HPC in small packages: Viability of Small Machines facing Big Data, a study in leveraging high-expressive data helps improve iterative algorithms.

Mehmet M. Dalkılıç
Dr. Hasan Kurban, Dr. Mark Jenne, Parichit Sharma
Indiana University, Bloomington, IN, USA

NIH NATIONAL CANCER INSTITUTE

Outline

I. Context

- Data Explosion and consequences

II. Data-Driven Problem Statement (Clustering)

- Using *k*-means and EM (iterative)
- Low Expression (LE), High Expression (HE)
  - HE affects the outcome the most

III. Solution

- Separate LE, HE, using the HE as much as possible, ignoring LE as much as possible
- Implement DCEM R package (Data Clustering using EM*)

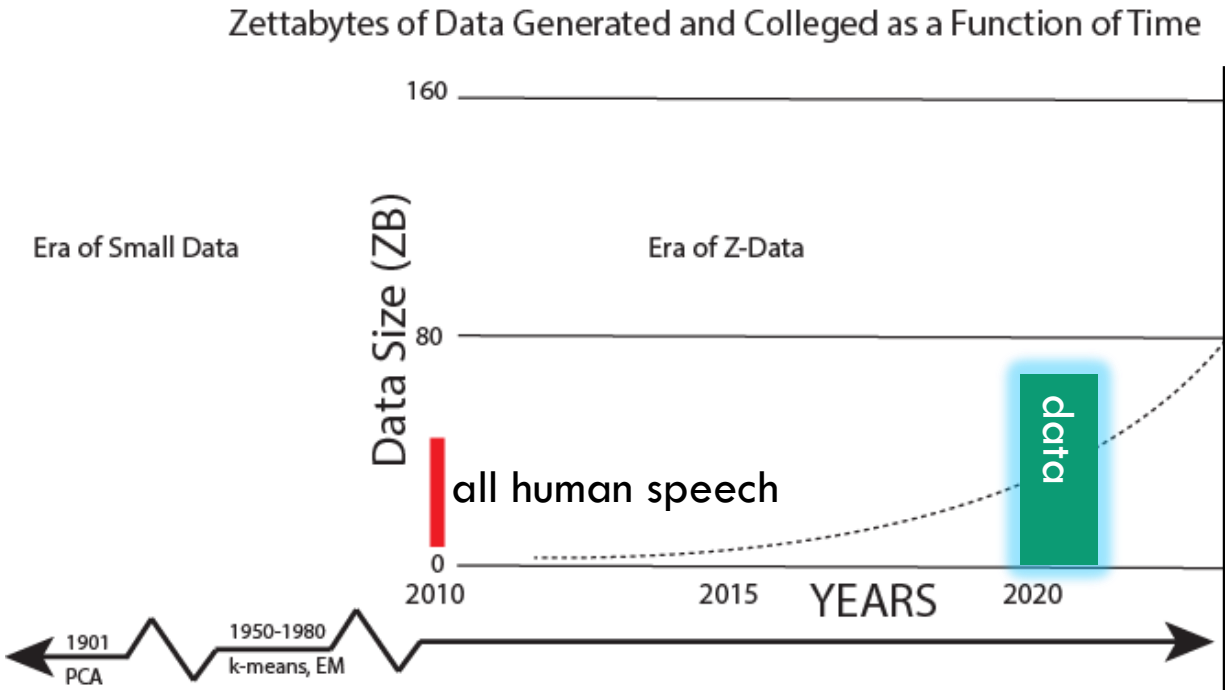
IV. Results

- EM* work on big data
- Case Studies

V. Future Work

- Parallelize, Distributed algorithm (Extend DCEM)

I. Context: Perspective—it's all about the data

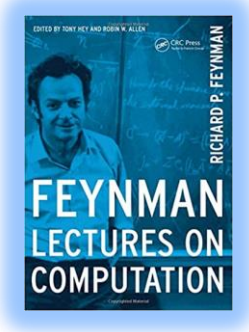


Multiples of bytes							
Decimal			Binary				
Value	Metric		Value	IEC		JEDEC	
1	B	byte	1	B	byte	B	byte
1000	kB	kilobyte	1024	KiB	kibibyte	KB	kilobyte
1000^2	MB	megabyte	1024^2	MiB	mebibyte	MB	megabyte
1000^3	GB	gigabyte	1024^3	GiB	gibibyte	GB	gigabyte
1000^4	TB	terabyte	1024^4	TiB	tebibyte		–
1000^5	PB	petabyte	1024^5	PiB	pebibyte		–
1000^6	EB	exabyte	1024^6	EiB	exbibyte		–
1000^7	ZB	**zettabyte**	1024^7	ZiB	zebibyte		–
1000^8	YB	yottabyte	1024^8	YiB	yobibyte		–

Orders of magnitude of data

wikipedia

$$\lim_{t\to\infty} \frac{Computing_t(\text{Data})}{Movement_t(\text{Data})} = 0$$

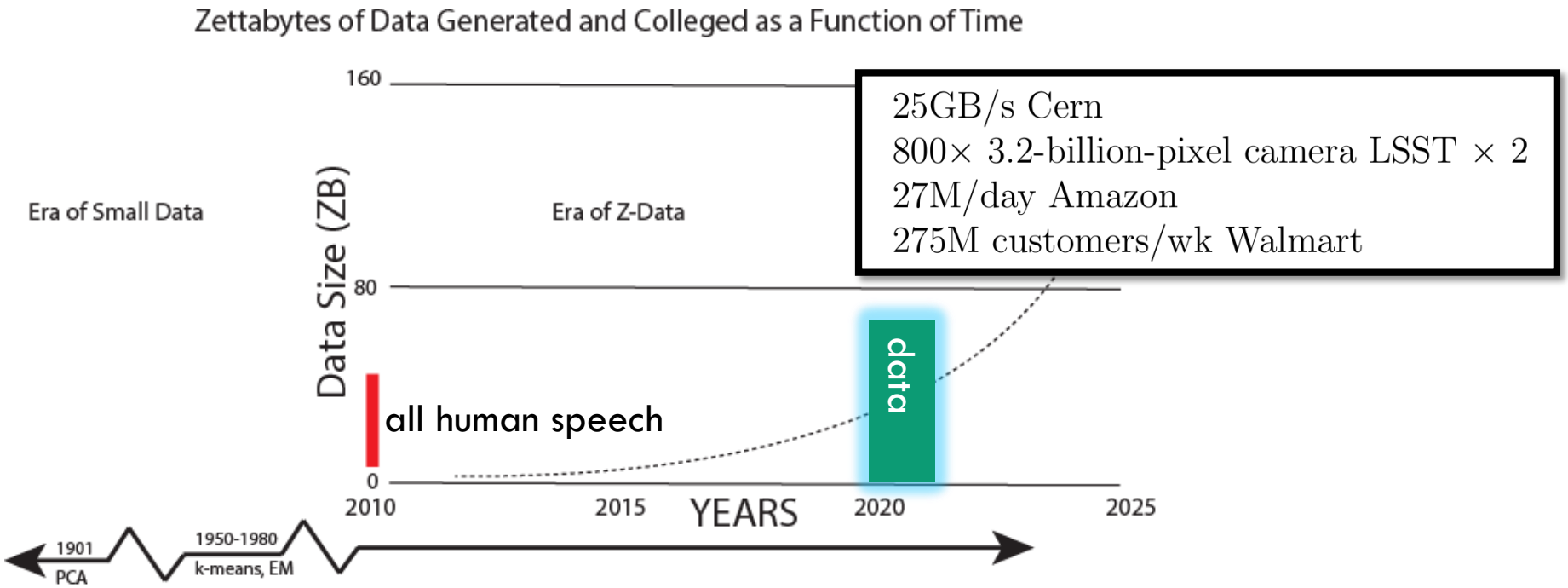


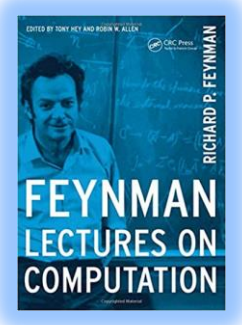
6
The Computing Machines in the Future

Richard P. Feynman

Bașarim 2020

I. Context: Perspective—it's all about the data

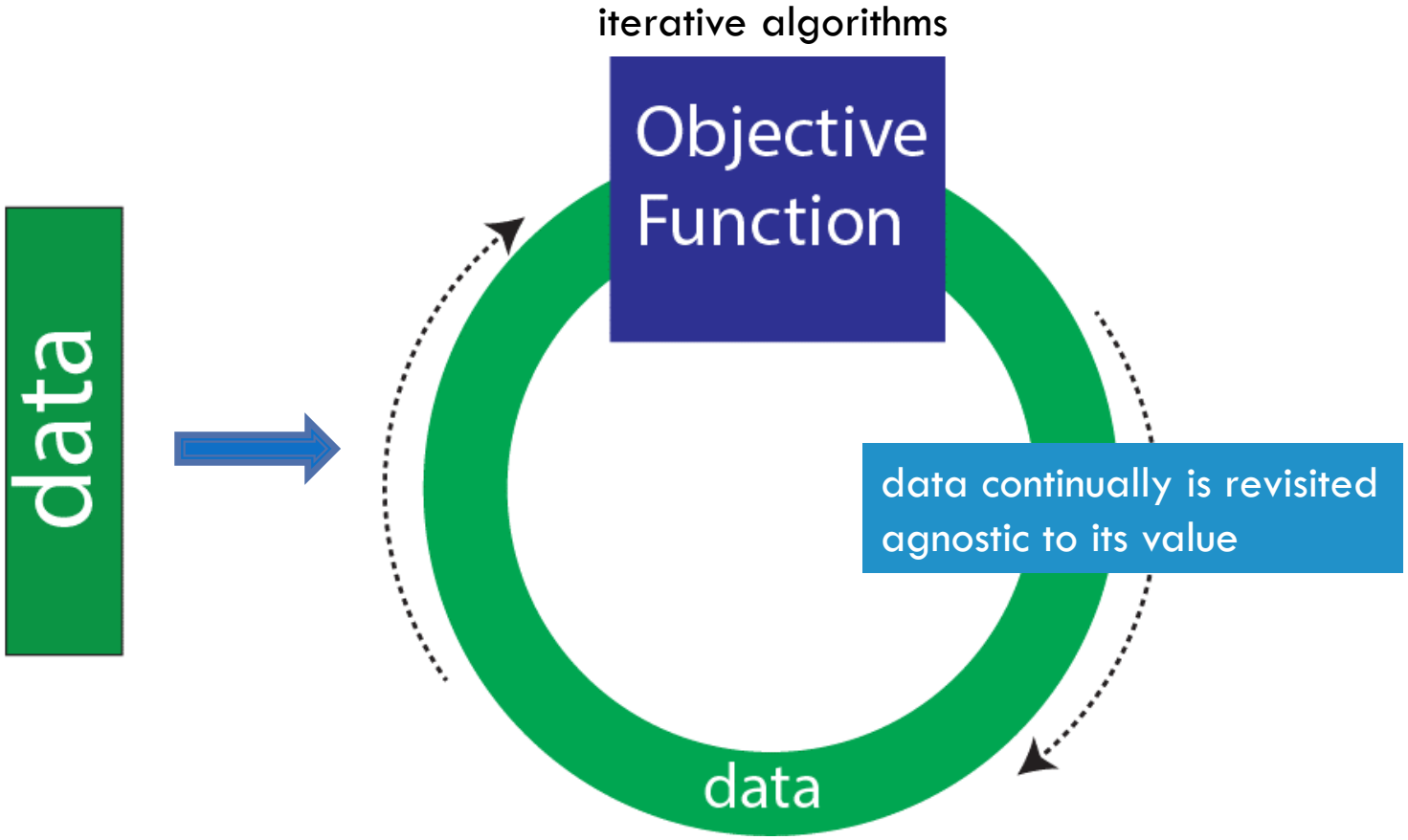




$$\lim_{t\to\infty} \frac{Computing_t(\text{Data})}{Movement_t(\text{Data})} = 0$$

Başarim 2020

I. Context: Perspective—it's all about the data



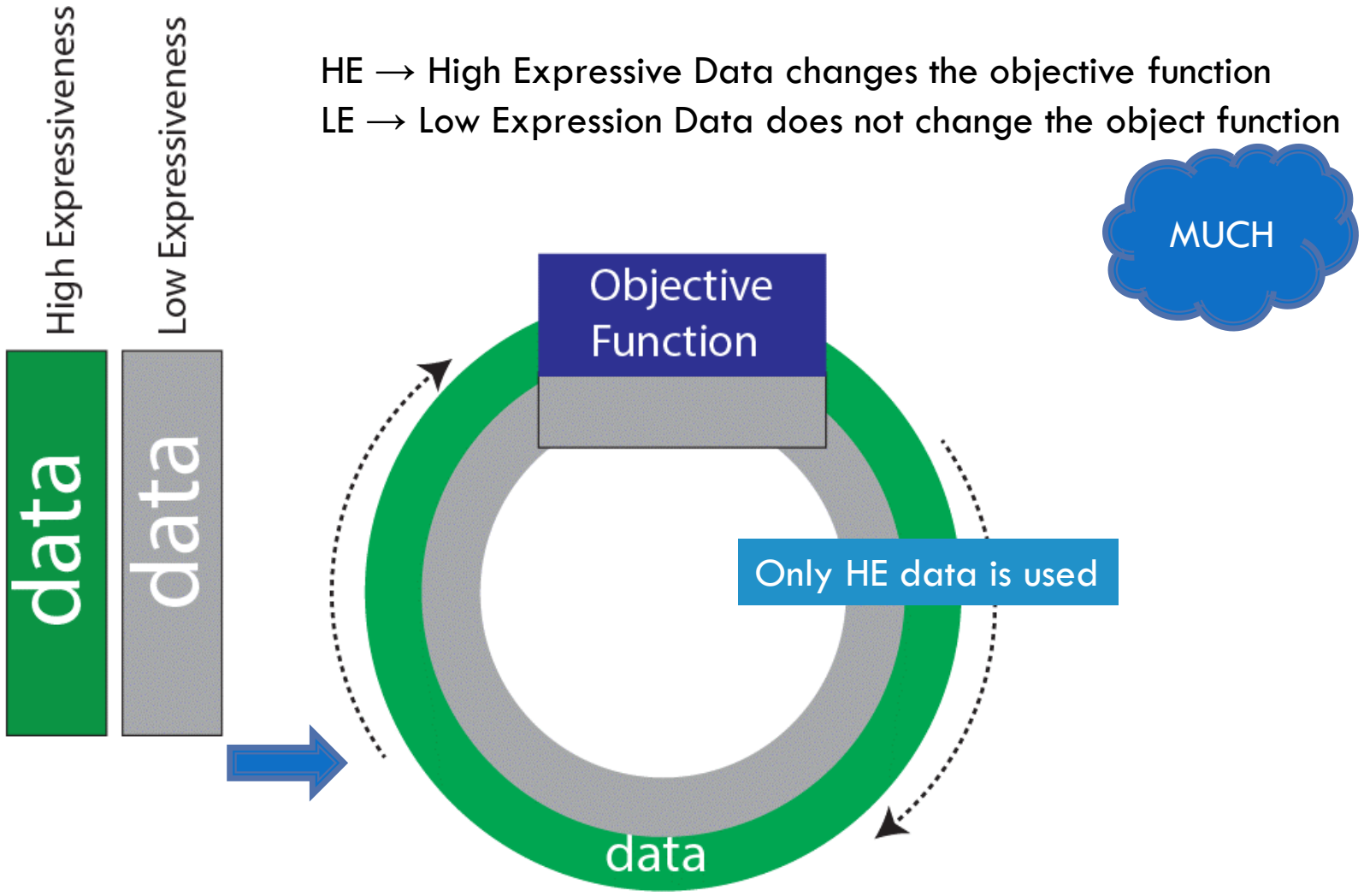
$$\lim_{t\to\infty} \frac{Computing_t(\text{Data})}{Movement_t(\text{Data})} = 0$$

Bașarim 2020

I. **Context:** Perspective—it's all about the data

HE → High Expressive Data changes the objective function

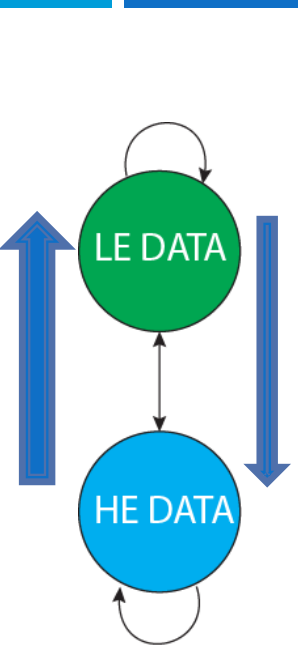
LE → Low Expression Data does not change the object function

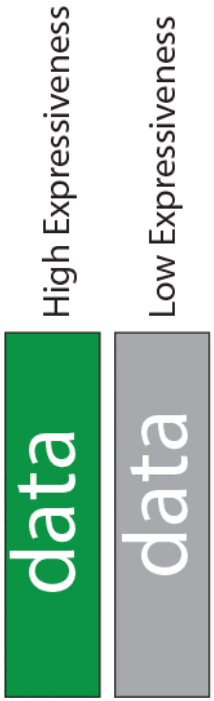


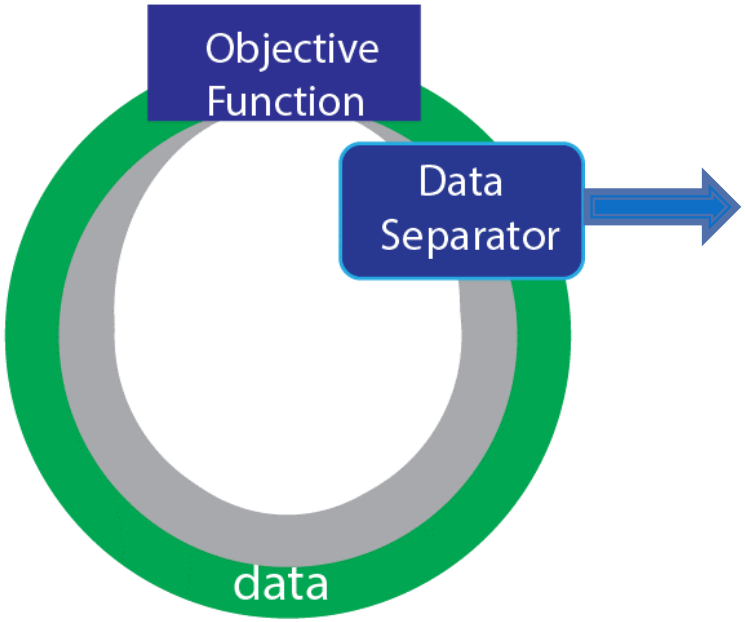
I. Context: Perspective—it's all about the data

