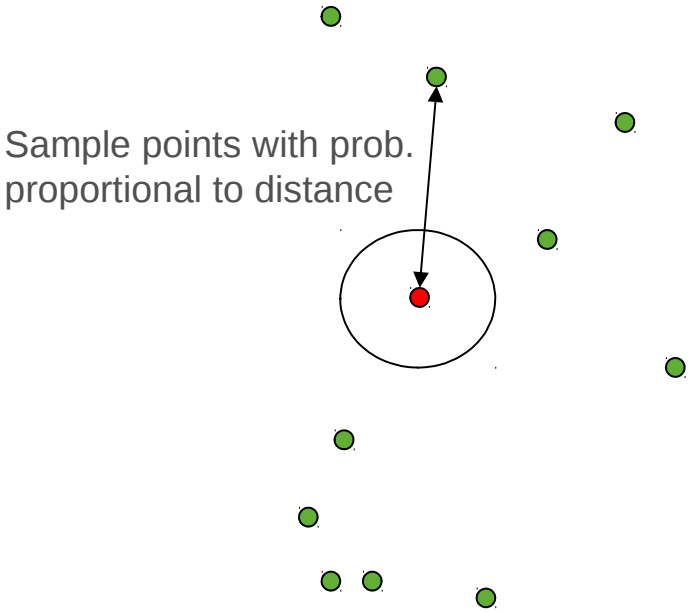
- Sample uniformly the inner ring
- Sample all remaining points according to Pr[p] = cost({p},COpt) / Opt and assign weight 1/Pr[p].



Technique 2.5: Importance Sampling

Idea [Feldman, Monemizadeh, S., 2005]

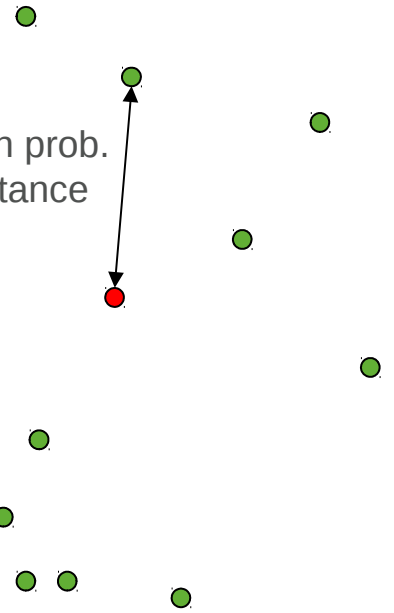
- Sample uniformly the inner ring
- Sample all remaining points according to Pr[p] = cost({p},COpt) / Opt and assign weight 1/Pr[p].
- Analyze using Hoeffding's bound for all solutions on an appropriate net



Technique 2.5: Importance Sampling

Idea [Feldman, Monemizadeh, S., 2005]

- Sample uniformly the inner ring
- Sample all remaining points according to Pr[p] = cost({p},COpt) / Opt and assign weight 1/Pr[p].
- Analyze using Hoeffding's bound for all solutions on an appropriate net



Removing the inner ring
[Schulman, Langberg, 2010]

- Pr[p] = cost({p},COpt) /(2 Opt) + 1/(2n)

Technique 2.5: Importance Sampling

Definition [Langberg, Schulman, 2010]

The *sensitivity $\gamma(p)$* of a point $p \in P$ with respect to k-means clustering is defined as

$$\gamma(p) = \sup_{C=\{c1,..,ck\} \subseteq Rd} \text{cost}(\{p\},C) / \text{cost}(P,C).*$$

The *total sensitivity $\Gamma(P)$* of P is defined as $\Gamma(P) = \sum_{p \in P} \gamma(p)$

* assuming that cost({p},C) / cost(P,C) = 0, if cost({p},C) = 0.

Technique 2.5: Importance Sampling

Theorem [Feldman, Langberg, 2011]

- If we choose $s \geq d^*/\varepsilon^2 \, \Gamma(P)^2$ points with probability proportional to $\gamma(p)$ then w.p. at least 2/3 the resulting point set is a (k,ε)-coreset.
- Here, d^* is the VC dimension of the range space of the family of k balls.

Third Technique: Reducing the Variance

Coresets with Offset for k-Means [Feldman, Schmidt, S., 2013]

A pair (S,Δ) of a small weighted set S and a value Δ is called ***(k,ε)-coreset with offset*** for P, if for every set C of k centers we have

$$(1-\varepsilon)\,\mathrm{cost}(P,C) \le \mathrm{cost}(S,C) + \Delta \le (1+\varepsilon)\,\mathrm{cost}(P,C)$$

In the above definition, the weight of a point is interpreted as its multiplicity.

Third Technique: Reducing the Variance

Coresets with Offset for k-Means [Feldman, Schmidt, S., 2013]

A pair (S,Δ) of a small weighted set S and a value Δ is called ***(k,ε)-coreset with offset*** for P, if for every set C of k centers we have

$$(1-\varepsilon)\,\mathrm{cost}(P,C) \le \mathrm{cost}(S,C) + \Delta \le (1+\varepsilon)\,\mathrm{cost}(P,C)$$

In the above definition, the weight of a point is interpreted as its multiplicity.

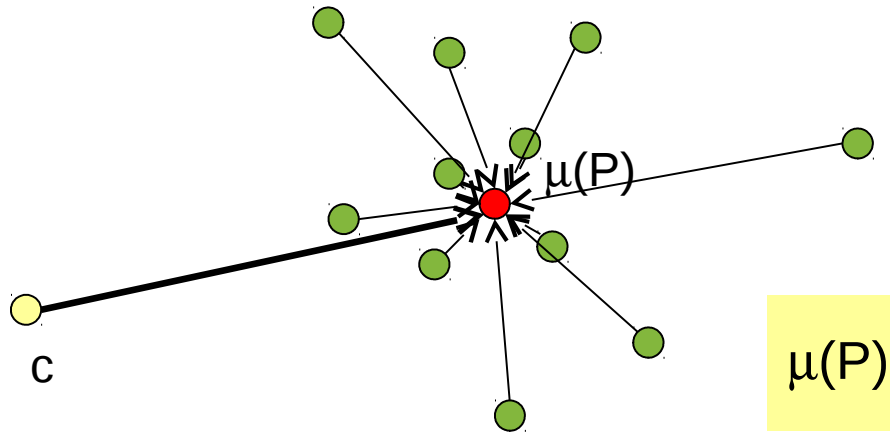
Remarks

- Δ may depend on P, k and ε
- This is **not** an additive approximation
- The variance in S is smaller than in P (by the amount of Δ)

Third Technique: Reducing the Variance

The „Magic Formula“ for k-Means [Zhang, Ramakrishnan, Livny, 96]

$$\sum_{p \in P} \|p-c\|^2 = |P| \cdot \|c-\mu(P)\|^2 + \sum_{p \in P} \|p-\mu(P)\|^2$$



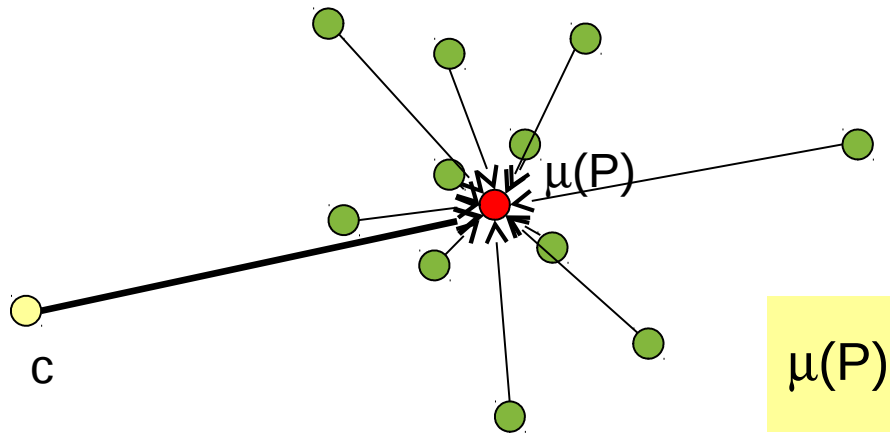
$$\mu(P) = \sum_{p \in P} p/|P|$$

Third Technique: Reducing the Variance

The „Magic Formula“ for k-Means [Zhang, Ramakrishnan, Livny, 96]

$$\sum_{p \in P} \|p-c\|^2 = |P| \cdot \|c-\mu(P)\|^2 + \sum_{p \in P} \|p-\mu(P)\|^2$$

Exact coreset for 1-means
Pair $\mu(P)$ with weight $|P|$ and the value $\Delta = \sum \|p-\mu(P)\|^2$



$$\mu(P) = \sum_{p \in P} p/|P|$$

Third Technique: Reducing the Variance

Definition

- Let Optk(P) denote the optimal clustering cost with k centers. A set P that satisfies Opt1(P)≤(1+ε)Optk(P) is called *pseudorandom*.

Intuition

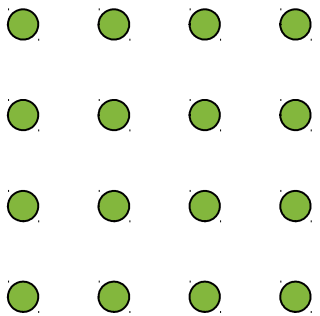
- A set is pseudorandom, if there is no good clustering (it behaves like a random point set).
- In particular, if we take an arbitrary partition into k clusters, their weighted means will be close to the mean of P

Third Technique: Reducing the Variance

A construction for k>1

- We recursively partition P into subsets {Pi} by applying the following rule:
- Let Pi be the set with the highest cost Opt1(Pi)
- **If** Pi is pseudorandom **then** replace Pi by $\mu(Pi)$
- **else** find an optimal clustering of Pi into k sets {Qj} and replace Pi by the {Qj}

  $[\sum_j \text{Opt1}(Qj)$ decreases by a factor of $1+\varepsilon]$
- If $\sum_j \text{Opt1}(Qj) \leq \varepsilon^2 \text{Optk}(P)$ then replace all {Qj} by {μj}

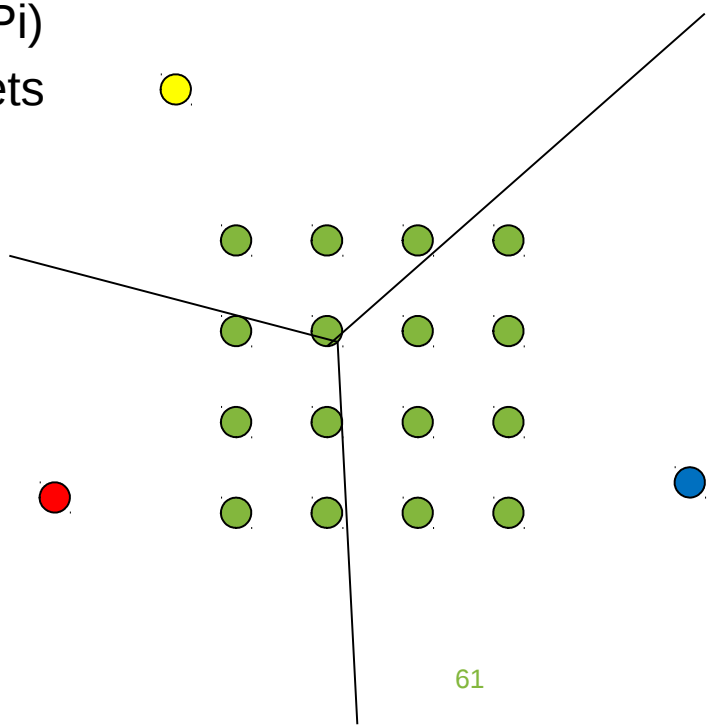


Third Technique: Reducing the Variance

A construction for k>1

- We recursively partition P into subsets {Pi} by applying the following rule:
- Let Pi be the set with the highest cost Opt1(Pi)
- **If** Pi is pseudorandom **then** replace Pi by μ(Pi)
- **else** find an optimal clustering of Pi into k sets {Qj} and replace Pi by the {Qj}

