Streaming k-means Approximation

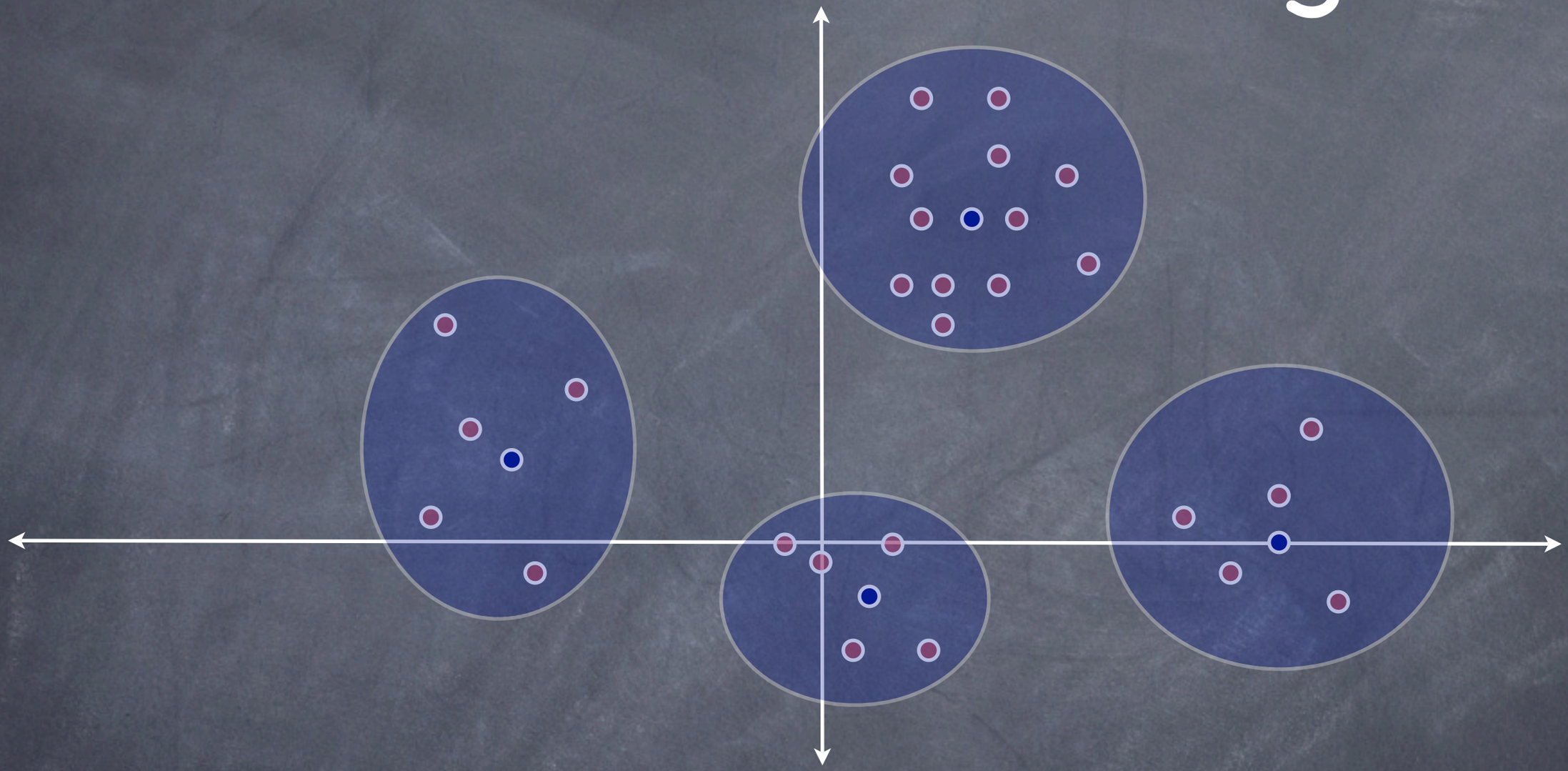
Ragesh Jaiswal
Columbia University

joint work with Claire Monteleoni (CCLS, Columbia) and Nir Ailon (Google)

Outline

- k-means problem:
  - (pseudo) approximation algorithm
  - Streaming algorithm

k-means Clustering



k-means clustering: Given a set of n points $X \subset R^d$, find a set of k points $C \subset R^d$ (called centers) such that the following potential function is minimized.

$$\Phi_C(X) = \Sigma_{x \in X} \min_{c \in C} ||x - c||^2$$

k-means Clustering

- k-means clustering problem is NP-hard even when $k = 2$ [DFKVV04].