HE → High Expressive Data changes the objective function

LE → Low Expression Data does not change the object function





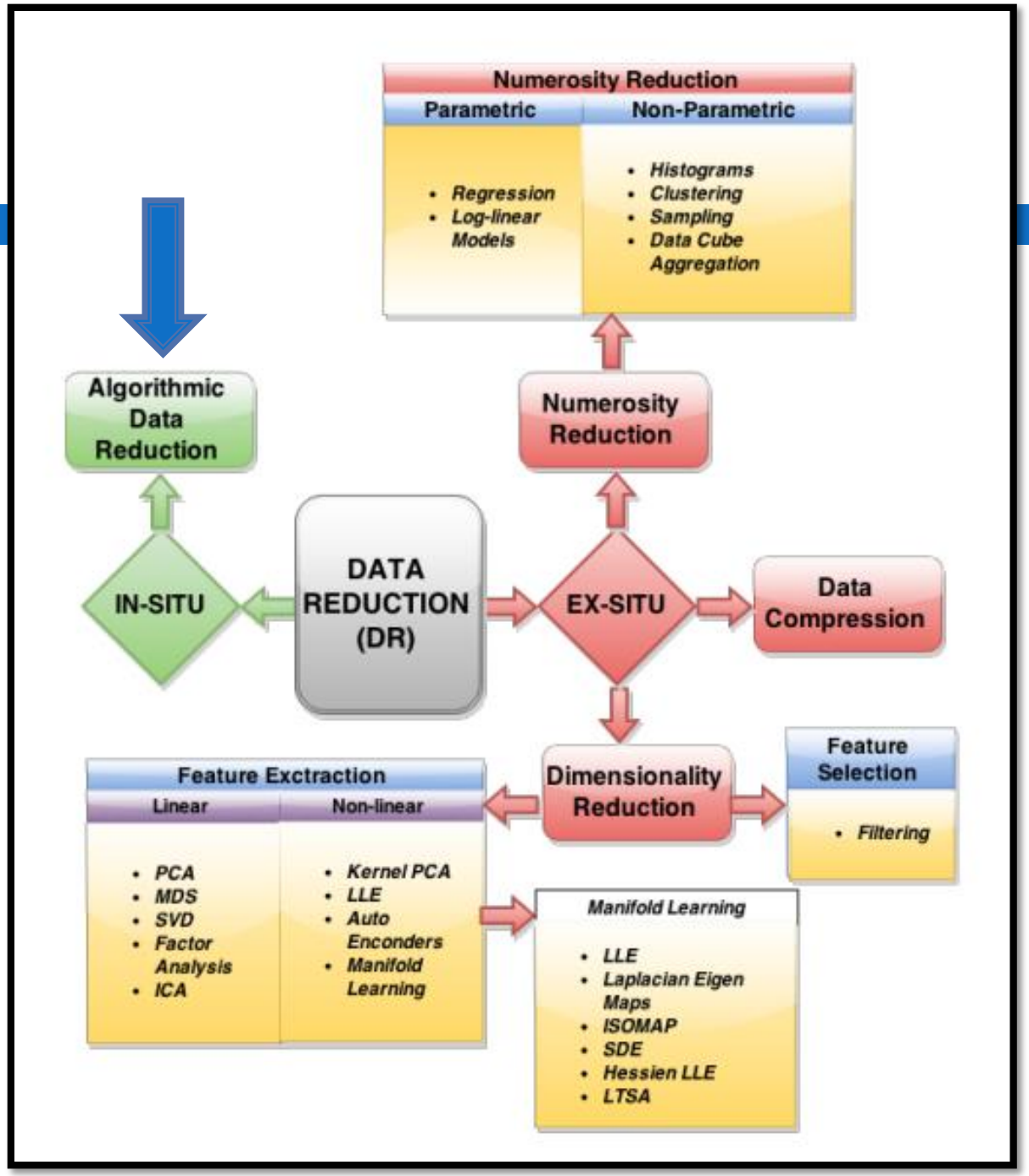


efficient actual "structure"

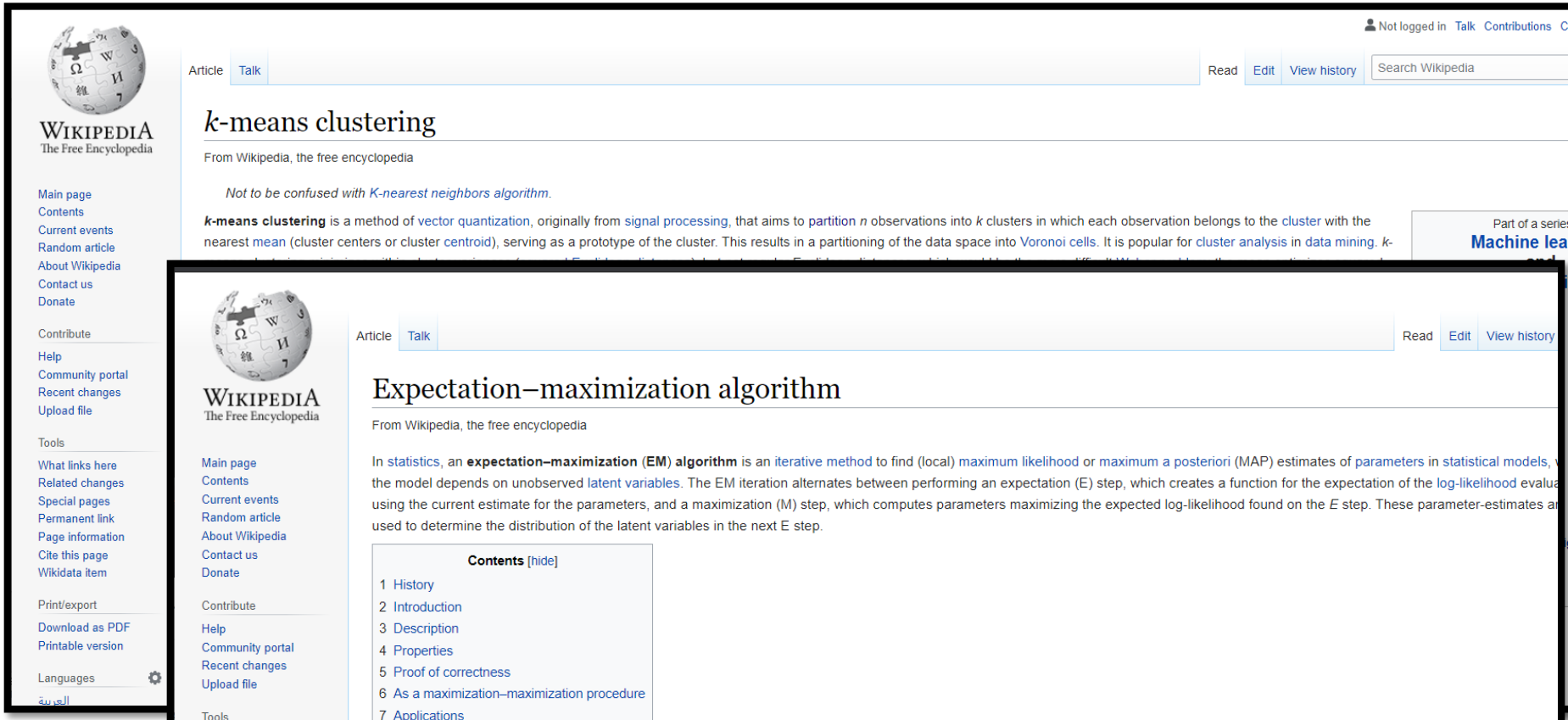
convergence on "structure"

MUCH

A taxonomy of data reduction techniques

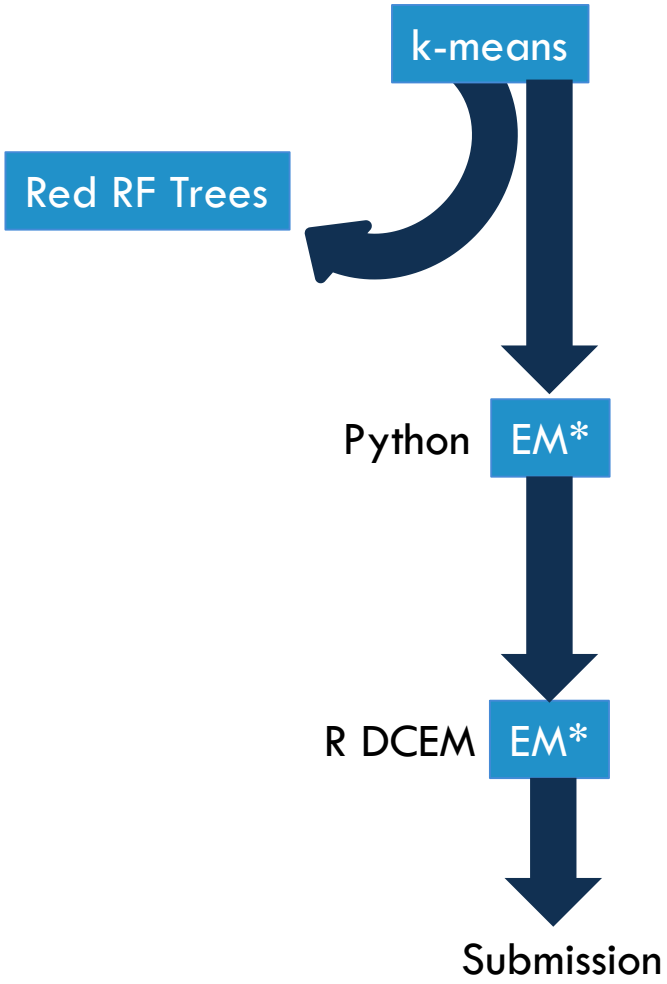


I. Wikipedia pages…



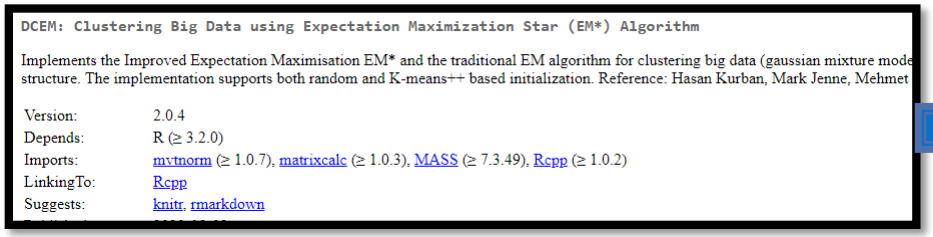
I. Related Publications

- Hussein Mohsen, Hasan Kurban, Mark Jenne, Mehmet Dalkilic: A new set of Random Forests with varying dynamic data reduction and voting techniques, IEEE International Conference on Data Science and Advanced Analytics (DSAA'14), Shanghai, China, 2014
- Mark Jenne, Owen Boberg, Hasan Kurban and Mehmet M. Dalkilic: Studying the Milky Way Galaxy using ParaHeap-k, a parallel heap-based k-means, Computer, vol.47, no.9, pp.26-33, 2014
- Hasan Kurban, Mark Jenne and Mehmet M. Dalkilic: EM*: An EM algorithm for Big Data, IEEE International Conference on Data Science and Advanced Analytics (DSAA'16), Montreal, Canada, 2016 (received "Honorable Mention Paper Award", best paper awards)
- Hussein Mohsen, Hasan Kurban, Kurt Zimmer, Mark Jenne, Mehmet Dalkilic: Red-RF: Reduced Random Forest for big data using priority voting & dynamic data reduction, International Congress on Big Data, IEEE BigData Congress 2015, New York, USA, 2015



I. Related Publications/Products

- Hasan Kurban, Mark Jenne and Mehmet M. Dalkilic: Using data to build a better EM: EM* for big data, International Journal of Data Science and Analytics (JDSA), pp.1-15, 2017
- H. Kurban, M.M. Dalkilic: A novel approach to optimization of iterative machine learning algorithms: Over heap structure, 2017 IEEE International Conference on Big Data (Big Data), Boston, MA, 2017, pp. 102-109, doi: 10.1109/BigData.2017.8257917
- Hasan Kurban, Can Kockan, Mark Jenne and Mehmet M. Dalkilic: Case study: clustering big stellar data with EM*, IEEE/ACM International Conference on Big Data Computing, Applications and Technologies (BDCAT 2017)



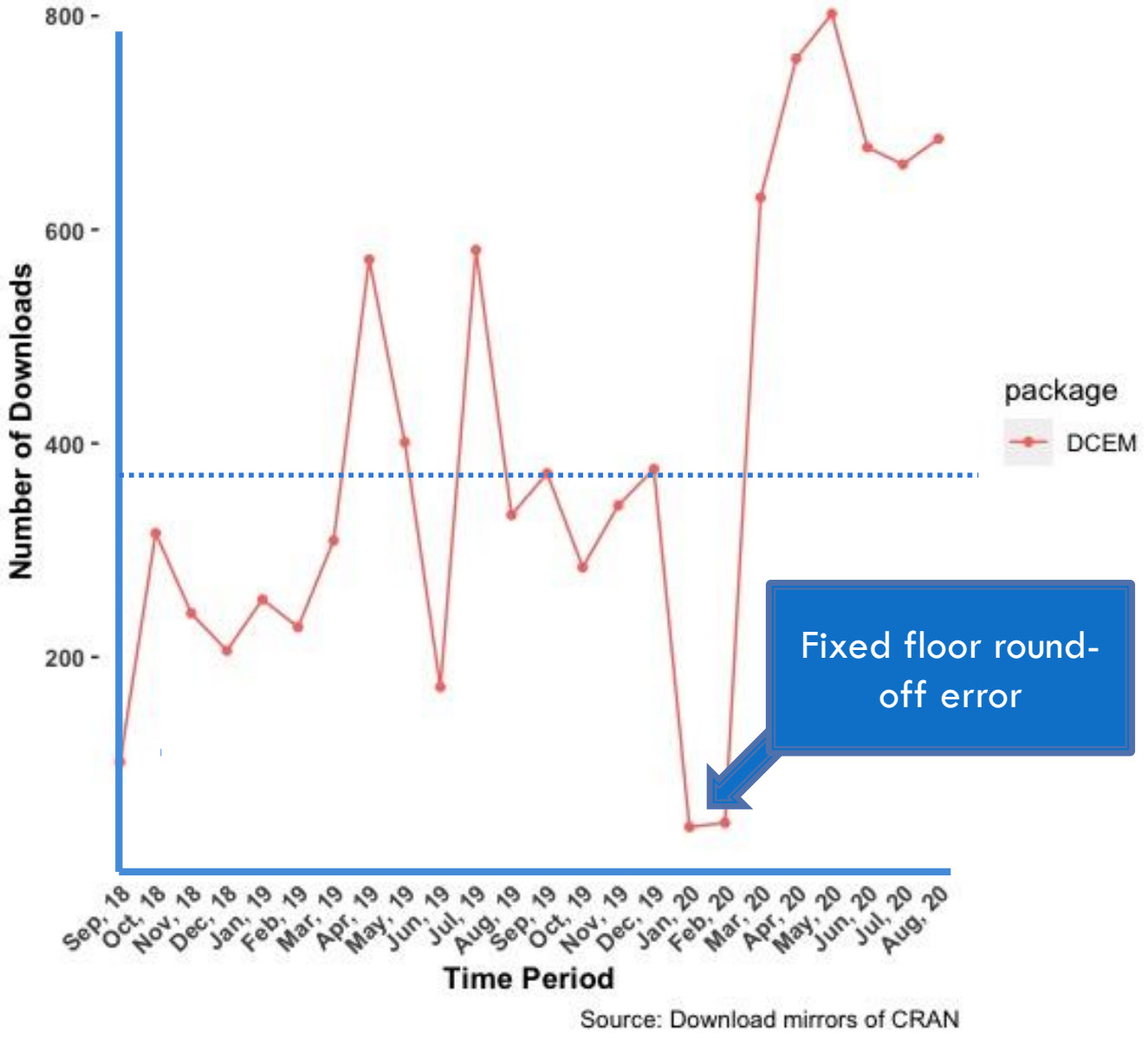
DCEM: Clustering Big Data using Expectation Maximization Star (EM*) Algorithm

Implements the Improved Expectation Maximisation EM* and the traditional EM algorithm for clustering big data (gaussian mixture mode structure. The implementation supports both random and K-means++ based initialization. Reference: Hasan Kurban, Mark Jenne, Mehmet

Version:	2.0.4
Depends:	R (≥ 3.2.0)
Imports:	mvtnorm (≥ 1.0.7), matrixcalc (≥ 1.0.3), MASS (≥ 7.3.49), Rcpp (≥ 1.0.2)
LinkingTo:	Rcpp
Suggests:	knitr, rmarkdown

Under Journal Review

# Download statistics for DCEM since Release (2018 September)
Max = 800, Ave = 390, Min = 41



II. Data-Driven Problem Statement (Clustering)

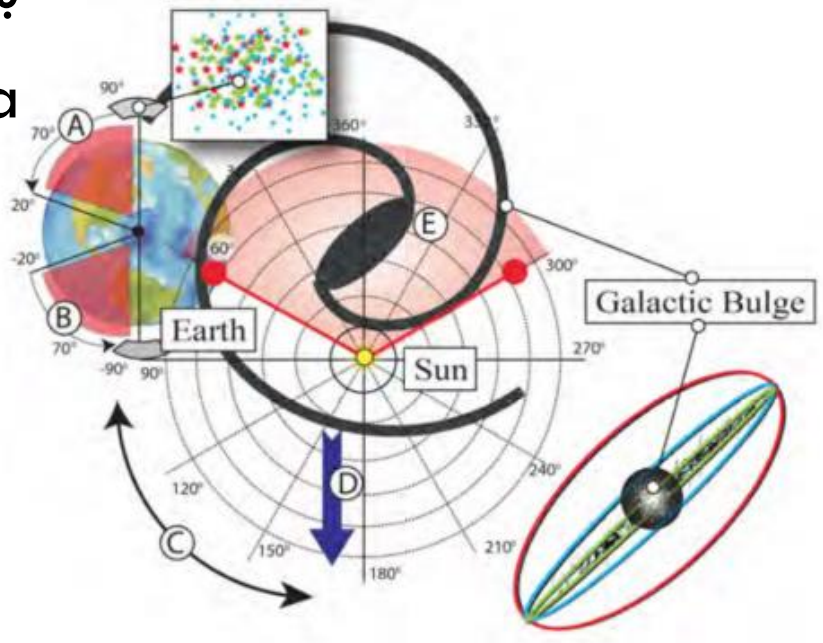
We conducted a series of experiments comparing

- KM* to KM
- EM* to EM
- Parallel implementations, ParaHeap-*k* to *k*-means++ and *k*-means||

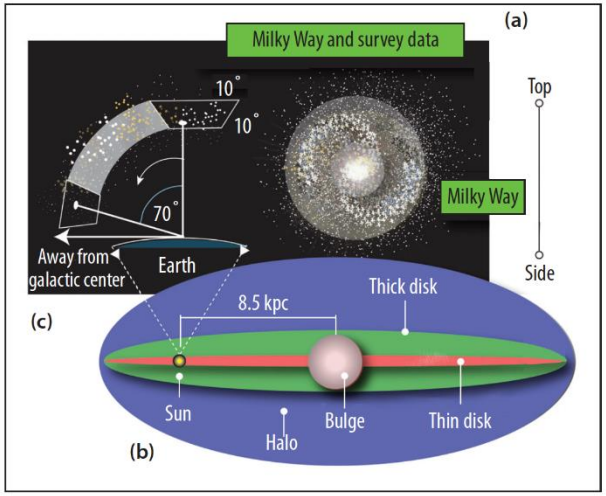
with both real world and synthetic data sets through different kinds of testing: scale, dimension, and separability.