  [$\sum_j$ Opt1(Qj) decreases by a factor of $1+\varepsilon$]
- If $\sum_j$ Opt1(Qj) ≤ ε^2 Optk(P) then replace all {Qj} by {μj}

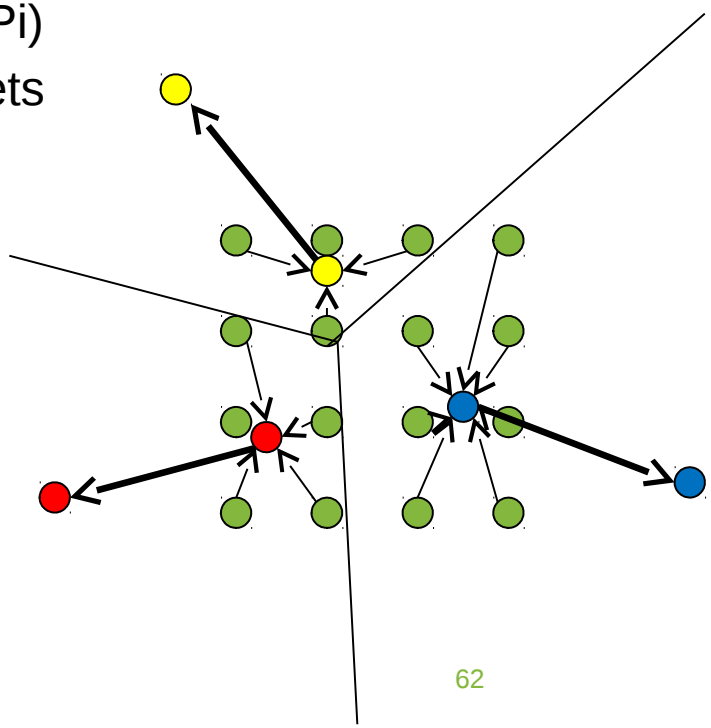


Third Technique: Reducing the Variance

A construction for k>1

- We recursively partition P into subsets {Pi} by applying the following rule:
- Let Pi be the set with the highest cost Opt1(Pi)
- **If** Pi is pseudorandom **then** replace Pi by μ(Pi)
- **else** find an optimal clustering of Pi into k sets {Qj} and replace Pi by the {Qj}

  $[\sum_j \text{Opt1}(Q_j)$ decreases by a factor of $1+\varepsilon]$
- If $\sum_j \text{Opt1}(Q_j) \leq \varepsilon^2 \text{Optk}(P)$ then replace all {Qj} by {μj}

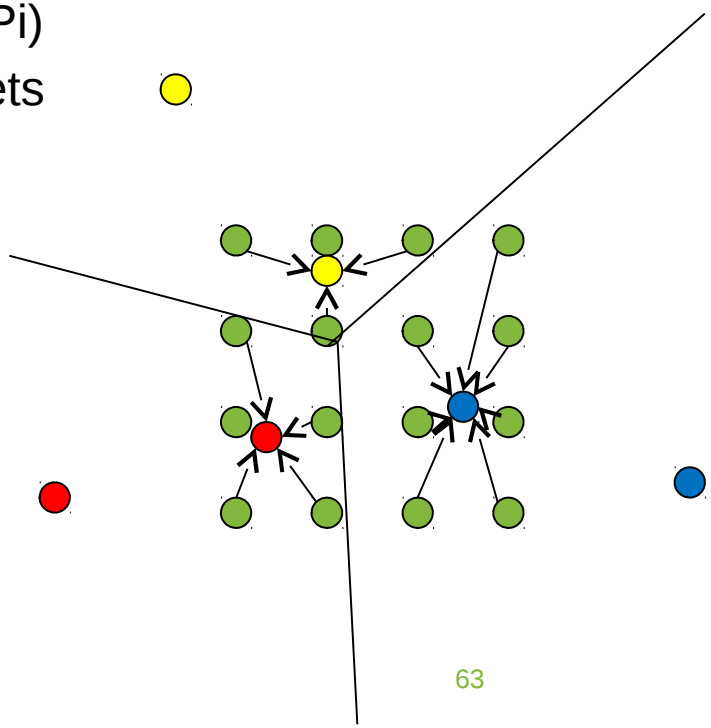


Third Technique: Reducing the Variance

A construction for k>1

- We recursively partition P into subsets {Pi} by applying the following rule:
- Let Pi be the set with the highest cost Opt1(Pi)
- **If** Pi is pseudorandom **then** replace Pi by μ(Pi)
- **else** find an optimal clustering of Pi into k sets {Qj} and replace Pi by the {Qj}

  $[\sum_j \text{Opt1}(Q_j)$ decreases by a factor of $1+\varepsilon]$
- If $\sum_j \text{Opt1}(Q_j) \leq \varepsilon^2 \text{Optk}(P)$ then replace all {Qj} by {μj}

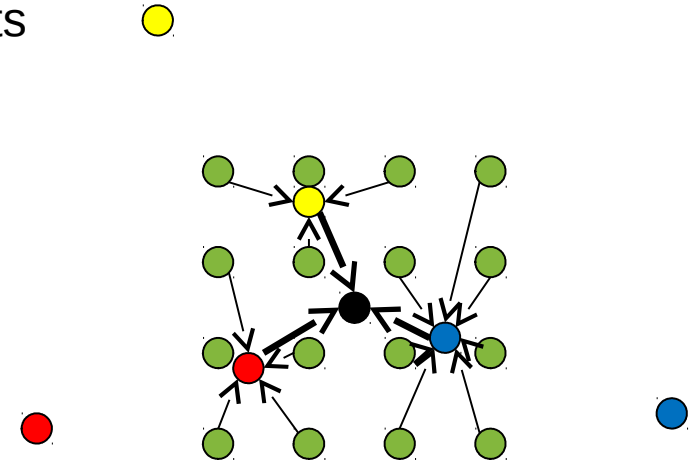


Third Technique: Reducing the Variance

A construction for k>1

- We recursively partition P into subsets {Pi} by applying the following rule:
- Let Pi be the set with the highest cost Opt1(Pi)
- **If** Pi is pseudorandom **then** replace Pi by μ(Pi)
- **else** find an optimal clustering of Pi into k sets {Qj} and replace Pi by the {Qj}

  $[\sum_j \text{Opt1}(Q_j)$ decreases by a factor of $1+\varepsilon]$
- If $\sum_j \text{Opt1}(Q_j) \leq \varepsilon^2 \text{Optk}(P)$ then replace all {Qj} by {μj}

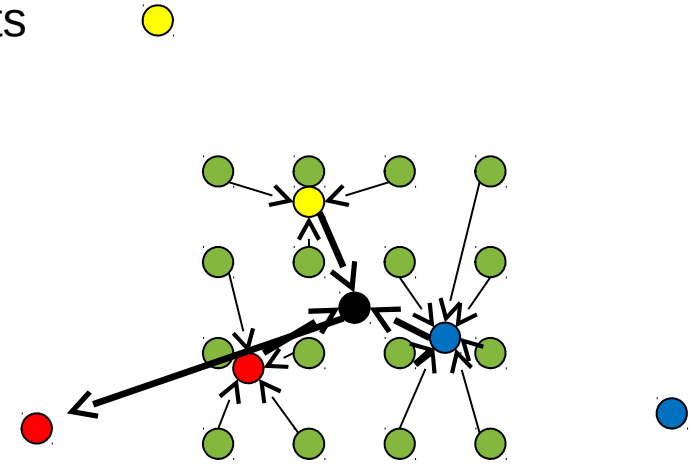


Third Technique: Reducing the Variance

A construction for k>1

- We recursively partition P into subsets {Pi} by applying the following rule:
- Let Pi be the set with the highest cost Opt1(Pi)
- **If** Pi is pseudorandom **then** replace Pi by $\mu(Pi)$
- **else** find an optimal clustering of Pi into k sets {Qj} and replace Pi by the {Qj}

  $[\sum_j \text{Opt1}(Qj)$ decreases by a factor of $1+\varepsilon]$
- If $\sum_j \text{Opt1}(Qj) \leq \varepsilon^2 \text{Optk}(P)$ then replace all {Qj} by {μj}



Third Technique: Reducing the Variance

Theorem [Feldman, Schmidt, S., 2013]

- The constructed coreset has size $k^{O(1/\varepsilon \log 1/\varepsilon)}$.

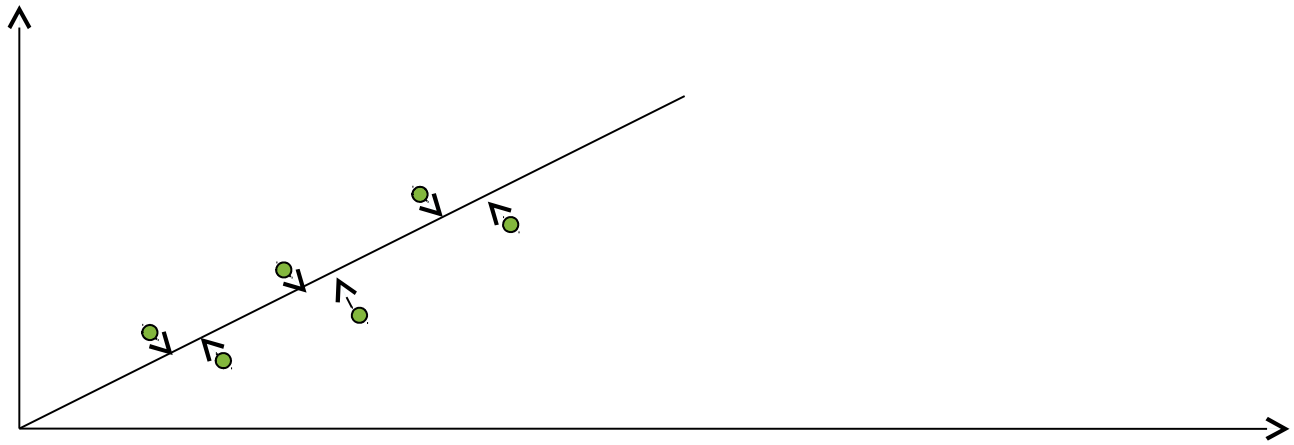
Fourth Technique: Linear Projections

Theorem [Feldman, Schmidt, S., 2013]

- There is a coreset with offset of size $(k/\varepsilon)O(1)$ for the k-means problem.

