UNIVERSITY OF
EASTERN FINLAND

How much k-means can be improved by using better initialization and repeats?

Pasi Fränti and Sami Sieranoja

11.4.2019

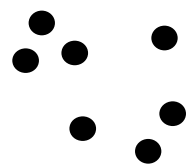
P. Fränti and S. Sieranoja, ”How much k-means can be improved by using better initialization and repeats?", *Pattern Recognition*, 2019.

Introduction

Goal of k-means

Input N points:

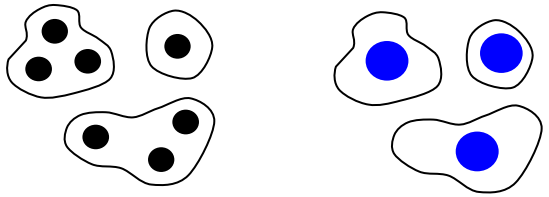
$X=\{x_1, x_2, \ldots, x_N\}$



Output partition and k centroids:

$P=\{p_1, p_2, \ldots, p_k\}$

$C=\{c_1, c_2, \ldots, c_k\}$



Objective function:

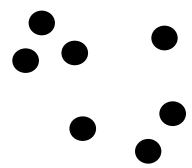
$$SSE = \sum_{i=1}^{N} \left\| x_i - c_j \right\|^2$$

SSE = **s**um-of-**s**quared **e**rrors

Goal of k-means

Input N points:

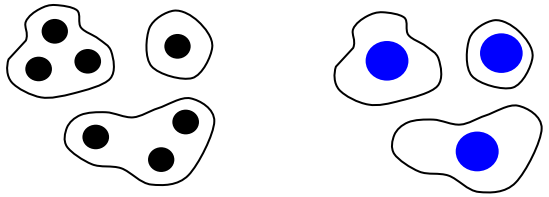
$X=\{x_1, x_2, \ldots, x_N\}$



Output partition and k centroids:

$P=\{p_1, p_2, \ldots, p_k\}$

$C=\{c_1, c_2, \ldots, c_k\}$



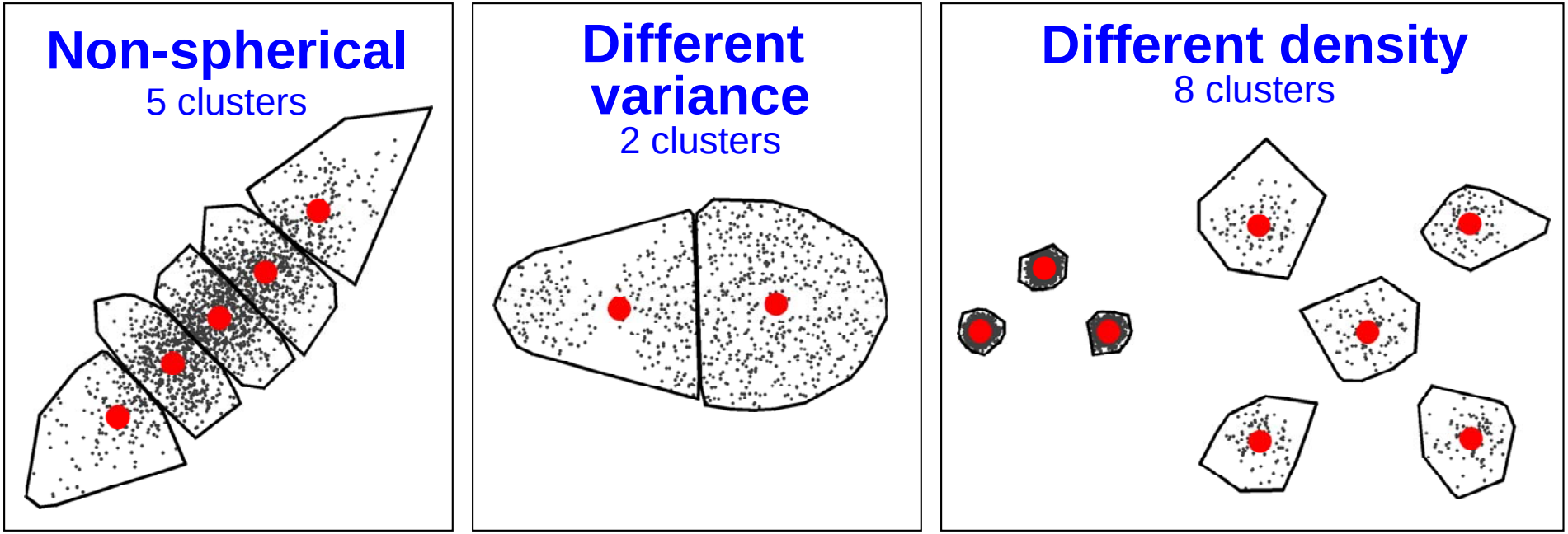
Objective function:

$$SSE = \sum_{i=1}^{N} \left\| x_i - c_j \right\|^2$$

Assumptions:

- SSE is suitable
- k is known

Using SSE objective function



K-means algorithm

http://cs.uef.fi/sipu/clustering/animator/

X = Data set
C = Cluster centroids
P = Partition

K-Means(X, C) → (C, P)

REPEAT

$C_{prev} \leftarrow C$;

FOR i=1 TO N DO
$p_i \leftarrow$ FindNearest(x_i, C);

Assignment step

FOR j=1 TO k DO
$c_j \leftarrow$ Average of $x_i \ \forall \ p_i = j$;

Centroid step

UNTIL $C = C_{prev}$

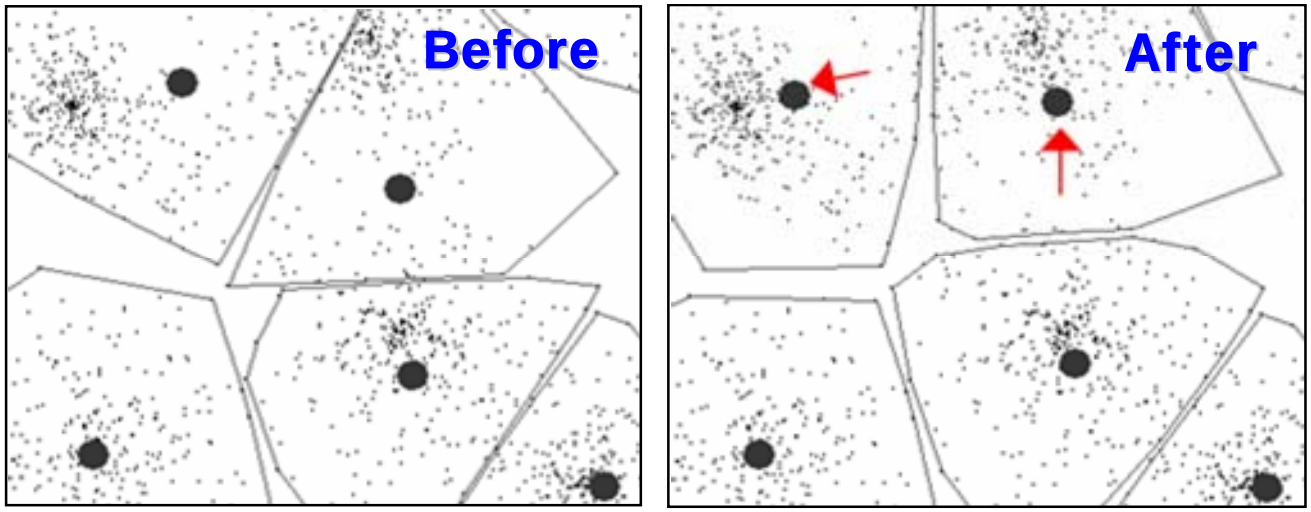
K-means optimization steps

Assignment step:

$$P_i = \arg\min_{1 \le j \le k} \left\| x_i - c_j \right\|^2 \qquad \forall i \in [1, N]$$

Centroid step:

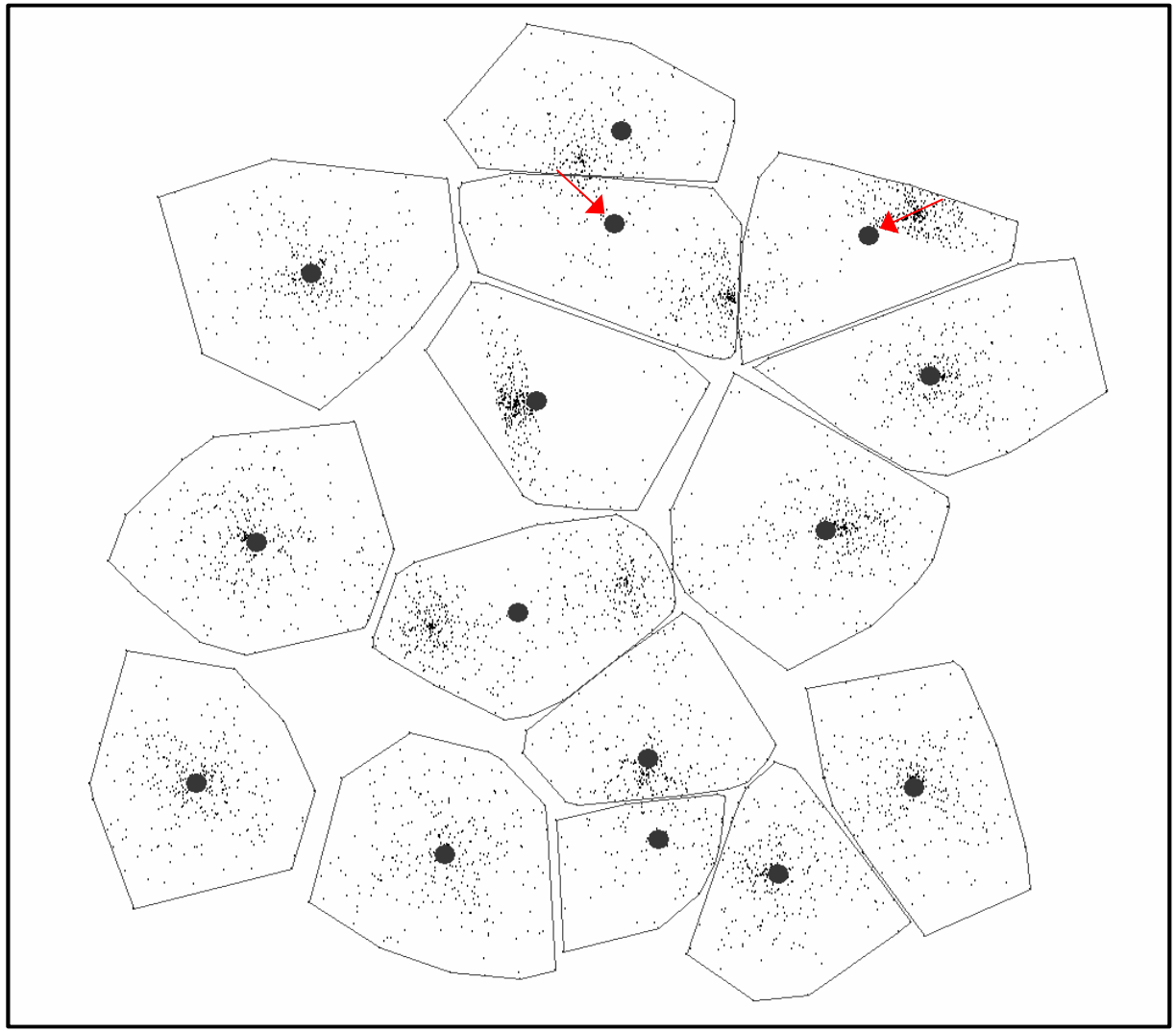
$$c_j = \sum_{P_i = j} x_i \Big/ \sum_{P_i = j} 1 \qquad \forall j \in [1, k]$$