II. Data-Driven Problem Statement (Clustering)

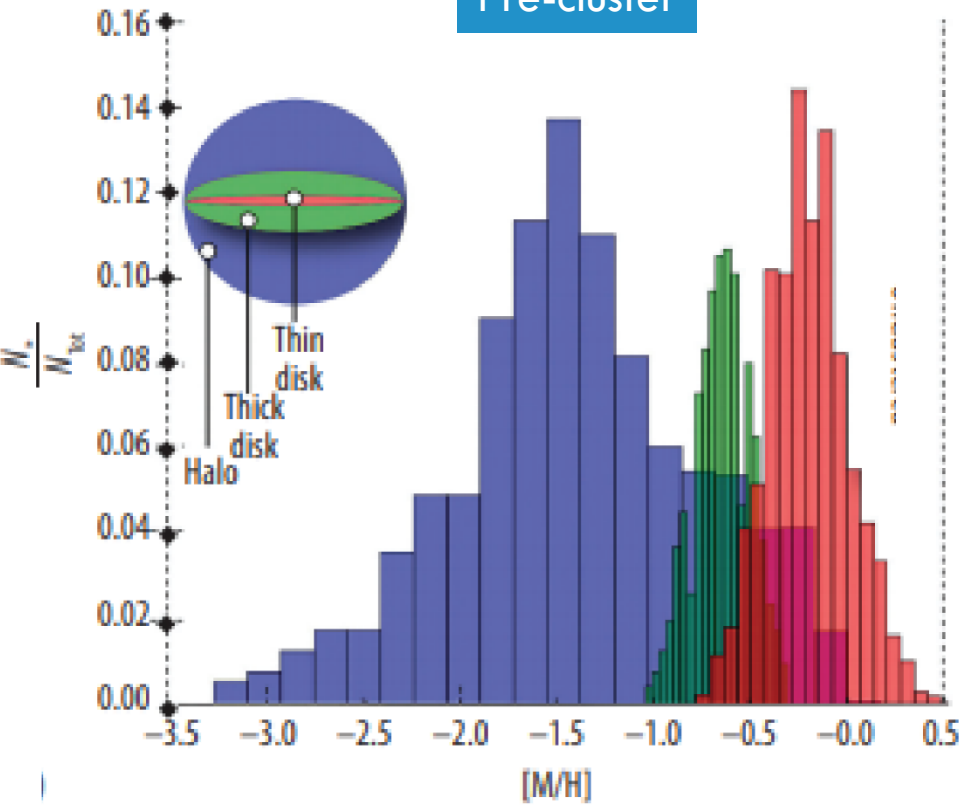
- 1st Call from IEEE Computer on Computational Astronomy
  - Question: Cluster the Milky Way?
  - Yes: Difficult because of the data
- Let's do it!
  - Each star is "given birth" from one of the galactic components
  - We can check the quality of the cluster using metallicity



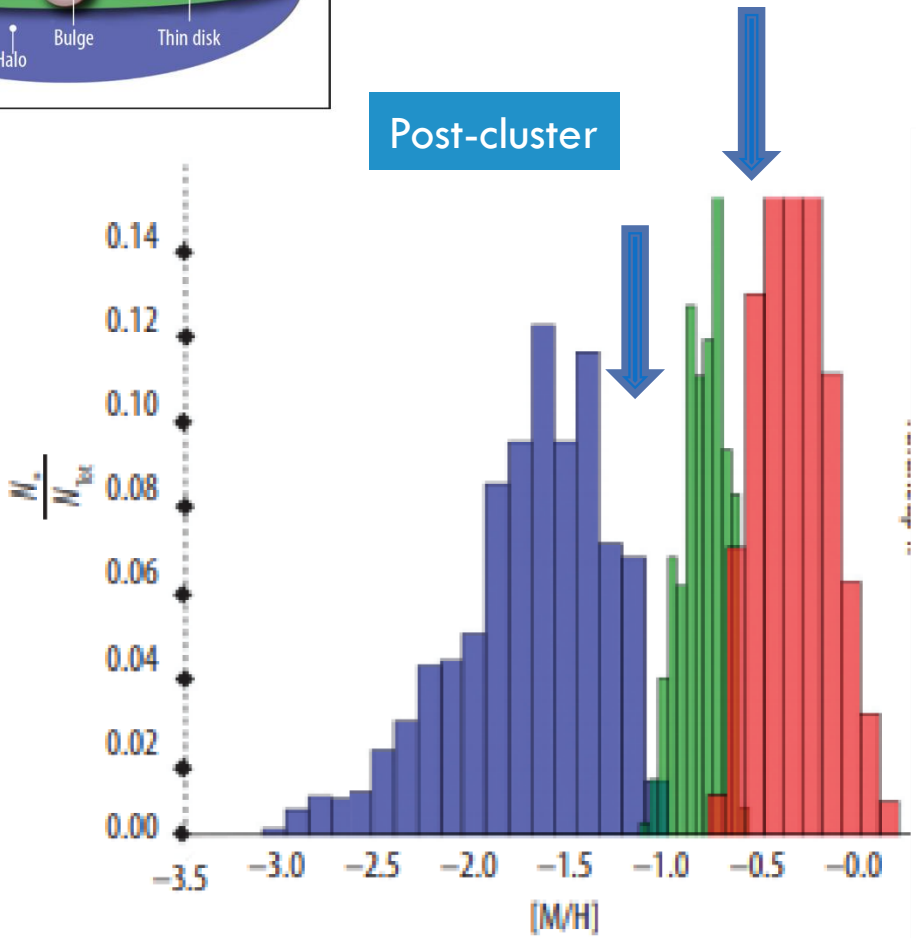
II. Data-Driven Problem Statement (Clustering)



Pre-cluster



Post-cluster



II. Data-Driven Problem Statement (Clustering)

- What, at that time, would be the best clustering
- Quick change in nomenclature

For a non-empty finite set X, a partition is $\{X_1, X_2, \ldots, X_n\}$ s.t.

- $\cup_{i=1}^{n} X_n = X$
- **for** $i, j = 1, 2, \ldots, n; i \neq j \rightarrow X_i \cap X_j = \emptyset$
- **for** $i = 1, 2, \ldots, n; X_i \neq \emptyset$
- **for** $i = 1, 2, \ldots, n; X_i \subseteq X$, X_i is called a block

- So the question becomes, can we partition the Milky Way so that each star belongs to its proper block (in this case, galactic component)

II. Data-Driven Problem Statement

- Can we materially improve *k*-means from a data-centric perspective to cluster big stellar data?
  - YES!
- Can this be generalized for other iterative machine learning algorithms?
  - YES!
    - Separating LE/HE data

II. Data-Driven Problem Statement

- *k*-means algorithm (60 yrs.) [hard]
  - Assign each datum to <u>one</u> block (Centroid ~ "Best Representative" ~ Ave)
  - Use simple Euclidean distance
  - Re-calculate Centroid until convergence

- Gaussian Mixture Model (50 yrs.) [soft]
  - Assign datum to <u>*every*</u> Gaussian (block) with a probability
  - Re-calculate properties of Gaussians
  - Iterate until convergence though it can be very slow (another ascent method)

II. Data-Driven Problem Statement (Clustering)

- We decided on *k*-means because of its general success and ubiquitous use and remains one of the most popular algorithms (Lloyd's algorithm)
  - Data Clustering: 50 Years Beyond K-Means [Jain] 2009
  - Initialization (seeding)
    - *k*-means++ [Arthur, Vassilvitskii] 2006, 2007
    - *k*-means|| (Scalable *k*-means) [Bahman *et al.*] 2012
    - kd-trees pre-filtering [Kanungu, *et al.*] 2000
    - kd-trees [Pelleg, Moore] 1999
  - Triangle Inequality + distance bounds
    - [Elkan] 2003 (Excellent paper for bibliography on speed-ups for *k*-means)
    - [Hamerly] 2010
    - [Drake] 2012
    - [Hamerly, Drake] 2014 use priority queues to prune points close to assigned centers; use heaps to store differences between lower and upper bounds of points
- Witness where it's used: ELKI, graphlab, Mahout, MATLAB, MLPACK, Octave, OpenCV, R (various), SciPy, Weka, Yael, *etc.* Başarim 2020

II. Data-Driven Problem Statement (Clustering)

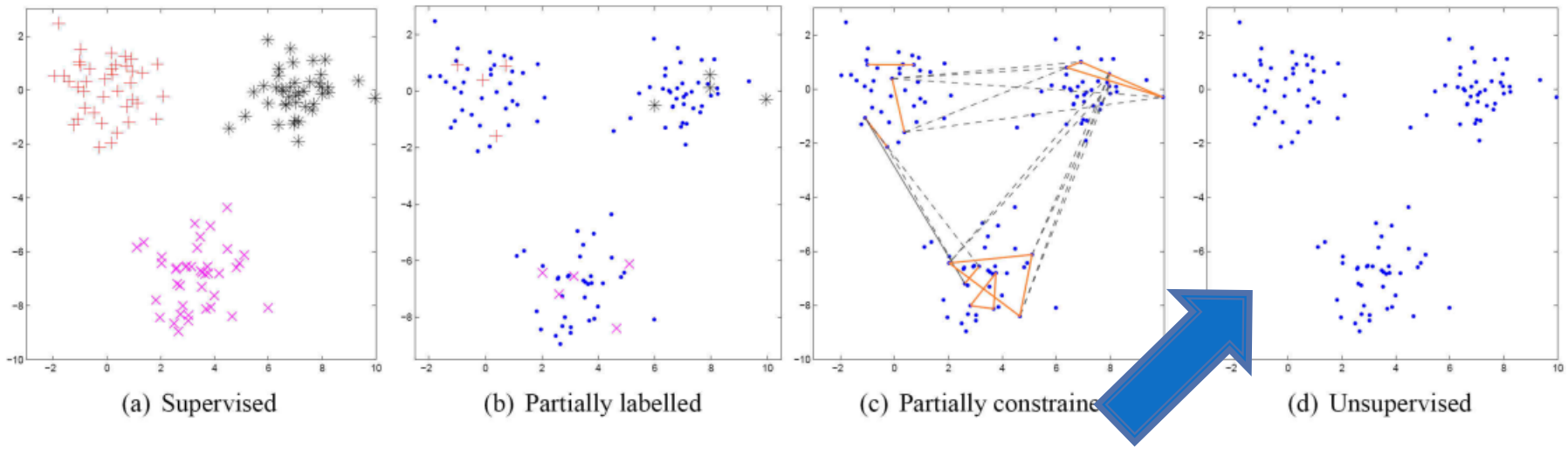


Figure 1 Learning problems: dots correspond to points without any labels. Points with labels are denoted by plus signs, asterisks, and crosses. In (c), the must-link and cannot-link constraints are denoted by solid and dashed lines, respectively (figure taken from [Lange et al., 2005]).

[Jain] 2009 *Pattern Recognition Letters*

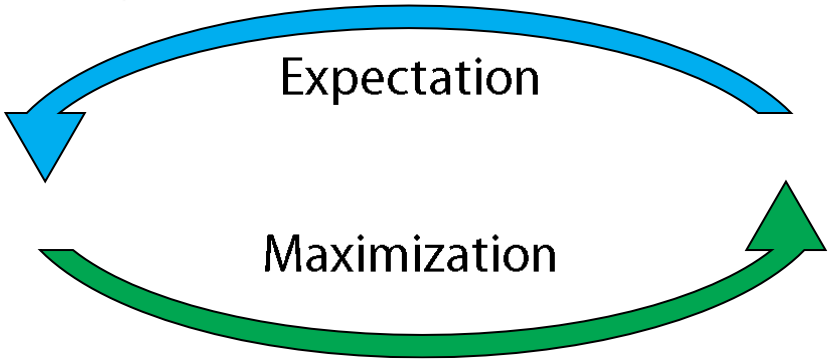
Similis simili gaudet

II. Data-Driven Problem Statement

Iteration continues until the set of centroids is stable; in other words, convergence is guaranteed in a finite number of steps by showing that for some non-negative error function, monotonically decreasing during each iteration

Algorithm 1: K-Means (KM) Algorithm

K-**Means**($\mathbf{D}$, k, ϵ):
$t \leftarrow 0$
// Set of Centroids $\mathbf{C}^t = \{C_1^t, \ldots, C_k^t\}$
// Assume Centroid is Class $C_i = (\bar{\mathbf{x}}_i, D_i)$, $D_i \subseteq D$
Randomly initialize $C_i.\bar{\mathbf{x}}_1, \ldots, C_k.\bar{\mathbf{x}}_k$
repeat
  $t \leftarrow t + 1$
  // Gathering (Expectation hard assignment)
  for $i = 1, \ldots, k$ *and* $j = 1, \ldots, n$ **do**
    $C_i^t.D_i \overset{\cup}{\leftarrow} \mathbf{x}_j$ s.t. $\min \|C_i^t.\bar{\mathbf{x}}_i - \mathbf{x}_j\|^2$
  end
  // Maximization Step
  for $i = 1, \ldots, k$ **do**
    $C_i^t.\bar{\mathbf{x}}_i \leftarrow \frac{\sum_{\mathbf{x} \in D_i} \mathbf{x}}{|D_i|}$ // re-estimate mean
  end
until $\sum_{i=1}^{k} \sum_{\mathbf{x} \in C_i} \|C_i^{t-1}.\bar{\mathbf{x}} - C_i^t.\bar{\mathbf{x}}\|^2 \leq \epsilon$



we always add an iterate i to ensure no loop if convergence is too slow

II. Data-Driven Problem Statement (Clustering)

- The run-time of *k*-means is $O(inkd)$ where

- i is the iteration

- n is data size

- k is the number of clusters

- d is the dimension of data, *i.e.*, $\mathbf{x} \in \mathcal{R}^d$

II. Data-Driven Problem Statement

- Each cluster $C_i = \{\mathbf{x}_{i1}, \ldots, \mathbf{x}_{i\ell}\}$ contributes to sum of pairwise dissimilarities to sum of total measure
- For cluster C_i, we can write total measure measure(i), where $|C_i.D_i| = \ell$ as:

$$J(i) = \frac{1}{2}\sum_{j=1}^{\ell}\sum_{l=1}^{\ell}\|\mathbf{x}_{ij} - \mathbf{x}_{il}\| \quad (1)$$

linear algebra

$$= \ell\sum_{j=1}^{\ell}\|\mathbf{x}_{ij} - C_i.\bar{\mathbf{x}}_i\| \quad (2)$$

II. Data-Driven Problem Statement

Cost: J is minimized monotonically

$$J = \sum_{i=1}^{k} \sum_{\mathbf{x} \in C_i} \min ||\mathbf{x} - C_i.\bar{\mathbf{x}}||^2$$

$$J = \sum_{i=1}^{k} \sum_{\mathbf{x} \in C_i.D^{LE}} \min ||\mathbf{x} - C_i.\bar{\mathbf{x}}||^2 + \sum_{i=1}^{k} \sum_{\mathbf{x} \in C_i.D^{HE}} \min ||\mathbf{x} - C_i.\bar{\mathbf{x}}||^2$$

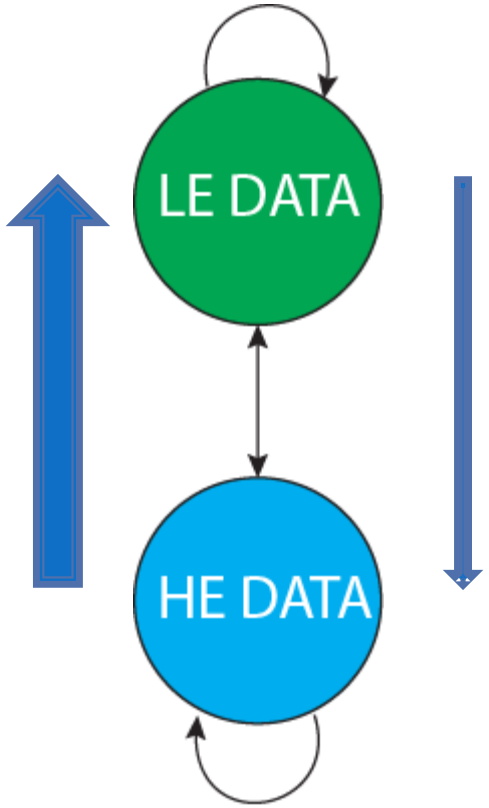
$$J_{OPT} = \sum_{i=1}^{k} \sum_{\mathbf{x} \in C_i.D^{HE}} \min ||\mathbf{x} - C_i.\bar{\mathbf{x}}||^2$$

II. Data-Driven Problem Statement

- **Can we materially improve *k*-means from a data-centric perspective to partition big stellar data?**
- **Can this be generalized for other iterative machine learning algorithms?**
- *YES!*
  - Observing that data, in the original algorithm, is visited continually no matter its affect on the algorithm i, n
  - **Data as it is used *changes*** in its "expressiveness"—this notion is that the affect of data is dependent on when it's used.

II. Data-Driven Problem Statement

- Low Expression (LE) and High Expression (HE) data affect the outcome minimally and significantly, respectively
- LE and HE can switch
- There is a tendency as *k*-means converges that most of the data becomes LE
- Can we capture this difference?



II. Failures to Capture LE and HE

- Transparency—there were many failures to capture this notion—which made us doubt, maybe, this was a sound approach
  - Entropy failed
  - Itemset (level set) failed
  - Triangulation failed—bounds
  - Eureka—heaps! Now we needed to re-examine heaps

II. Taste of Using LE,HE vs. *k*-||

	data set		$k = 1000$	$k = 5000$	$k = 10000$
ParaHeap-k	910K GS1	init time (s)	0	0	0
		total run time (s)	66	242	354
		cluster error	0.097	0.089	0.084
	95M GS2	init time (s)	0	0	0
		total run time (s)	2540	6226	11785
		cluster error	0.096	0.102	0.098
	data set		$k = 1000$	$k = 5000$	$k = 10000$
***k*-means||**	910K GS1	init time (s)	46	158	313
		total run time (s)	69	214	412
		cluster error	0.084	0.081	0.077
	95M GS2	init time (s)	6934	**failed**	**failed**
		total run time (s)	8713		
		cluster error	0.092		

II. HPC (system architecture)

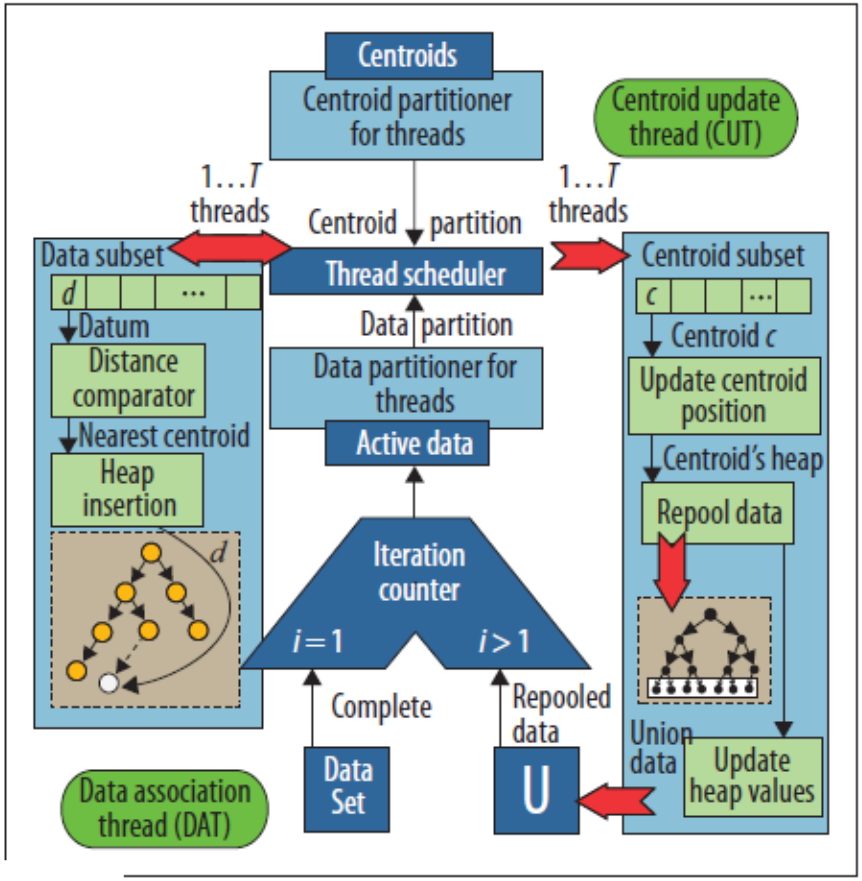
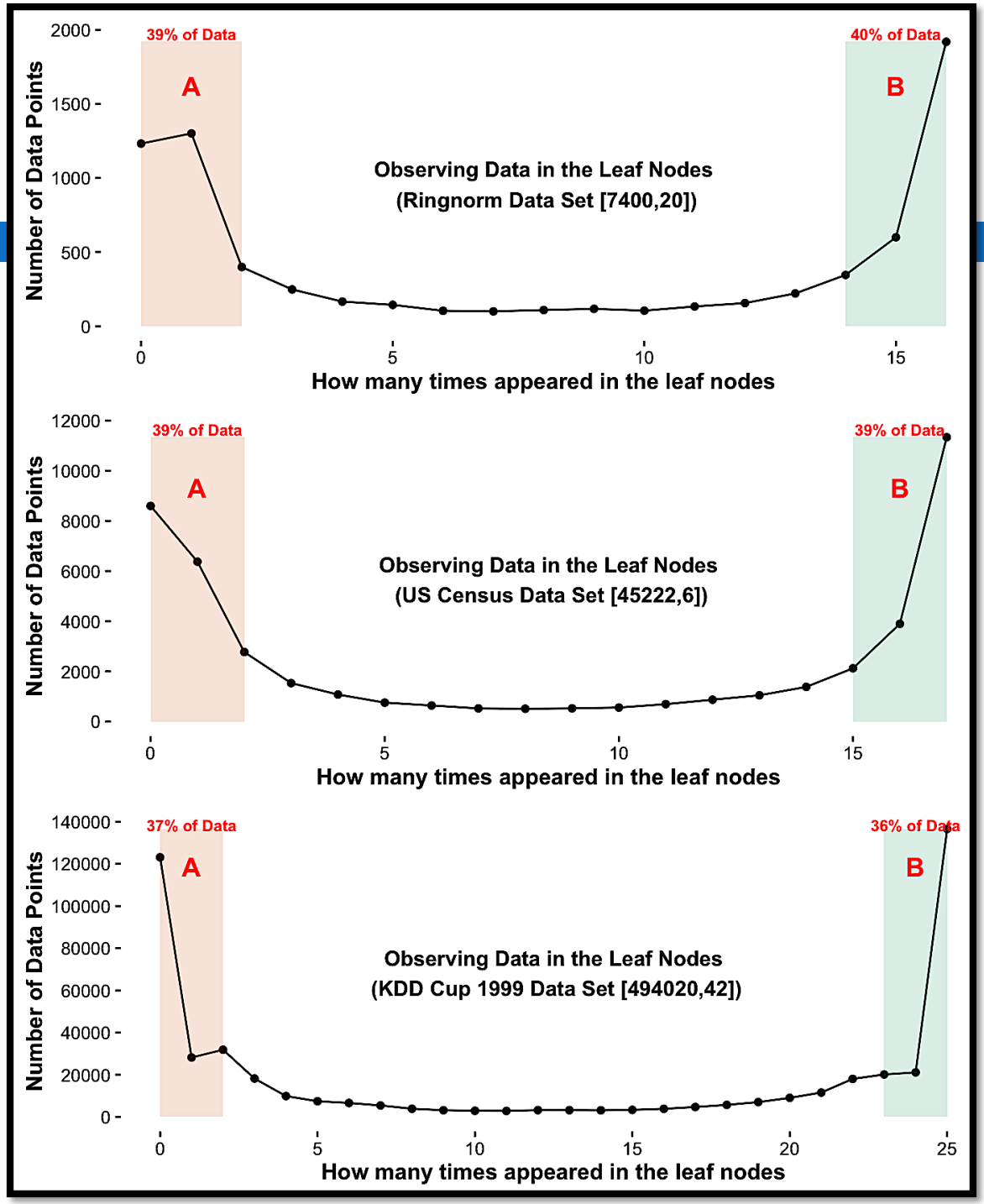