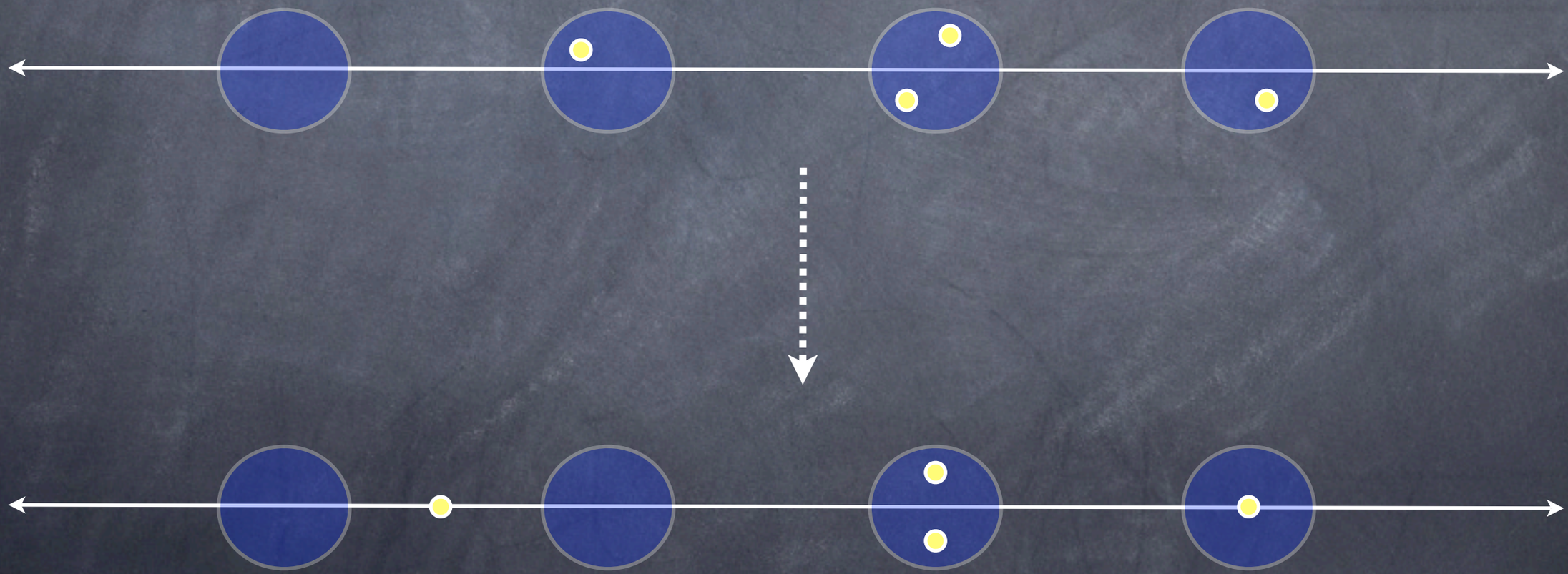
Problem	Algorithm	Approximation	Time (dependence on n)
k-means	Kanungo et al.'04 (k-means using Local Search)	$O(1)$	n^3
k-median	Jain & Vazirani'99 (k-median using Primal Dual)	$O(1)$	n^2
k-median	Charikar & Guha'99 (k-median using Primal Dual)	$O(1)$	n^2
k-median	Indyk'99 (Sampling Technique using above as black box)	$(O(1), O(1))$	n

k-means Clustering
(Lloyd's Algorithm)

- The most popular batch algorithm used is the Lloyd's algorithm [L57] (also known as k-means algorithm)
  - Initialize C (|C|=k) arbitrarily.
  - Cluster the points based on the nearest center.
  - Update C by picking the centroid of each cluster. Repeat.
- Does well in practice but has no theoretical approximation guarantees.

k-means Clustering
(Lloyd's Algorithm)

- Lloyd's algorithm suffers from bad initialization.



k-means Clustering
(k-means++)

- k-means++: Initialization algorithm (choosing points with D^2 weighting)
  - Step 1: Pick the first point randomly.
  - Step i ($1<i\leq k$): Choose a center by picking a point with probability proportional to the square of the distance of the point from its nearest $(i-1)$ previously chosen centers.

- [AV07]: The above algorithm gives an expected $O(\log(k))$ approximation guarantee.

k-means Clustering
(Bi-criteria approximation)

- Bi-criteria approximation or pseudo approximation
  - (α,β)-pseudo approximation algorithm: We are allowed to pick $\alpha * k$ centers with approximation factor β.

- Advantages:
  - Better approximation guarantees.
  - Streaming application.

k-means Clustering (k-means#)

- k-means#
  - Step 1: Pick the first $3*\log(k)$ points randomly.
  - Step i ($1<i\leq k$): Choose $3*\log(k)$ centers independently by picking a point with probability proportional to the square of the distance of the point from its nearest $3*\log(k)*(i-1)$ previously chosen centers.
- Theorem [AJM09]: With probability at least 1/4, k-means# gives $(O(\log(k)), O(1))$-pseudo approximation.
- Advantages:
  - The analysis becomes simple.
  - Streaming application.

k-means# (proof sketch)

- Definition:
  - Given a subset of points $Y \subset X$ and a set of centers C, let $\Phi_C(Y)$ denote the cost of C with respect to subset Y. More formally,