Main idea

- Replace the point set by its projection on an optimal fit $O(k/\varepsilon^2)$-subspace and compute a coreset for this subspace (actually, a slightly larger space)



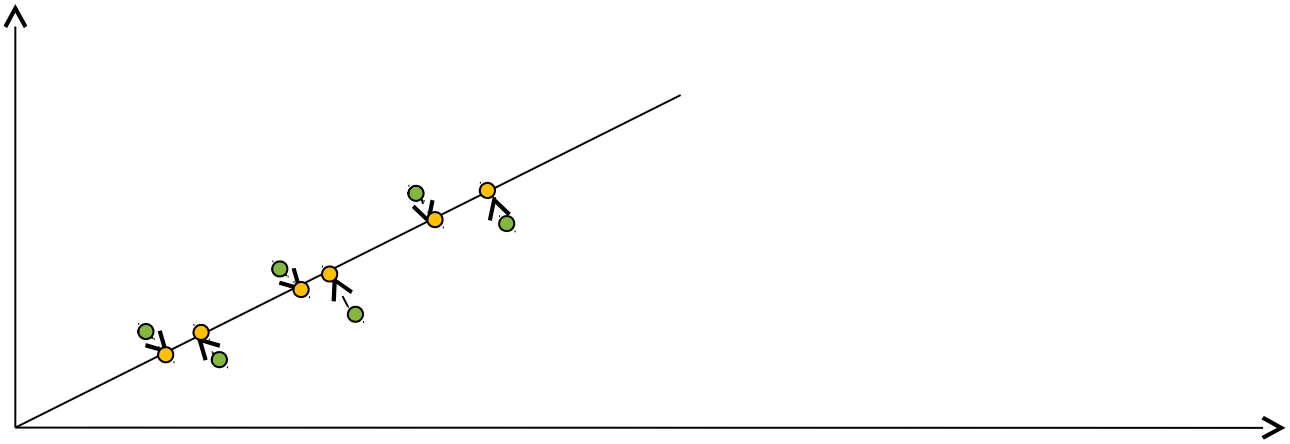
Fourth Technique: Linear Projections

Theorem [Feldman, Schmidt, S., 2013]

- There is a coreset with offset of size $(k/\varepsilon)O(1)$ for the k-means problem.

Main idea

- Replace the point set by its projection on an optimal fit $O(k/\varepsilon^2)$-subspace and compute a coreset for this subspace (actually, a slightly larger space)



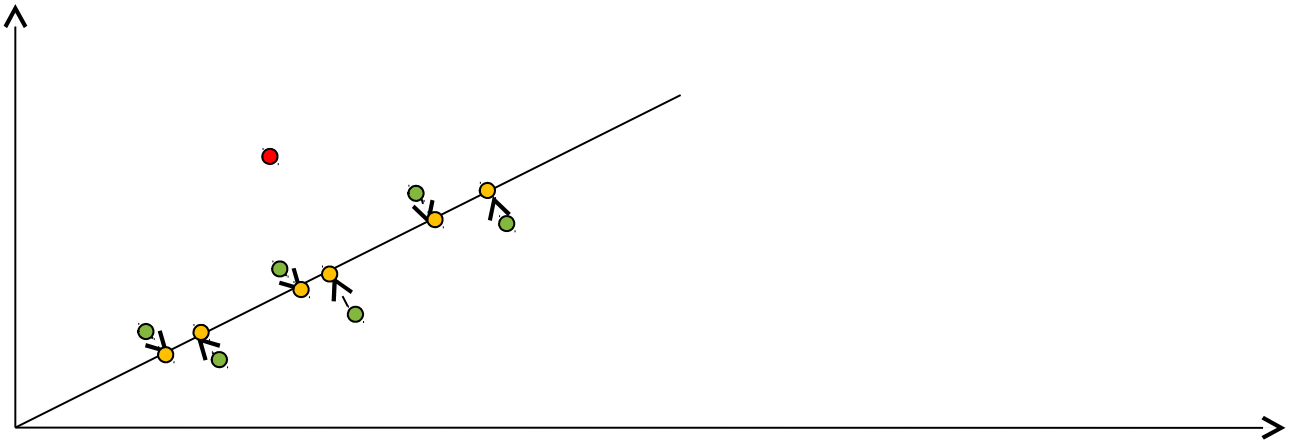
Fourth Technique: Linear Projections

Theorem [Feldman, Schmidt, S., 2013]

- There is a coreset with offset of size $(k/\varepsilon)O(1)$ for the k-means problem.

Main idea

- Replace the point set by its projection on an optimal fit $O(k/\varepsilon^2)$-subspace and compute a coreset for this subspace (actually, a slightly larger space)



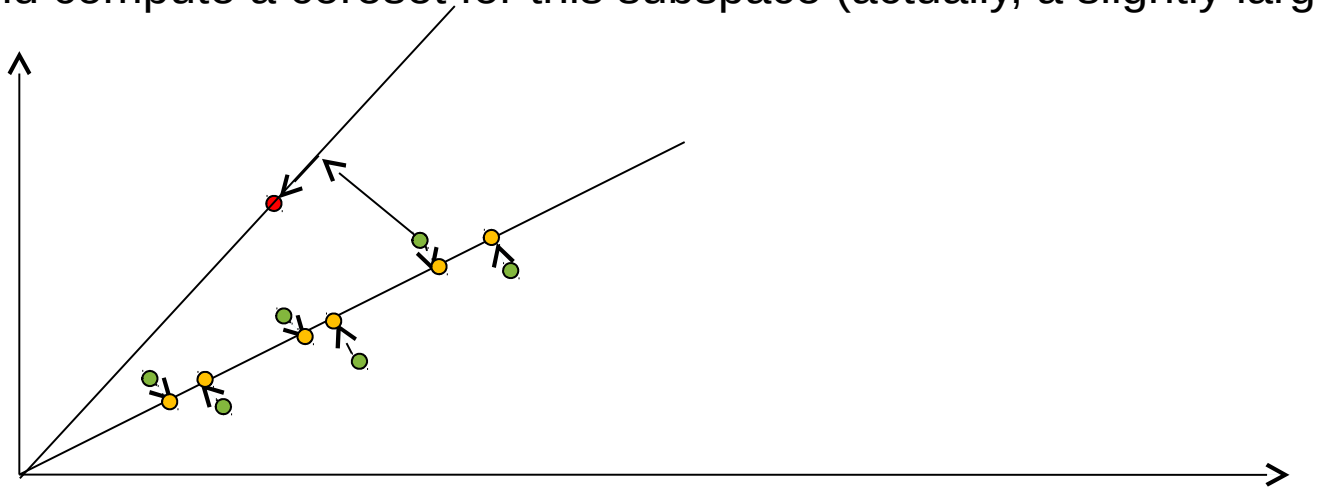
Fourth Technique: Linear Projections

Theorem [Feldman, Schmidt, S., 2013]

- There is a coreset with offset of size $(k/\varepsilon)O(1)$ for the k-means problem.

Main idea

- Replace the point set by its projection on an optimal fit $O(k/\varepsilon^2)$-subspace and compute a coreset for this subspace (actually, a slightly larger space)



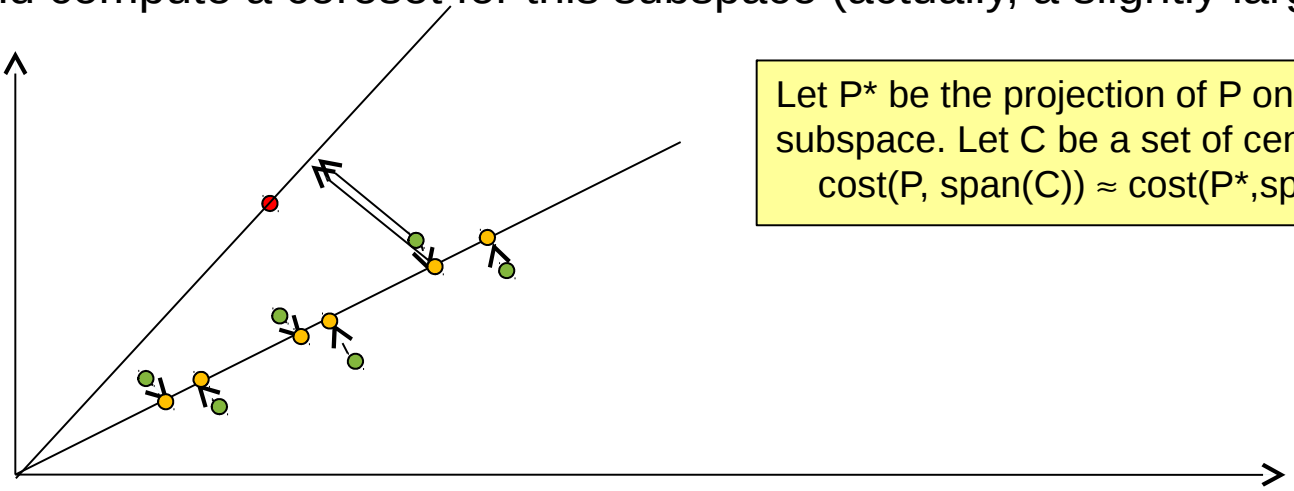
Fourth Technique: Linear Projections

Theorem [Feldman, Schmidt, S., 2013]

- There is a coreset with offset of size $(k/\varepsilon)O(1)$ for the k-means problem.

Main idea

- Replace the point set by its projection on an optimal fit $O(k/\varepsilon^2)$-subspace and compute a coreset for this subspace (actually, a slightly larger space)



Let P* be the projection of P on the best fit subspace. Let C be a set of centers. Then
$\text{cost}(P, \text{span}(C)) \approx \text{cost}(P^*, \text{span}(C)) + \Delta$.

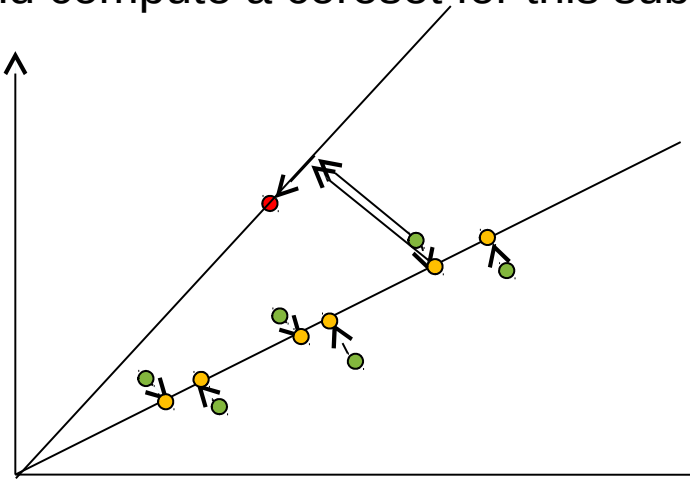
Fourth Technique: Linear Projections

Theorem [Feldman, Schmidt, S., 2013]

- There is a coreset with offset of size $(k/\varepsilon)O(1)$ for the k-means problem.