Figure 2. An overview of the system architecture highlighting the heap data structures and parallel implementation for a single ParaHeap-*k* iteration in its entirety. Each iteration includes data association threads (DATs) and centroid update threads (CUTs).


39% of Data
A
40% of Data
B
Observing Data in the Leaf Nodes
(Ringnorm Data Set [7400,20])
Number of Data Points
How many times appeared in the leaf nodes
39% of Data
A
39% of Data
B
Observing Data in the Leaf Nodes
(US Census Data Set [45222,6])
Number of Data Points
How many times appeared in the leaf nodes
37% of Data
A
36% of Data
B
Observing Data in the Leaf Nodes
(KDD Cup 1999 Data Set [494020,42])
Number of Data Points
How many times appeared in the leaf nodes

II. Expectation Maximization (EM)

- Like *k*-means lots of work done (mostly by statisticians) to improve movement
- Unlike *k*-means (hard assignment), EM is soft assignment—each datum belongs to typically a Gaussian.
- Used everywhere like *k*-means
- Expectation-maximization algorithm as a clustering algorithm(EM-T): Each block can be characterized as a probability distribution and the goal is to iteratively find the MLE of unknown parameters of blocks.

II. Expectation Maximization (EM)

Algorithm 2: Expectation-Maximization (EM) Algorithm

Expectation-Maximization($\mathbf{D}$, k, ϵ):

$t \leftarrow 0$

Randomly initialize $\boldsymbol{\mu}_1^t, \ldots, \boldsymbol{\mu}_k^t$

$\boldsymbol{\Sigma}_i^t \leftarrow \mathbf{I}, \ \forall i = 1, \ldots, k$

$P^t(C_i) \leftarrow \frac{1}{k}, \forall i = 1, \ldots, k$

repeat

  $t \leftarrow t + 1$

  // Expectation Step

  for $i = 1, \ldots, k$ *and* $j = 1, \ldots, n$ **do**

    $w_{ij} \leftarrow \frac{f(\mathbf{x}_j|\boldsymbol{\mu}_i\boldsymbol{\Sigma}_i)\cdot P(C_i)}{\sum_{a=1}^{k} f(\mathbf{x}_j|\boldsymbol{\mu}_a,\boldsymbol{\Sigma}_a)\cdot P(C_a)}$ // Posterior probability $P^t(C_i|\mathbf{x}_j)$

  end

  // Maximization Step

  for $i = 1, \ldots, k$ **do**

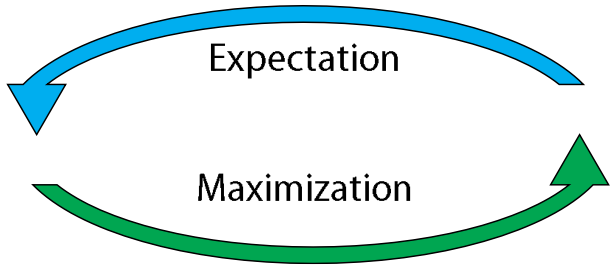
    $\boldsymbol{\mu}_i^t \leftarrow \frac{\sum_{j=1}^{n} w_{ij}\cdot\mathbf{x}_j}{\sum_{j=1}^{n} w_{ij}}$ // re-estimate mean

    $\boldsymbol{\Sigma}_i^t \leftarrow \frac{\sum_{j=1}^{n} w_{ij}(\mathbf{x}_j-\boldsymbol{\mu}_i)(\mathbf{x}_j-\boldsymbol{\mu}_i)^\mathsf{T}}{\sum_{j=1}^{n} w_{ij}}$ // re-estimate covariance matrix

    $P^t(C_i) \leftarrow \frac{\sum_{j=1}^{n} w_{ij}}{n}$ // re-estimate priors

  end

until $\sum_{i=1}^{k} \|\boldsymbol{\mu}_i^t - \boldsymbol{\mu}_i^{t-1}\|^2 \leq \epsilon$



II. Expectation Maximization (EM)

EM-T general run-time: $O(ik(d^3 + nd^2)$; best run-time $O(inkd)$

E-step invert $\boldsymbol{\Sigma}$; compute $|\boldsymbol{\Sigma}_i|$; $O(d^3) \xrightarrow{k} O(kd^3)$

evaluate density; $O(d^2) \xrightarrow{k,n} O(knd^2)$

M-step update $\boldsymbol{\Sigma} \xrightarrow{k} O(knd^2)$

Use D EM-T

Use D^{HE} EM*

$$
\begin{aligned}
\mathbf{D} &= \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}, \mathbf{x}_i \in \mathcal{R}^d \\
C_i^t &= (\boldsymbol{\mu}_i \in \mathcal{R}^d, \boldsymbol{\Sigma}_i \in \mathcal{R}^{d \times d}, Pr(C_i) \in [0,1], C_i.D_i \subseteq \mathbf{D}) \\
C_i &\sim \mathsf{N}(C_i.\boldsymbol{\mu}, C_i.\boldsymbol{\Sigma}) \\
f_i(\mathbf{x}|C_i) &= (2\pi)^{-\frac{d}{2}} \det(\boldsymbol{\Sigma})^{-\frac{1}{2}} e^{-\frac{1}{2}(\mathbf{x}-\bar{\mathbf{x}})^\mathsf{T} \boldsymbol{\Sigma}^{-1} (\mathbf{x}-\bar{\mathbf{x}})}
\end{aligned}
$$

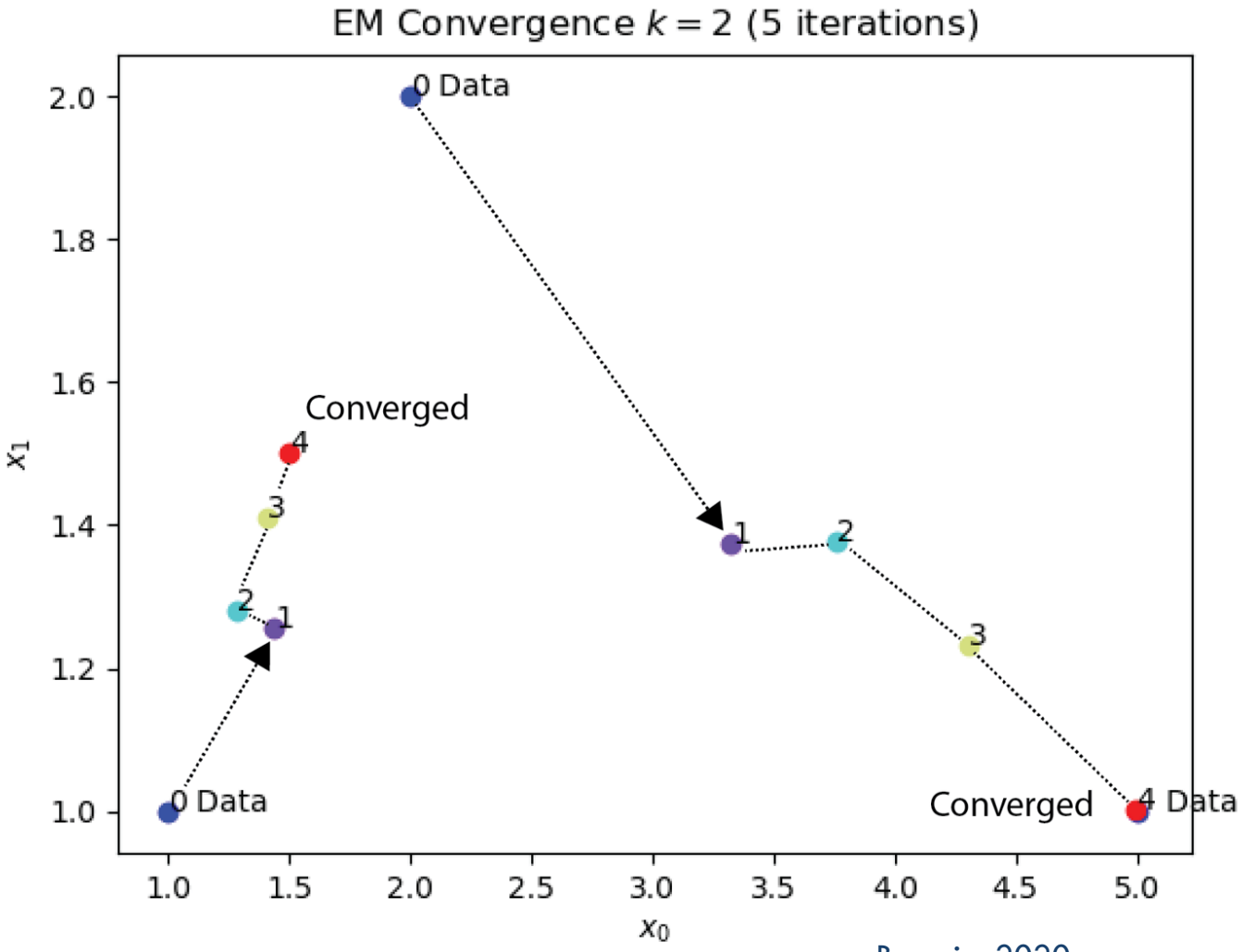
Maximum Likelihood Estimate (Calculus and Linear Algebra):

$\hat{\mathbf{x}} = \frac{1}{n} \sum_i \mathbf{x}_i$

$\hat{\boldsymbol{\Sigma}} = \Sigma_i \, (\mathbf{x}_i - \hat{\boldsymbol{\mu}})^\mathsf{T} (\mathbf{x}_i - \hat{\boldsymbol{\mu}})$

Başarim 2020

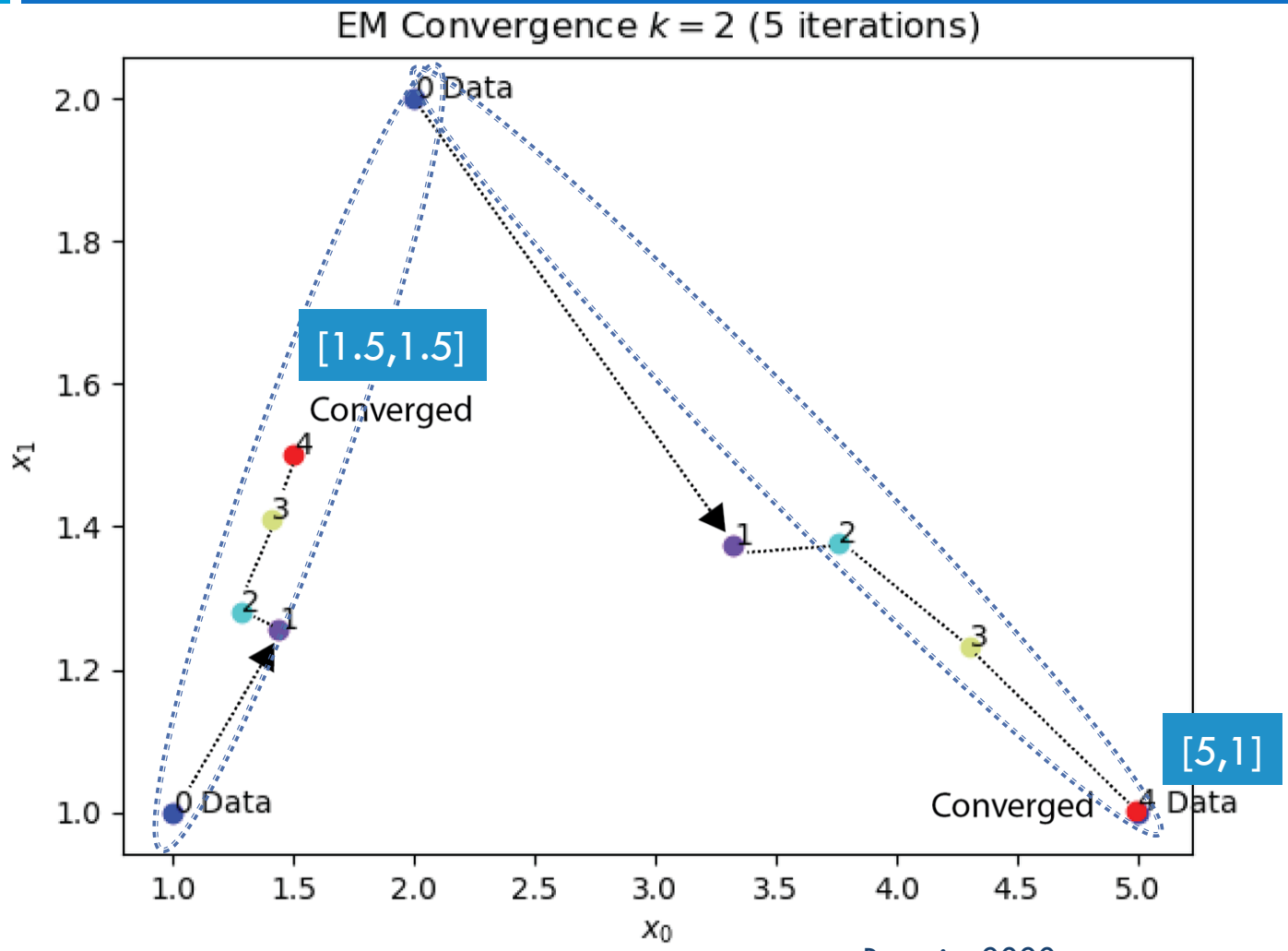
II. Expectation Maximization (EM)



Data

1,1
2,2
5,1

II. Expectation Maximization (EM)

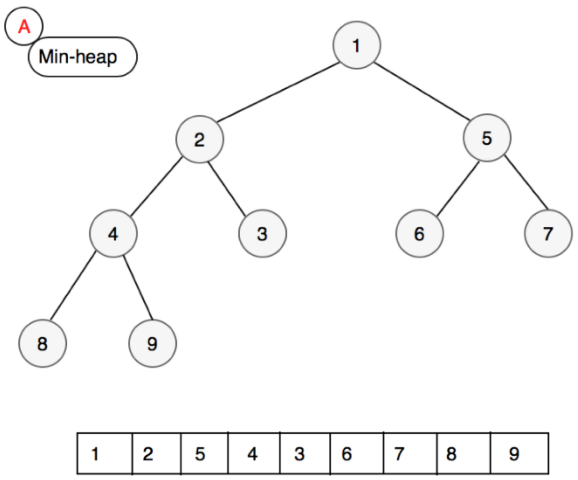


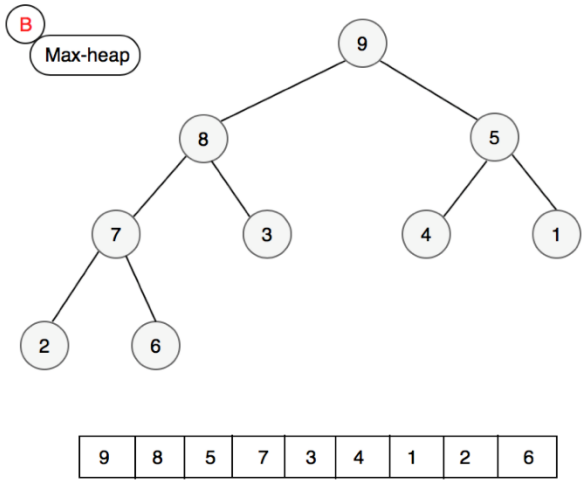
Data

1,1
2,2
5,1

III. Solution

- Heaps have excellent computational properties
- Use these to store and separate LE, HE
- A complete/nearly complete balanced binary tree with the "heap" property





Binary (min) heap

Type	binary tree/heap
Invented	1964
Invented by	J. W. J. Williams

Time complexity in big O notation

Algorithm	Average	Worst case
Space	O(n)	O(n)
Search	O(n)	O(n)
Insert	O(1)	O(log n)
Find-min	O(1)	O(1)
Delete-min	O(log n)	O(log n)

wikipedia

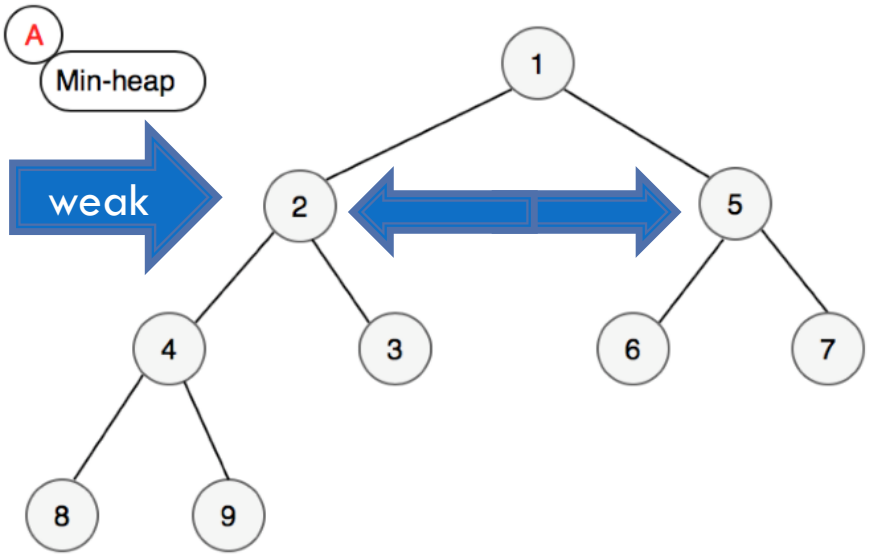
III. Solution

- Intuition was LE resides at top, HE resides at bottom of heap (min for *k*-means, max for EM) (investigating Laplacian)
- Why should LE remain at the top if it's LE?
- We identified a new property of heaps that we call *strong* and *weak*
  - Strong means from the root to some level ℓ each level can be permuted, and the heap property remains true.
  - Weak means from some level ℓ below the root to the leaves, one or more levels cannot be permuted without violating the heap property
  - Strong heaps are "semi"-monotonic—tends to want to be monotonic