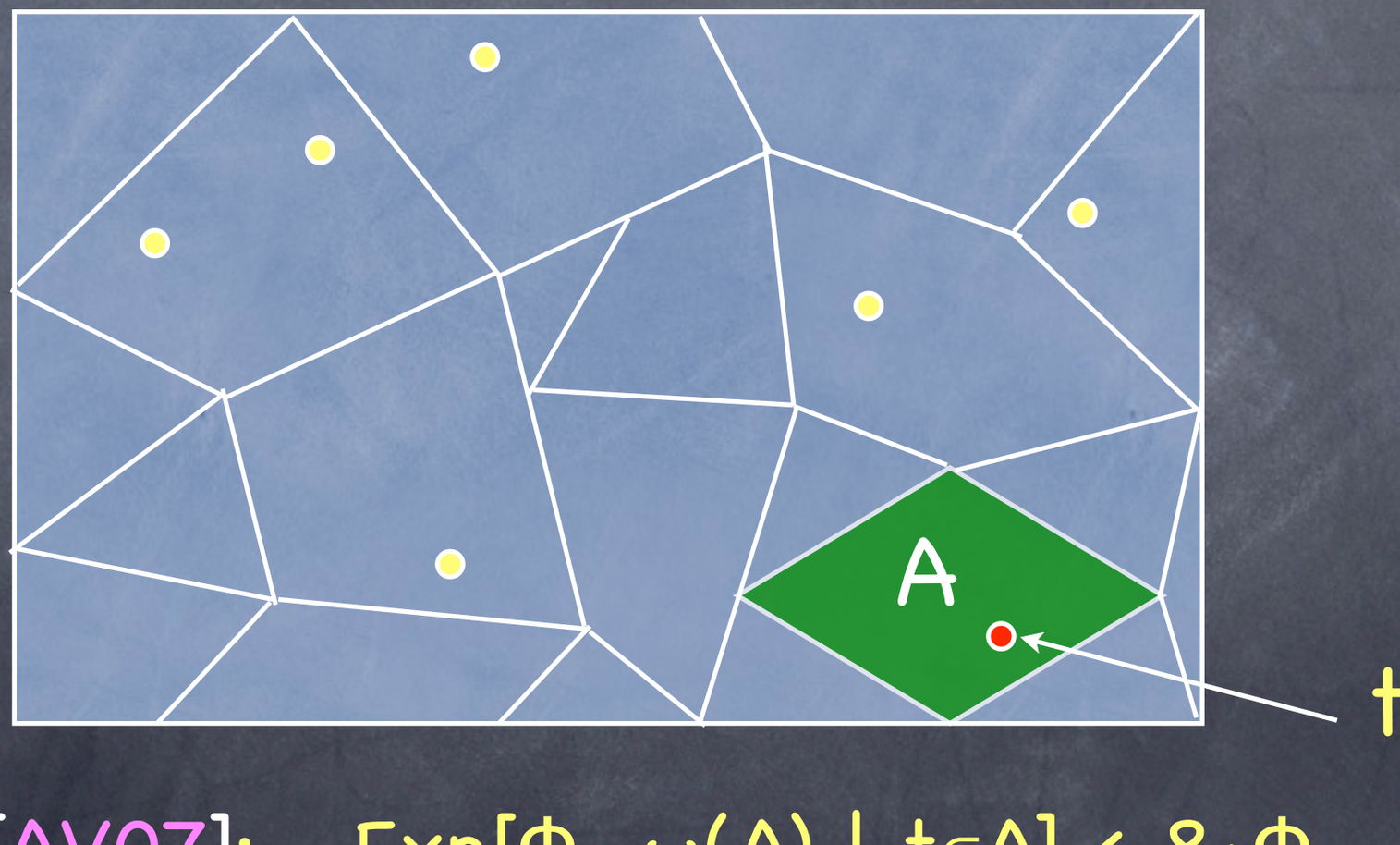
$$\Phi_C(Y) = \Sigma_{x \in Y} \min_{c \in C} \|x - c\|^2$$

- So, for any partition $Y_1, \ldots, Y_m$ of X

$$\Phi_C(X) = \Sigma_{i \in [1..m]} \Phi_C(Y_i)$$

k-means# (proof sketch)

- Let the partitions denote the optimal clusters.
- Let • denote an arbitrary set of centers C.
- Let A be a set of points in an optimal cluster.
- Let • denote a point chosen using D^2 weighting.