Main idea

- Replace the point set by its projection on an optimal fit $O(k/\varepsilon^2)$-subspace and compute a coreset for this subspace (actually, a slightly larger space)



Let P* be the projection of P on the best fit subspace. Let C be a set of centers. Then
$\text{cost}(P, \text{span}(C)) \approx \text{cost}(P^*, \text{span}(C)) + \Delta$.

Let P* be the projection of P on the best fit subspace. For any set of centers C, the projections of P and P* on the linear span of C have „distance“ at most ε cost(P,span(C)).

Extensions

Other types of centers

- Most techniques extend to different types of centers like subspace or any shapes that are contained in a low dimensional subspace

Other distance measures

- l_2^2- distance can be replaced by l_p – distance for constant p and q

Coresets and Approximation Algorithms

Theorem

- The k-means problem can be approximated within a factor of $(1+\varepsilon)$ in time $O(T(n,k,\varepsilon) + S(n,k,\varepsilon)O(dk^2))$, where

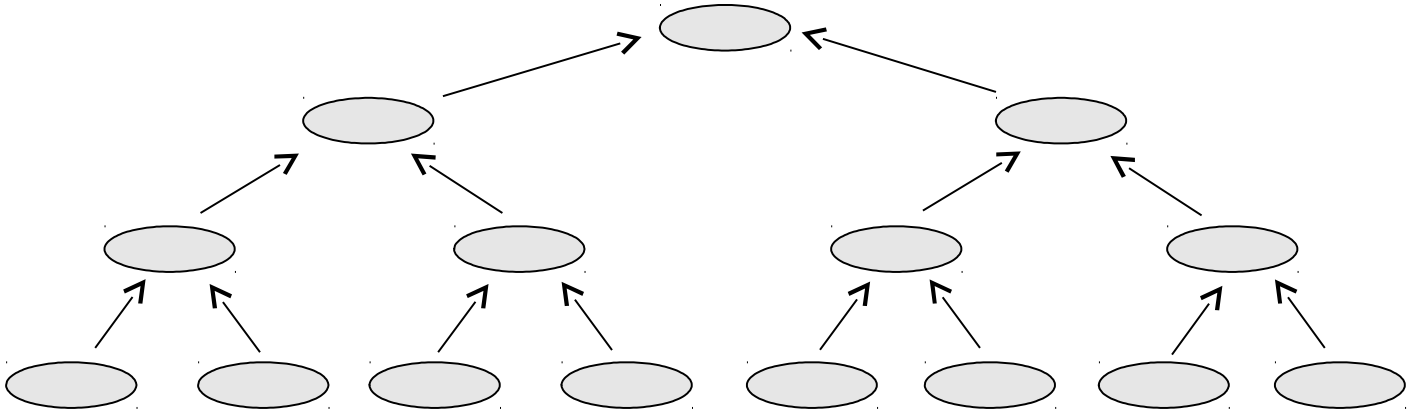
- $T(n,k,\varepsilon)$ is the time to compute a (k,ε)-coreset on an input of size n and
- $S(n,k,\varepsilon)$ is the size of the coreset.

Coresets and Streaming Algorithms

[Bentley, Saxe, 80; Agarwal, Har-Peled, Varadarajan, 2004]

Assumption

- The union of two coresets for A and B is a coreset of A∪B
- It is possible to compute a coreset of a coreset

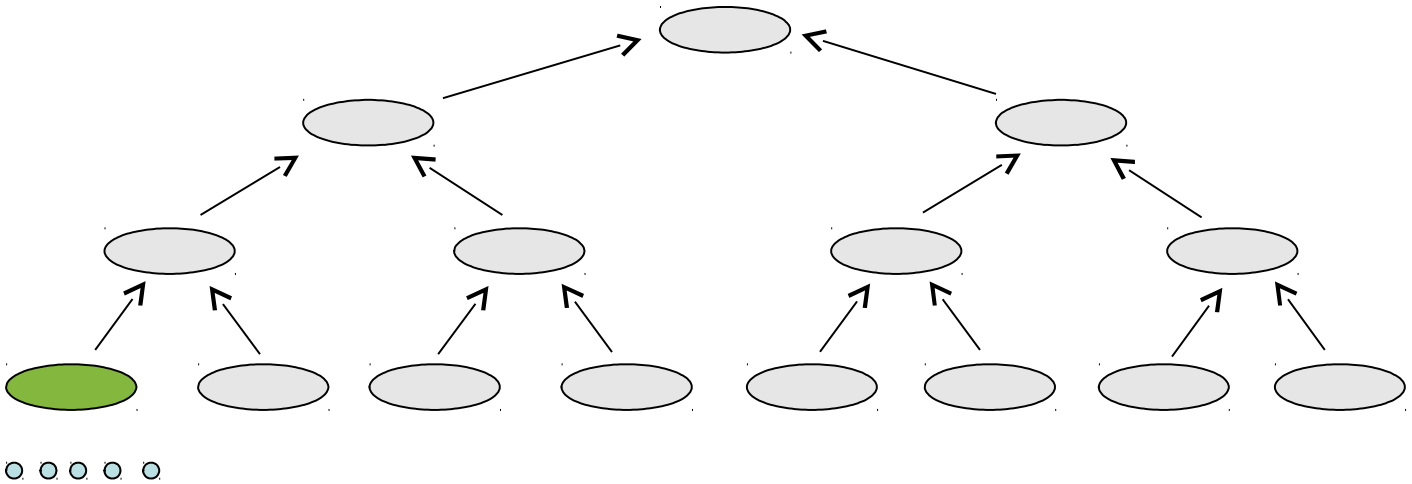


Coresets and Streaming Algorithms

[Bentley, Saxe, 80; Agarwal, Har-Peled, Varadarajan, 2004]

Assumption

- The union of two coresets for A and B is a coreset of A∪B
- It is possible to compute a coreset of a coreset

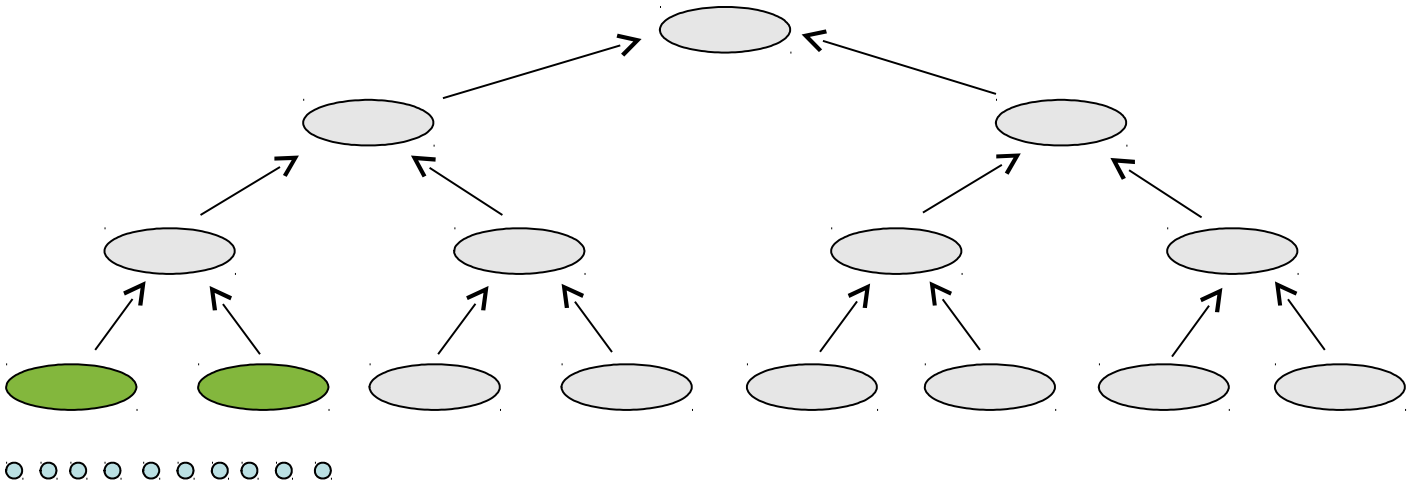


Coresets and Streaming Algorithms

[Bentley, Saxe, 80; Agarwal, Har-Peled, Varadarajan, 2004]

Assumption

- The union of two coresets for A and B is a coreset of A∪B
- It is possible to compute a coreset of a coreset

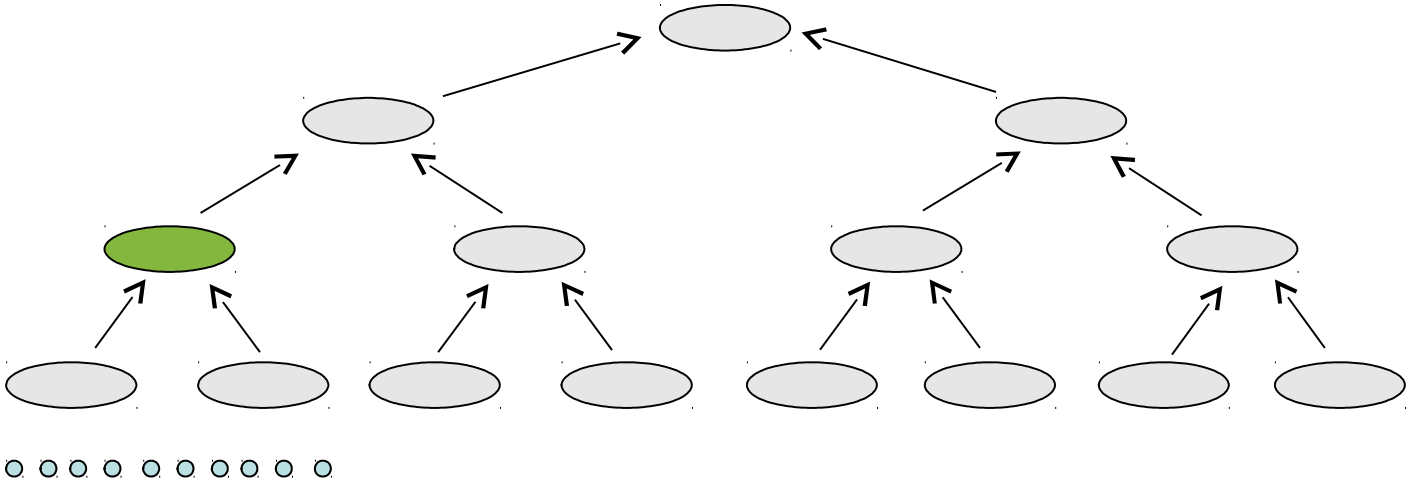


Coresets and Streaming Algorithms

[Bentley, Saxe, 80; Agarwal, Har-Peled, Varadarajan, 2004]

Assumption

- The union of two coresets for A and B is a coreset of A∪B
- It is possible to compute a coreset of a coreset