III. Solution

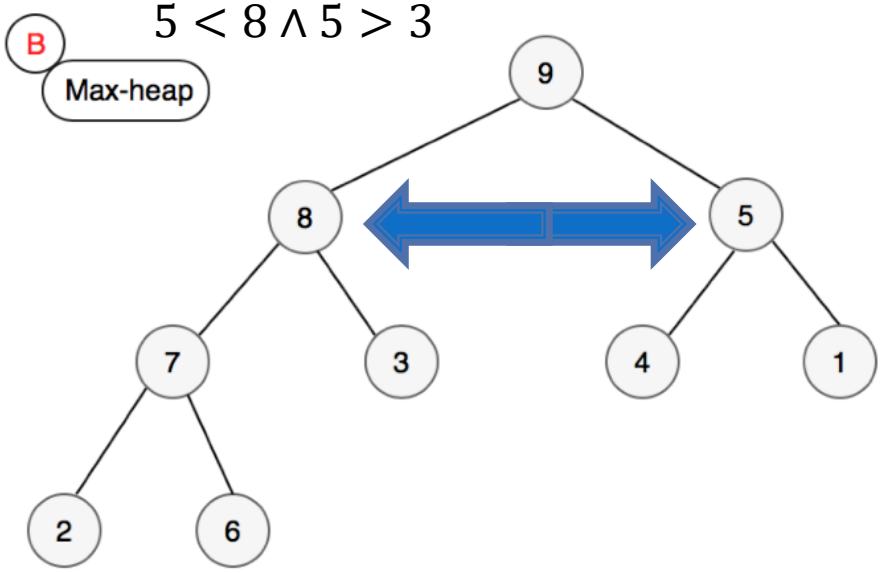
for *k*-means min heap
for EM max heap

$5 > 4 \wedge 5 > 3$



1	2	5	4	3	6	7	8	9

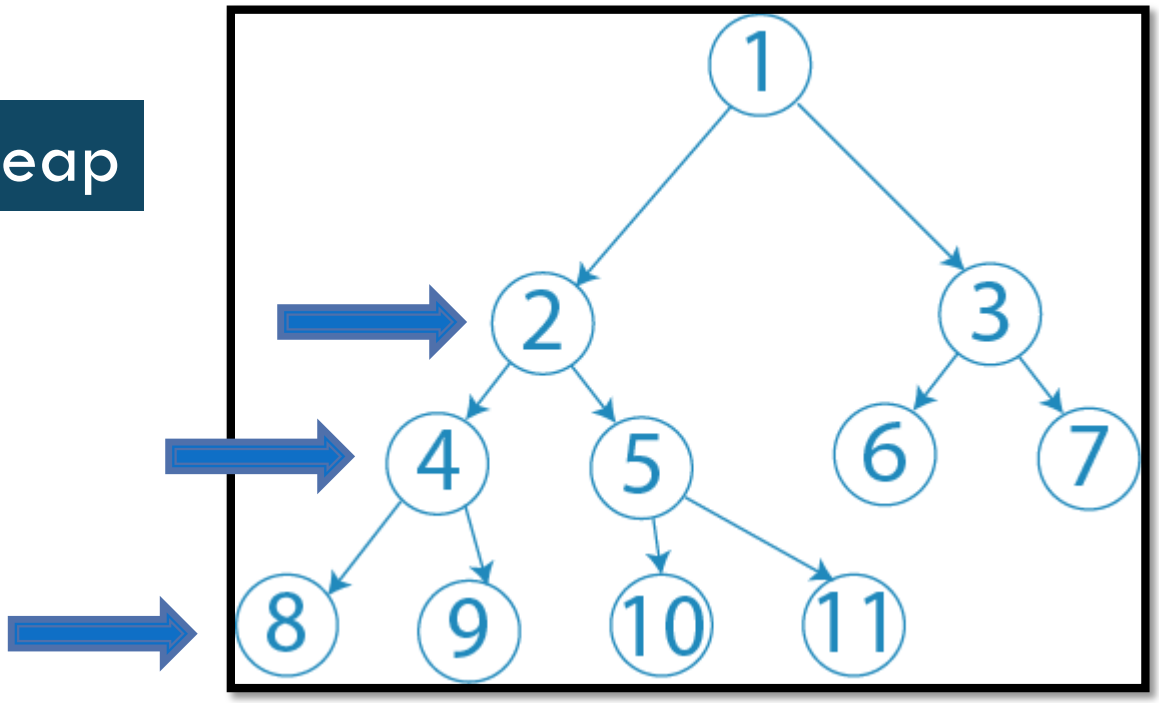
$5 < 8 \wedge 5 > 3$



9	8	5	7	3	4	1	2	6

III. Solution

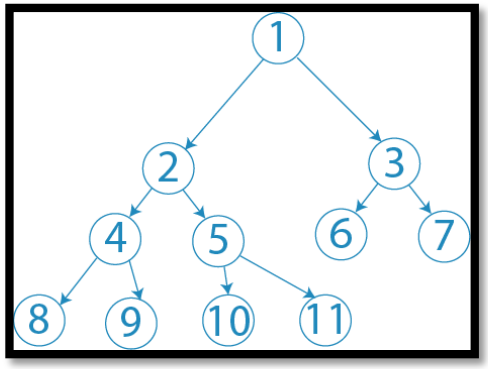
Strong Heap



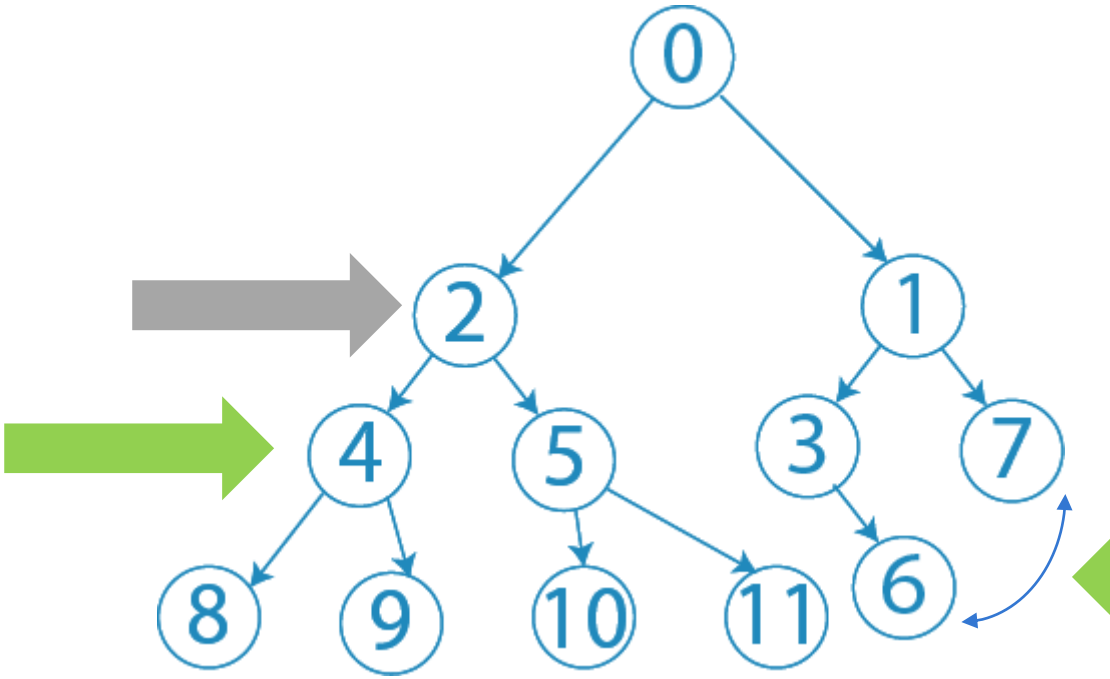
1	2	3	4	5	6	7	8	9	10	11

III. Solution

Strong and Weak Heap



1	2	3	4	5	6	7	8	9	10	11

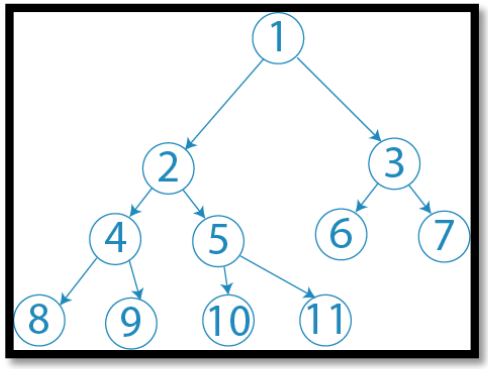


might be indicating that 6 is HE data masquerading as LE

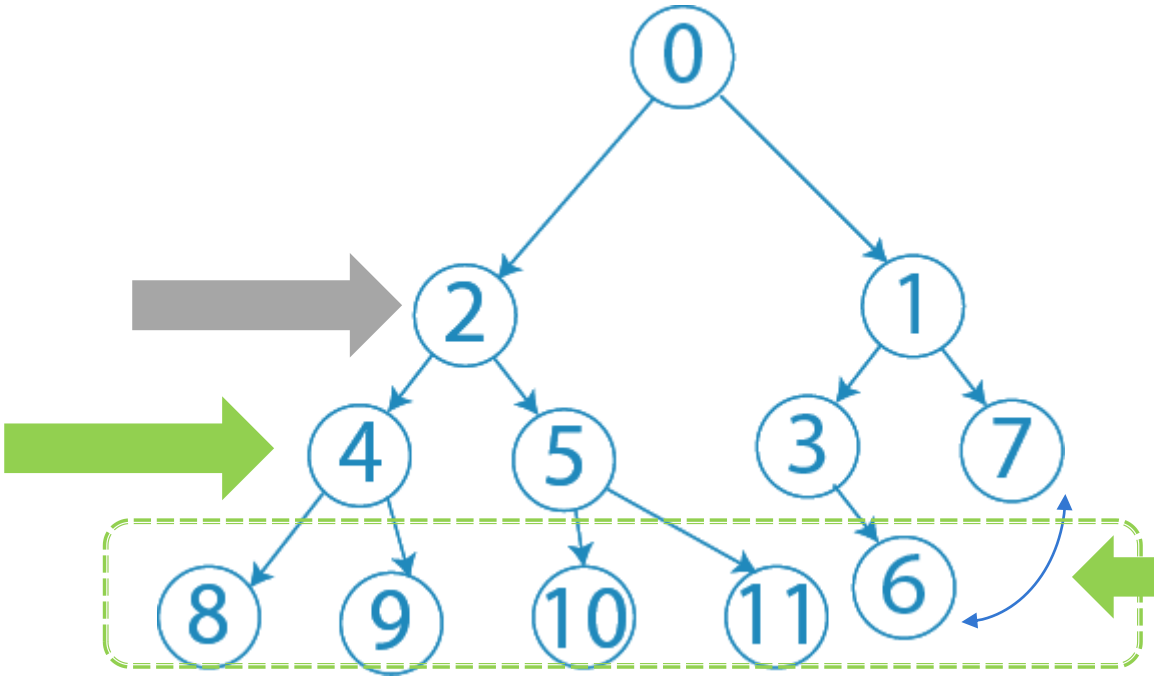
0	2	1	4	5	3	7	8	9	10	11	6

III. Solution

Strong and Weak Heap



1	2	3	4	5	6	7	8	9	10	11

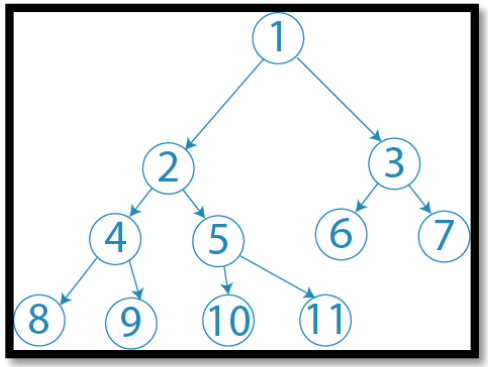


might be indicating that 6 is HE data masquerading as LE

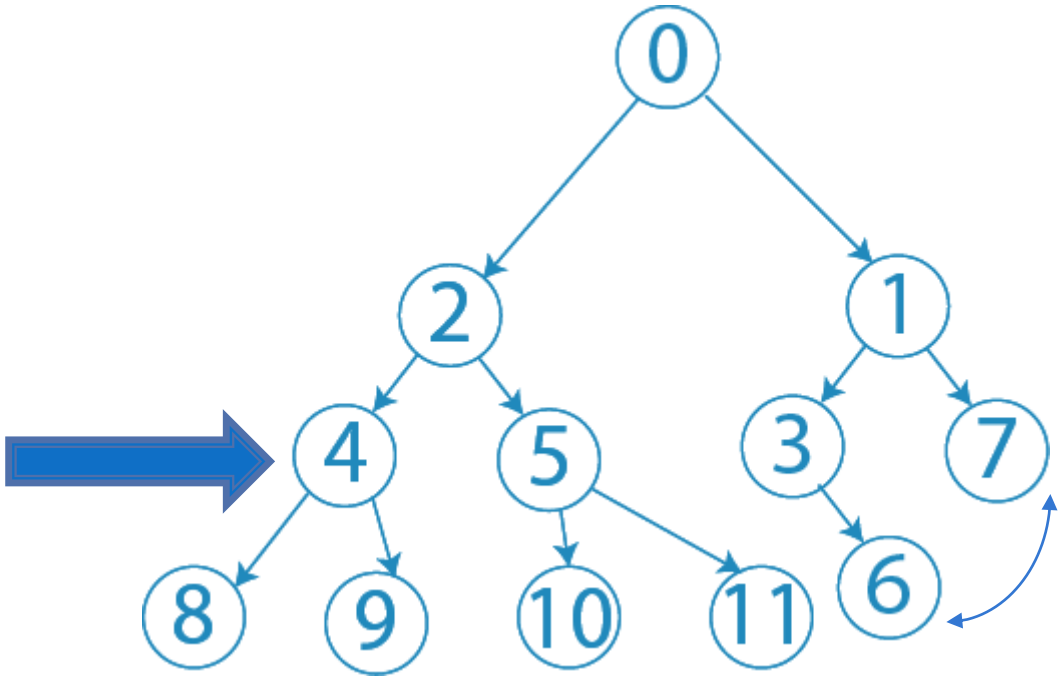
0	2	1	4	5	3	7	8	9	10	11	6

III. Solution

Strong and Weak Heap



1	2	3	4	5	6	7	8	9	10	11



Our stopping criteria

0	2	1	4	5	3	7	8	9	10	11	6

% change in Hamming Distance

I. **Context:** Perspective—it's all about the data

Observe that $\mu_s h$ exist from the root to some level ℓ and $\mu_w h$ exist from $\ell + 1$ to the root. For N distinct elements, the limit of the ratio of $\mu_s h$ to μh goes to zero:

Lemma

$$\lim_{N \to \infty} \frac{f(\mu_s h)}{f(N)} = 0$$

As an example, for seven distinct elements, 48 of the 80 different heaps (60%) are $\mu_s h$. Adding only one more distinct element drops the percentage to less than 23% for possible 210 heaps.

I. **Context:** Perspective—it's all about the data

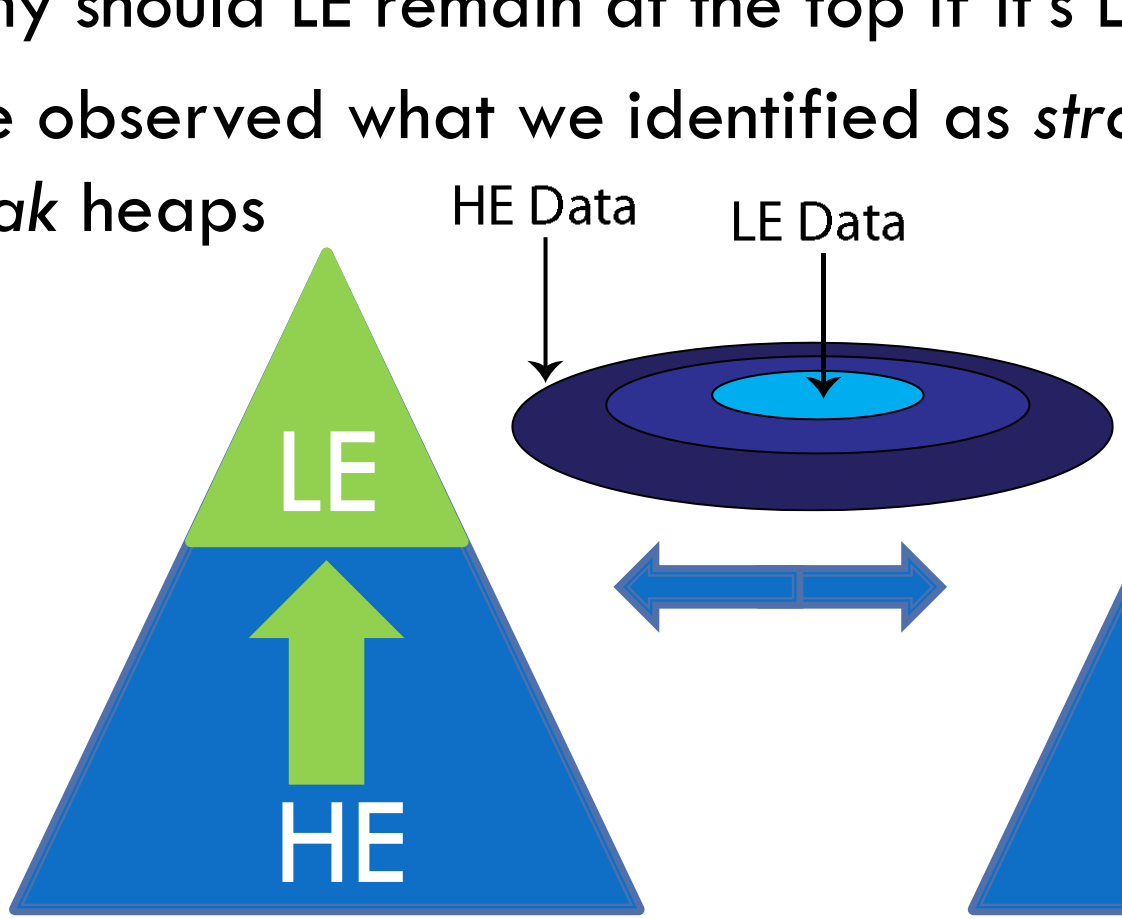
KM*

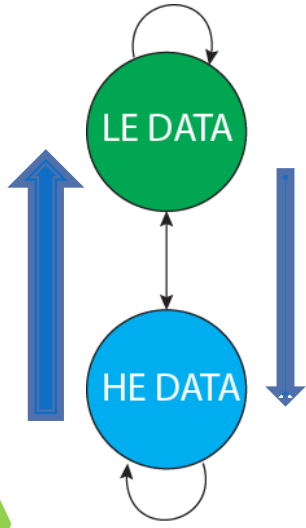
Data set	Never	Once	All But Once	Always
Ringnorm	17%	18%	8%	26%
US Census	19%	14%	9%	25%
KDD Cup	25%	6%	4%	28%

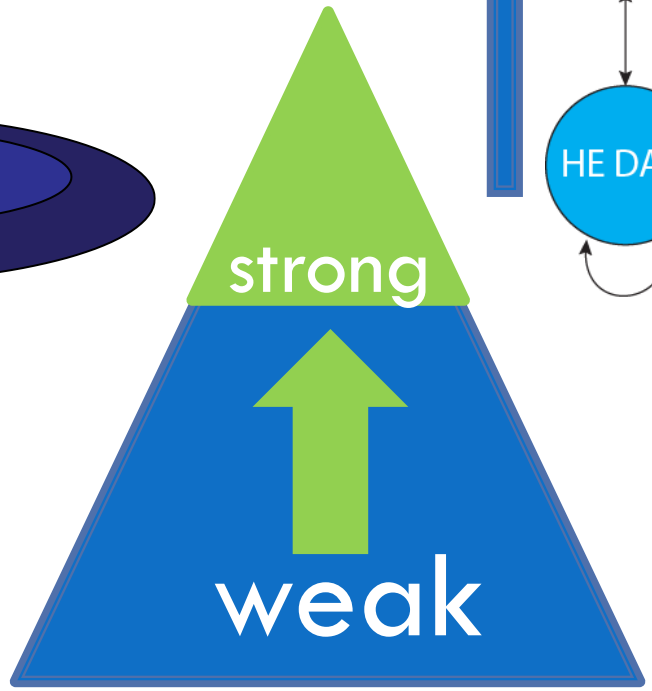
Table of what percentage of data vs. how many times showed up in the leaf nodes. This table reveals that a significant portion of the data can be clustered without any computation after first iteration.