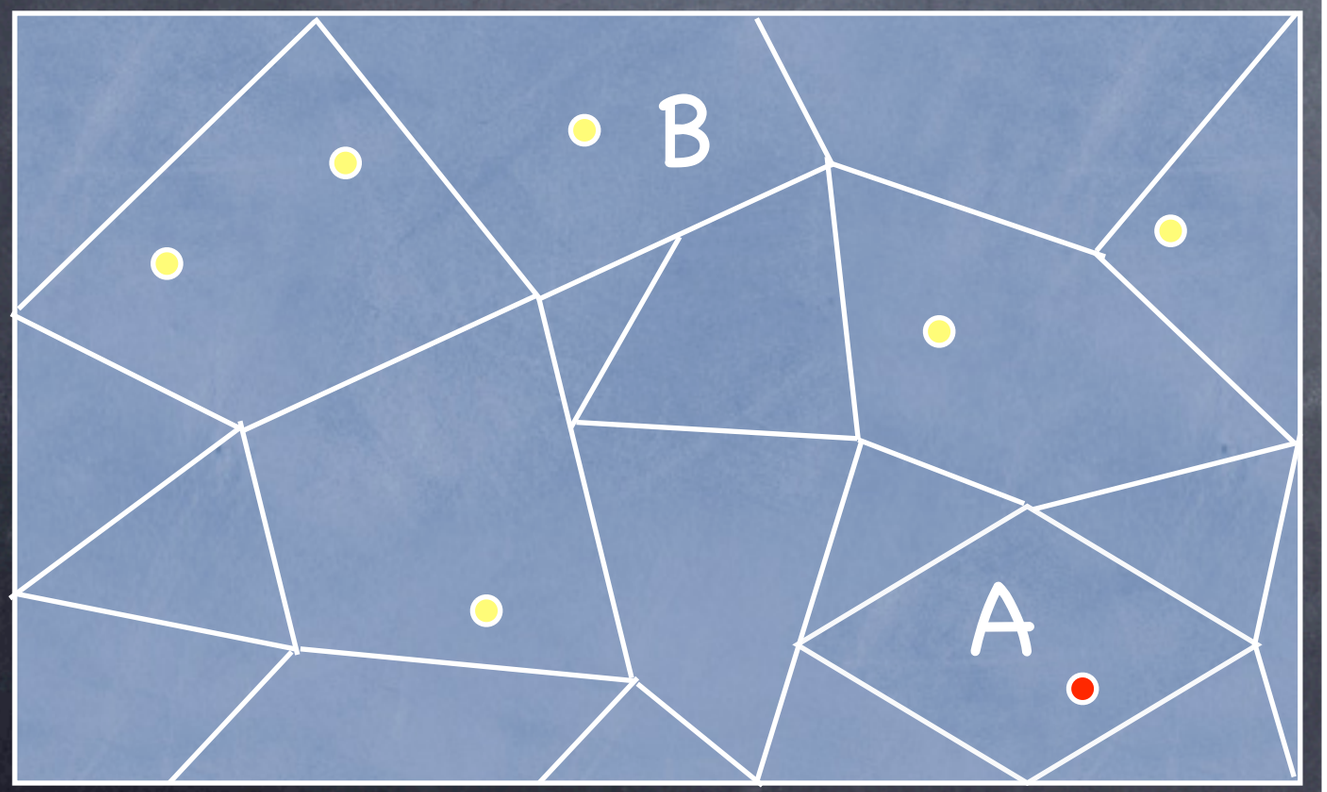
Lemma [AV07]: $\mathrm{Exp}[\Phi_{C\cup\{t\}}(A) \mid t\in A] \le 8*\Phi_{OPT}(A)$

Lemma (by markov): $\Pr[\Phi_{C\cup\{t\}}(A) \ge 16*\Phi_{OPT}(A) \mid t\in A] \le 1/2$

k-means# (proof sketch)

- Proof is by induction on the step number i of the algorithm.
- Base case is simple so we will skip that.
- Consider a set of centers C.
- An optimal cluster B is called "covered" if $\Phi_C(B) \leq const.\Phi_{OPT}(B)$.

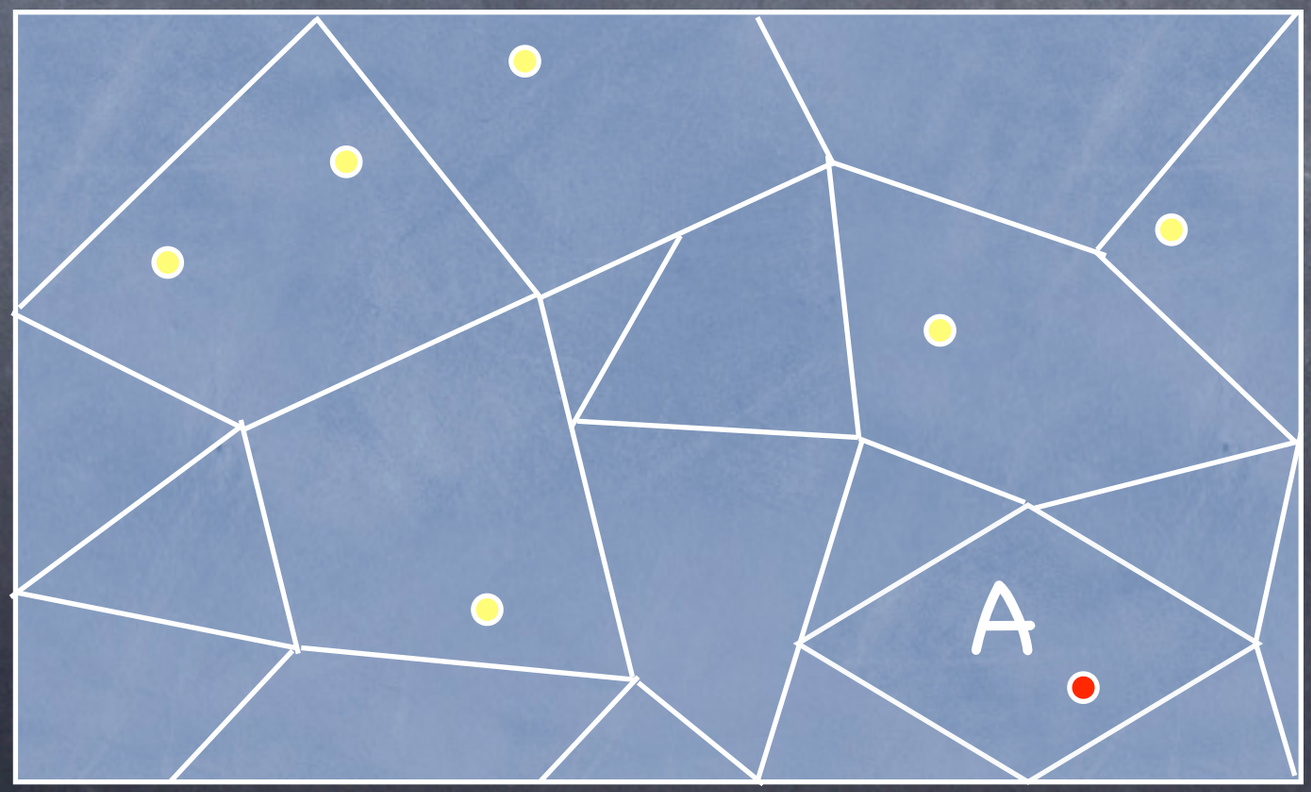
- Let X_u denote uncovered clusters
- X_c denote covered clusters