Examples

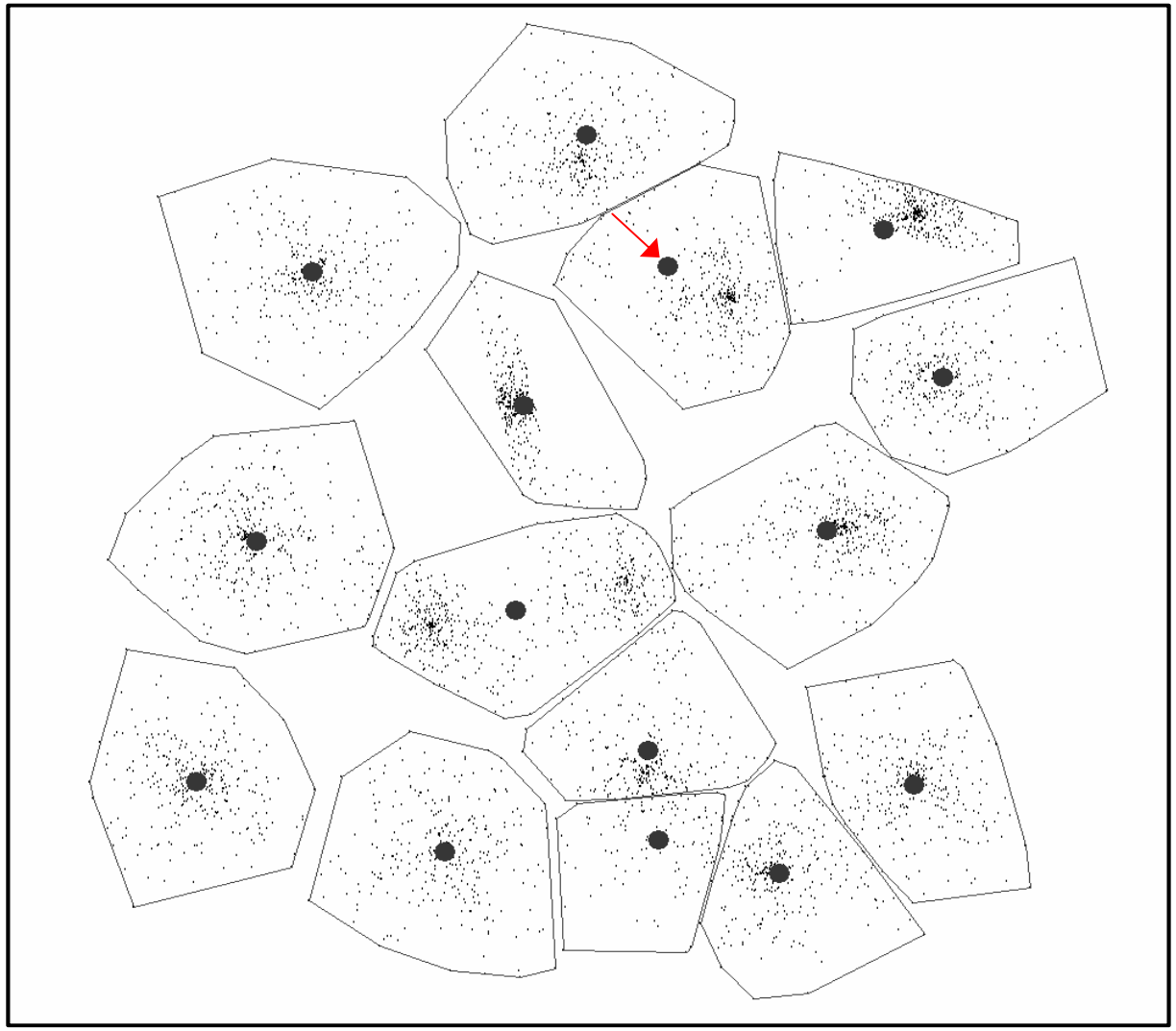
Iterate by k-means

1st iteration



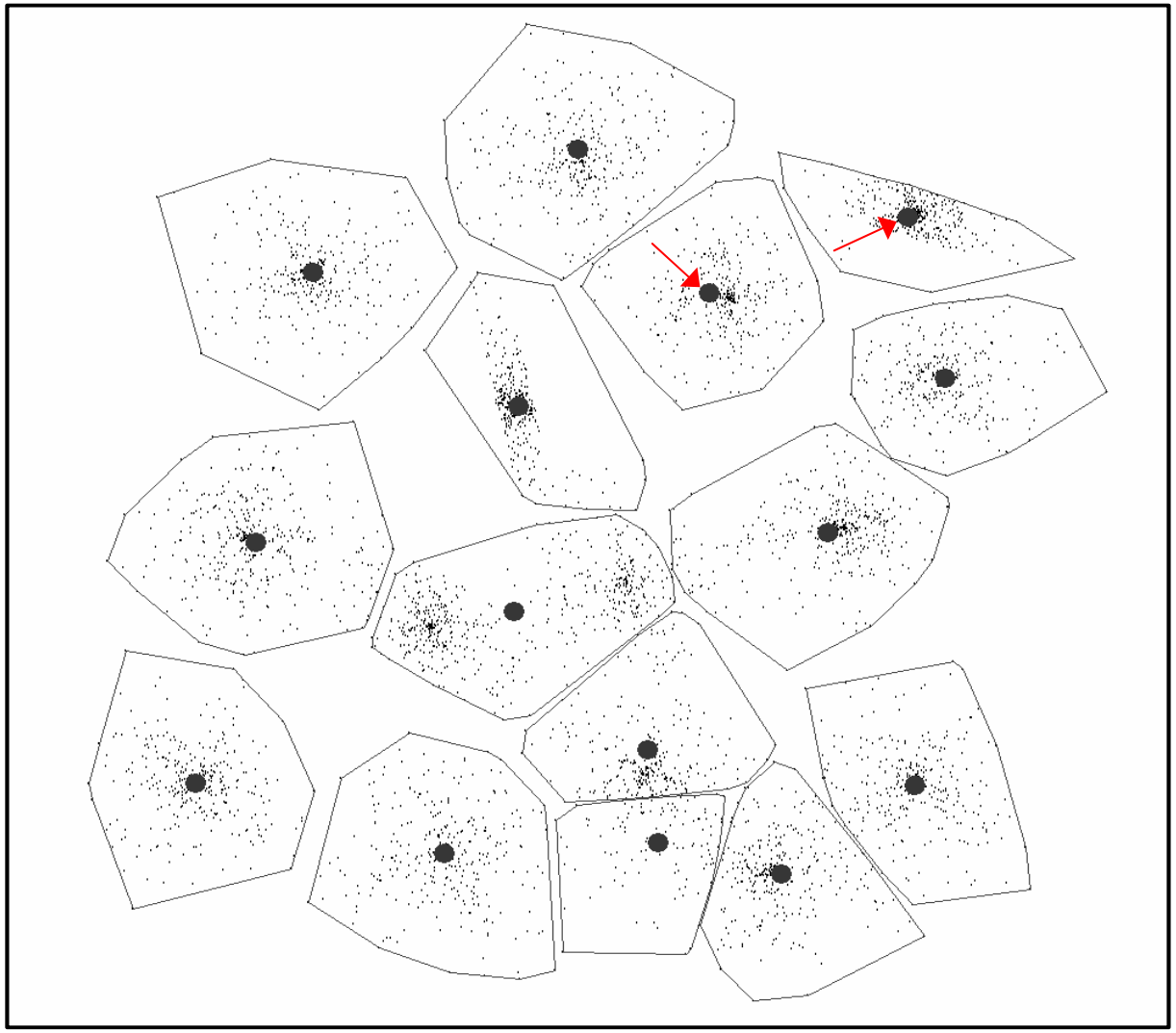
Iterate by k-means

2nd iteration



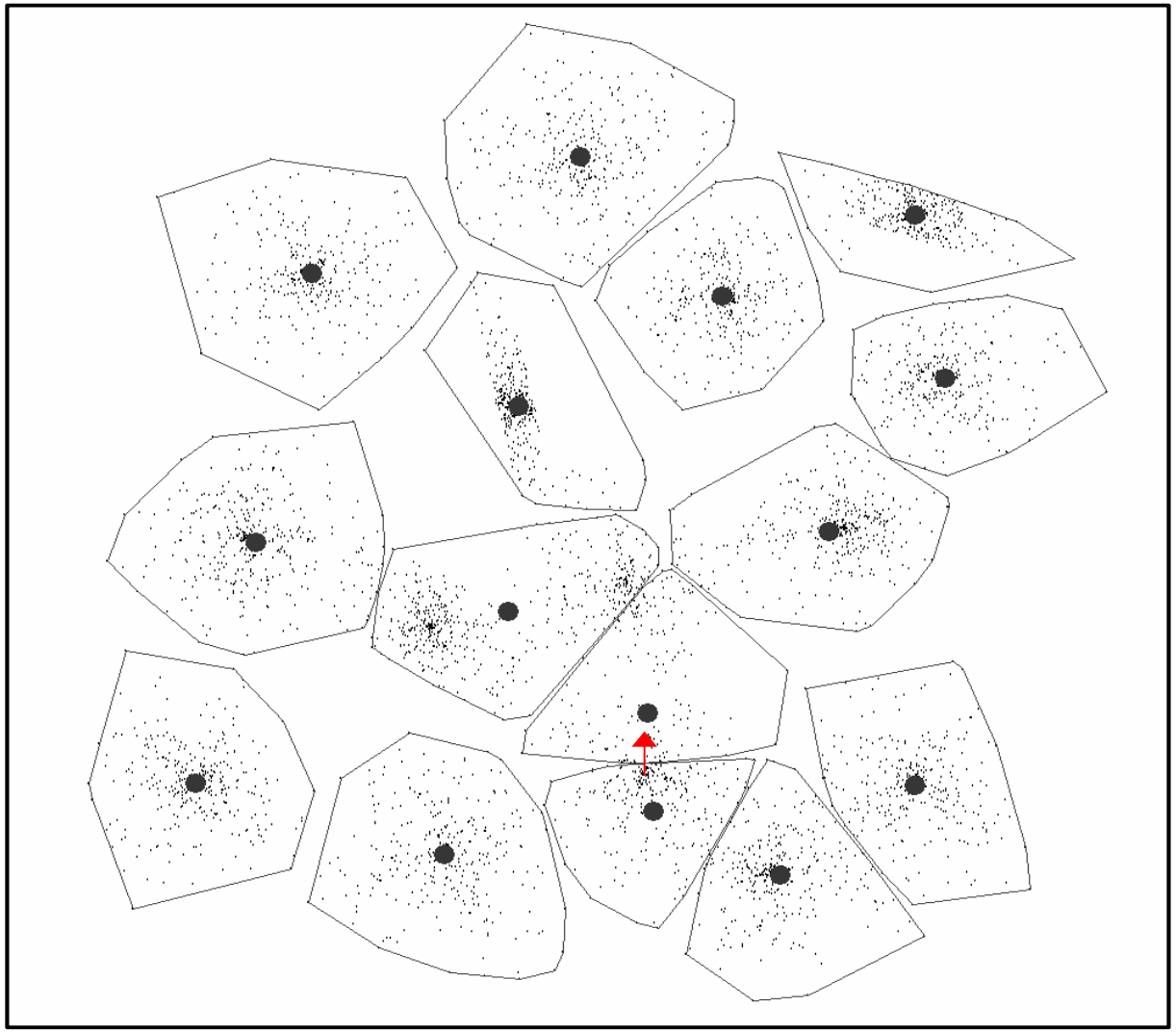
Iterate by k-means

3rd iteration



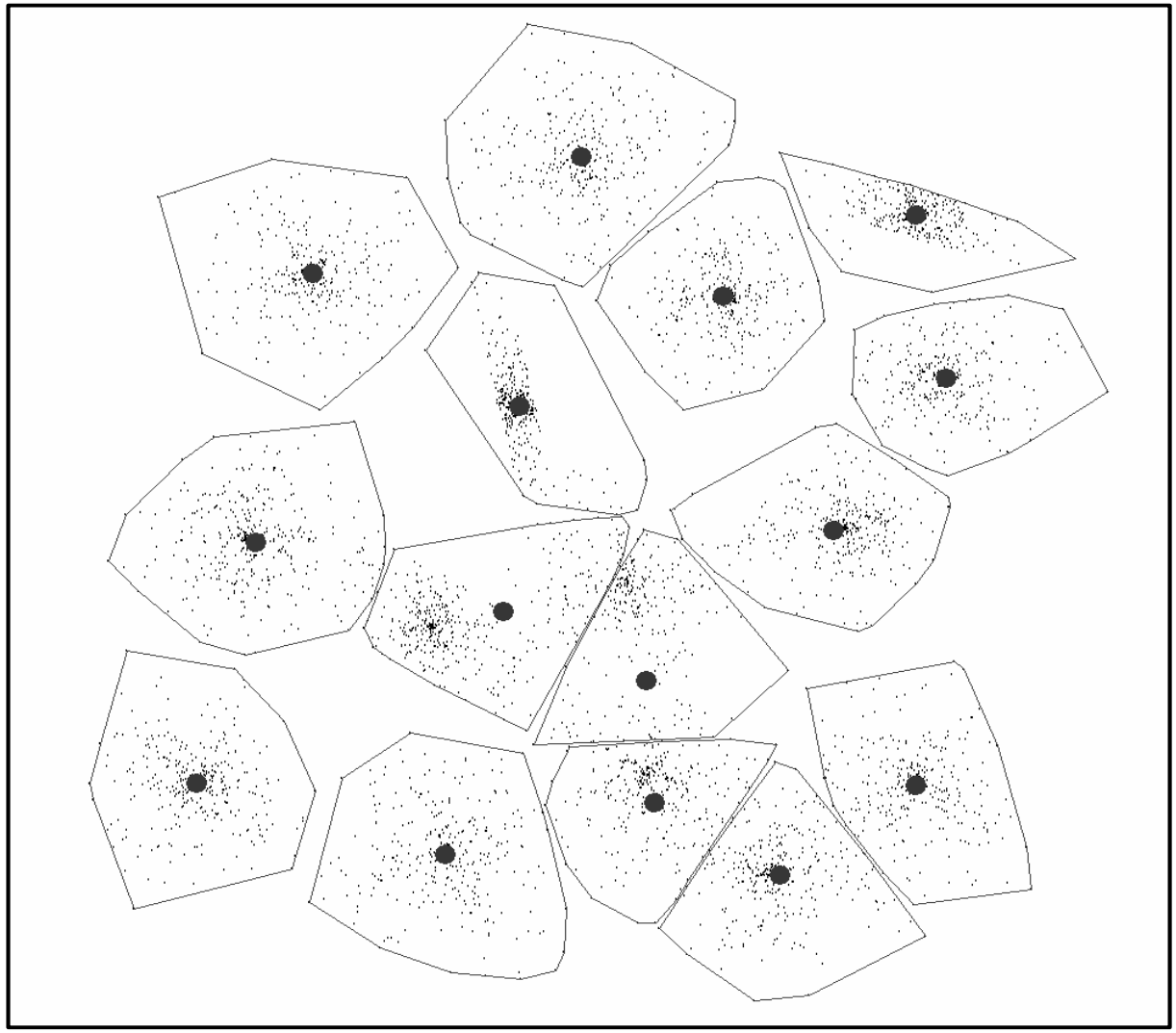
Iterate by k-means

16th iteration



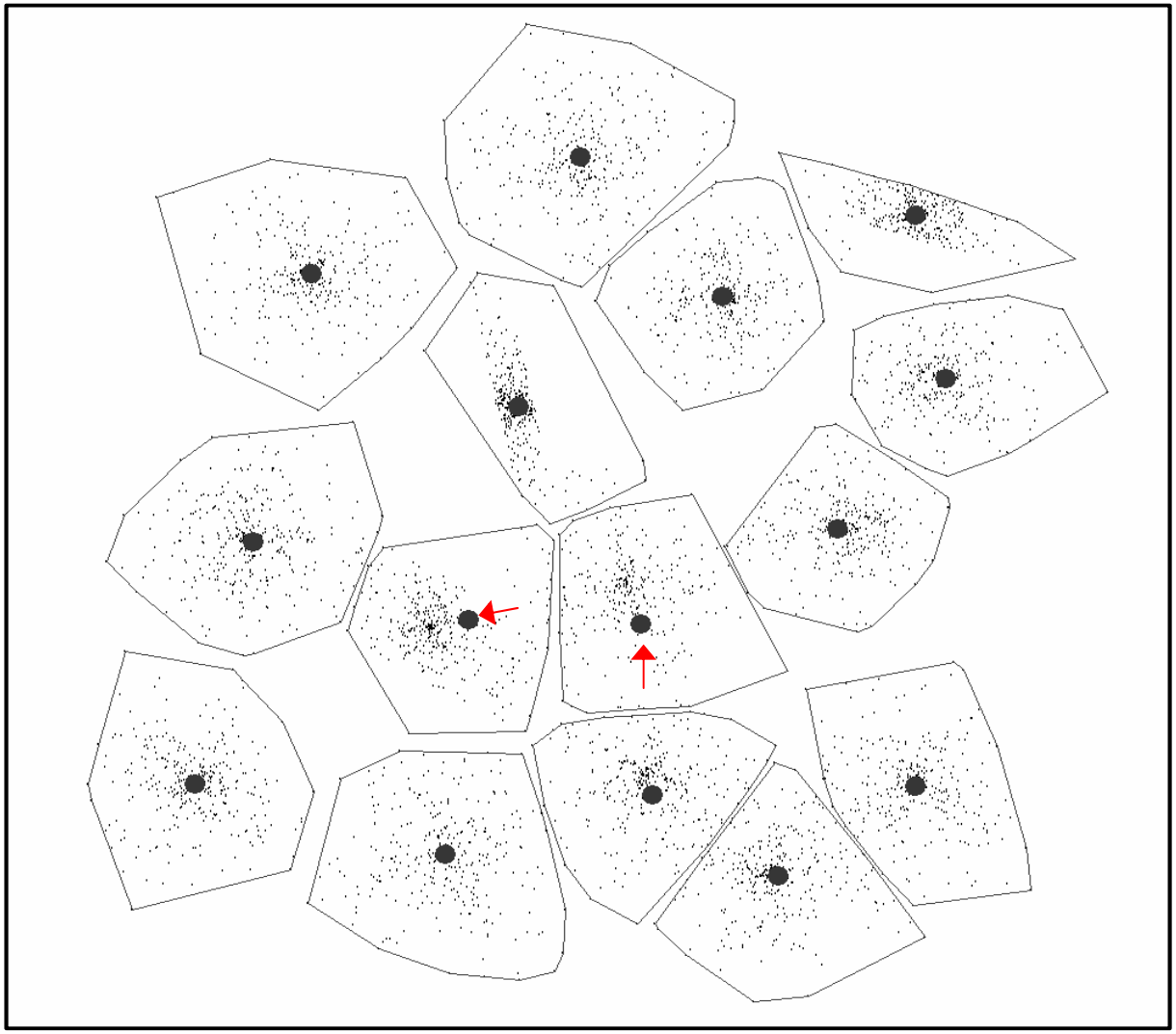
Iterate by k-means

17th iteration



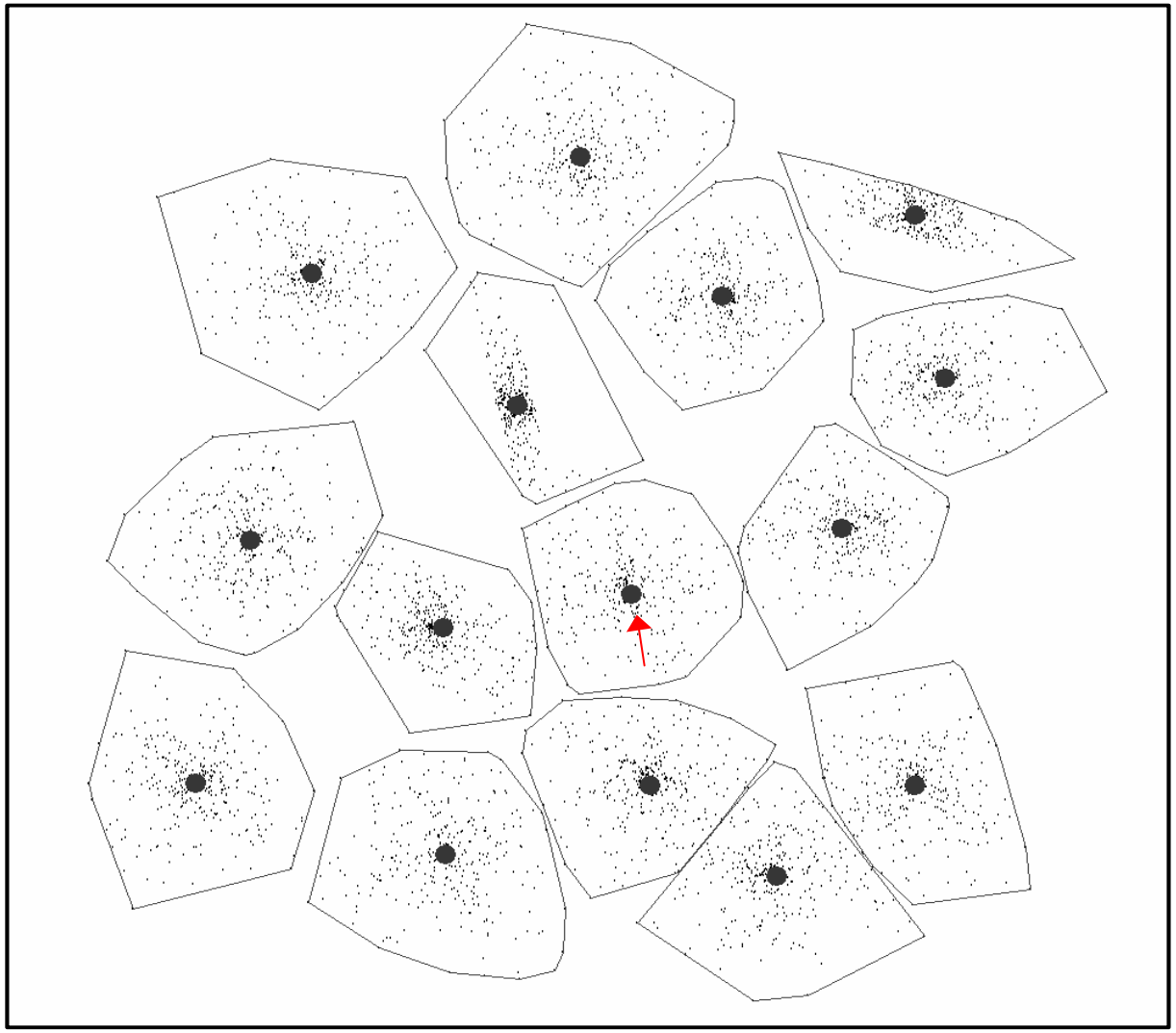
Iterate by k-means

18th iteration



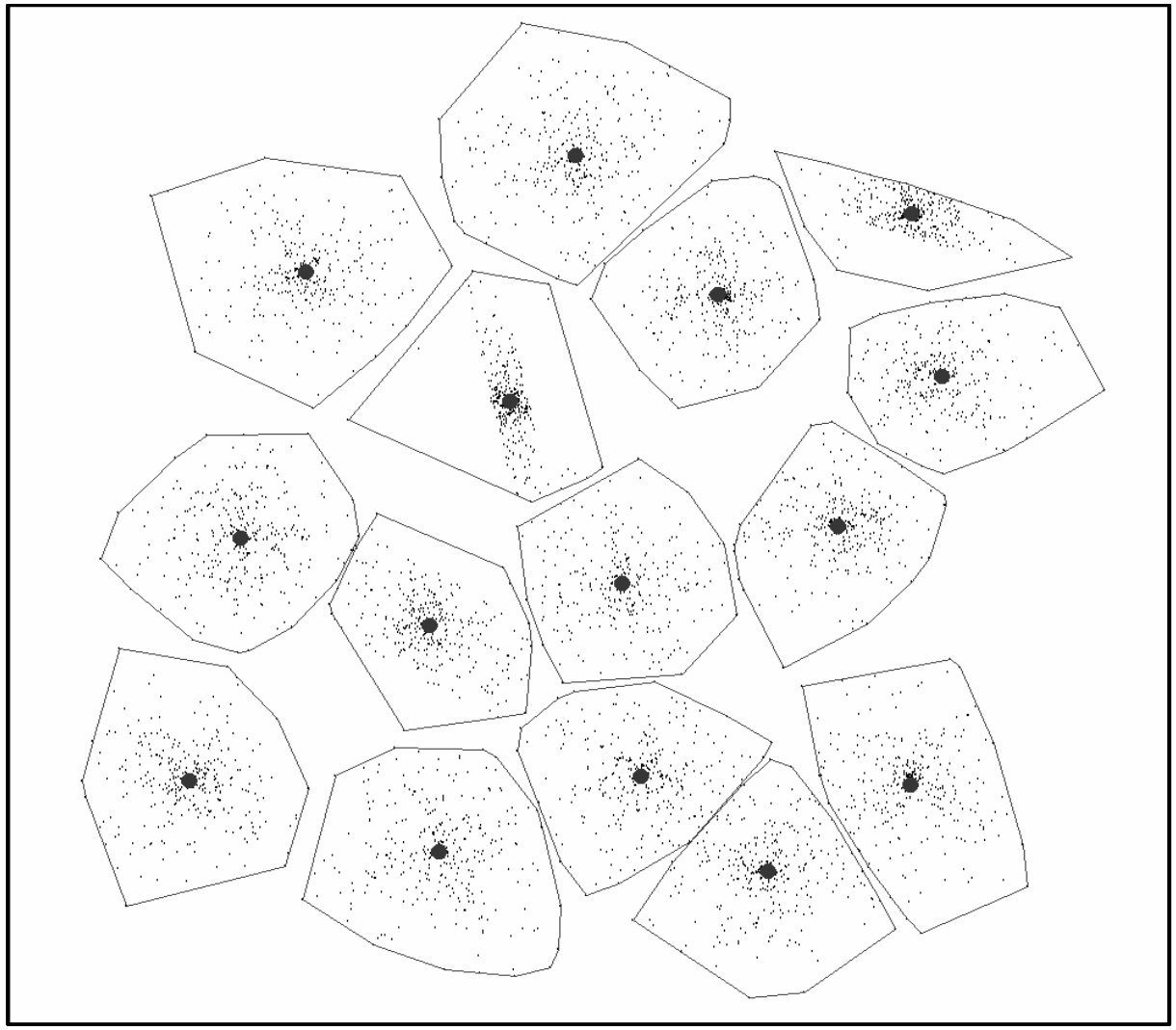
Iterate by k-means

19th iteration



Final result

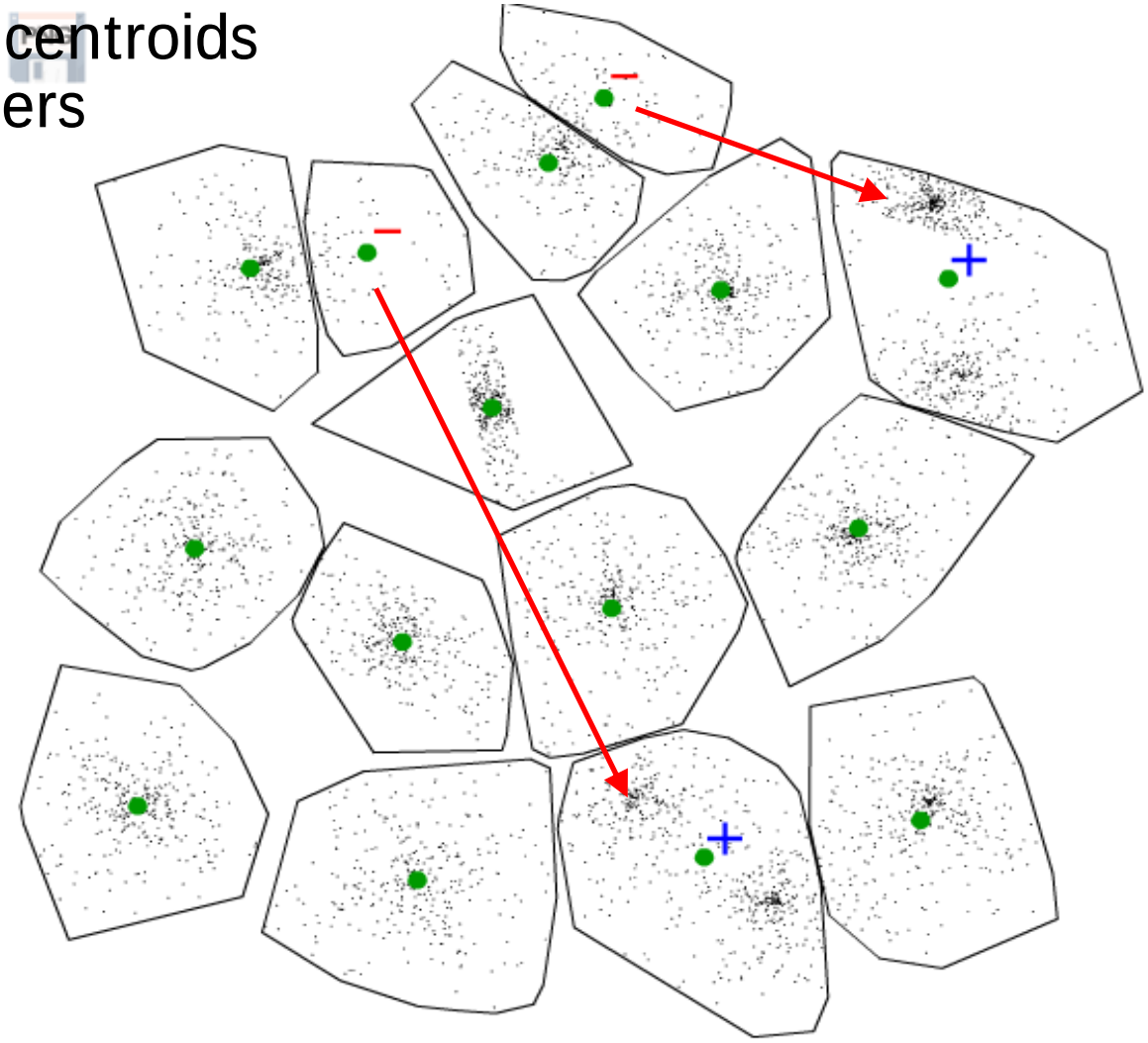
25 iterations



Problems of k-means

Distance of clusters

Cannot move centroids between clusters far away

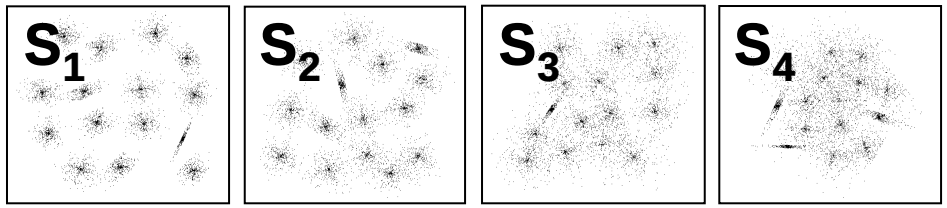


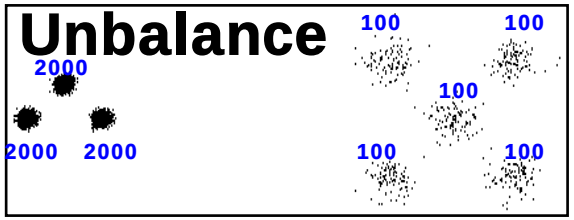
Data and methodology

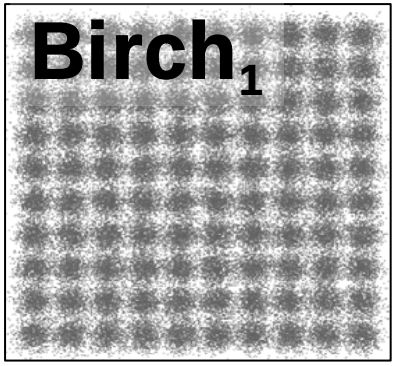
Clustering basic benchmark