III. Solution

- Why should LE remain at the top if it's LE?
- We observed what we identified as *strong* and *weak* heaps

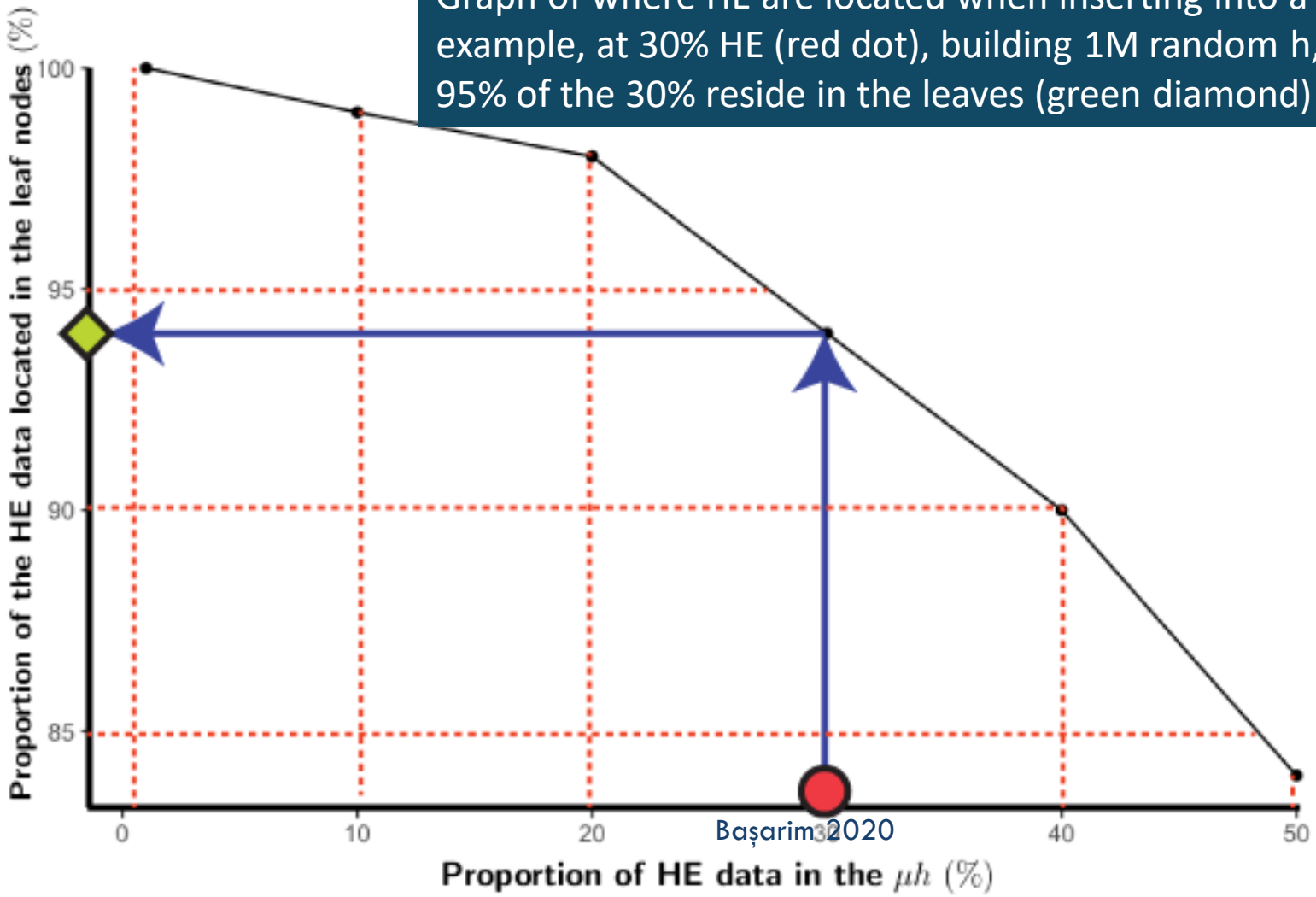






II. Where is HE mostly?

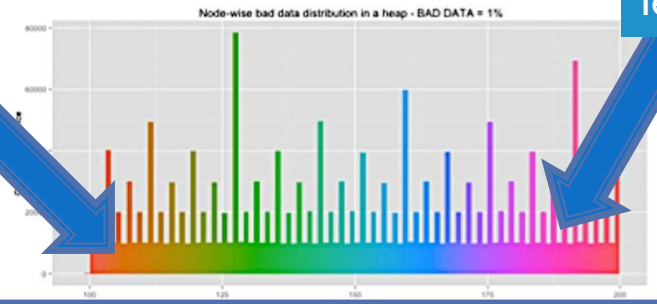
Graph of where HE are located when inserting into a h. For example, at 30% HE (red dot), building 1M random h, almost 95% of the 30% reside in the leaves (green diamond)

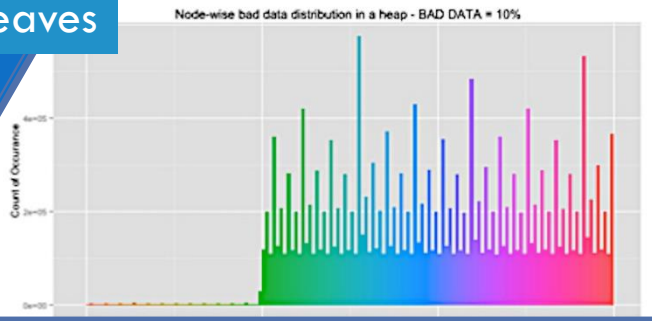


II. Where is HE mostly?

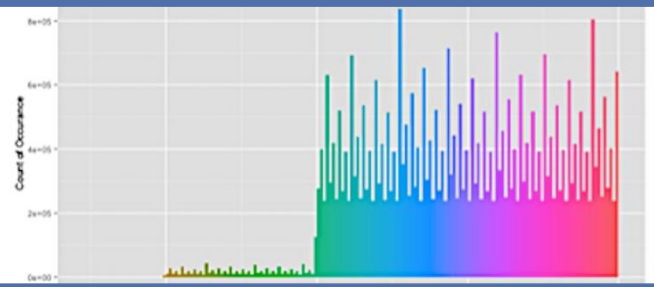
top of heap

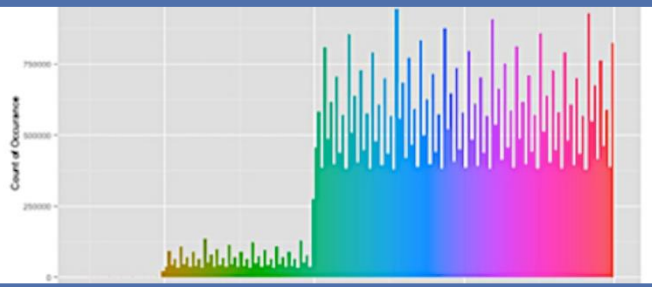
leaves



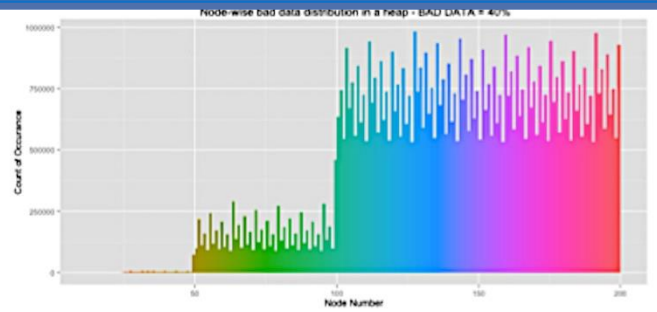


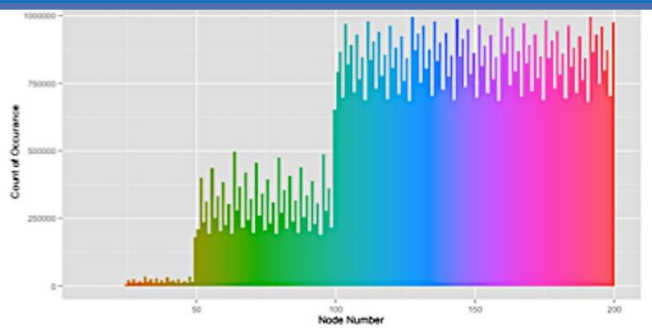
Increasing HE data shows, experimentally resides in leaves





Last image is 50/50 mixture





III. Solutions (Model based)

There are several variants of the initial expectation-maximization algorithm (EM-T for *traditional* EM) that exist and most of them have been designed by the statistical community. Broadly, we can aggregate the strategies in three categories:

(1) Employing new models Langari et al. (1997)

(2) Improving the accuracy Tang et al. (2007)

(3) Expediting convergence rates and reducing computing times Neal and Hinton (1998), Booth and Hobert (1999)

Ann. Statist.
Volume 11, Number 1 (1983), 95-103.
On the Convergence Properties of the EM Algorithm
C. F. Jeff Wu

J. R. Statist. Soc. B (1999)
61, *Part* 1, *pp.* 265–285

Maximizing generalized linear mixed model likelihoods with an automated Monte Carlo EM algorithm

James G. Booth and James P. Hob
University of Florida, Gainesville, USA

A VIEW OF THE EM ALGORITHM THAT JUSTIFIES INCREMENTAL, SPARSE, AND OTHER VARIANTS

Algorithm 3: Expectation-Maximization* (EM*) Algorithm

Expectation-Maximization*($\mathbf{D}$, k, ϵ, τ_1, τ_2):

$t \leftarrow 0$

Randomly initialize $\boldsymbol{\mu}_1^t, \ldots, \boldsymbol{\mu}_k^t$

$\boldsymbol{\Sigma}_i^t \leftarrow \mathbf{I},\ \forall i = 1, \ldots, k$

$P^t(C_i) \leftarrow \frac{1}{k}, \forall i = 1, \ldots, k$

// Heaps and data $\mathsf{H}^1 = \{H_1^1, H_2^1, \ldots, H_k^1\}$

// Individual maxheap $H_i = ((\mathbf{x}_1, w_{i1}), \ldots, (\mathbf{x}_\ell, w_{i\ell})),\ \ell \leq n$

// Gaussians $\mathsf{G}^1 = \{G_1^1, G_2^1, \ldots, G_k^1\}$

// Gaussain is a class $G_i = (\boldsymbol{\mu}_i,\ \boldsymbol{\Sigma}_i, P(C_i), H_i)$

$\mathbf{D}_{HE} \leftarrow \mathbf{D}$ // Initial Step Use All Data

repeat

  // Expectation Step

  // Using only $\mathbf{D}_{HE}$

  for $i = 1, \ldots, k$ *and* $j = 1, \ldots, n$ **do**

    $G_{ij}^t.w_{ij} \leftarrow \frac{f(\mathbf{x}_j|\boldsymbol{\mu}_i\boldsymbol{\Sigma}_i)\cdot P(C_i)}{\sum_{a=1}^k f(\mathbf{x}_j|\boldsymbol{\mu}_a,\boldsymbol{\Sigma}_a)\cdot P(C_a)}$ // Posterior probability $P^t(C_i|\mathbf{x}_j)$

  end

  // Build Heaps and then only update

  for $i = 1, \ldots, k$ *and* $j = 1, \ldots, n$ **do**

    $G_{ij}^t.H.insert(\mathbf{x}_i, w_{ij})$, where $\max\{\Pr(G_{ij}^t.w_{ij})\}$

  end

  // Maximization Step

  // Prepare to update with HE data

  $\mathbf{D}'_{HE} \leftarrow \emptyset$

  for $i = 1, \ldots, k$ **do**

    $\boldsymbol{\mu}_i^t \leftarrow \frac{\sum_{j=1}^n w_{ij}\cdot\mathbf{x}_j}{\sum_{j=1}^n w_{ij}}$ // re-estimate mean

    $\boldsymbol{\Sigma}_i^t \leftarrow \frac{\sum_{j=1}^n w_{ij}(\mathbf{x}_j-\boldsymbol{\mu}_i)^\intercal(\mathbf{x}_j-\boldsymbol{\mu}_i)}{\sum_{j=1}^n w_{ij}}$ // re-estimate covariance matrix

    $P^t(C_i) \leftarrow \frac{\sum_{j=1}^n w_{ij}}{n}$ // re-estimate priors

    $\mathbf{D}'_{HE} \leftarrow G_i^t.H_i.\mathsf{Get}(\tau_2)$ // Default leaves

    $\mathsf{G}^{t+1} \overset{\cup}{\leftarrow} \{G_i^{t+1}\}$

  end

  $t \leftarrow t + 1$

  $\mathbf{D}_{HE} \leftarrow \mathbf{D}'_{HE}$

until threshold on $\sum_{i=1}^k \mathsf{Hamming}(G_i^{t-1}.H) \leq \tau_1$

D^{HE}

Arrows represent adding heap structures

We are using only the HE (which are the leaves) in the expectation step; we use LE and HE in the maximization step.

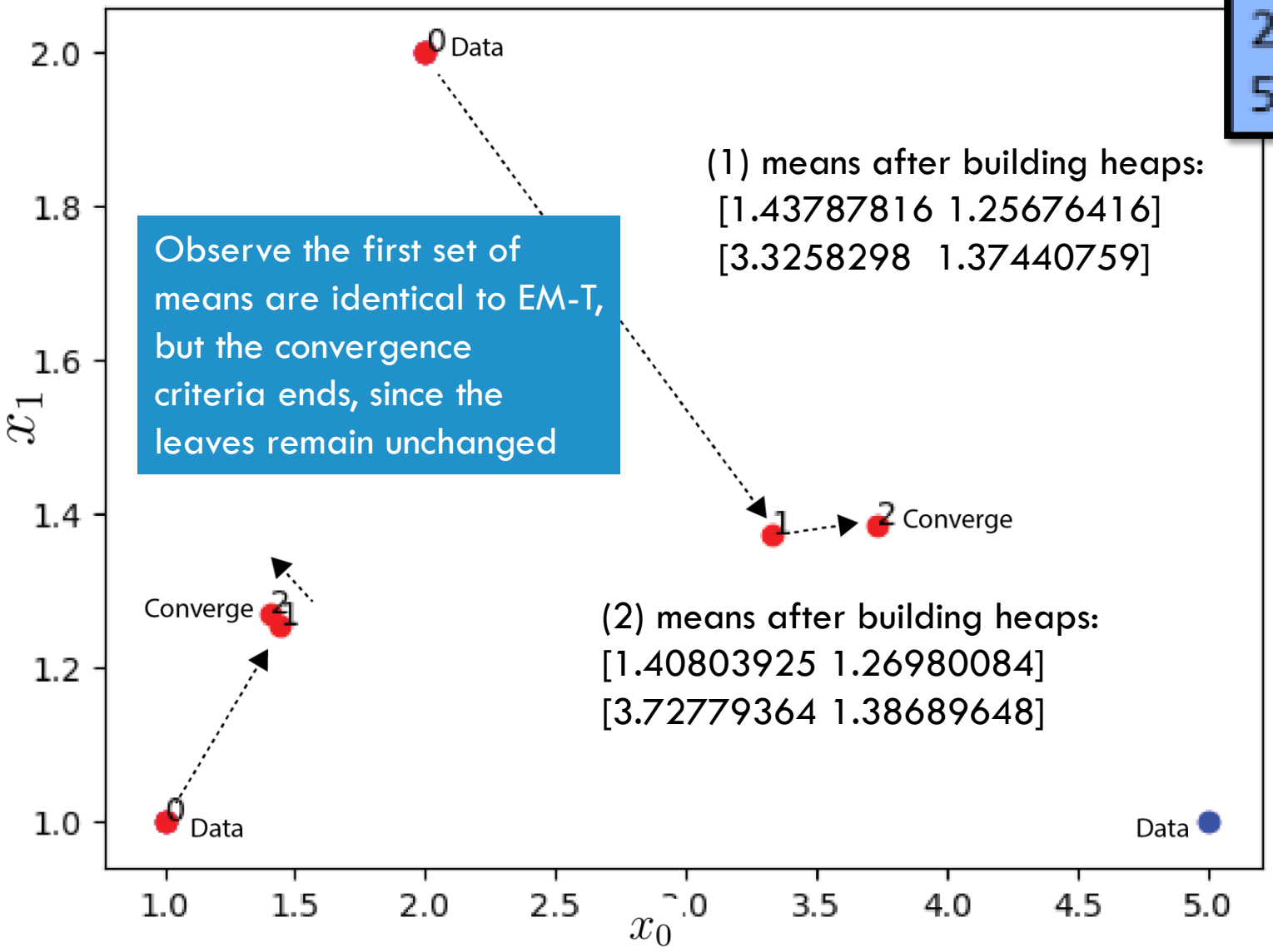
$D^{HE} \cup D^{LE}$

Başarim 2020

EM*

Data

1,1
2,2
5,1

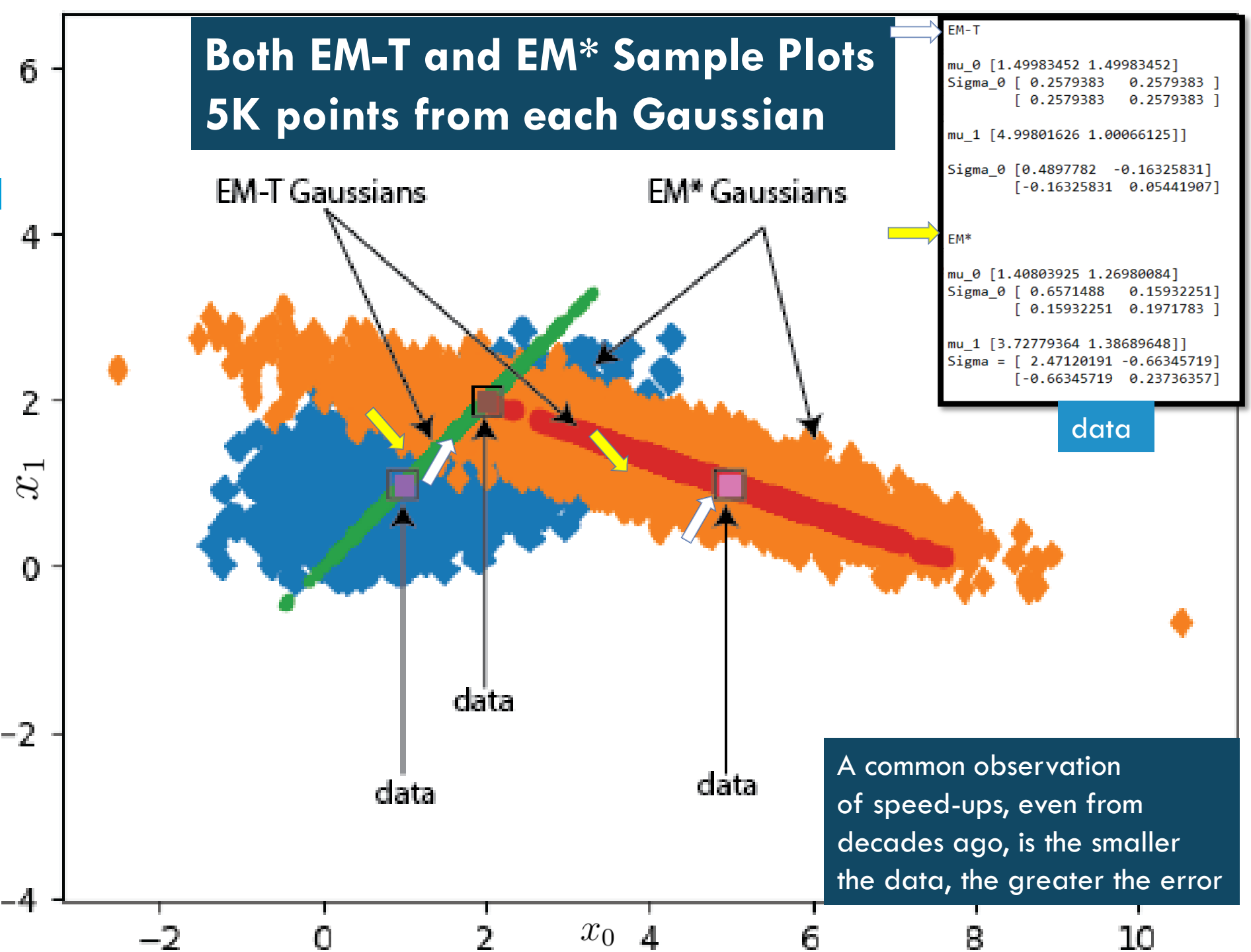


Observe the first set of means are identical to EM-T, but the convergence criteria ends, since the leaves remain unchanged