- Induction assumption: After (i-1) steps,
  1. there are (i-1) covered clusters, or
  2. the current set of centers gives constant approximation.
- Let p be the probability that next chosen center is from one of the uncovered clusters.
- If p is small, say p<1/2. Then $\Phi_C(X_u)/(\Phi_C(X_u) + \Phi_C(X_c)) < 1/2 \Rightarrow \Phi_C(X_u) < \Phi_C(X_c)$,
- but then, $\Phi_C(X) = \Phi_C(X_u) + \Phi_C(X_c) < 2.\Phi_C(X_c) < const.\Phi_{OPT}(X_c) < const.\Phi_{OPT}(X)$. This cannot happen since we are not in case (2).

k-means# (proof sketch)

- So, with probability > 1/2 the next center is chosen from one of the uncovered clusters,
- conditioned on this event, we know from the [AV07] lemma that the chosen center covers this previously uncovered clusters with constant probability.
- Since we pick 3.log(k) centers in step i we cover a new cluster with probability at least (1 - 1/k).
- So we cover all clusters with constant probability.



Better bound on number of centers

- k-means##: Seeding algorithm (choosing points with D^2 weighting)
  - Step 1: Pick the first point randomly.
  - Step i ($1<i\leq t$): Choose a center by picking a point with probability proportional to the square of the distance of the point from its nearest $(i-1)$ previously chosen centers.
- $t = O(k)$ centers is enough to show constant approximation [ADK09].

Super Martingales

- Definition (Super Martingale): A sequence of random variables $J_0,\ldots, J_t$ is called a super martingale if

  for all $i>0$, $E[J_i \mid J_0,\ldots,J_{i-1}] \leq J_{i-1}$

- Theorem (Azuma-Hoeffding inequality): If $J_0,\ldots,J_t$ is a super martingale and for all $t>i\geq 0$, $(J_{i+1} - J_i)\leq 1$, then

  $\Pr[J_t \geq J_0 + \delta] \leq \exp(-\delta^2/2t)$

Better bounds using super martingales [ADK09]

- Consider the random variable X_i
  - $X_i = 0$ if a new cluster is covered in step i and 1 otherwise.
- We have: $\forall i \geq 1$, $\Pr[X_i = 0] \geq c$, where c is some fixed constant.
- Consider the random variables
  - $J_0 = 0$, and $\forall i > 0$, $J_i = \sum_{j \leq i} (X_j - (1-c))$
- We have
  - $J_{i+1} - J_i \leq 1$
  - $E[J_i \mid J_0, \ldots, J_{i-1}] \leq J_{i-1}$