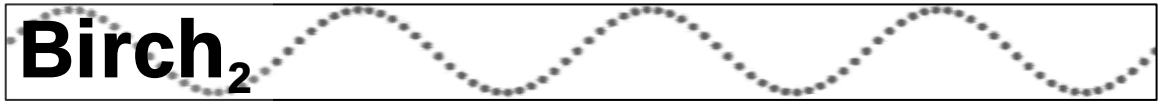
Fränti and Sieranoja
K-means properties on six clustering benchmark datasets
Applied Intelligence, 2018.

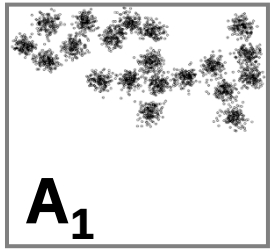
Dataset	Varying	Size	Dimensions	Clusters	Per cluster
A	Number of clusters	3000-7500	2	20-50	150
S	Overlap	5000	2	15	333
Dim	Dimensions	1024	32-1024	16	64
G2	Dimensions + overlap	2048	2-1024	2	1024
Birch	Structure	100,000	2	100	1000
Unbalance	Balance	6500	2	8	100-2000

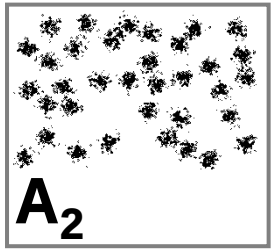


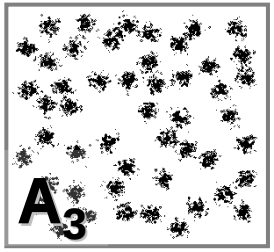


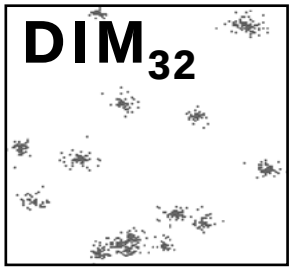


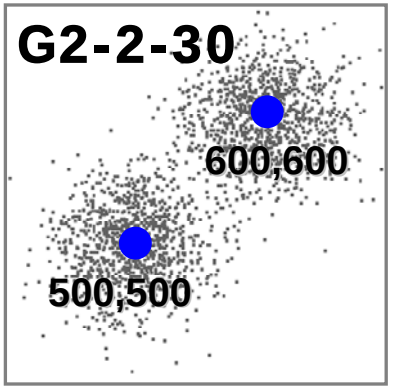






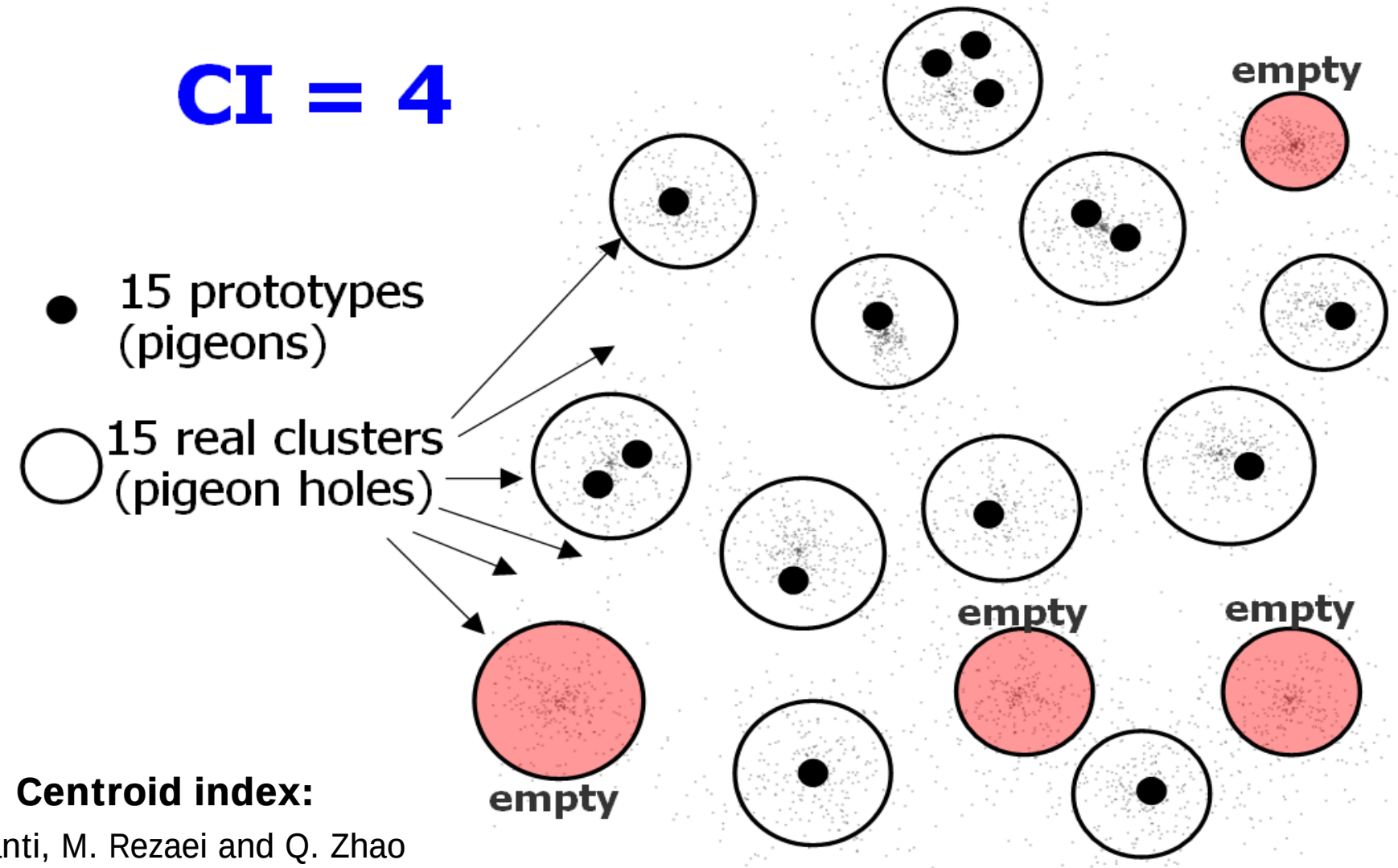






Centroid index

Requires ground truth

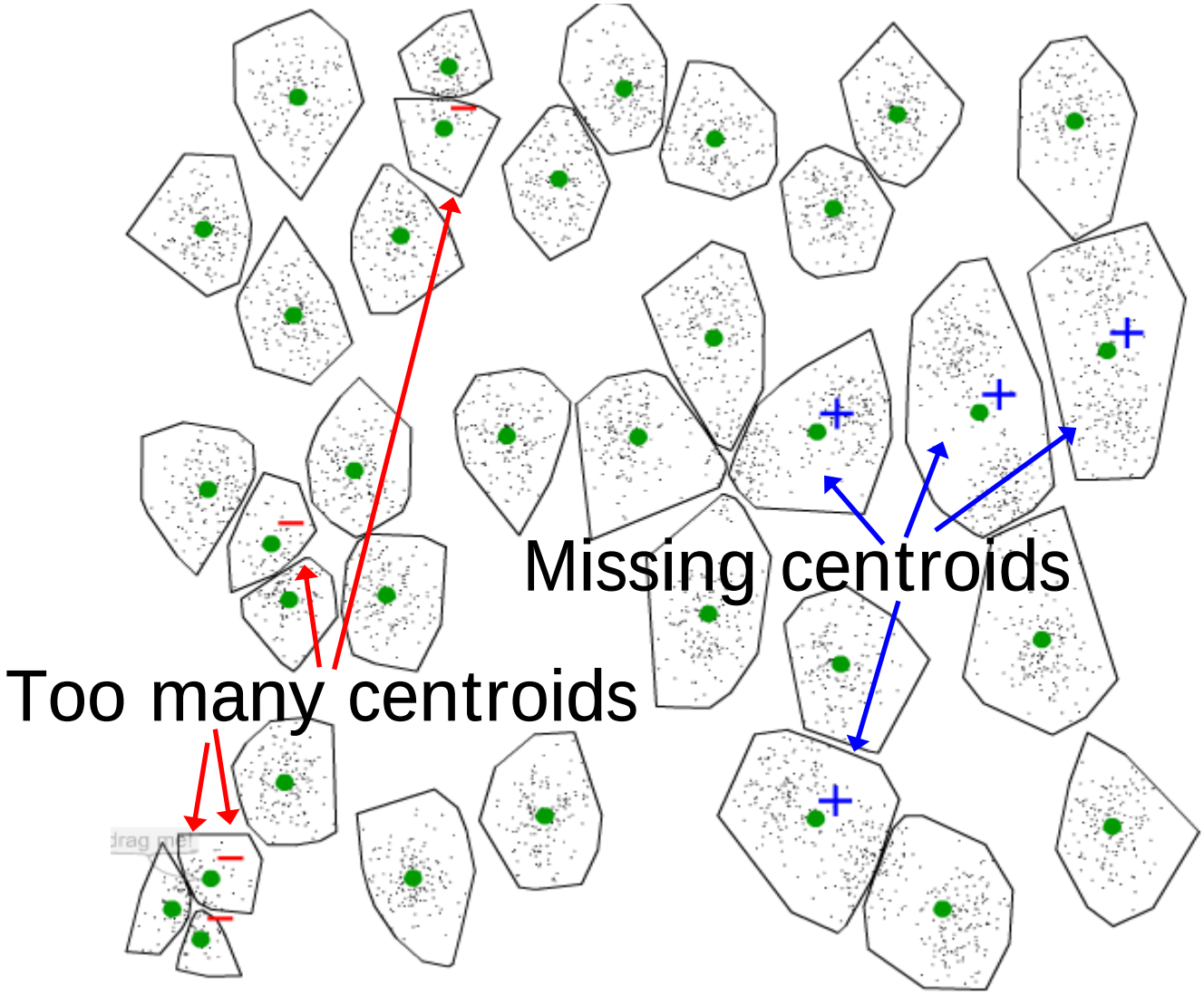


CI = Centroid index:

P. Fränti, M. Rezaei and Q. Zhao
"Centroid index: cluster level similarity measure"
Pattern Recognition, 47 (9), 3034-3045, September 2014.