(1) means after building heaps:
[1.43787816 1.25676416]
[3.3258298 1.37440759]

(2) means after building heaps:
[1.40803925 1.26980084]
[3.72779364 1.38689648]

Both EM-T and EM* Sample Plots 5K points from each Gaussian



EM-T

```
mu_0 [1.49983452 1.49983452]
Sigma_0 [ 0.2579383   0.2579383 ]
        [ 0.2579383   0.2579383 ]

mu_1 [4.99801626 1.00066125]]

Sigma_0 [0.4897782  -0.16325831]
        [-0.16325831  0.05441907]
```

EM*

```
mu_0 [1.40803925 1.26980084]
Sigma_0 [ 0.6571488   0.15932251]
        [ 0.15932251  0.1971783 ]

mu_1 [3.72779364 1.38689648]]
Sigma = [ 2.47120191 -0.66345719]
        [-0.66345719  0.23736357]
```

data

A common observation of speed-ups, even from decades ago, is the smaller the data, the greater the error

IV. Results

The CRAN landscape

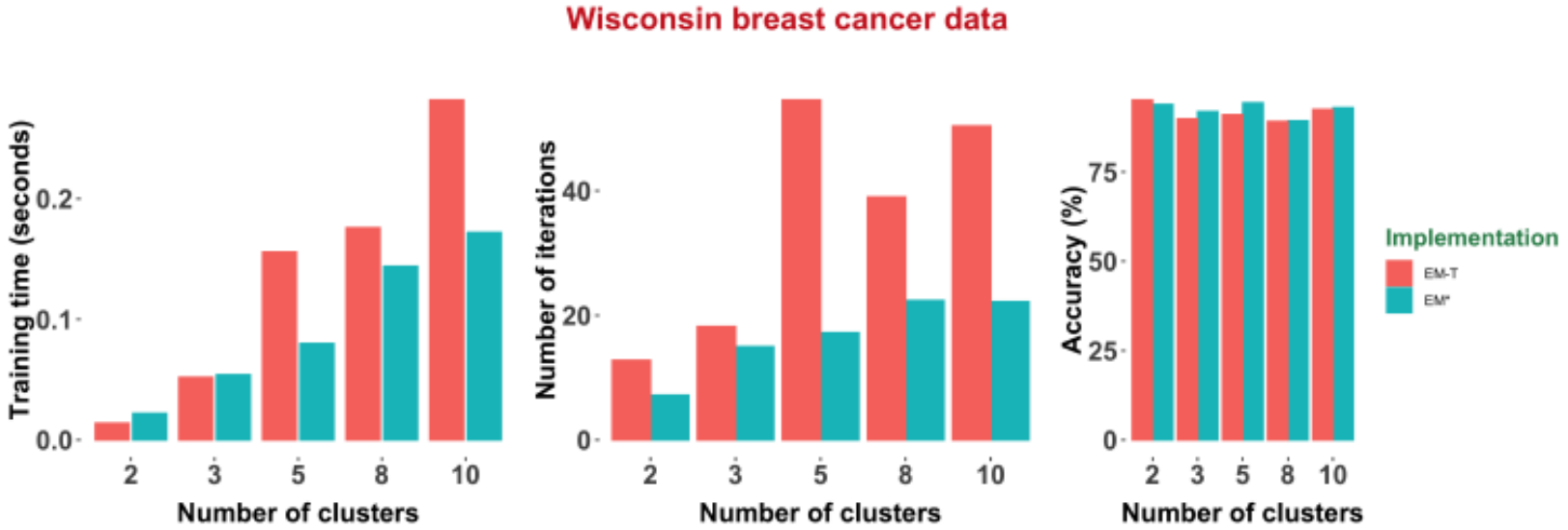
Pkg.	Vrs.	Clust./Class	Density Est.	MB Clust.	Lang.	Parallel
mclust	5.4.6	Y/Y	Y	Y	R	N
mixture	1.5	Y/Y	N	Y	R	N
Rmixmod	2.1.5	Y/Y	Y	Y	R	N
flexmix	2.3-15	Y/N	N	Y	R	N
EMCluster	0.2-12	Y/Y	N	Y	R	N
mixtools	1.2.0	Y/N	Y	N	R	N
ClusterR	1.2.2	Y/Y	N	N	R,C++	Y
DCEM (EM*)	2.0.2	Y/Y	N	N	R	N

IV. Results 700 points, 11 features

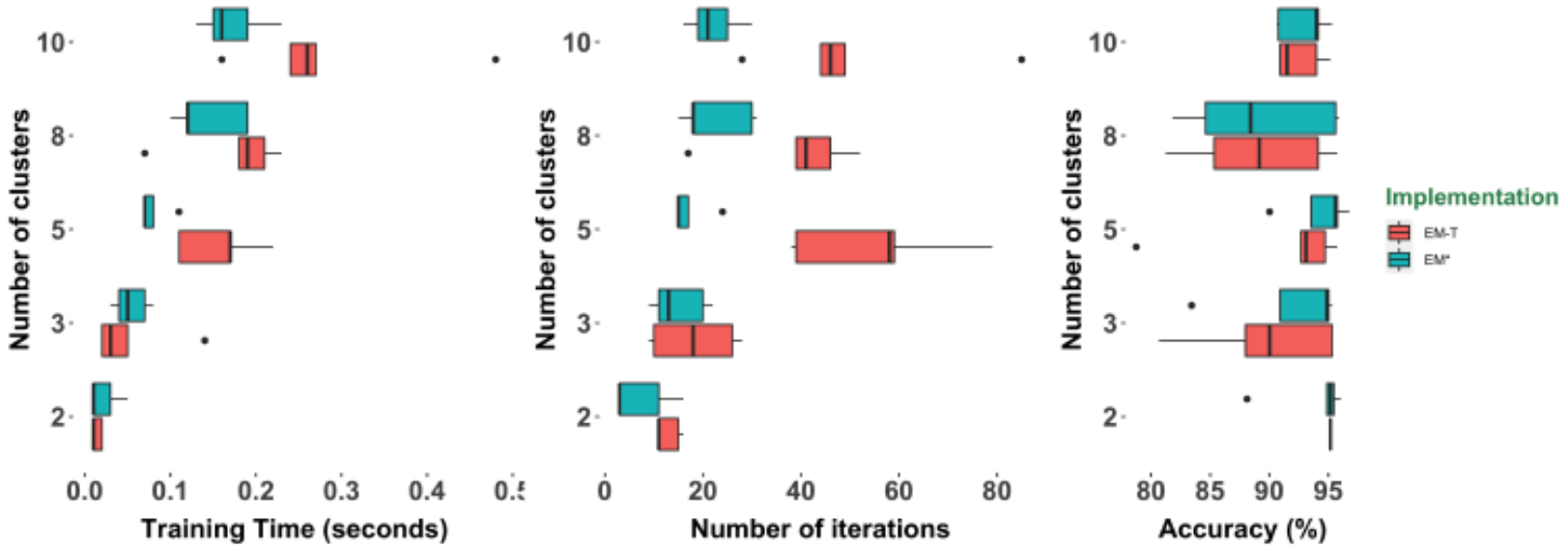
Wisconsin breast cancer data

The first experiment is done with the Wisconsin breast cancer data Wolberg et al. (1995). This data set is small: 699 records and 11 attributes. It is publicly available from the UCI (University of California, Irvine) machine learning repository Dua and Graff (2017) and was created to identify features that indicate tumor classification as either malignant or benign. Originally, the two groups have 444 and 239 data points, respectively. After removing missing records (a minor, insignificant number), EM∗ and EM-T algorithms are running over the data size (683 tuples using 9 features).

The results show that EM-T is marginally better for small values of k but, EM∗ takes less time and fewer number of iterations, as the number of clusters are increased beyond $k = 2$ (Figure 2 and Table 3). Results are similar in terms of clustering error and the accuracy of EM∗ is in the range $81.82 - 96.77\%$ and for EM-T, it is $78.74 - 95.75\%$. However, EM∗ has slightly higher accuracy than EM-T for true numbers of clusters $(k = 2)$ (Table 3).



(A): Plots showing averaged results of 5 runs for training time, iterations and accuracy.



(B): Boxplots for training time, number of iterations and accuracy.

IV. Results

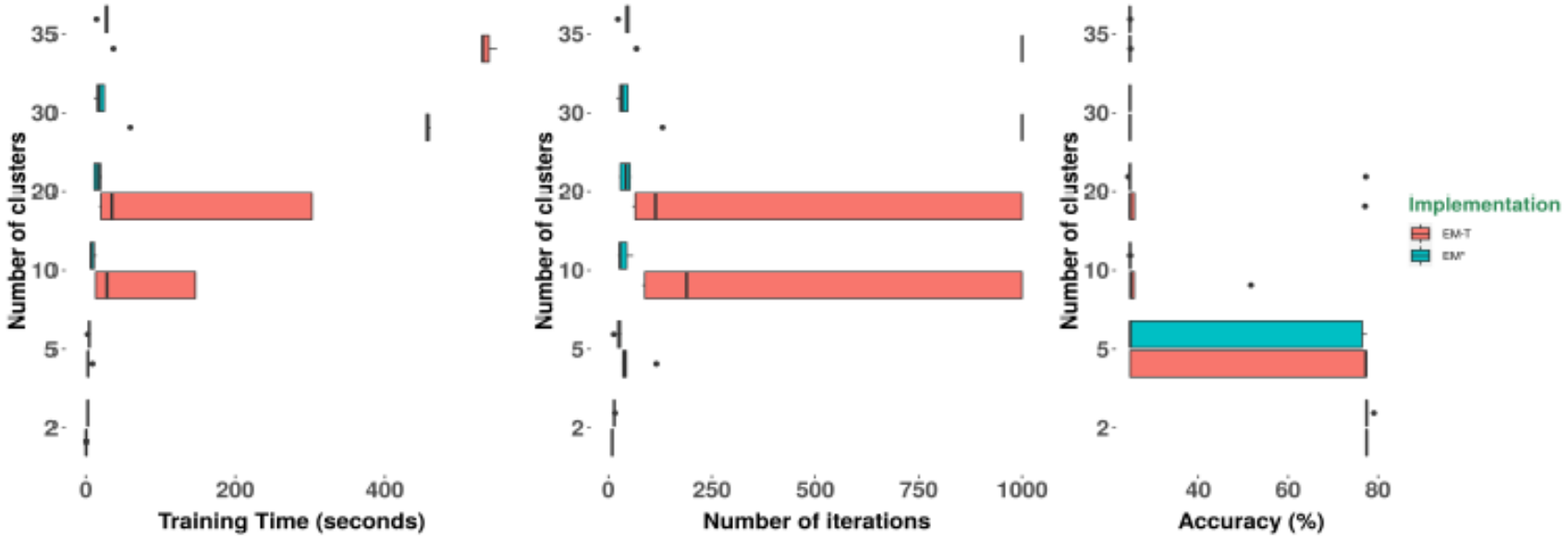
50K points, 14 features

US Census data

The second experiment is done on the US census income data Meek et al. (2002). This data set contains approximately 50K points with 14 features (both continuous and discrete) and is available for download from the UCI (University of California, Irvine) machine learning repository Dua and Graff (2017). After removing missing values, the data had 45K points and 6 dimensions (only continuous attributes). The original classification objective was to separate people who earn more than 50K from those who do not.

The results indicate that EM$*$ quickly outperforms EM-T (Figure 3) as the number of clusters are increased, particularly beyond $k = 2$ (Table 4). EM-T fails to converge in 1000

(A): Plots showing averaged results of 5 runs for training time, iterations and accuracy.

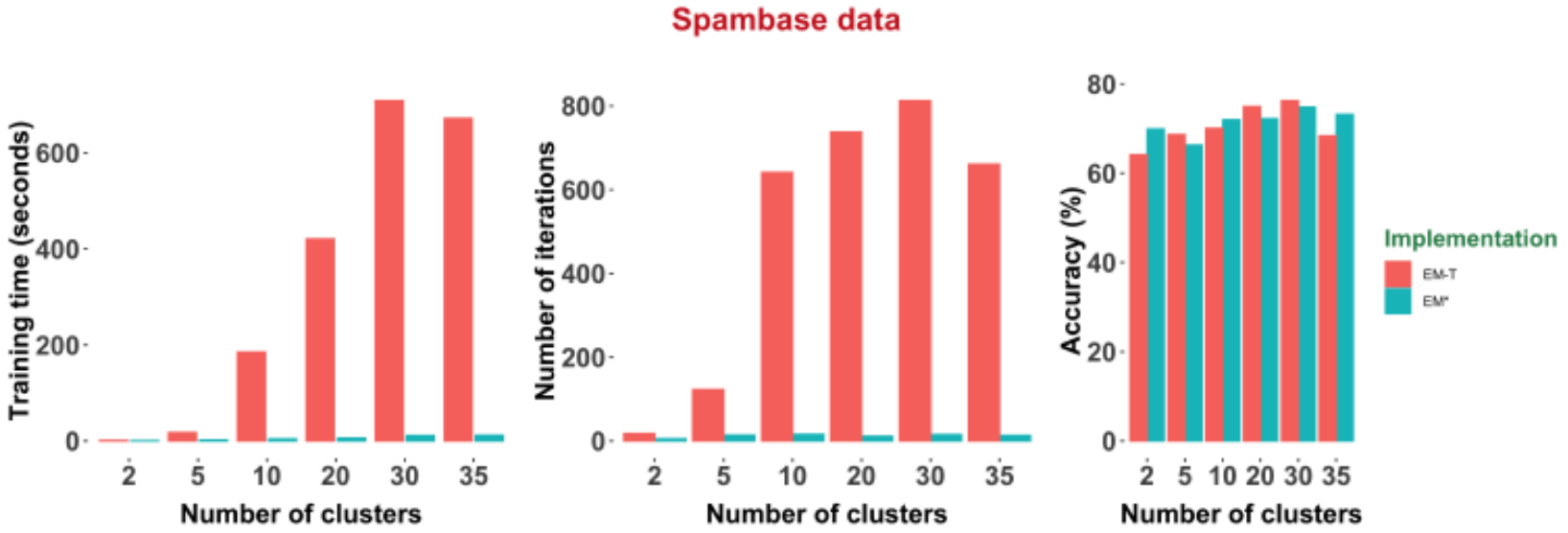


(B): Boxplots for training time, number of iterations and accuracy.

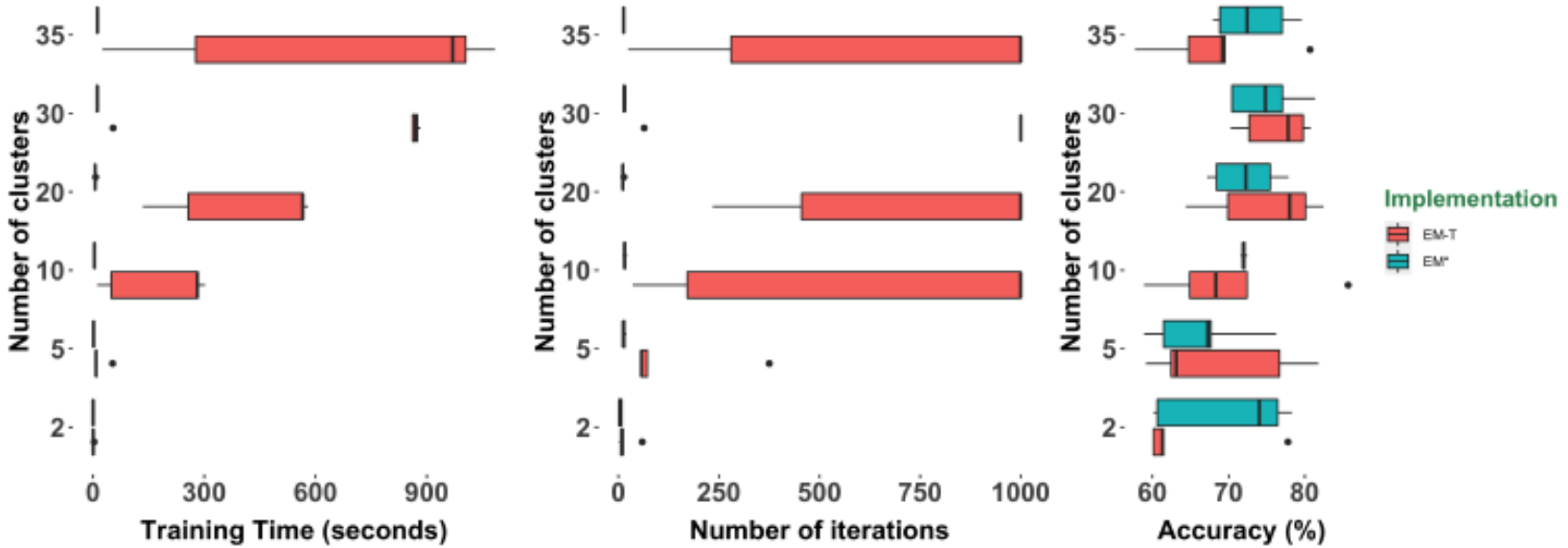
IV. Results 5K points, 57 features

Spambase data

In the third experiment, we used the Spambase data set Hopkins et al. (1999). The original purpose of this data was to create a spam filter by identifying the characteristic patterns (words, sequences *etc.*) that can distinguish between spam and non-spam emails. This data set has 4601 records with 57 features and in spite of its moderate size, presents a sharp increase in number of dimensions, scale and type of features, as compared to the earlier experiments. It includes 2788 spam and 1813 non-spam emails and is publicly available from the UCI (University of California, Irvine) machine learning repository Dua and Graff (2017).



(A): Plots showing averaged results of 5 runs for training time, iterations and accuracy.