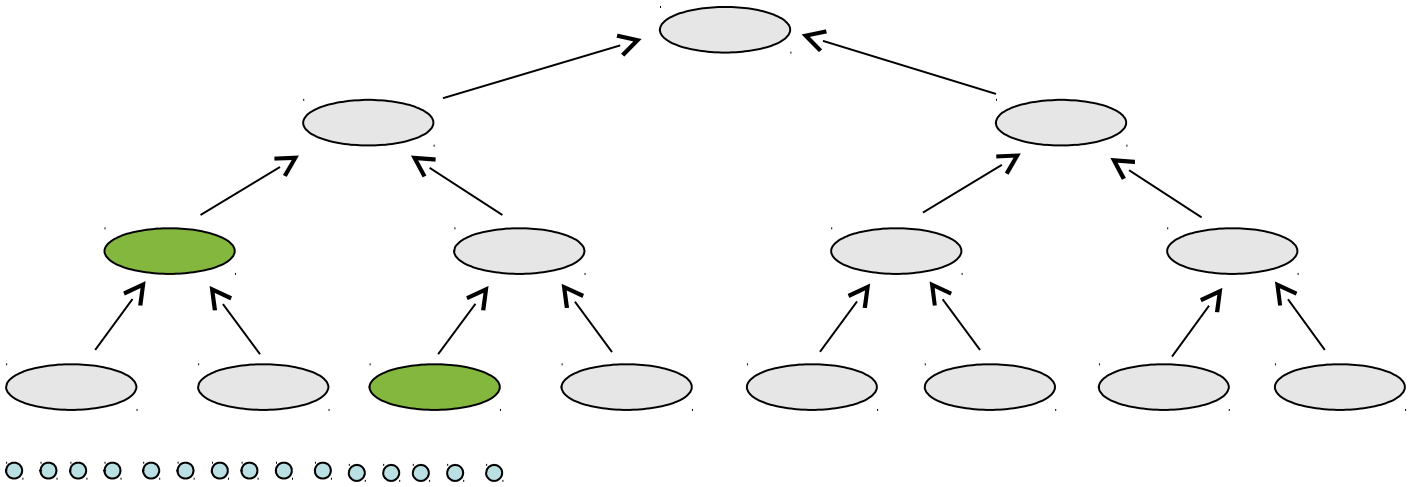


Coresets and Streaming Algorithms

[Bentley, Saxe, 80; Agarwal, Har-Peled, Varadarajan, 2004]

Assumption

- The union of two coresets for A and B is a coreset of A∪B
- It is possible to compute a coreset of a coreset

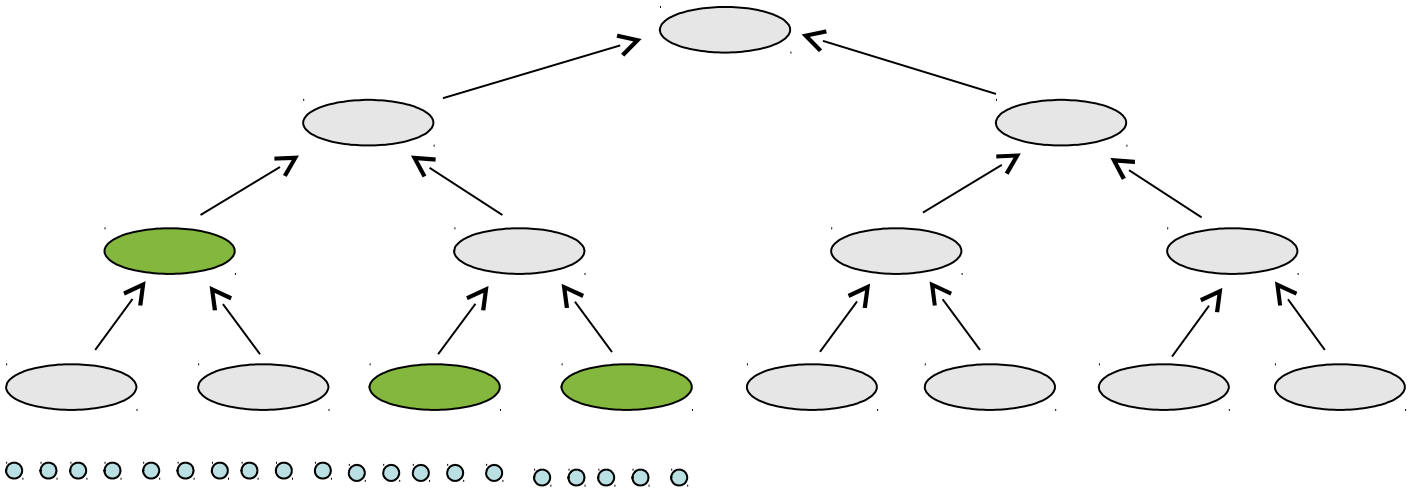


Coresets and Streaming Algorithms

[Bentley, Saxe, 80; Agarwal, Har-Peled, Varadarajan, 2004]

Assumption

- The union of two coresets for A and B is a coreset of A∪B
- It is possible to compute a coreset of a coreset

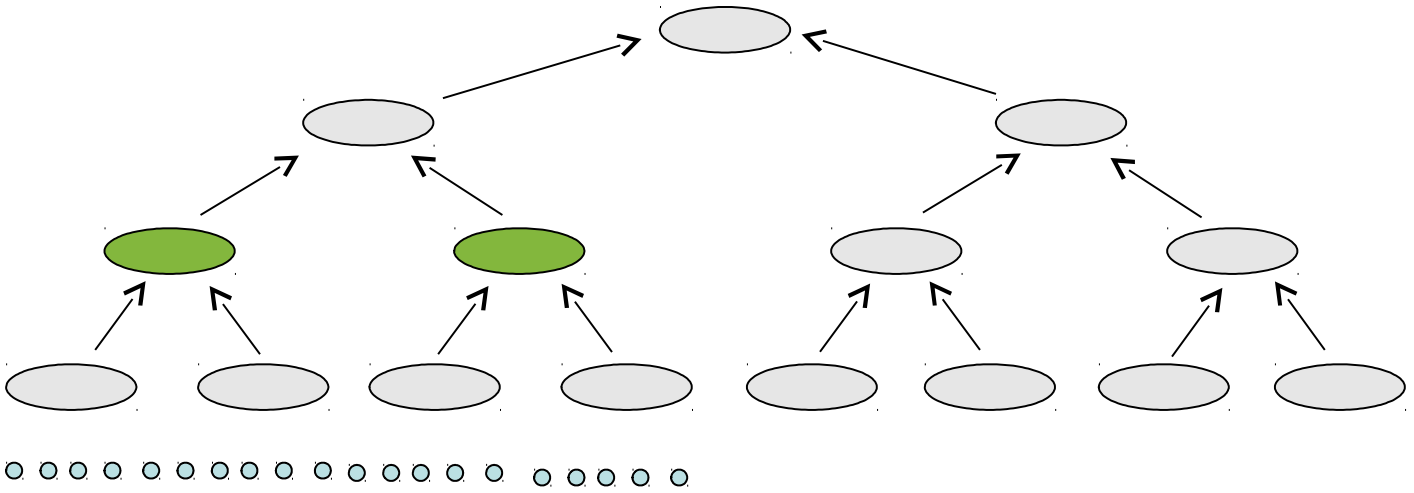


Coresets and Streaming Algorithms

[Bentley, Saxe, 80; Agarwal, Har-Peled, Varadarajan, 2004]

Assumption

- The union of two coresets for A and B is a coreset of A∪B
- It is possible to compute a coreset of a coreset

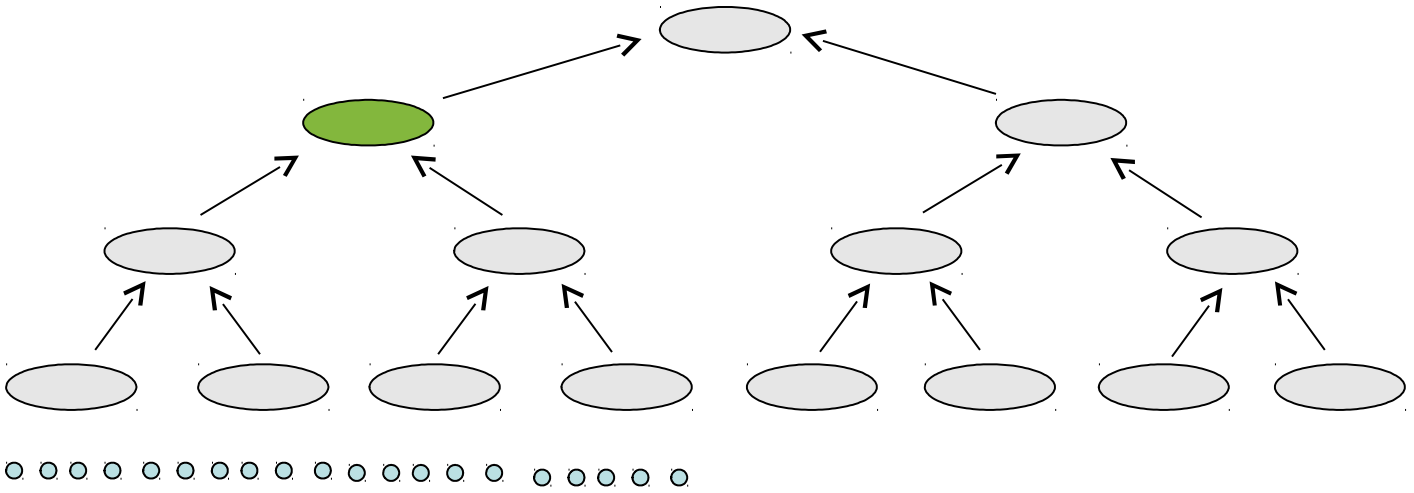


Coresets and Streaming Algorithms

[Bentley, Saxe, 80; Agarwal, Har-Peled, Varadarajan, 2004]

Assumption

- The union of two coresets for A and B is a coreset of A∪B
- It is possible to compute a coreset of a coreset

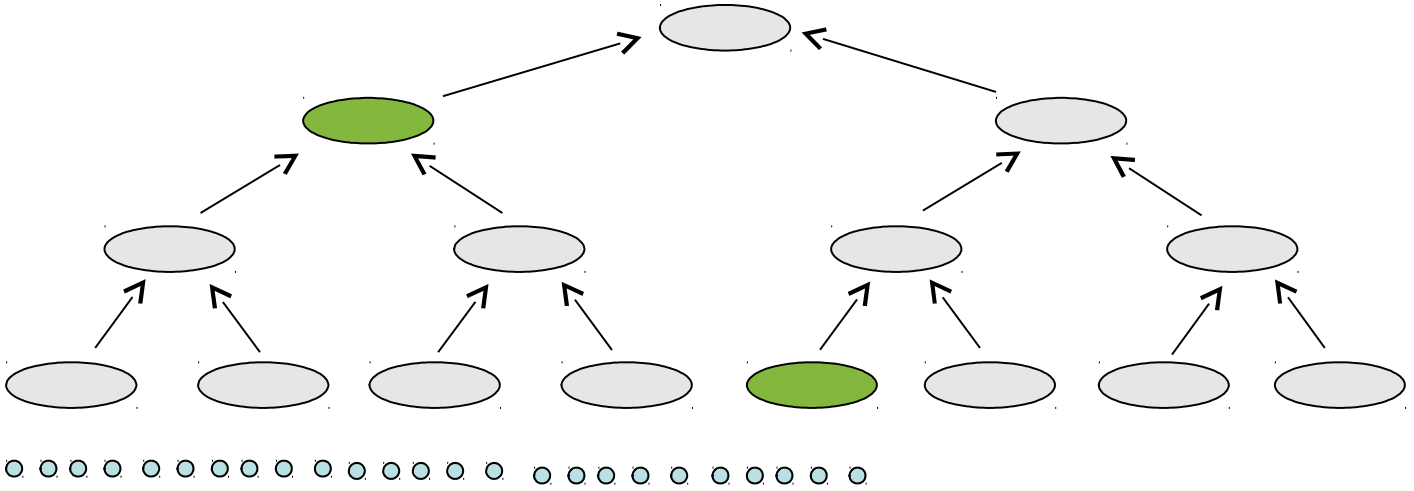


Coresets and Streaming Algorithms

[Bentley, Saxe, 80; Agarwal, Har-Peled, Varadarajan, 2004]