Centroid index example

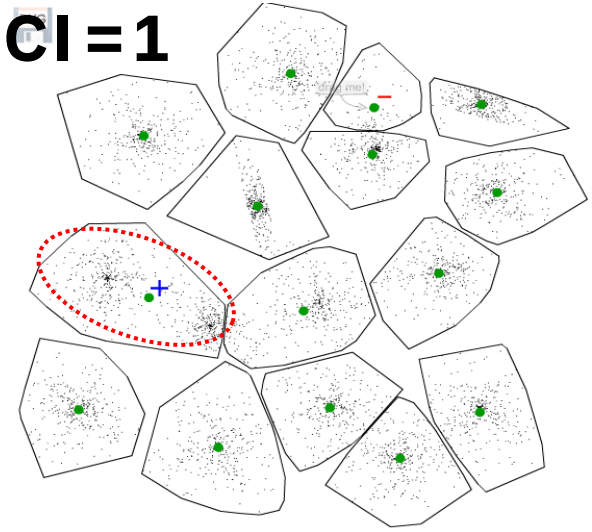
CI = 4

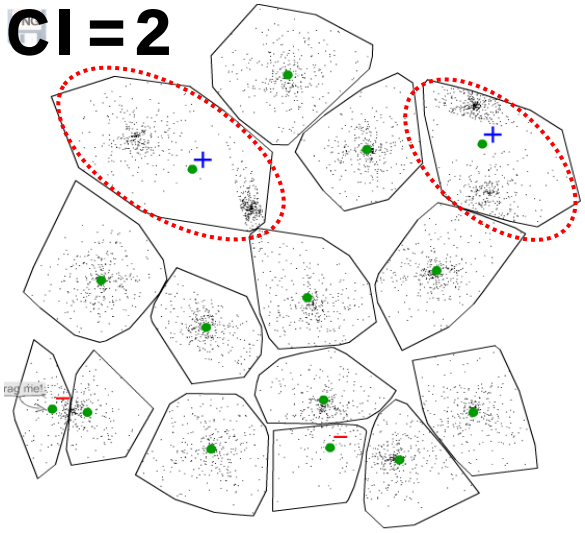


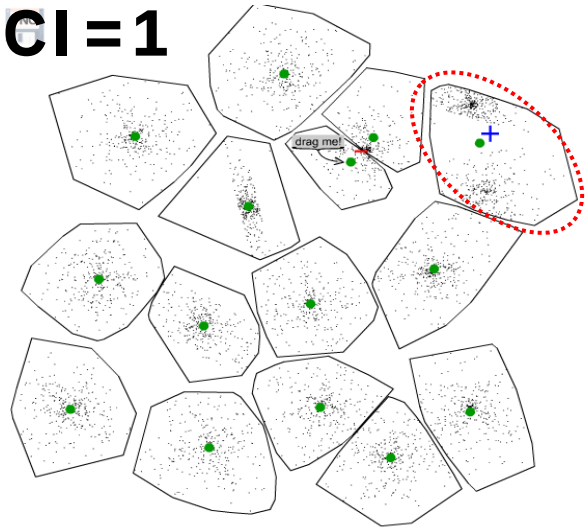
Success rate

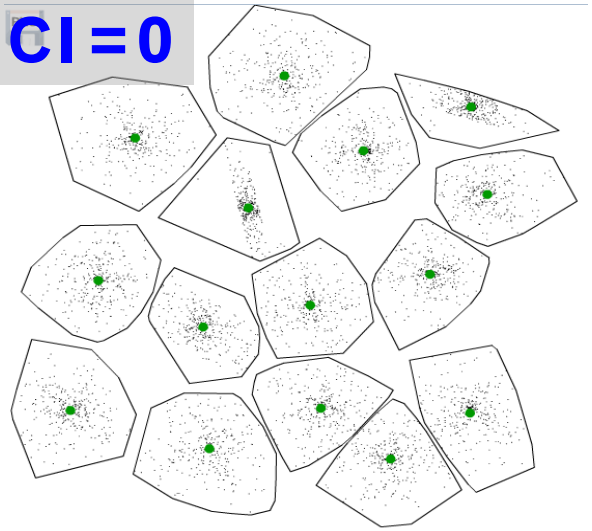
How often CI=0?

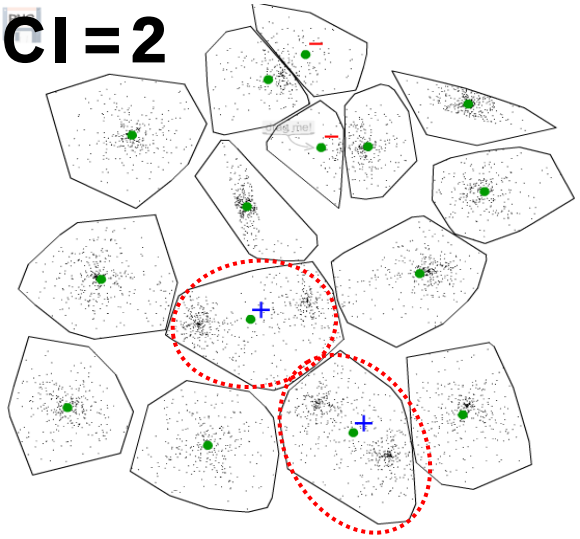
17 %

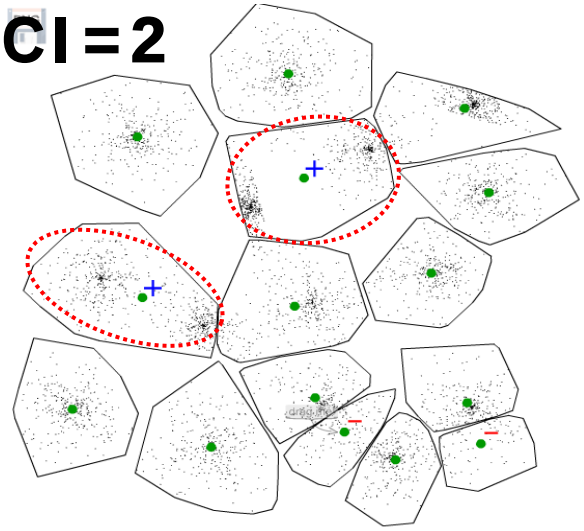












Properties of k-means

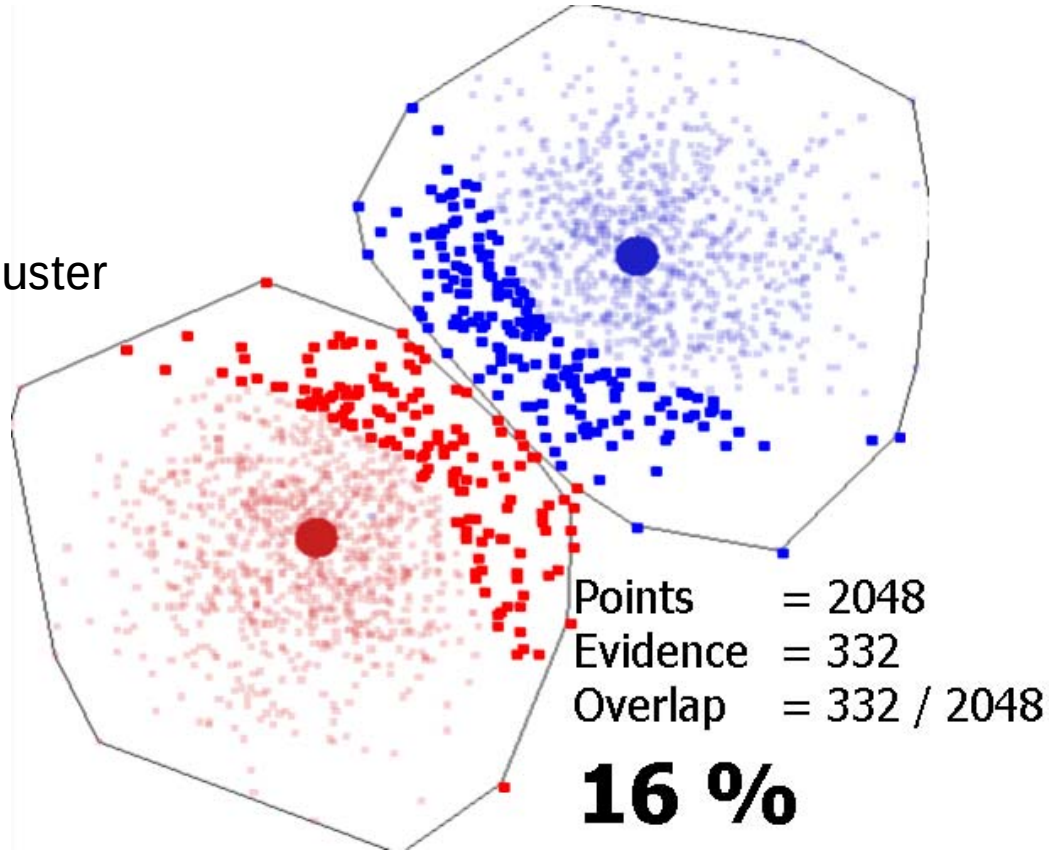
Cluster overlap

Definition

$$overlap = \frac{1}{N} \cdot \sum ov(d_1, d_2)$$

$$ov(d_1, d_2) = \begin{cases} 1, & d_1 > d_2 \\ 0, & d_1 \leq d_2 \end{cases}$$

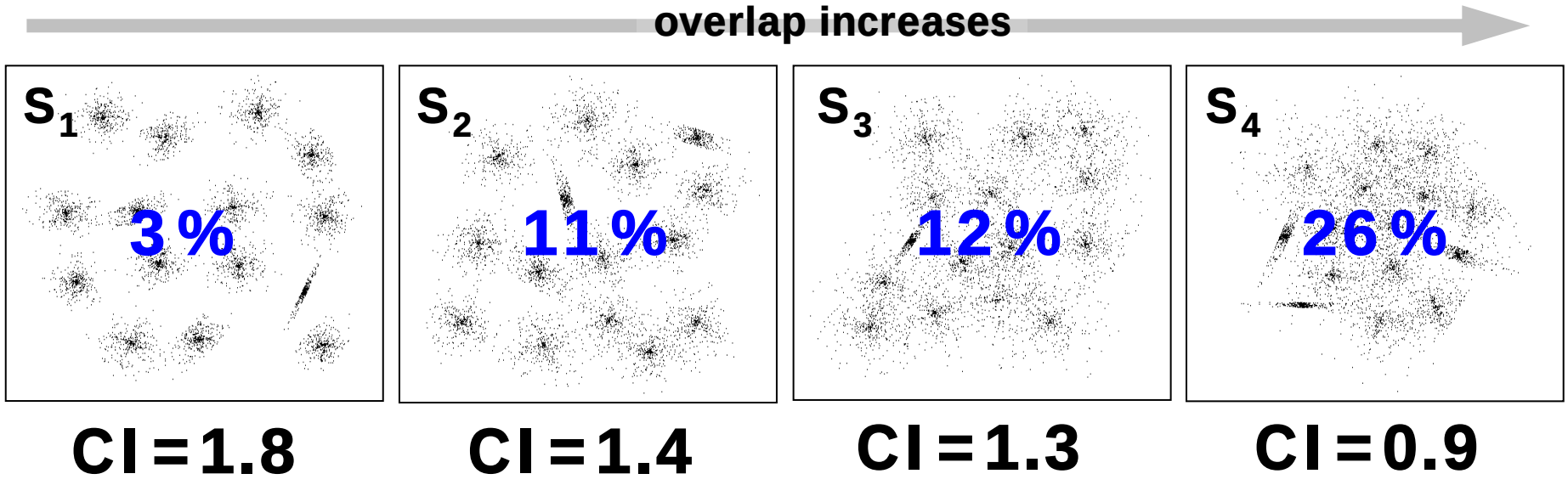
d_1 = distance to nearest centroid
d_2 = distance to nearest in **other** cluster



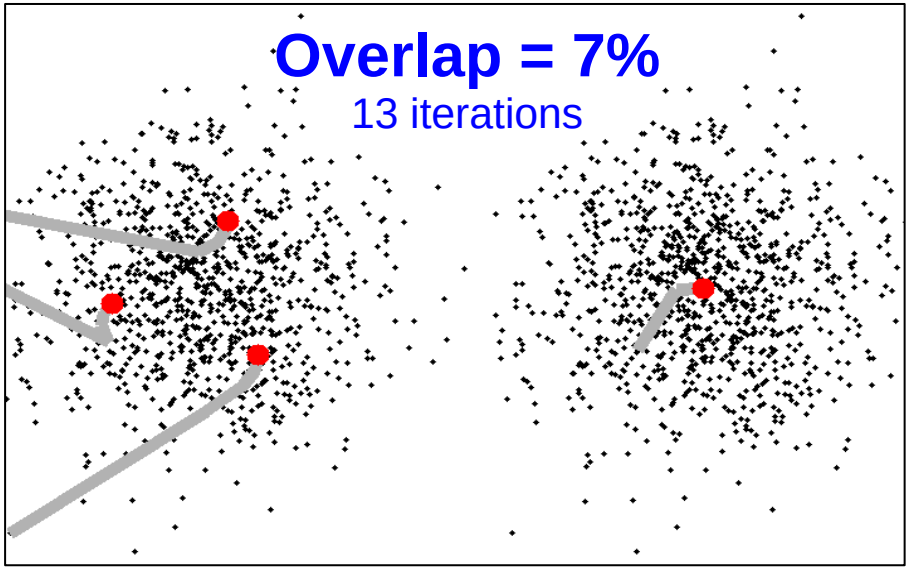
Dependency on overlap

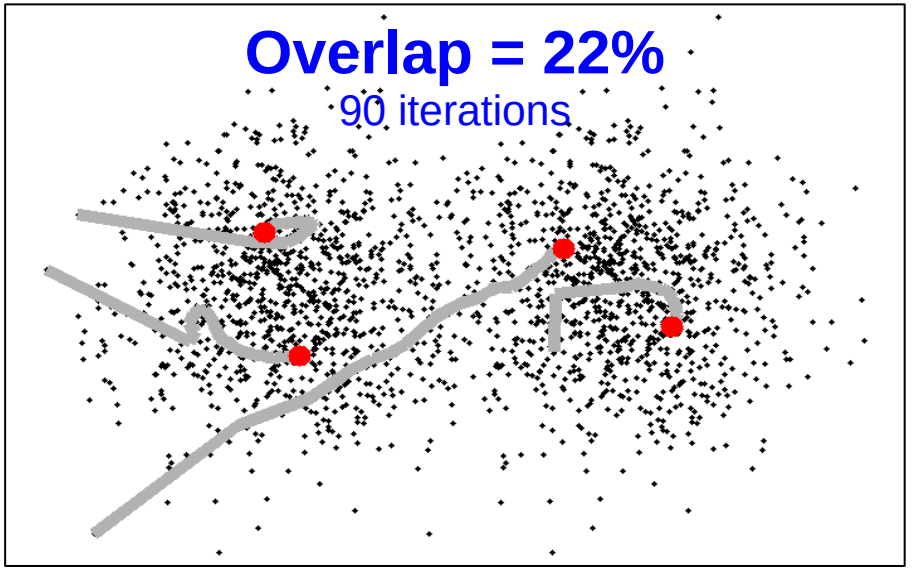
S datasets

Success rates and CI-values:



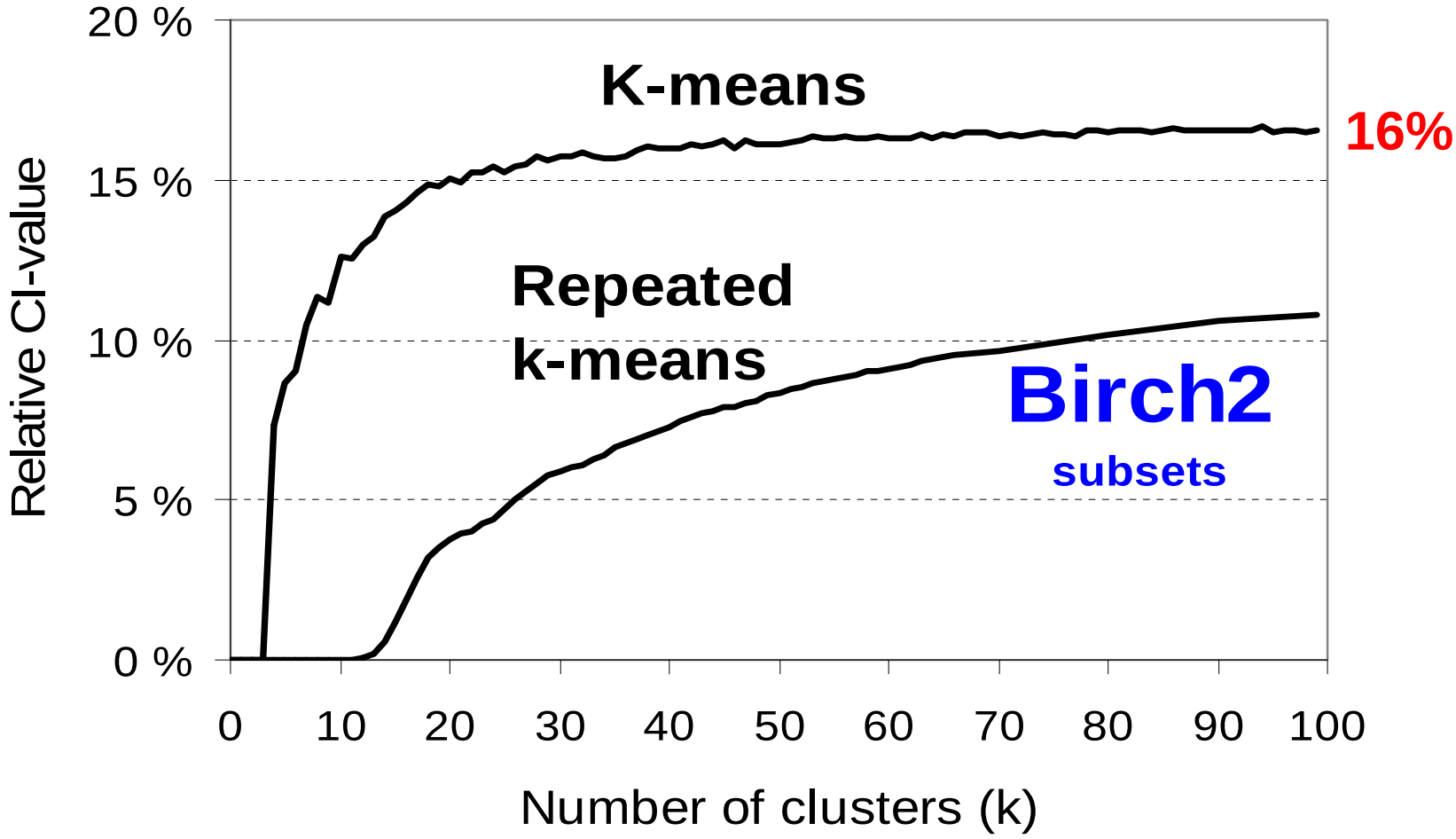
Why overlap helps?





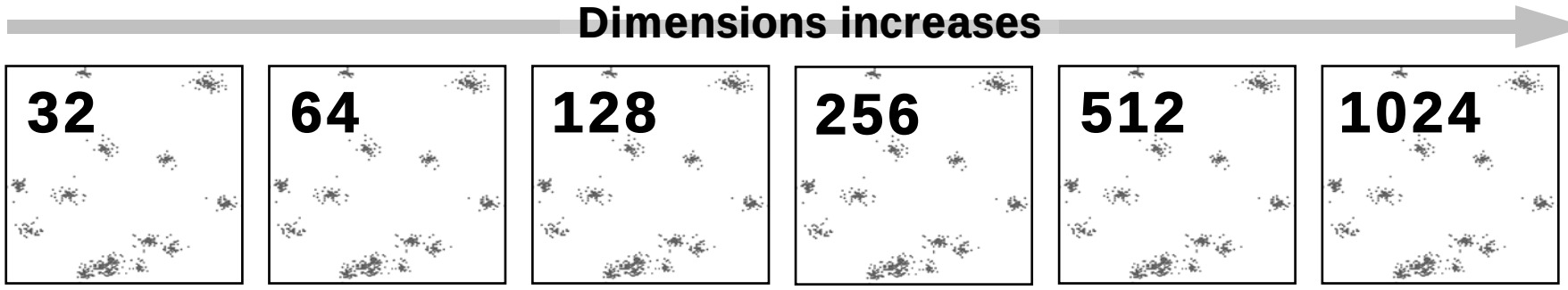
Linear dependency on clusters (k)

Birch2 dataset



Dependency on dimensions

DIM datasets

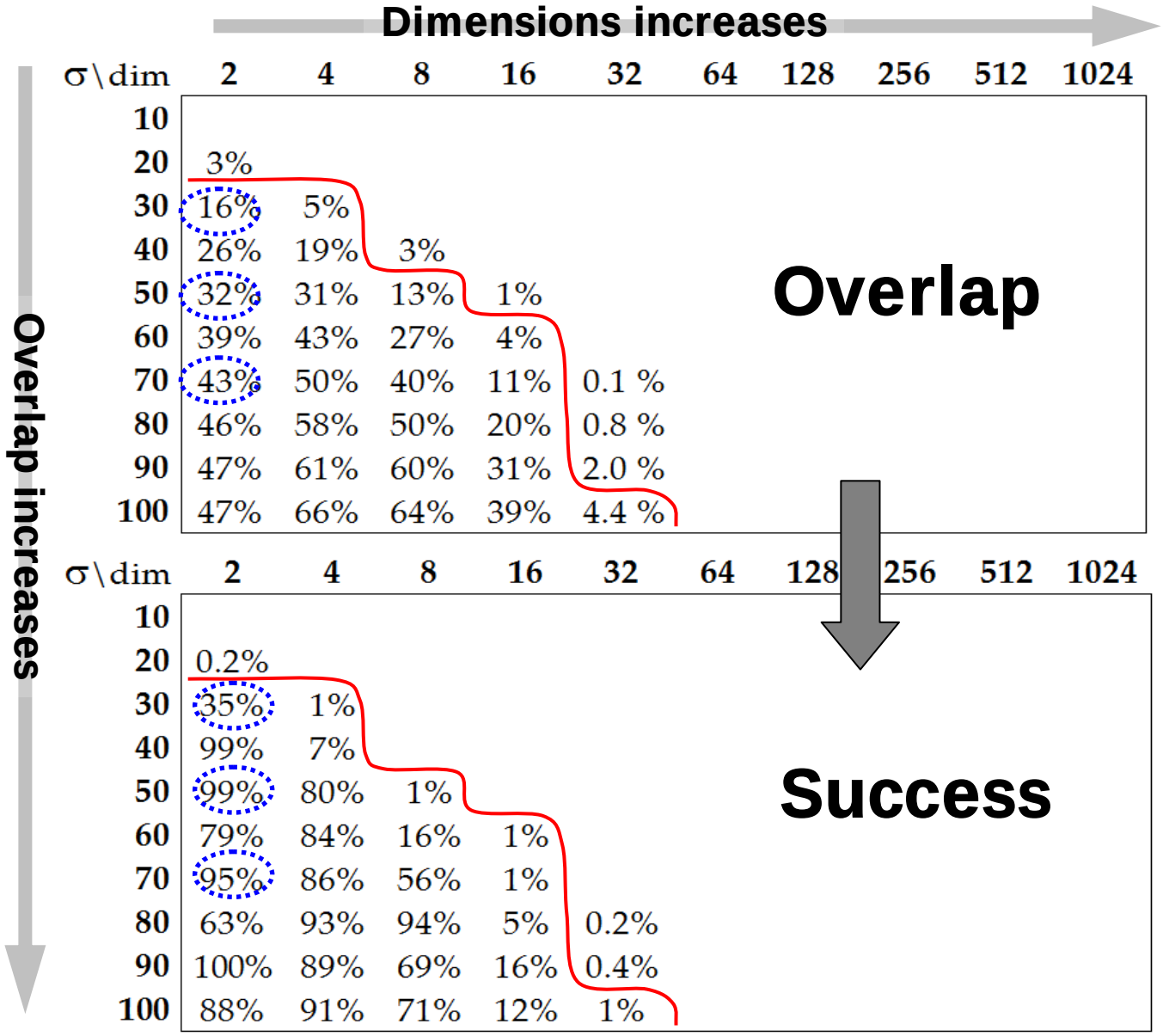


CI:	3.6	3.5	3.8	3.8	3.9	3.7

Success rate: **0 %**

Lack of overlap is the cause!

G2 datasets

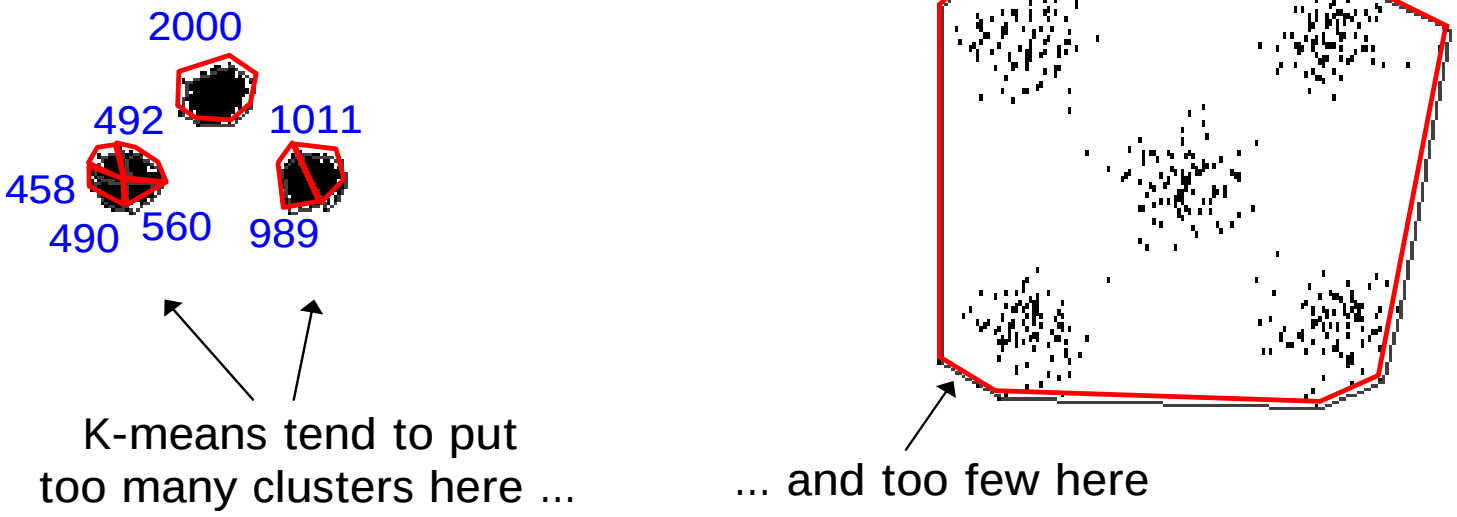


Correlation:

0.91

Effect of unbalance

Unbalabce datasets



Success:

0 %

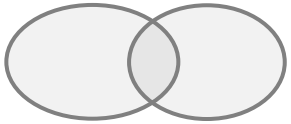
Average CI:

3.9

→ Problem originates from the random initialization.

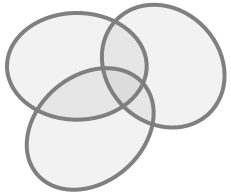
Summary of k-means properties

1. Overlap



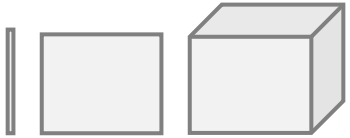
Good!

2. Number of clusters



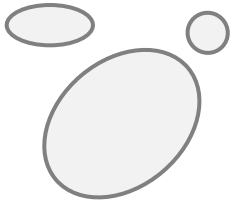
Linear dependency

3. Dimensionality



No direct effect

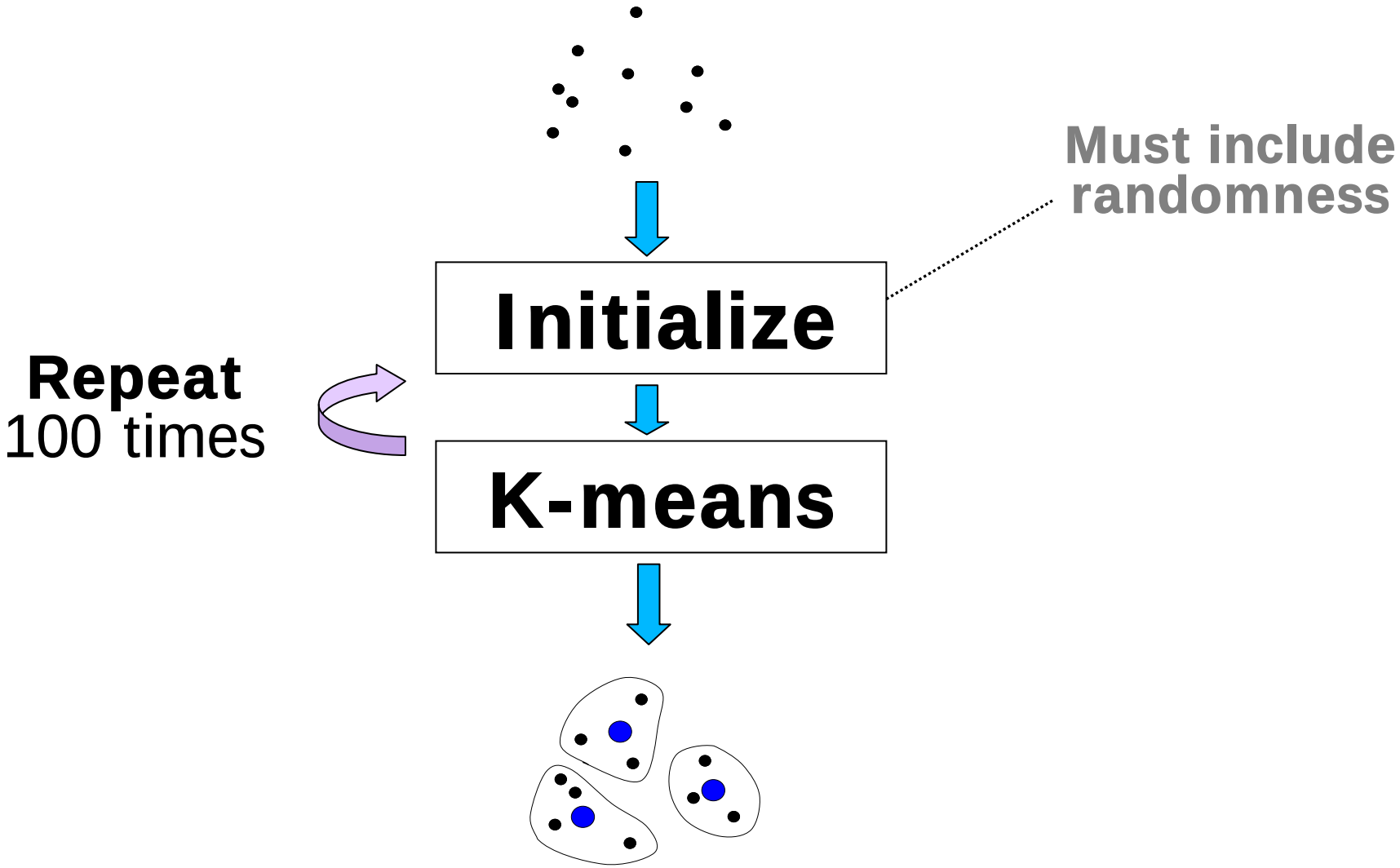
4. Unbalance of cluster sizes



Bad!

How to improve?

Repeated k-means (RKM)



How to initialize?

Some obvious heuristics:

- Furthest point
- Sorting
- Density
- Projection

Clear state-of-the-art is missing:

- No single technique outperforms others in all cases.
- Initialization not significantly easier than the clustering itself.
- K-means can be used as fine-tuner with almost anything.

Another desperate effort:

- Repeat it **LOTS OF** times

Initialization techniques

Criteria

1. Simple to implement
2. Lower (or equal) time complexity than k-means
3. No additional parameters
4. Include randomness

Requirements for initialization

1. **Simple** to implement

- Random centroids has 2 functions + **26 lines** of code
- Repeated k-means 5 functions + **162 lines** of code
- Simpler than k-means itself

2. **Faster** than k-means

- K-means real-time (**<1 s**) up to N≈10,000
- $O(IkN) \approx 25 \cdot N^{1.5}$ --- assuming I≈25 and $k=\sqrt{N}$
- Must be faster than **$O(N^2)$**

Simplicity of algorithms

	A	B	C
Random	0	2	26
Rep. K-Means	1	5	162
Random swap	1	7	226
Agglomerative	0	12	317
SPLIT	0	22	947
GA	2	21	573
FCM	2	11	295
GMM	2	44	1111

A. Parameters
B. Functions
C. Lines of code

Kinnunen, Sidoroff, Tuononen and Fränti,
"Comparison of clustering methods: a case study of text-independent speaker modeling"
Pattern Recognition Letters, 2011.