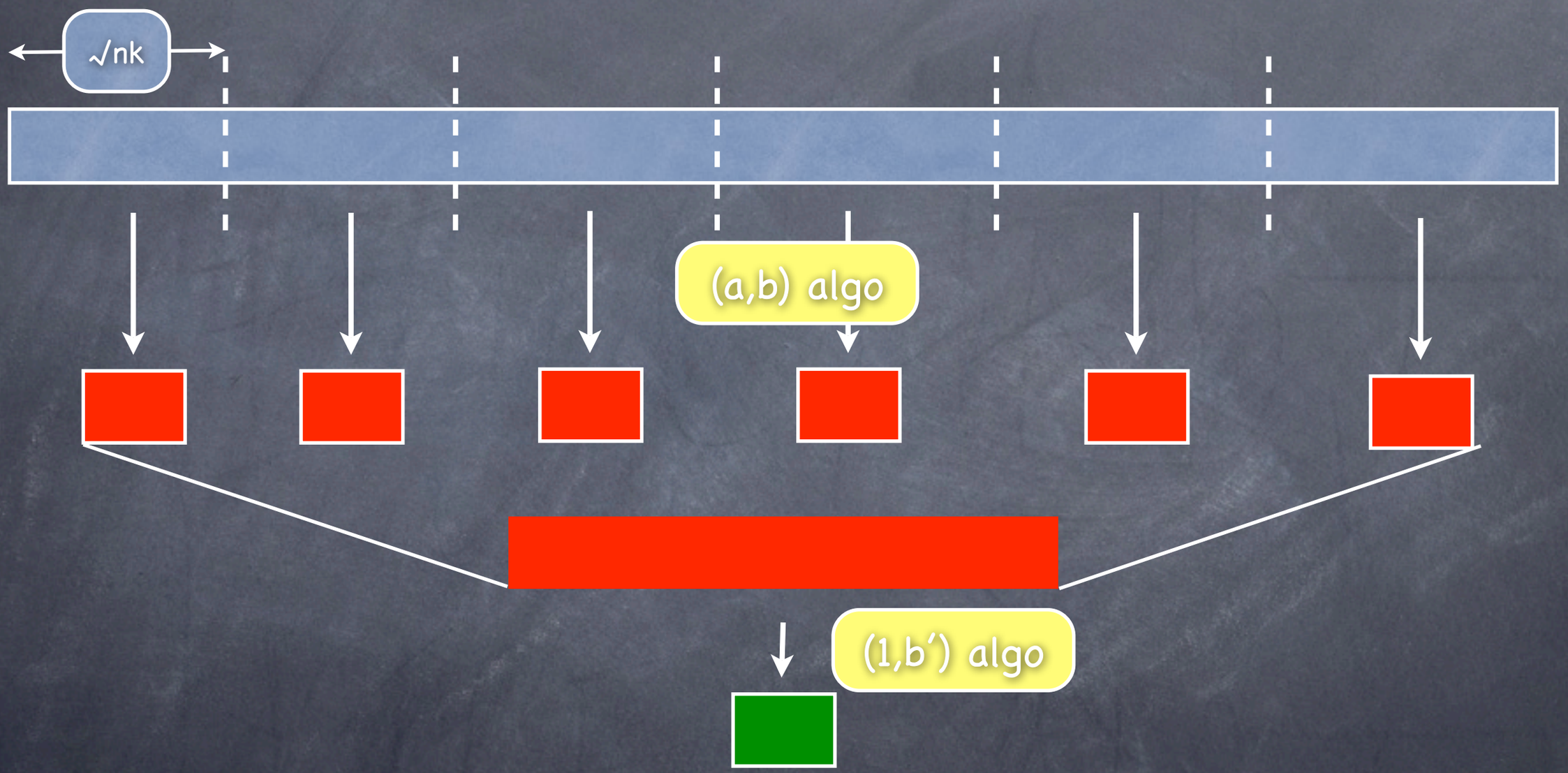
Better bounds using super martingales [ADK09]

- $E[J_i \mid J_0,\ldots,J_{i-1}] = E[J_{i-1}+X_i-(1-p) \mid J_0,\ldots,J_{i-1}]$
  $= E[J_{i-1} \mid J_0,\ldots,J_{i-1}] - (1-c) +$
  $E[X_i \mid J_0,\ldots,J_{i-1}]$
  $\leq J_{i-1}$
- So, $J_0,\ldots J_t$ is a super martingale and we can use the Azuma-Hoeffding bound which gives:
  $\Pr[J_t \geq J_0+\delta] \leq \exp(-\delta^2/2t)$
  $\Rightarrow \Pr[\Sigma_{i\leq t}\ (1-X_i) \geq ct-\delta] \geq 1 - \exp(-\delta^2/2t)$
  $\Rightarrow \Pr[\Sigma_{i\leq t}\ (1-X_i) \geq k] \geq 1 - \exp(-\text{const})$
  (using $t = (k+\sqrt{k})/c$ and $\delta=\sqrt{k}$)
- [ADK09]: With constant probability k-means## gives a (O(1), O(1)) pseudo-approximation.

Streaming Algorithms

- The data is very large compared to the memory available. So, we cannot hold the data in the memory while executing an algorithm.
- Allowed a few passes (preferably one) over the data.
- Performance measure:
  - Memory
  - Processing time per item
  - Compute time

Streaming Algorithm using Divide and Conquer [Guha+03]



[Guha+03]: The above approach gives O(bb') approximation with a√(nk) memory.

Streaming Algorithm using Divide and Conquer

- Approximation algorithms used in Divide & Conquer:
  - Level 0: k-means#: (O(log(k)), O(1))
  - Level 1: k-means++: (1, O(log(k)))
- [AJM09]: With one level divide and conquer, we get a one pass streaming algorithm with approximation ratio O(log(k)) and memory log(k)√(nk).

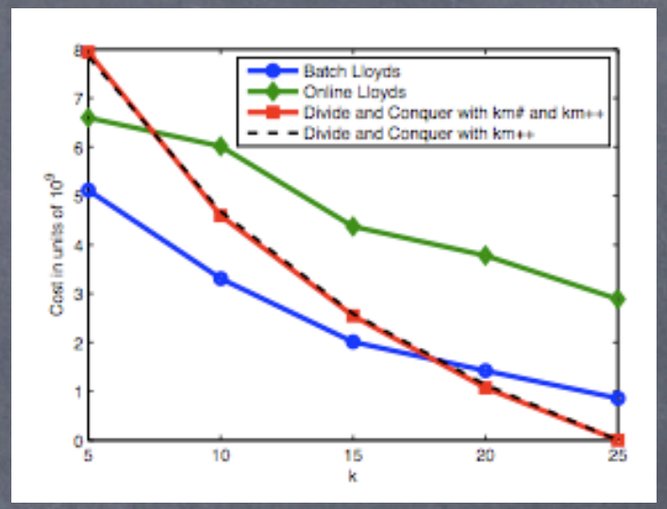
Experimental Results

(comparison with online/batch lloyd's)