Assumption

- The union of two coresets for A and B is a coreset of A∪B
- It is possible to compute a coreset of a coreset

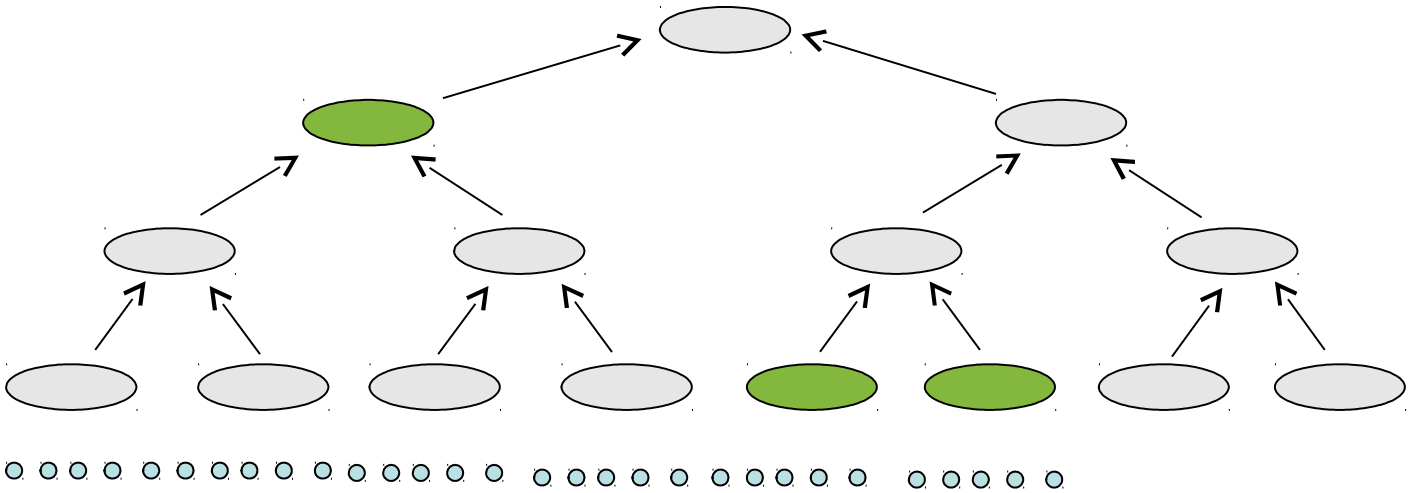


Coresets and Streaming Algorithms

[Bentley, Saxe, 80; Agarwal, Har-Peled, Varadarajan, 2004]

Assumption

- The union of two coresets for A and B is a coreset of A∪B
- It is possible to compute a coreset of a coreset

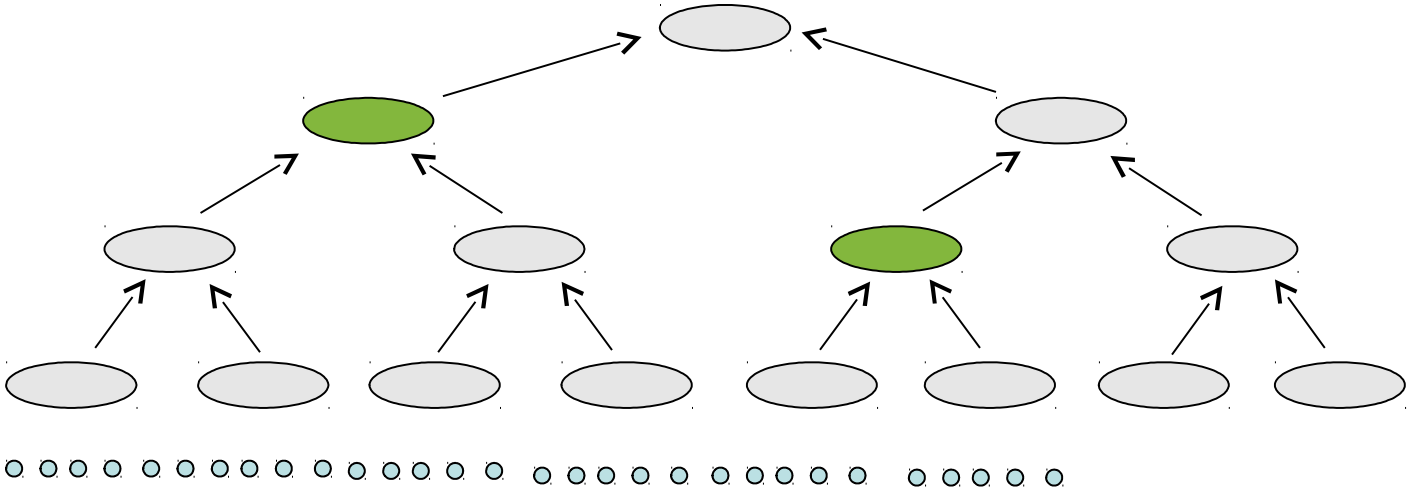


Coresets and Streaming Algorithms

[Bentley, Saxe, 80; Agarwal, Har-Peled, Varadarajan, 2004]

Assumption

- The union of two coresets for A and B is a coreset of A∪B
- It is possible to compute a coreset of a coreset

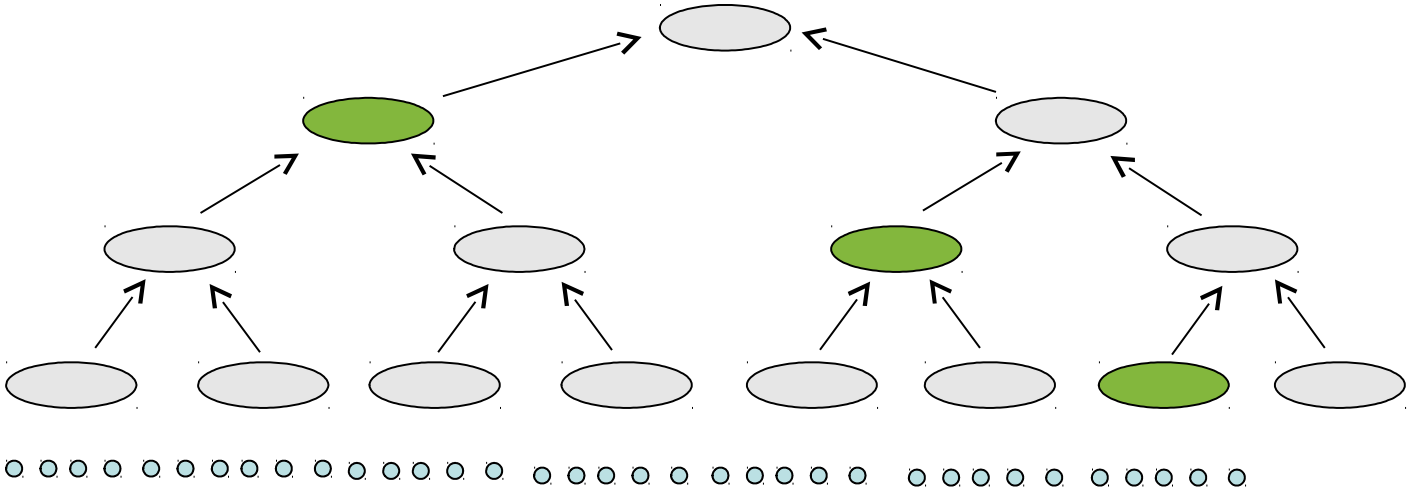


Coresets and Streaming Algorithms

[Bentley, Saxe, 80; Agarwal, Har-Peled, Varadarajan, 2004]

Assumption

- The union of two coresets for A and B is a coreset of A∪B
- It is possible to compute a coreset of a coreset

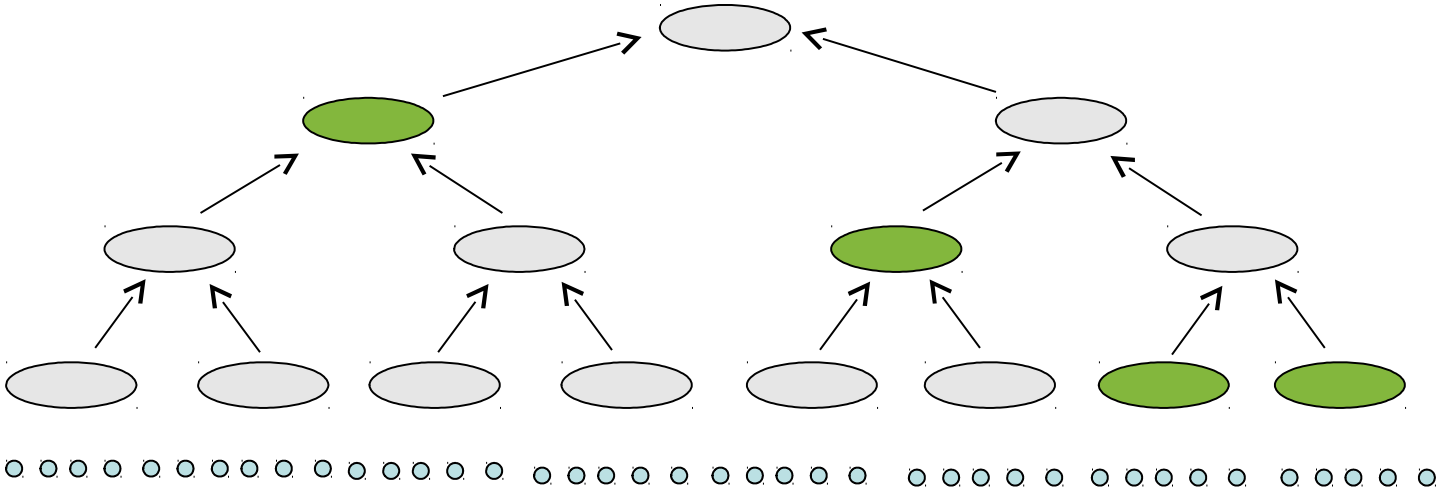


Coresets and Streaming Algorithms

[Bentley, Saxe, 80; Agarwal, Har-Peled, Varadarajan, 2004]

Assumption

- The union of two coresets for A and B is a coreset of A∪B
- It is possible to compute a coreset of a coreset