Alternative:
A better algorithm

Random Swap (RS)

Random Swap(X) $\rightarrow$ C, P

$C \leftarrow$ Select random representatives(X);
$P \leftarrow$ Optimal partition(X, C);
REPEAT T times
  (C^{new},j) $\leftarrow$ Random swap(X, C);
  $P^{new} \leftarrow$ Local repartition(X, C^{new}, P, j);
  C^{new}, $P^{new} \leftarrow$ *Kmeans*(X, C^{new}, P^{new});
  IF $f(C^{new}, P^{new}) < f(C, P)$ THEN
    (C, P) $\leftarrow$ C^{new}, P^{new};
RETURN (C, P);

CI = 0

P. Fränti, "Efficiency of random swap clustering", *Journal of Big Data*, 2018

Genetic Algorithm (GA)

GeneticAlgorithm(X) $\rightarrow$ (C, P)
  FOR $i \leftarrow$1 TO Z DO
    $C^i \leftarrow$ RandomCodebook(X);
    $P^i \leftarrow$ OptimalPartition(X, C^i);
  SortSolutions(C,P);
  REPEAT
    {C,P} $\leftarrow$ CreateNewSolutions({C,P});
    SortSolutions(C,P);
  UNTIL no improvement;

CreateNewSolutions({C, P}) $\rightarrow$ {C^{new}, P^{new} }
  $C^{new\text{-}1}$, $P^{new\text{-}1} \leftarrow C^1$, P^1;
  FOR $i \leftarrow$2 TO Z DO
    (a,b) $\leftarrow$ SelectNextPair;
    $C^{new\text{-}i}$, $P^{new\text{-}i} \leftarrow$ Cross(C^a, P^a, C^b, P^b);
    IterateK-Means($C^{new\text{-}i}$, $P^{new\text{-}i}$);

Cross(C^1, P^1, C^2, P^2) $\rightarrow$ (C^{new}, P^{new})
  $C^{new} \leftarrow$ CombineCentroids(C^1, C^2);
  $P^{new} \leftarrow$ CombinePartitions(P^1, P^2);
  $C^{new} \leftarrow$ UpdateCentroids(C^{new}, P^{new});
  RemoveEmptyClusters(C^{new}, P^{new});
  IS(C^{new}, P^{new});

CombineCentroids(C^1, C^2) $\rightarrow$ C^{new}
  $C^{new} \leftarrow C^1 \cup C^2$

CombinePartitions(C^{new}, P^1, P^2) $\rightarrow$ P^{new}
  FOR $i \leftarrow$1 TO N DO
    IF $\left\|x_i - c_{p_i^1}\right\|^2 \leq \left\|x_i - c_{p_i^2}\right\|^2$ THEN
      $p_i^{new} \leftarrow p_i^1$
    ELSE
      $p_i^{new} \leftarrow p_i^2$
  END-FOR

UpdateCentroids(C^1, C^2) $\rightarrow$ C^{new}
  FOR $j \leftarrow$1 TO $|C^{new}|$ DO
    $c_j^{new} \leftarrow$ CalculateCentroid(P^{new}, j);

CI = 0

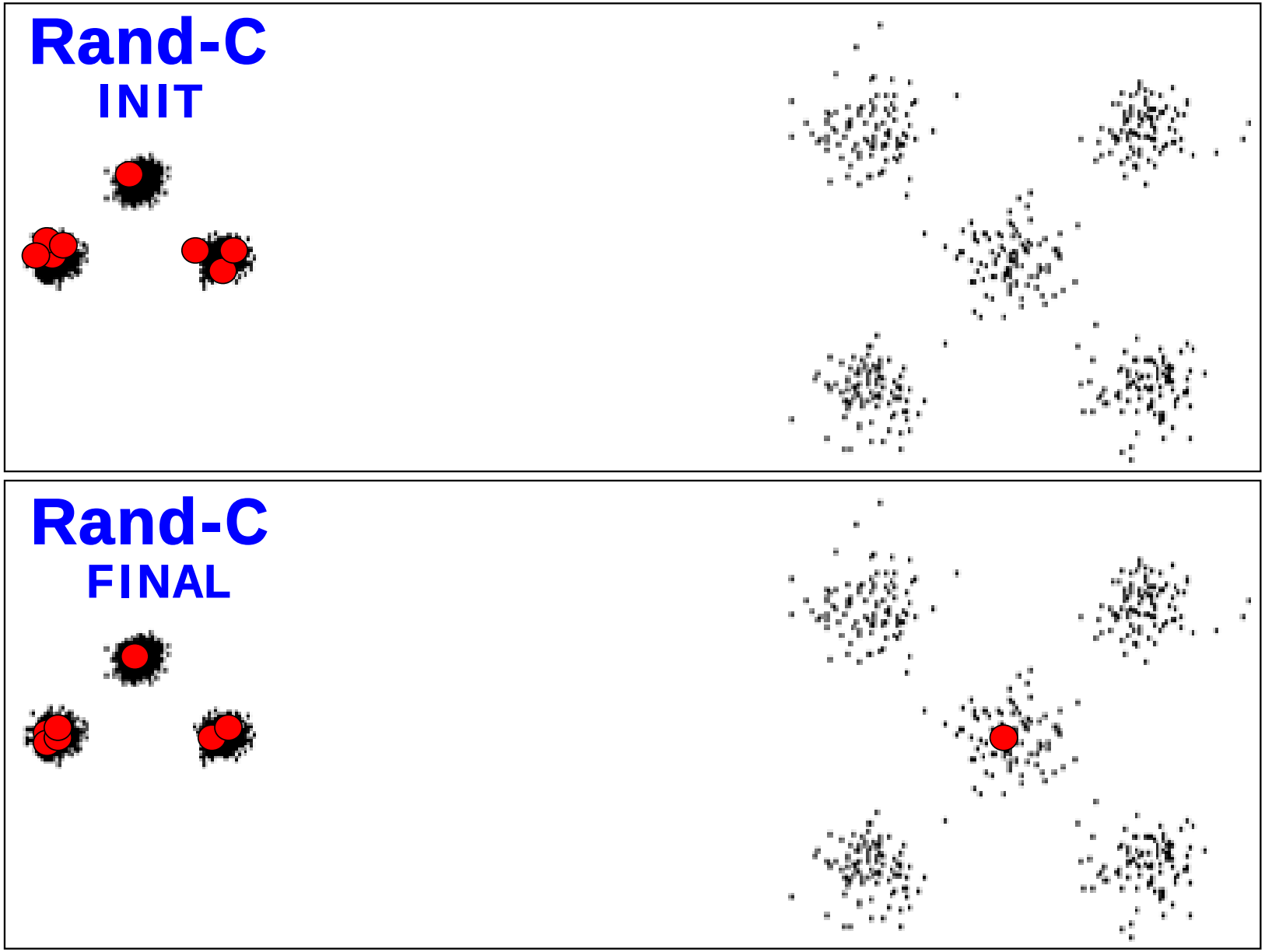
P. Fränti, "Genetic algorithm with deterministic crossover for vector quantization", *Pattern Recognition Letters*, 2000.

Initialization techniques

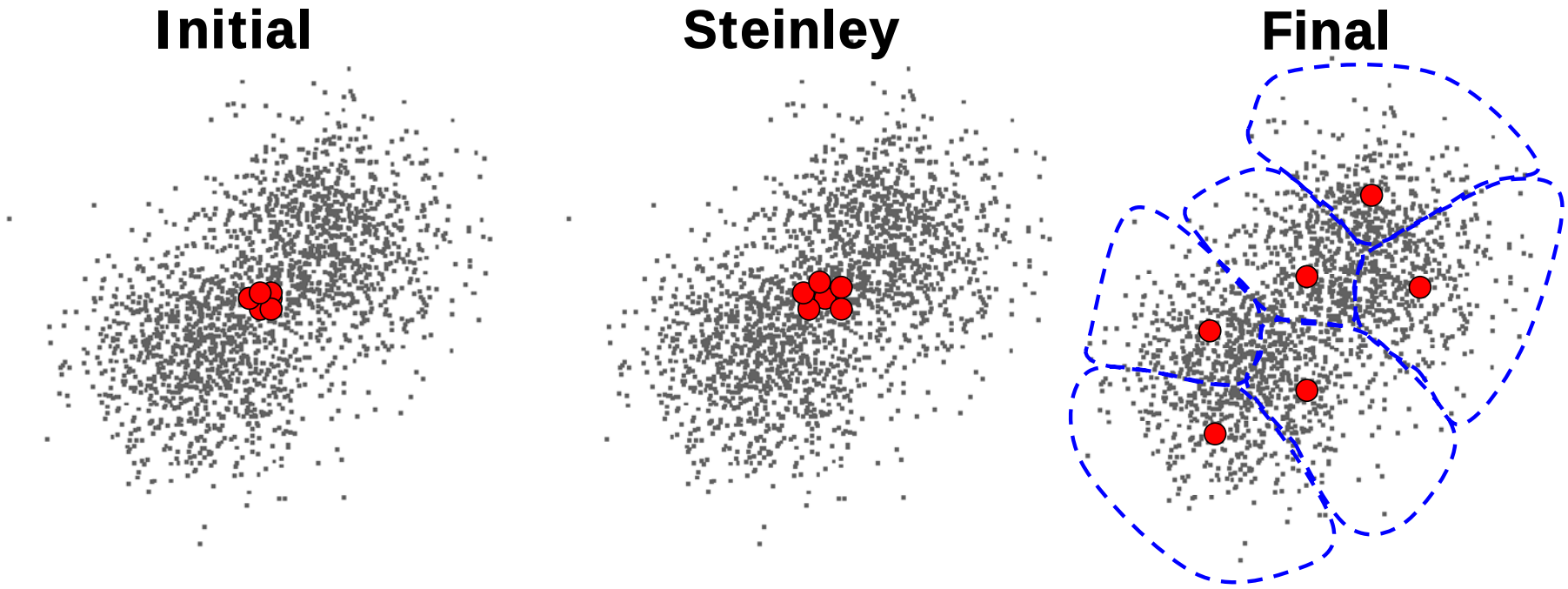
Techniques considered

Technique	Complexity	Time	Random-ized	Parameters
Random partitions	$O(N)$	10 ms	Yes	-
Random centroids	$O(N)$	13 ms	Yes	-
Maxmin	$O(kN)$	16 ms	Modified	-
kmeans++	$O(kN)$	19 ms	Yes	-
Bradley	$O(kN+Rk^2)$	41 ms	Yes	R=10, s=10%
Sorting heuristic	$O(N \log N)$	13 ms	Modified	-
Projection-based	$O(N \log N)$	14 ms	Yes	-
Luxburg	$O(kN \log k)$	29 ms	Yes	-
Split	$O(N \log N)$	67 ms	Yes	k=2