k	BL	OL	DC-1	DC-2
5	$5.1154 \cdot 10^9$	$6.5967 \cdot 10^9$	$7.9398 \cdot 10^9$	$7.8474 \cdot 10^9$
10	$3.3080 \cdot 10^9$	$6.0146 \cdot 10^9$	$4.5954 \cdot 10^9$	$4.6829 \cdot 10^9$
15	$2.0123 \cdot 10^9$	$4.3743 \cdot 10^9$	$2.5468 \cdot 10^9$	$2.5898 \cdot 10^9$
20	$1.4225 \cdot 10^9$	$3.7794 \cdot 10^9$	$1.0718 \cdot 10^9$	$1.1403 \cdot 10^9$
25	$0.8602 \cdot 10^9$	$2.8859 \cdot 10^9$	$2.7842 \cdot 10^5$	$2.7298 \cdot 10^5$

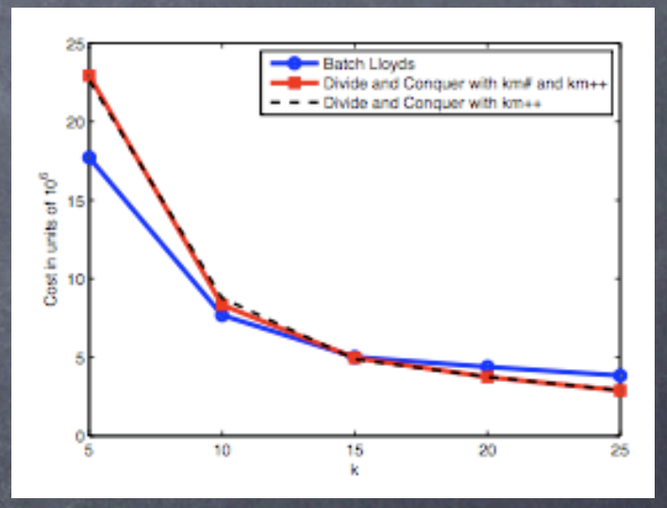
norm25, mixture of 25 separated gaussians

(10K points, 15 dimensions)



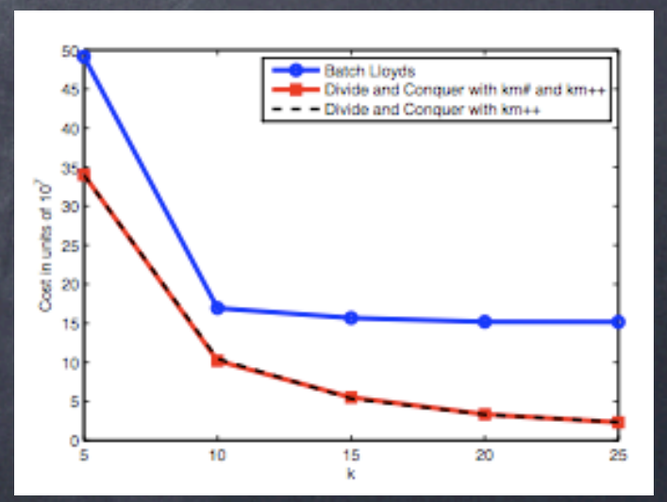
k	BL	OL	DC-1	DC-2
5	$1.7707 \cdot 10^7$	$1.2401 \cdot 10^8$	$2.2924 \cdot 10^7$	$2.2617 \cdot 10^7$
10	$0.7683 \cdot 10^7$	$8.5684 \cdot 10^7$	$8.3363 \cdot 10^6$	$8.7788 \cdot 10^6$
15	$0.5012 \cdot 10^7$	$8.4633 \cdot 10^7$	$4.9667 \cdot 10^6$	$4.8806 \cdot 10^6$
20	$0.4388 \cdot 10^7$	$6.5110 \cdot 10^7$	$3.7479 \cdot 10^6$	$3.7536 \cdot 10^6$
25	$0.3839 \cdot 10^7$	$6.3758 \cdot 10^7$	$2.8895 \cdot 10^6$	$2.9014 \cdot 10^6$

cloud (1024 points, 10 dimensions)