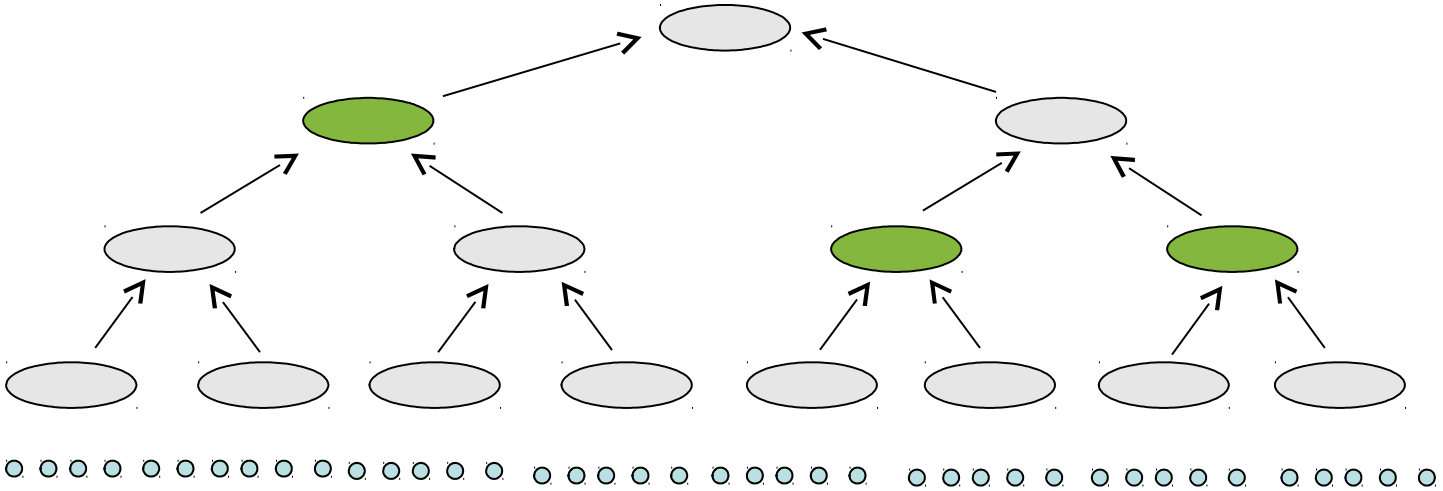


Coresets and Streaming Algorithms

[Bentley, Saxe, 80; Agarwal, Har-Peled, Varadarajan, 2004]

Assumption

- The union of two coresets for A and B is a coreset of A∪B
- It is possible to compute a coreset of a coreset

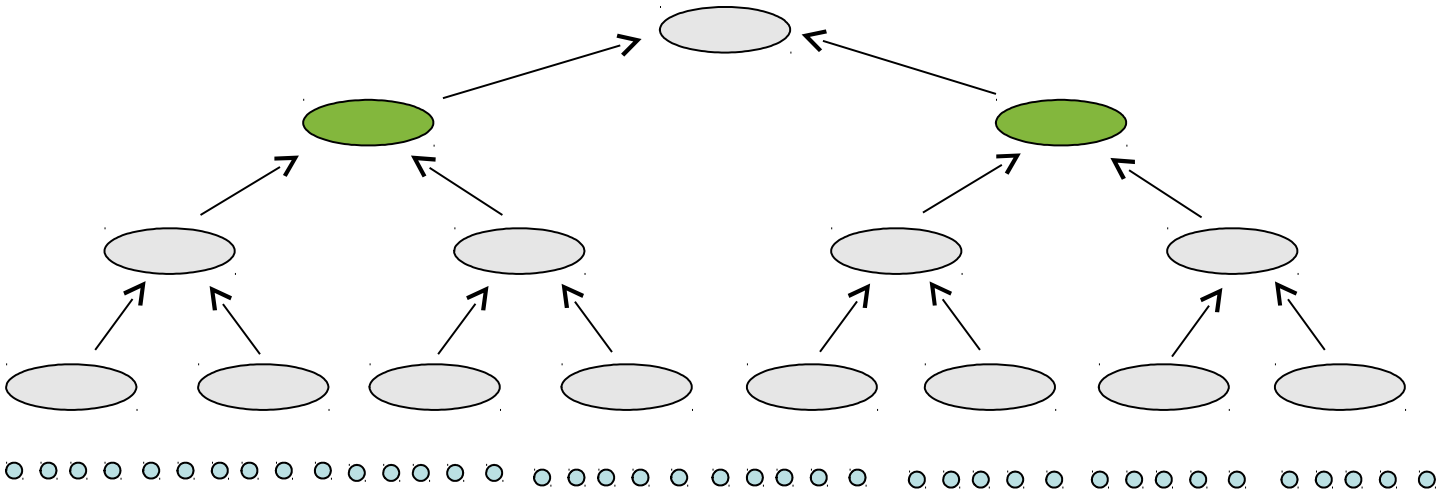


Coresets and Streaming Algorithms

[Bentley, Saxe, 80; Agarwal, Har-Peled, Varadarajan, 2004]

Assumption

- The union of two coresets for A and B is a coreset of A∪B
- It is possible to compute a coreset of a coreset

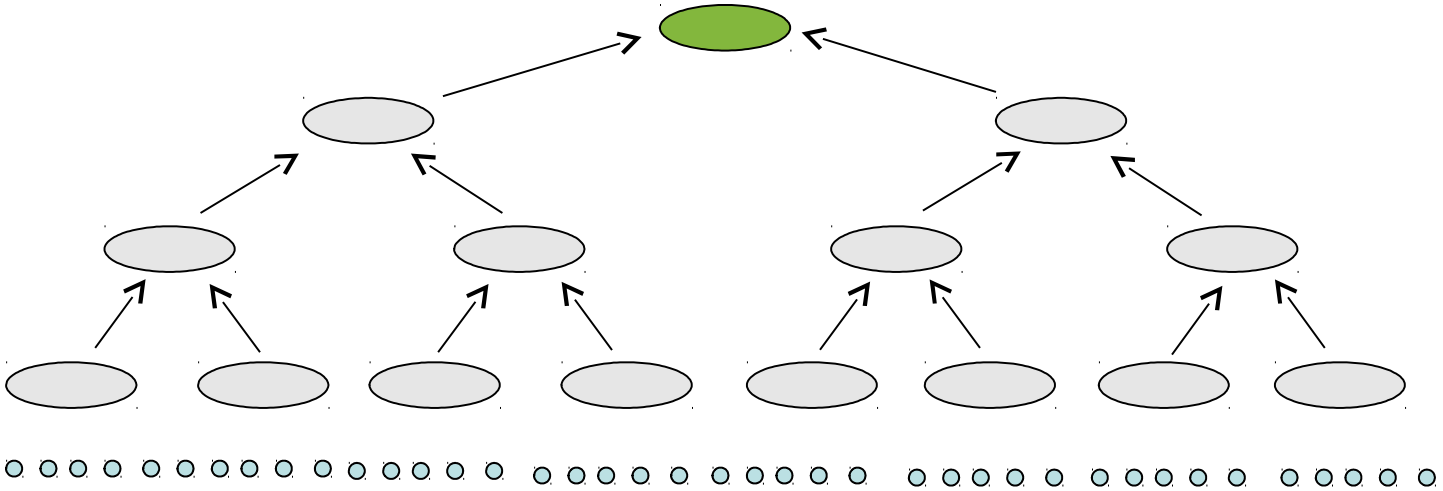


Coresets and Streaming Algorithms

[Bentley, Saxe, 80; Agarwal, Har-Peled, Varadarajan, 2004]

Assumption

- The union of two coresets for A and B is a coreset of A∪B
- It is possible to compute a coreset of a coreset



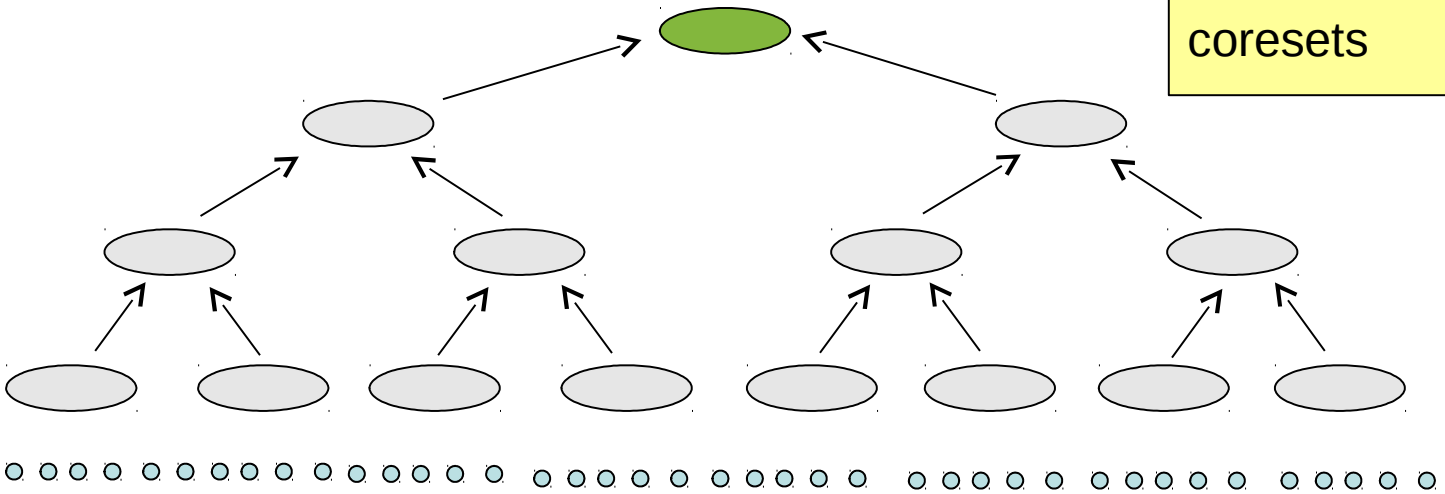
Coresets and Streaming Algorithms

[Bentley, Saxe, 80; Agarwal, Har-Peled, Varadarajan, 2004]

Assumption

- The union of two coresets for A and B is a coreset of A∪B
- It is possible to compute a coreset of a coreset