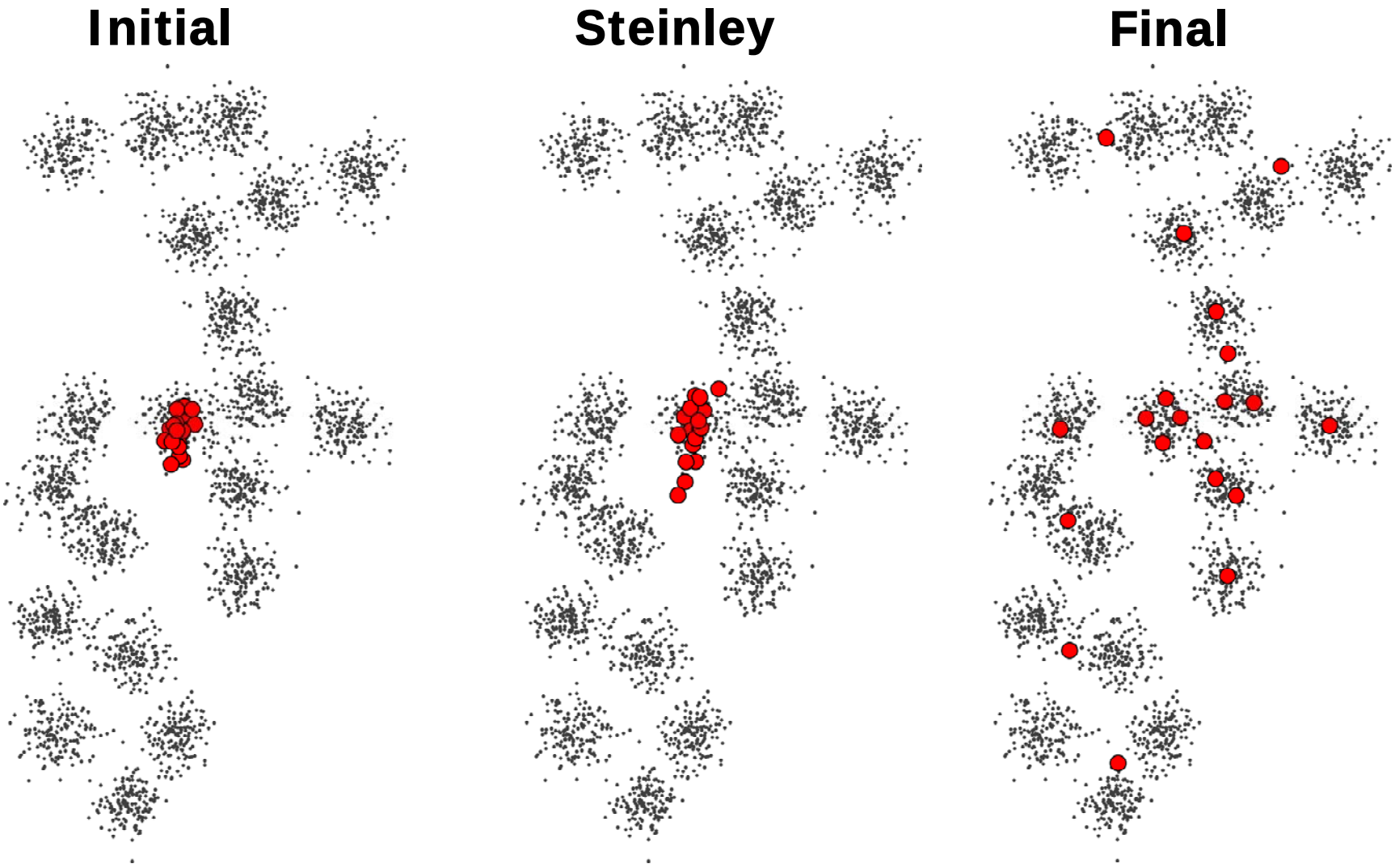
Random centroids



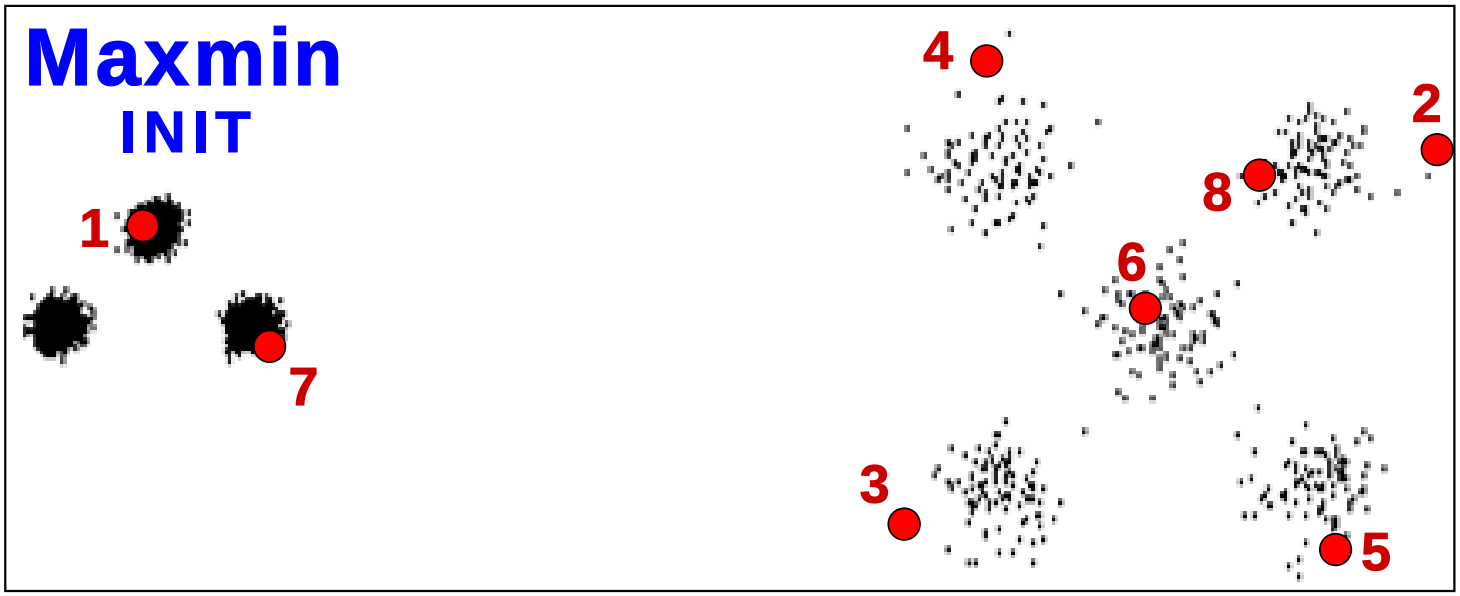
Random partitions

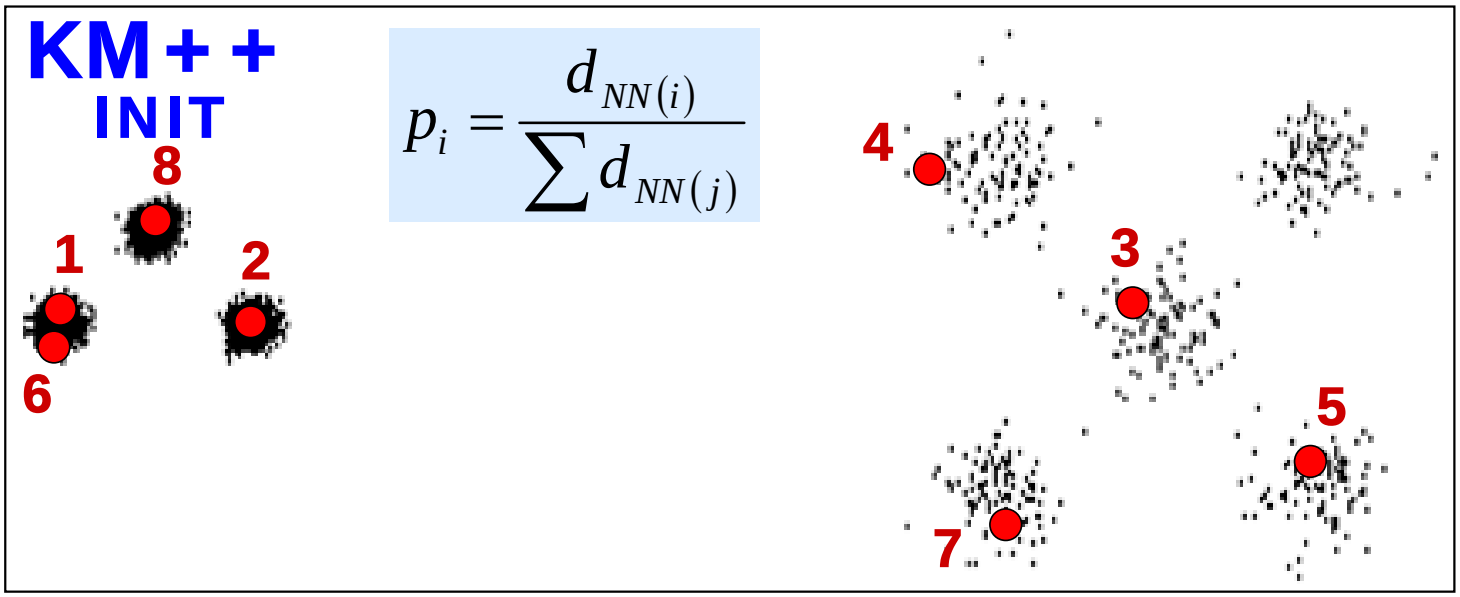


Random partitions

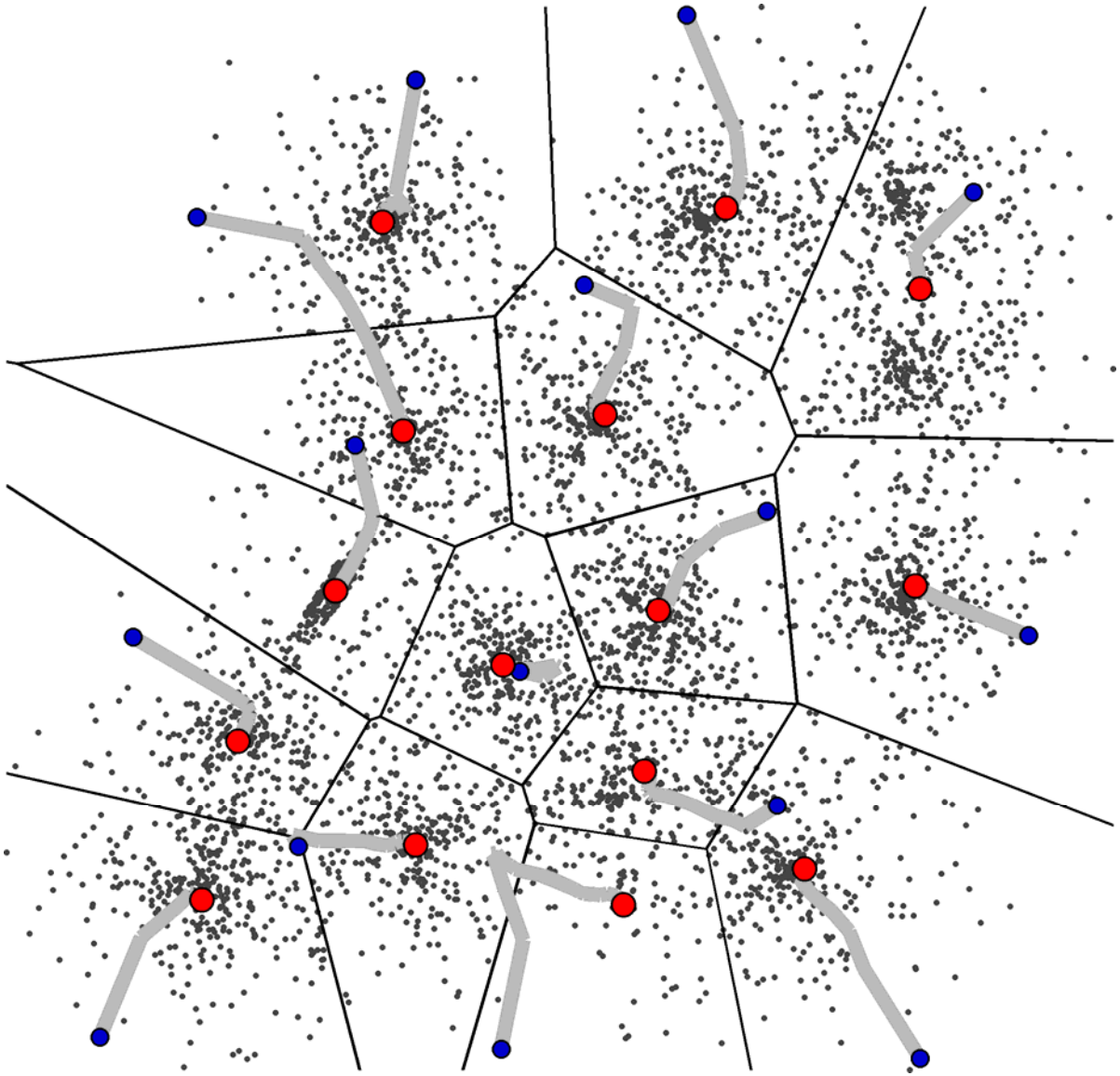


Furthest points (maxmin)





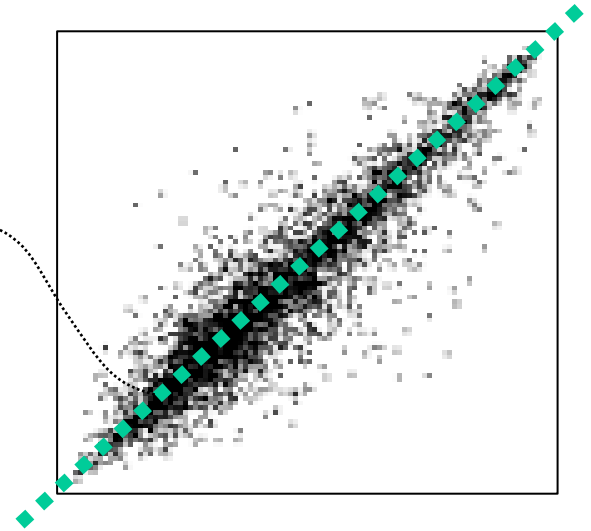
Furthest points (maxmin)



Projection-based initialization

Most common projection axis:

- Diagonal
- Principal axis (PCA)
- Principle curves



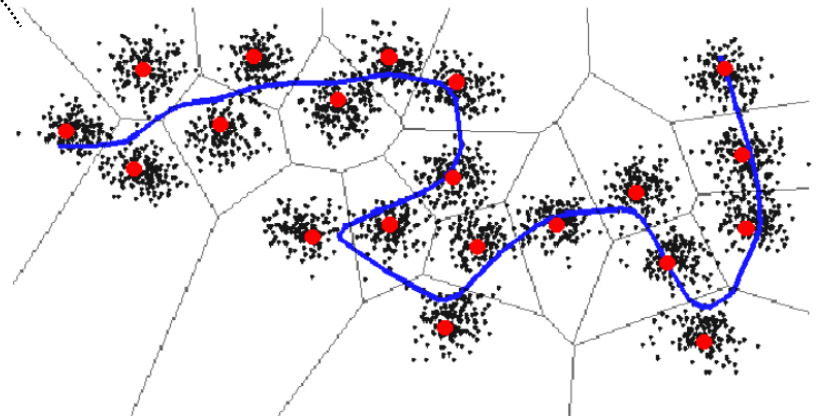
PCA = O(DN)-O(D^2N)

Initialization:

- Uniform partition along axis

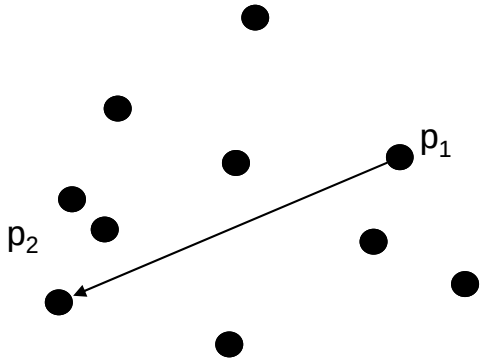
Used in divisive clustering:

- Iterative split

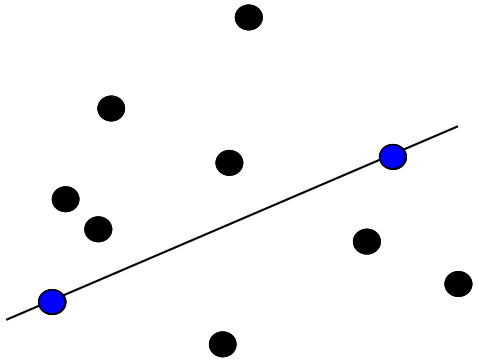


Furthest point projection

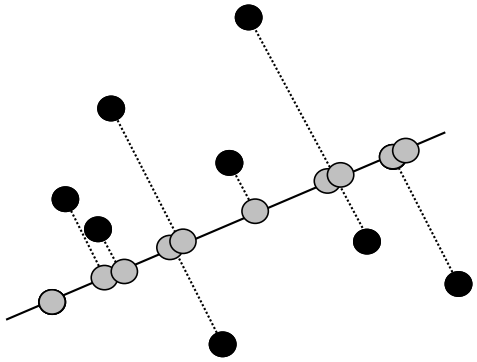
Furthest point



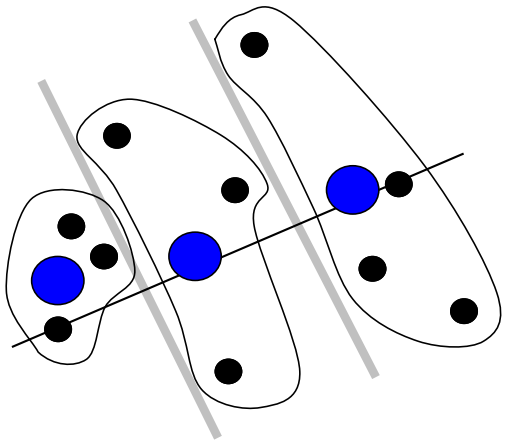
Projection axis



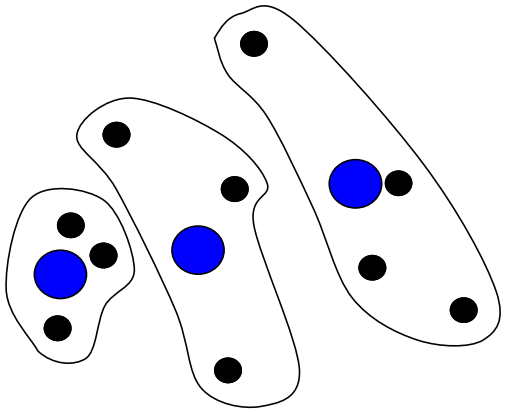
Projected points



Initial clustering



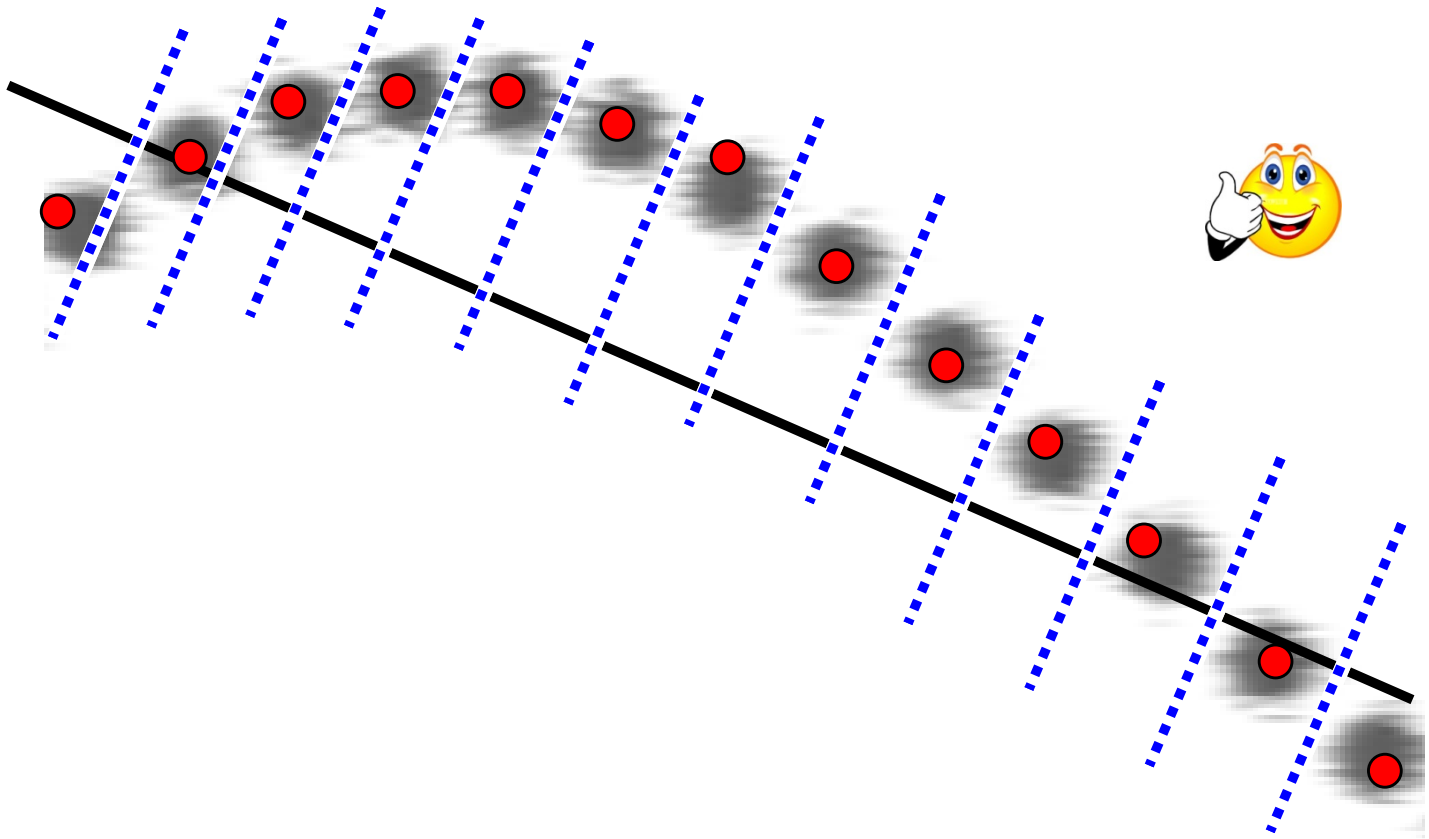
After k-means



Projection example (1)

Good projection!