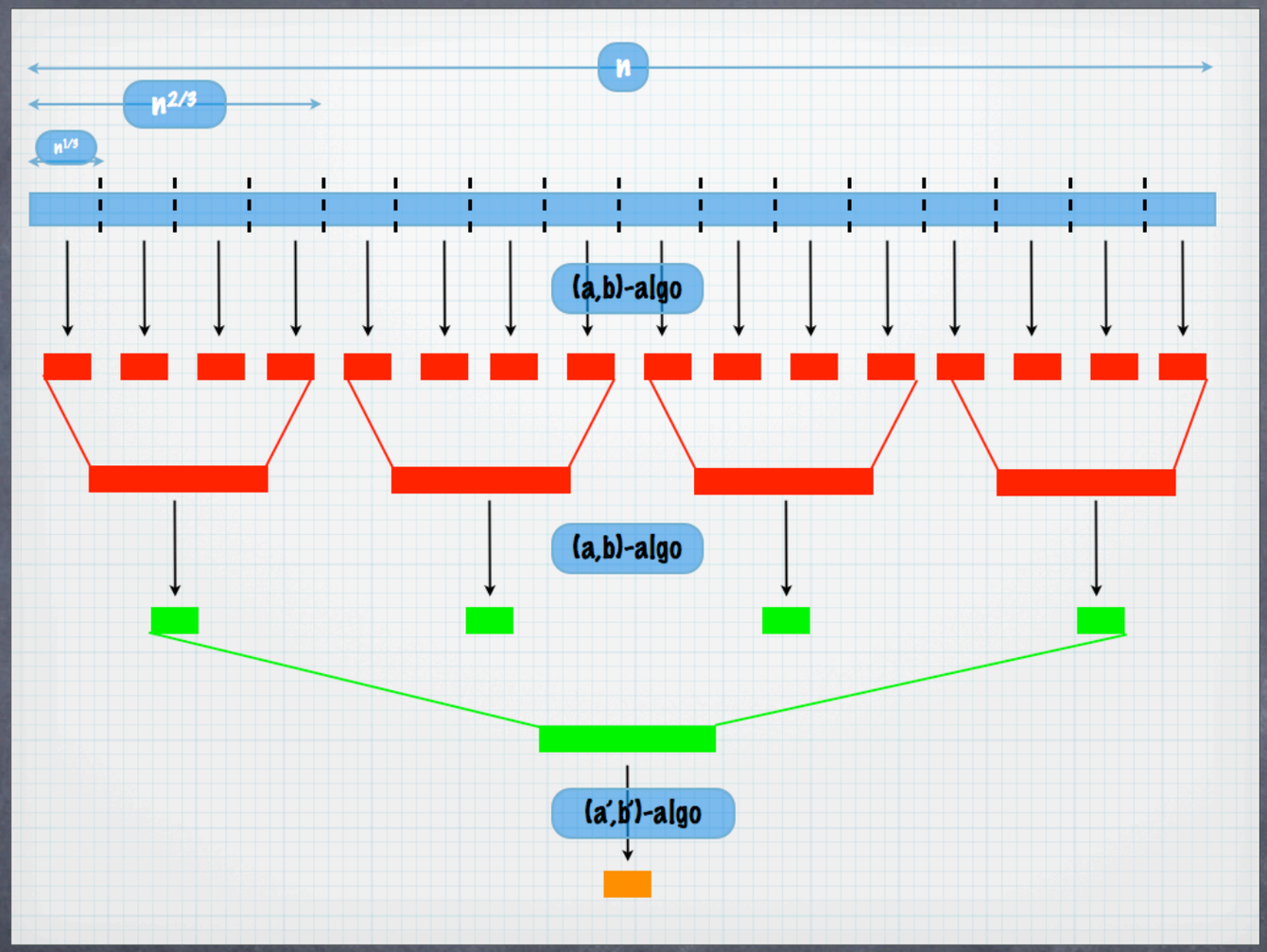


k	BL	OL	DC-1	DC-2
5	$4.9139 \cdot 10^8$	$1.7001 \cdot 10^9$	$3.4021 \cdot 10^8$	$3.3963 \cdot 10^8$
10	$1.6952 \cdot 10^8$	$1.6930 \cdot 10^9$	$1.0206 \cdot 10^8$	$1.0463 \cdot 10^8$
15	$1.5670 \cdot 10^8$	$1.4762 \cdot 10^9$	$5.5095 \cdot 10^7$	$5.3557 \cdot 10^7$
20	$1.5196 \cdot 10^8$	$1.4766 \cdot 10^9$	$3.3400 \cdot 10^7$	$3.2994 \cdot 10^7$
25	$1.5168 \cdot 10^8$	$1.4754 \cdot 10^9$	$2.3151 \cdot 10^7$	$2.3391 \cdot 10^7$

spam (4601 points, 58 dimensions)



Streaming Algorithm using Divide and Conquer (multi-level)



k-means#

k-means#

k-means++

- [AJM09]: For any fixed constant $1>\alpha>0$, let $r=1/\alpha$. With multi-level divide and conquer we get a $O(c^r \log(k))$ approximation algorithm with n^α memory.
- [CCP03]: They give a constant approximation algorithm which uses $O(k.poly(\log(n)))$ space.
  - Disadvantage: Per item processing time is large.

Experimental Results

(memory/approximation tradeoff)

- Use multi-level divide and conquer with k-means# in all levels except the last one where k-means++ is used.

Memory/ #levels	Cost	Memory/ #levels	Cost	Memory/ #levels	Cost
1024/0	8.74×10^6	2048/0	5.78×10^4	4601/0	1.06×10^8
480/1	8.59×10^6	1250/1	5.36×10^4	880/1	0.99×10^8
360/2	8.61×10^6	1125/2	5.15×10^4	600/2	1.03×10^8

Cloud
(k=10, n=1024)

norm25
(k=25, n=2048)

Spam
(k=10, n=4601)

- Approximation factor does not degrade as memory goes down which is counterintuitive.

Future Directions

- Further experimentation
- Lower Bounds
- Online Algorithms

Future Directions

- k-means###:
  - Step 1: Pick the first point randomly.
  - Step i (1<i≤k): Pick r centers independently by picking a point with probability proportional to the square of the distance of the point from its nearest (i-1) previously chosen centers and then choose the one which reduces the potential most.
- [AV07]: Empirical evidence of performance (r=const)
- Question: Show a constant approximation guarantee.

Questions?

Thank You