Compute the clustering on the union of the stored coresets

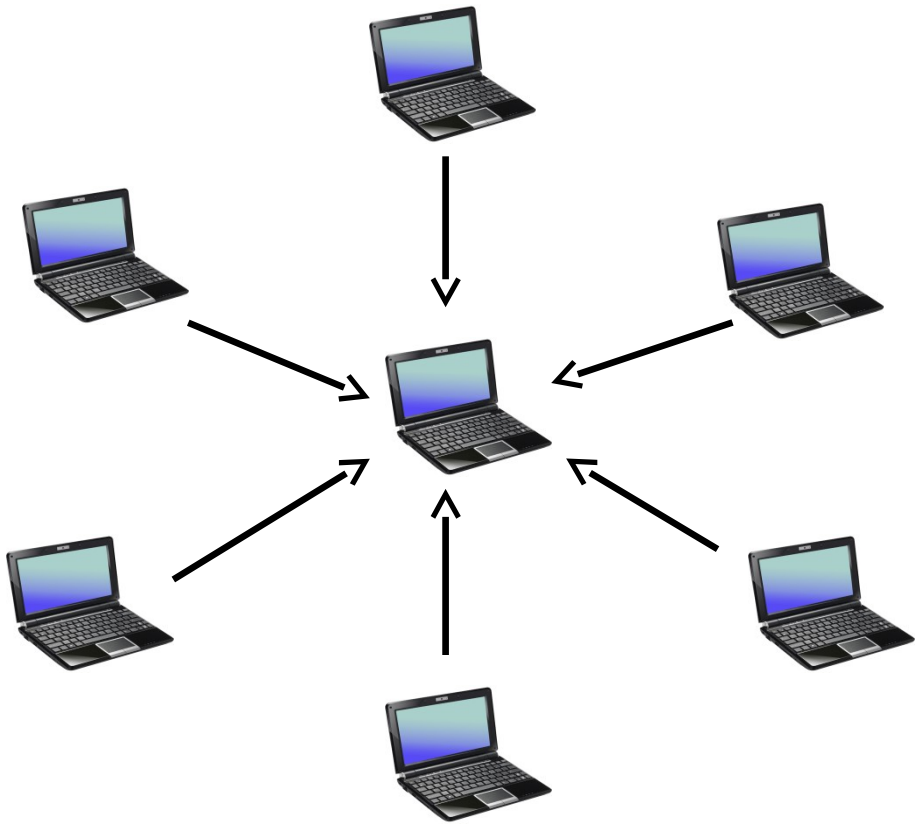


Coresets and Distributed Algorithms / Cloud Computing

Scenario:Distributed Input Data

- Data is collected locally
- Coreset will be computed locally

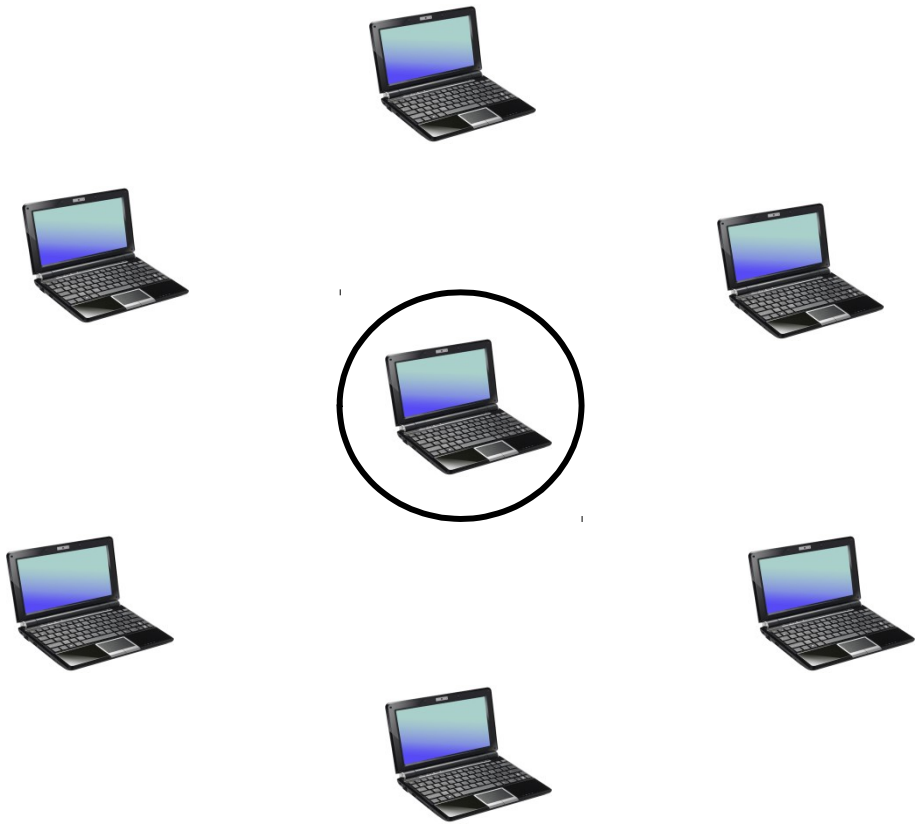
Coresets and Distributed Algorithms / Cloud Computing



Scenario:Distributed Input Data

- Data is collected locally
- Coreset will be computed locally
- Only the coresets are sent to a central computer

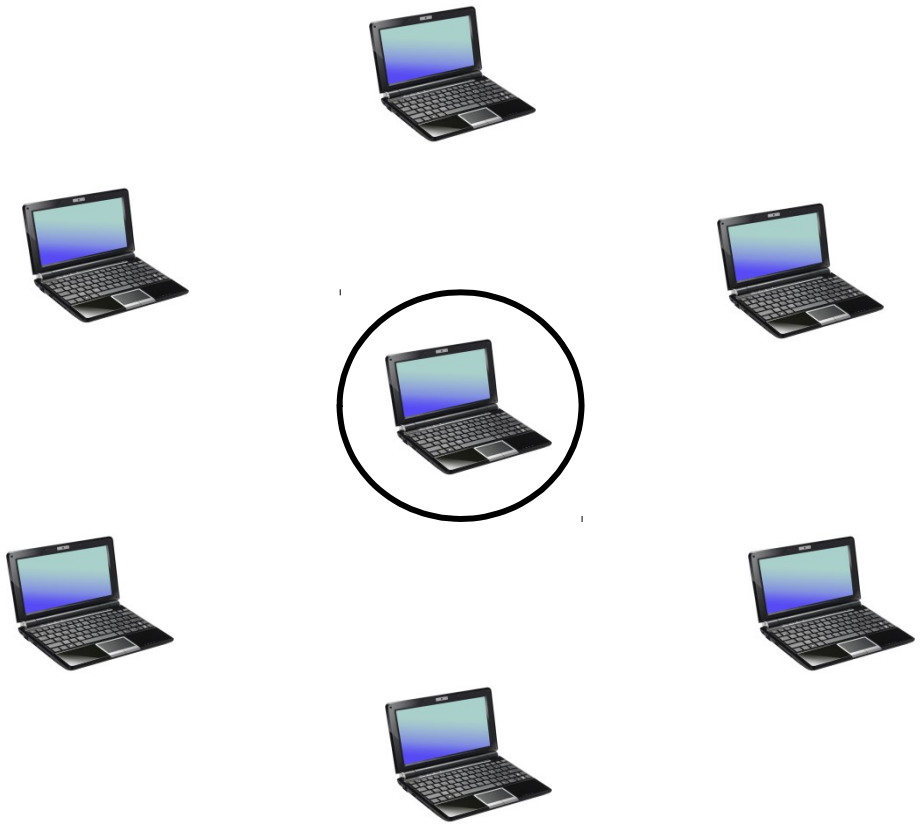
Coresets and Distributed Algorithms / Cloud Computing



Scenario:Distributed Input Data

- Data is collected locally
- Coreset will be computed locally
- Only the coresets are sent to a central computer
- Compute the clustering on this single machine

Coresets and Distributed Algorithms / Cloud Computing



Scenario:Distributed Input Data

- Data is collected locally
- Coreset will be computed locally
- Only the coresets are sent to a central computer
- Compute the clustering on this single machine

Improvements

- Implement merge&reduce like structure to avoid congestion
- Distributed computation of clustering

Practicality

StreamKM++
[Ackermann, Märtens, Raupach, Swierkot, Lammersen, S., 2011]

- Merge&Reduce based algorithm
- Coresets based on k-means++ seeding
- Coreset tree: Allows fast approximate k-means++ sampling

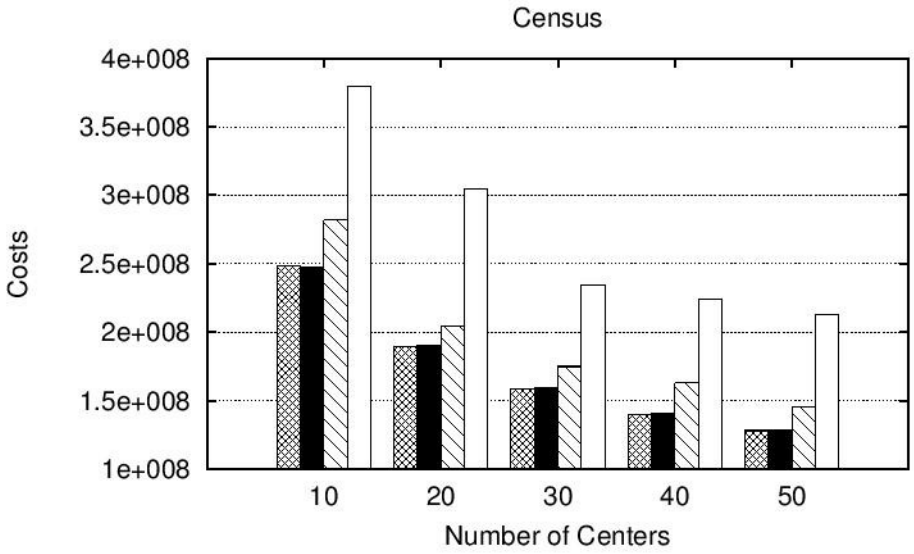
BICO [Fichtenberger, Gillé, Schmidt, Schwiegelshohn, S., 2013]

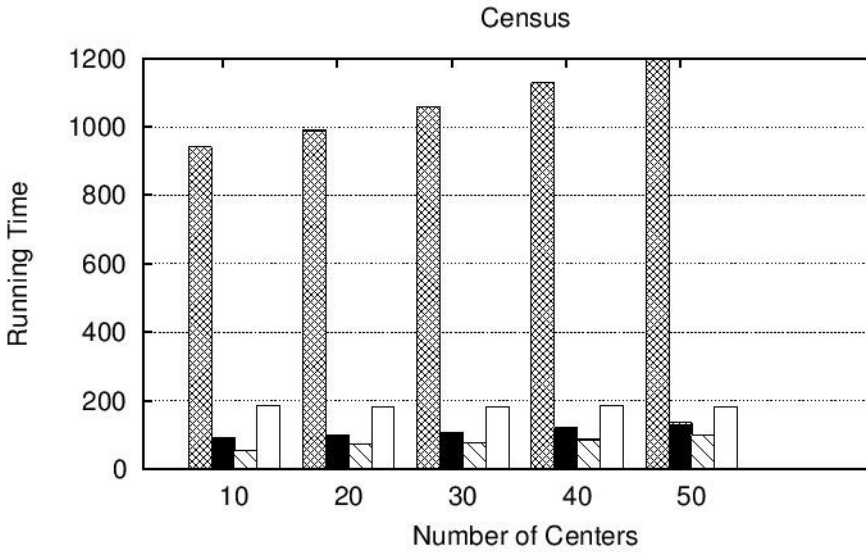
- Based on BIRCH‘s architecture
- Modifications, so that the algorithm computes a coreset
- A lot of heuristics to speed up performance

Practicality

Comparison of streaming clustering algorithms

- Census data set from UCI library
- 2.4 mio data points
- 68 dimensions





Summary

Coresets

- Bounded movement
- Sampling / Importance sampling
- Variance reduction
- Linear projections
- Easy-to-implement streaming & distributed algorithms
- High practicality