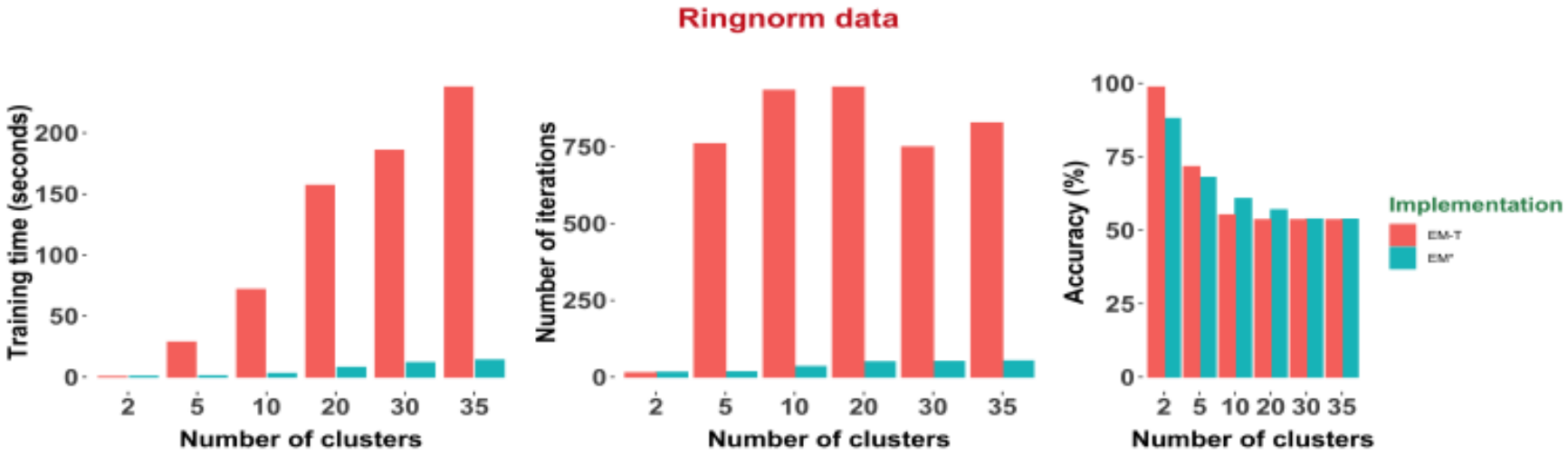
(B): Boxplots for training time, number of iterations and accuracy.

IV. Results 7.5K points, 20 features

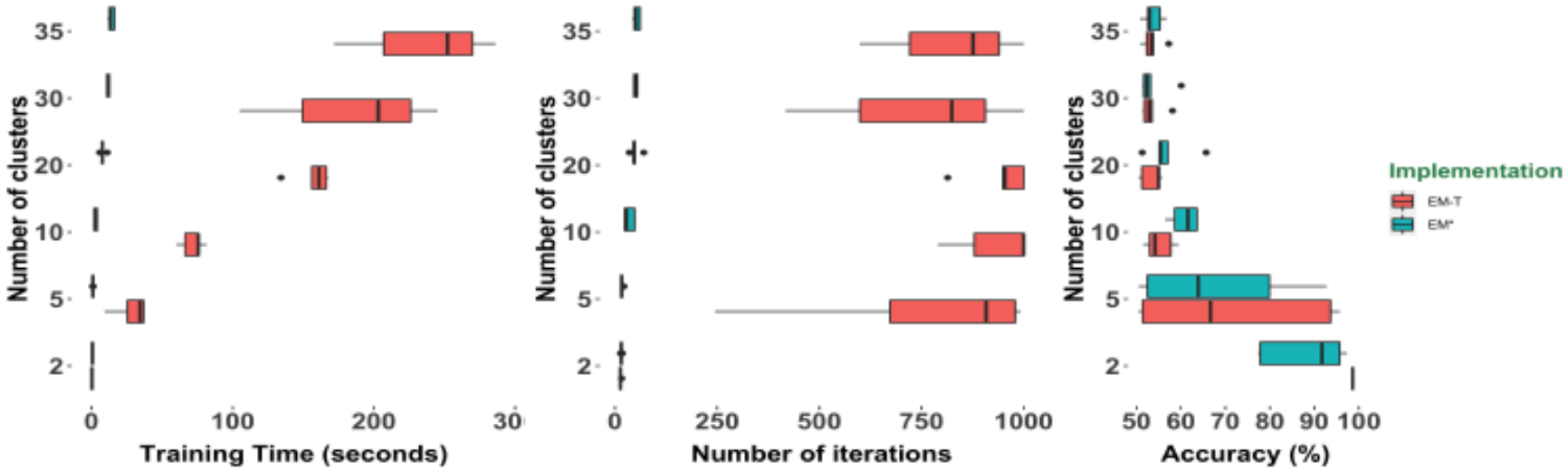
Ringnorm data

The first experiment was done on the ringnorm data Breiman (1996). This data set was initially used to study the bias and variance properties in arcing classifiers Breiman (1996) and is openly available from the UCI (University of California, Irvine) machine learning repository Dua and Graff (2017). It contains 7400 data points with 20 features. There are two groups that have 3664 and 3736 instances, respectively and both are drawn from multivariate normal distribution.

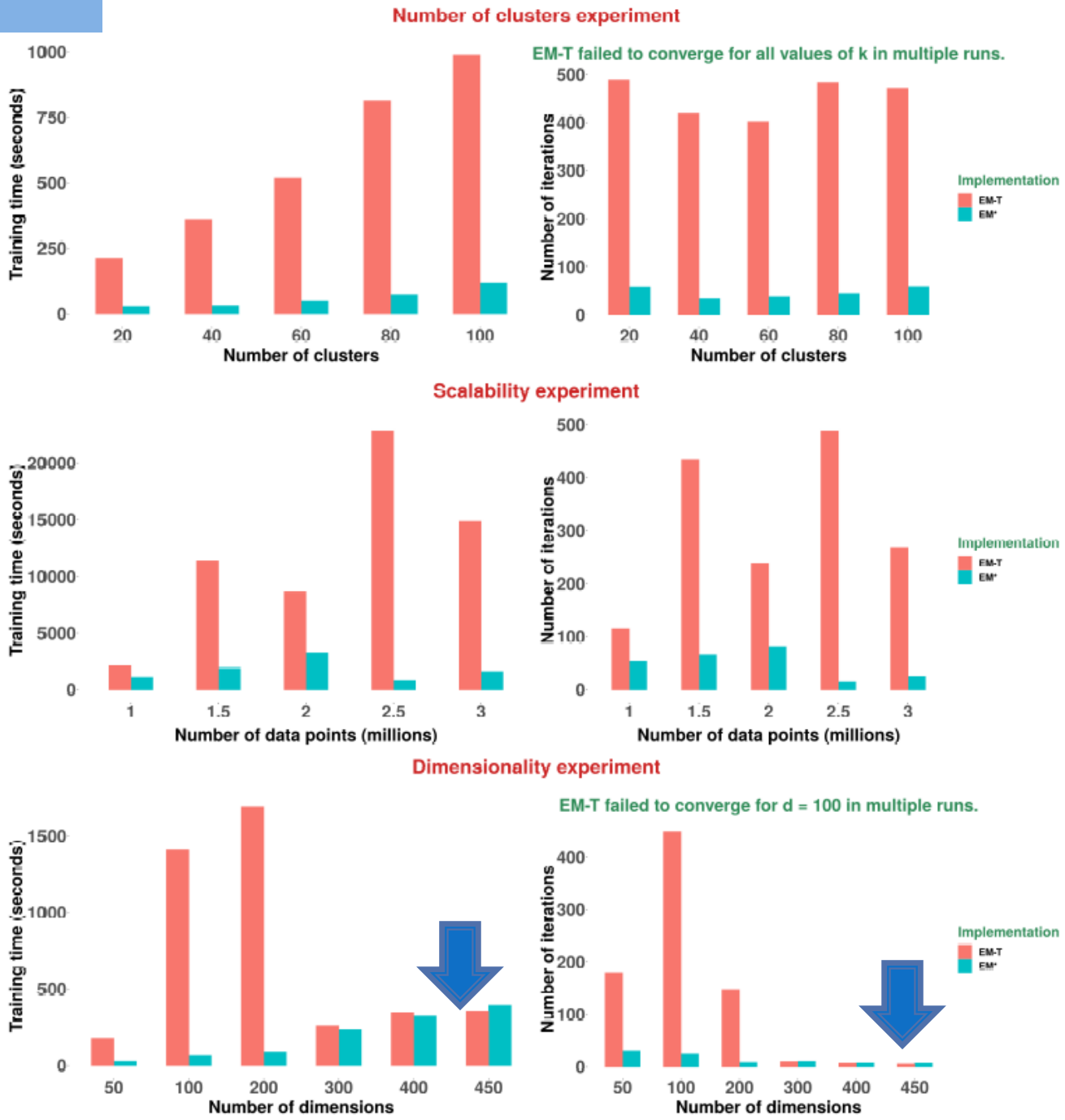
adaptively, resample, and combine (arc)



(A): Plots showing averaged results of 5 runs for training time, iterations and accuracy.



(B): Boxplots for training time, number of iterations and accuracy.

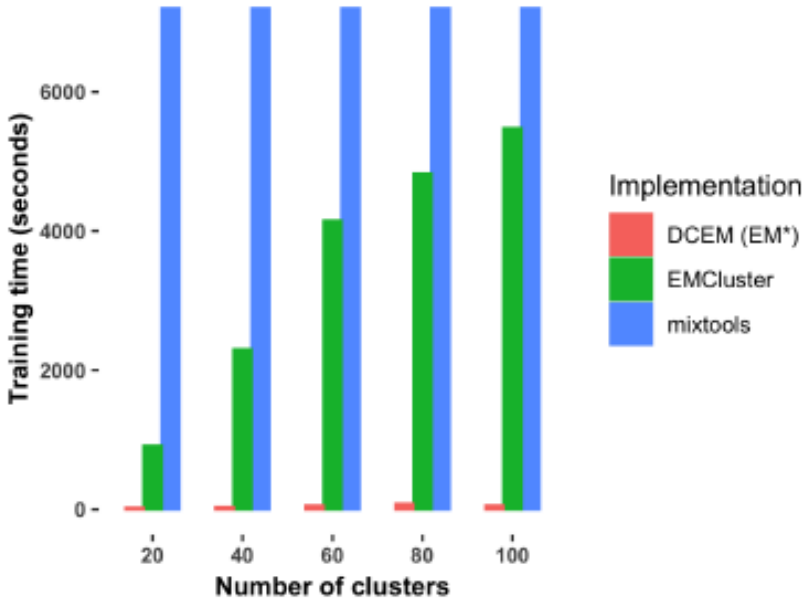

Number of clusters experiment
EM-T failed to converge for all values of k in multiple runs.
Training time (seconds)
Number of iterations
Number of clusters
Implementation
EM-T
EM*
Scalability experiment
Number of data points (millions)
Dimensionality experiment
EM-T failed to converge for d = 100 in multiple runs.
Number of dimensions

IV. Results

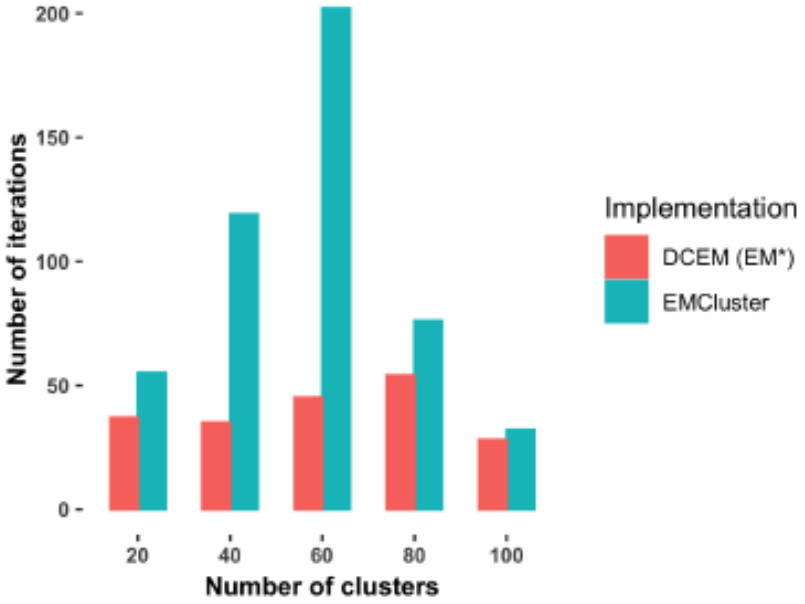
- Comparison to other packages in case we did “something special” to our code
- We used large synthetic data sets so no package would have an advantage
- We demonstrate some of the metrics we used to compare EM-T with EM*

Experiment on number of clusters

mixtools failed to converge for k = <20, 40, 60, 80, 100>

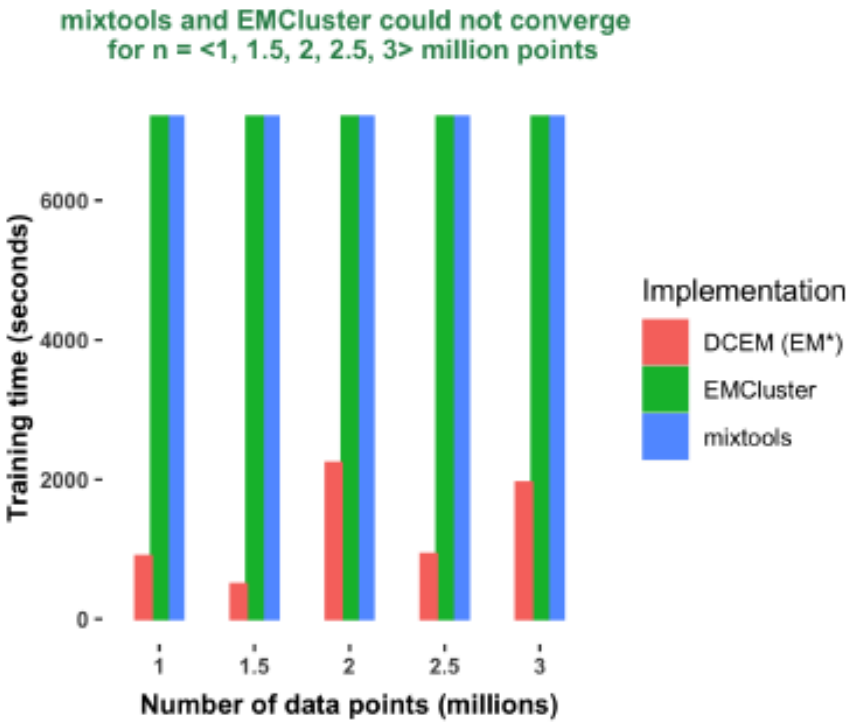


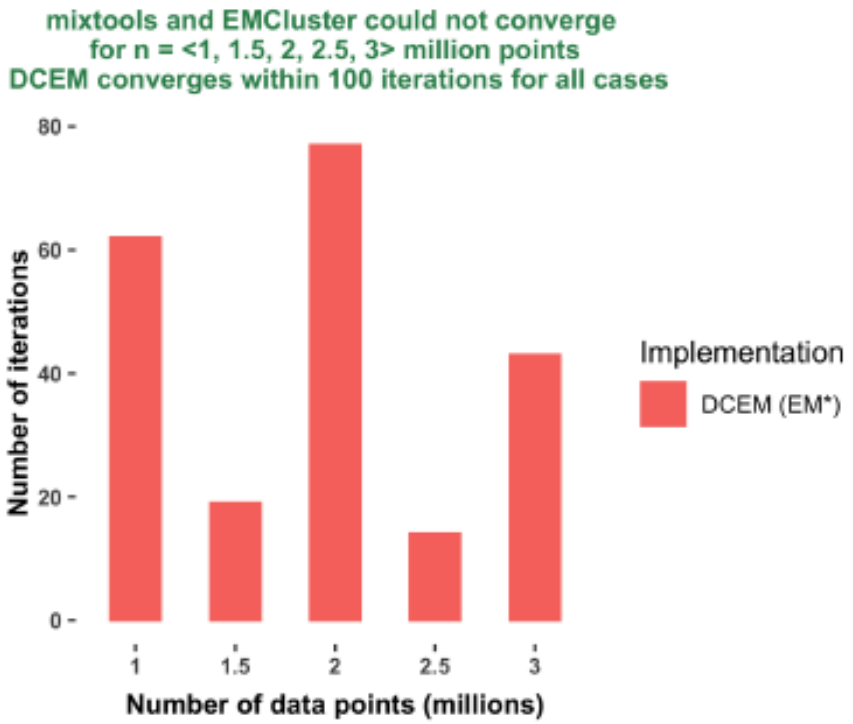
mixtools failed to converge, iterations not available



Comparing DCEM with mixtools and EMCluster on data with large number of clusters. Plot for execution time show that DCEM is significantly faster than EMCluster and mixtools. In terms of iterations, DCEM requires fewer iterations than EMCluster, iterations for mixtools are not reported as it could not converge in the threshold of 2 hours.

Scalability experiment





Comparing DCEM with **mixtools** and **EMCluster** on large data. DCEM always converges (maximum execution time of ~33 minutes for 2 million points) in all cases whereas **EMCluster** and **mixtools** failed to converge within the time limit of 2 hours. Iterations are not reported for **mixtools** and **EMCluster** because they failed to converge for all values of n.

Başarim 2020

IV. HPC in small packages: Viability of Small Machines facing Big Data, a study in leveraging high-expressive data helps improve iterative algorithms.

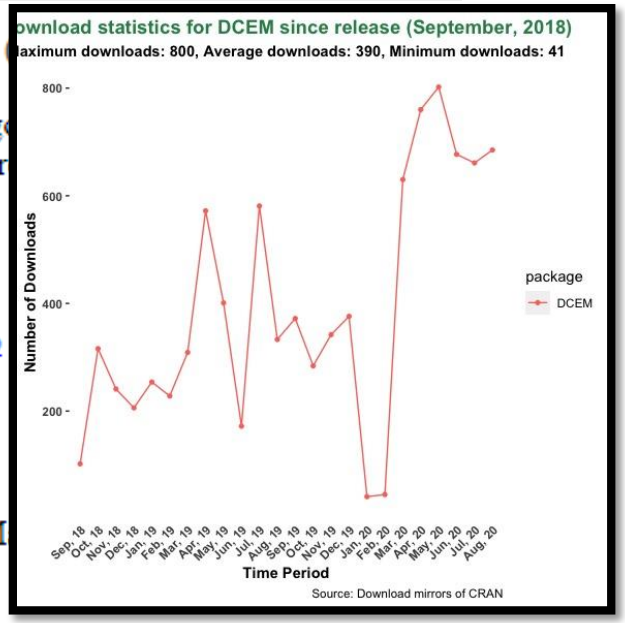
Computational details

All experiments were done on a dedicated SICE (School of Informatics, Computing and Engineering) VM instance - Schmuck. Schmuck is a 64-bit system running the bionic-18.04 release of the Ubuntu desktop. It has two intel Xeon cores (2.70GHz) with 16 GB of main memory and 8GB of swap partition. To ensure precision of results, the system was maintained in ideal conditions i.e., no other resource intensive jobs were running during the experiments. DCEM is implemented and tested in R 3.6.3 (2020-02-29).

DCEM: Clustering Big Data using Expectation Maximization Star

Implements the Improved Expectation Maximisation EM* and the traditional EM alg
The implementation supports both random and K-means++ based initialization. Refer

Version:	2.0.4
Depends:	R (≥ 3.2.0)
Imports:	mvtnorm (≥ 1.0.7), matrixcalc (≥ 1.0.3), MASS (≥ 7.3.49), Rcpp
LinkingTo:	Rcpp
Suggests:	knitr, rmarkdown
Published:	2020-08-02
Author:	Sharma Parichit [aut, cre, ctb], Kurban Hasan [aut, ctb], Jenne M
Maintainer:	Sharma Parichit <parishar at iu.edu>



V. Future Work

- Implement parallel/distributed heap-based optimization
- Examine other structures (maybe total order is better than partial order)
- Add additional functionality to EM* (Specifically k-means*, other iterative algorithms
- Maybe (maybe) make the Python upgradable to 3.8 instead of 2.7 (we have a good number of downloads)

Questions?