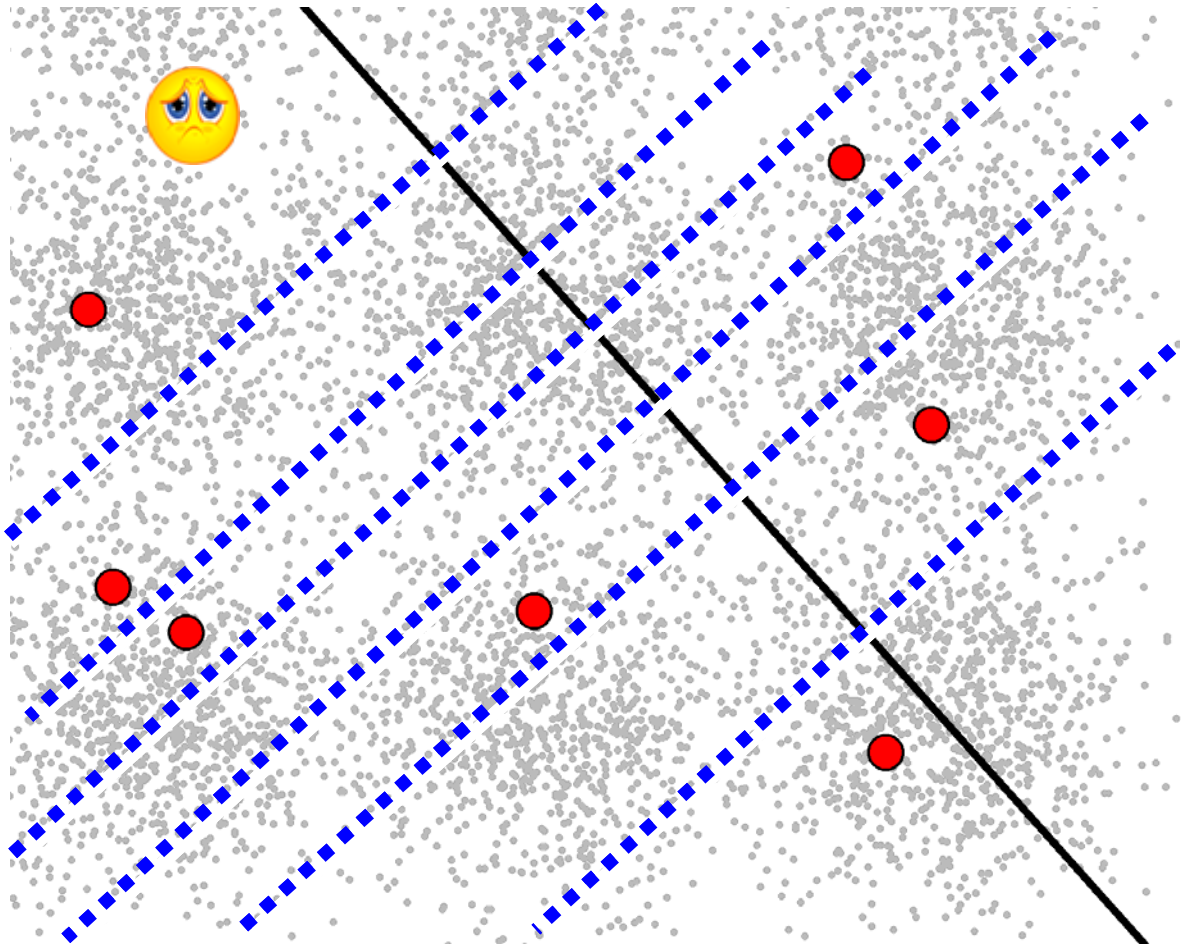
Birch2



Projection example (2)

Bad projection!

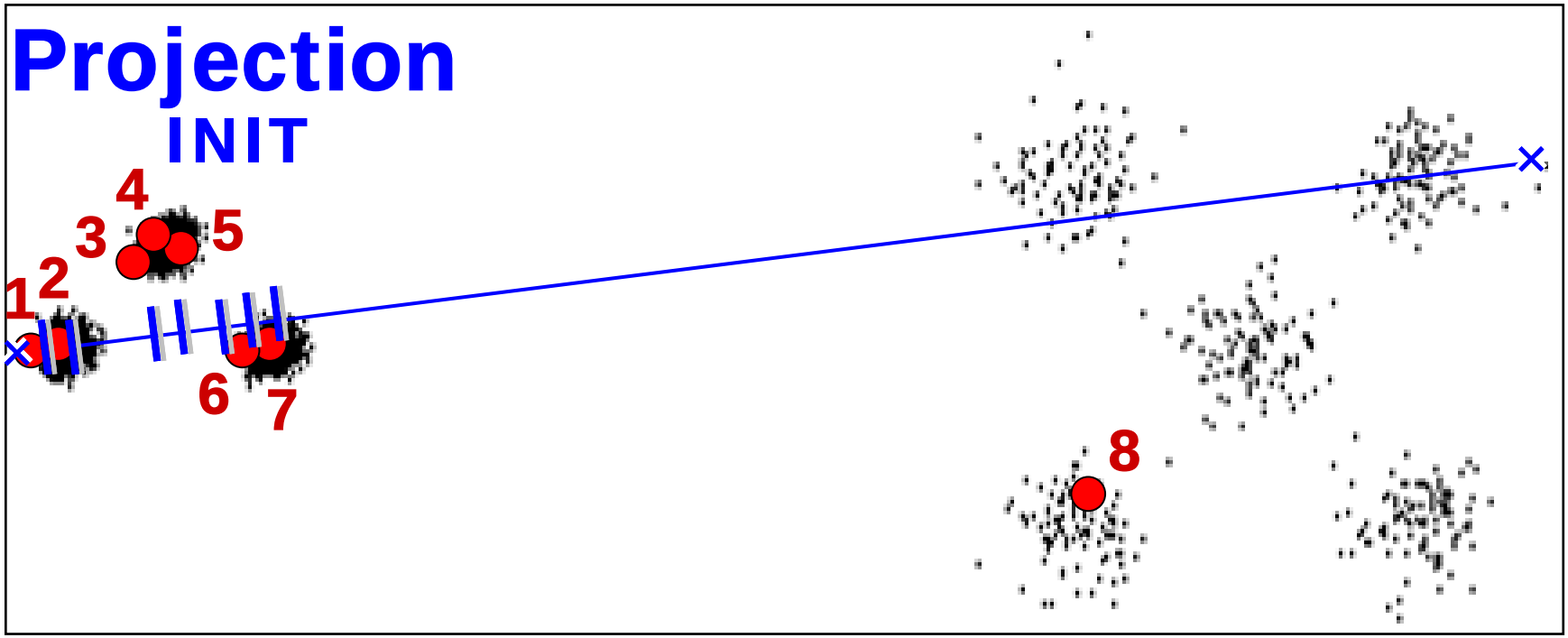
Birch1



Projection example (3)

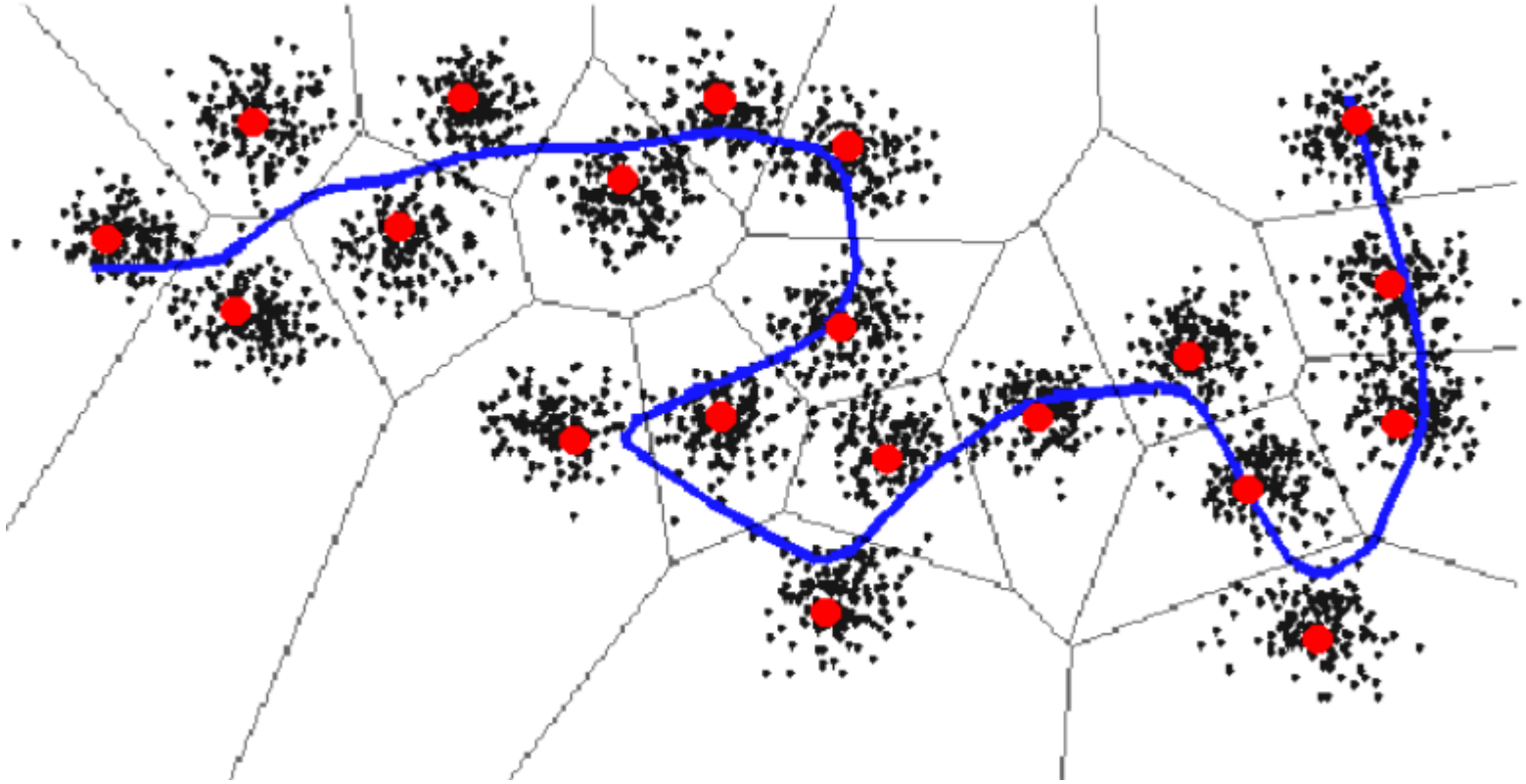
Bad projection!

Unbalance



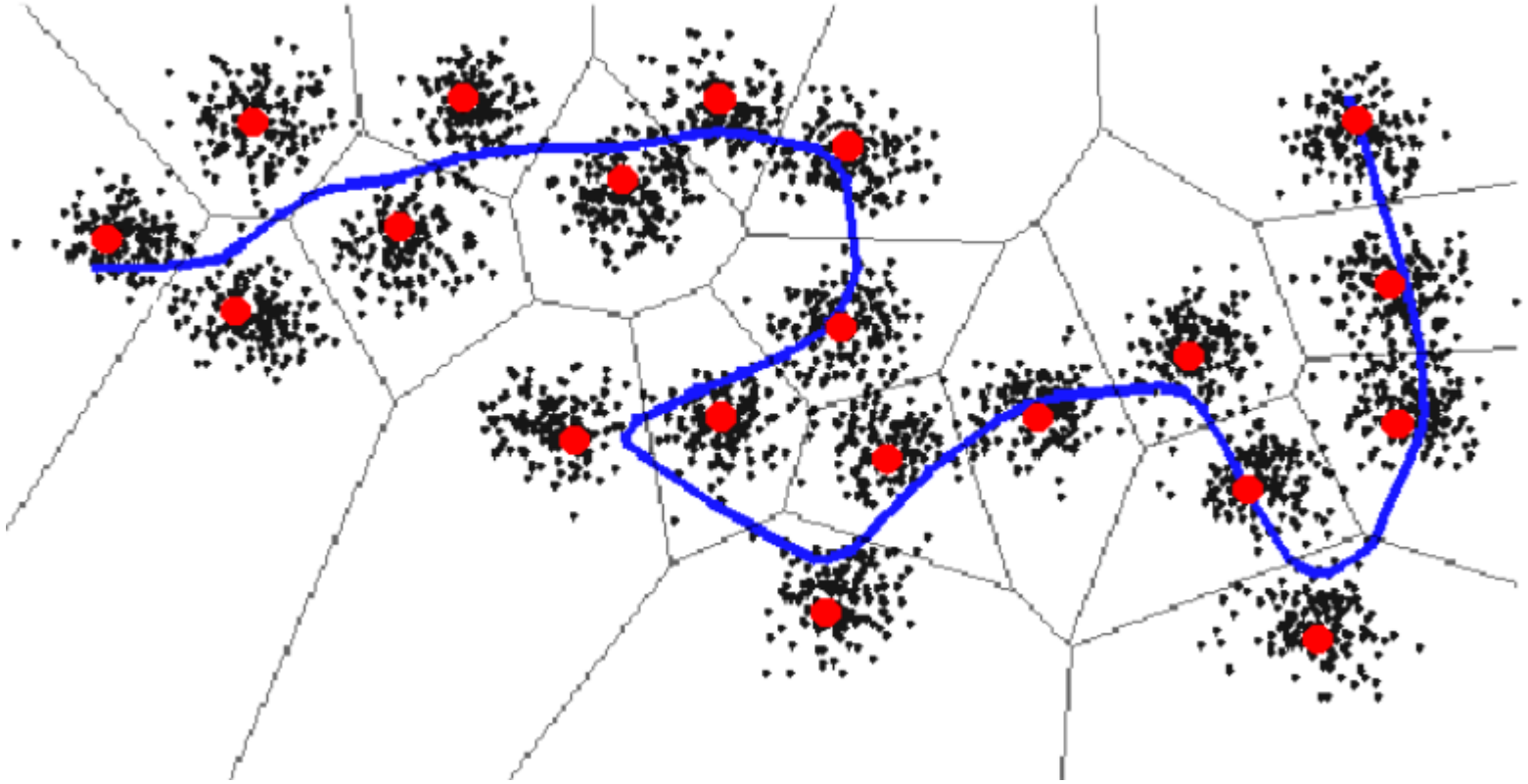
More complex projections

I. Cleju, P. Fränti, X. Wu, "Clustering based on principal curve",
Scandinavian Conf. on Image Analysis, LNCS vol. 3540, June 2005.



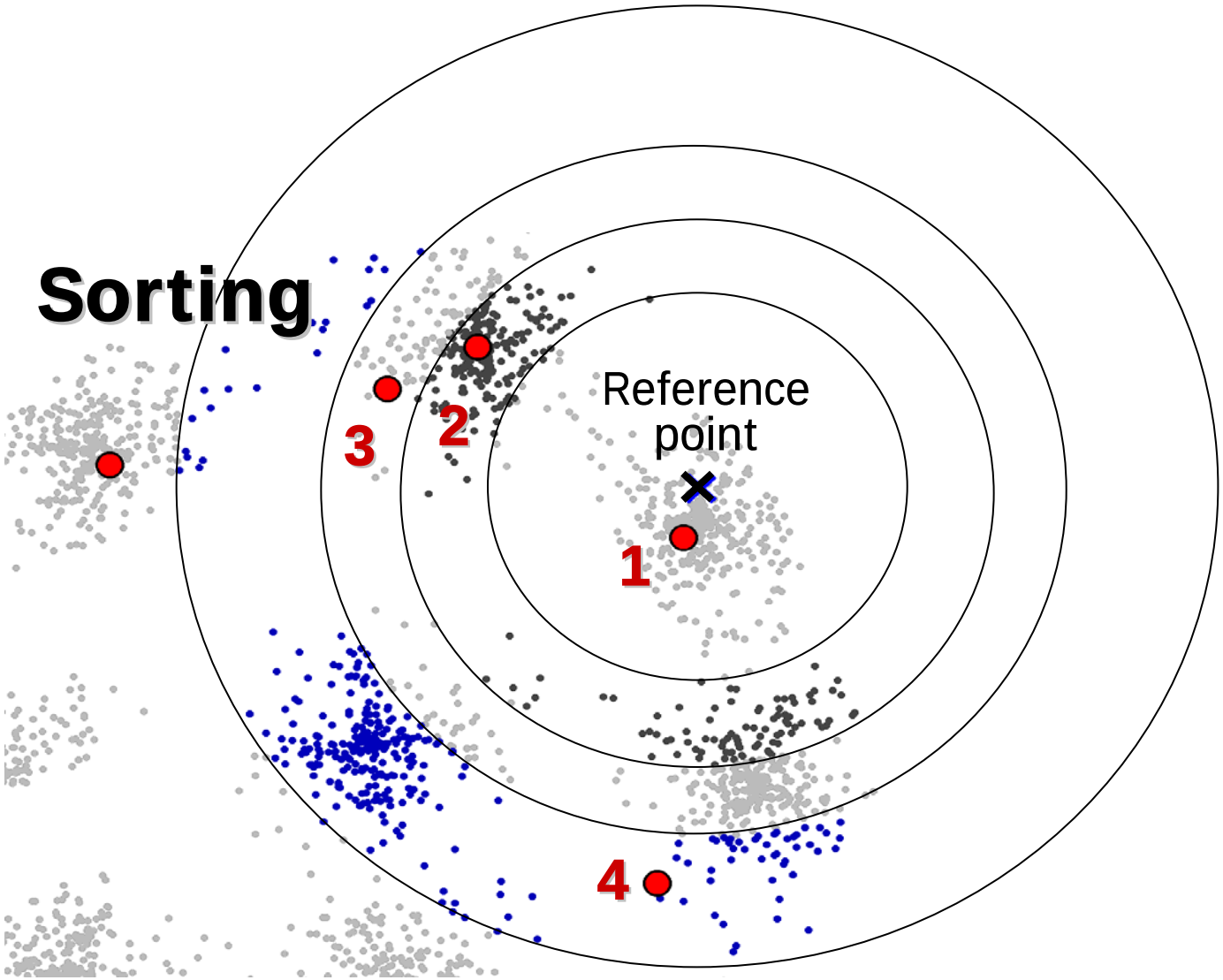
More complex projections

I. Cleju, P. Fränti, X. Wu, "Clustering based on principal curve",
Scandinavian Conf. on Image Analysis, LNCS vol. 3540, June 2005.



Travelling salesman problem!

Sorting heuristic

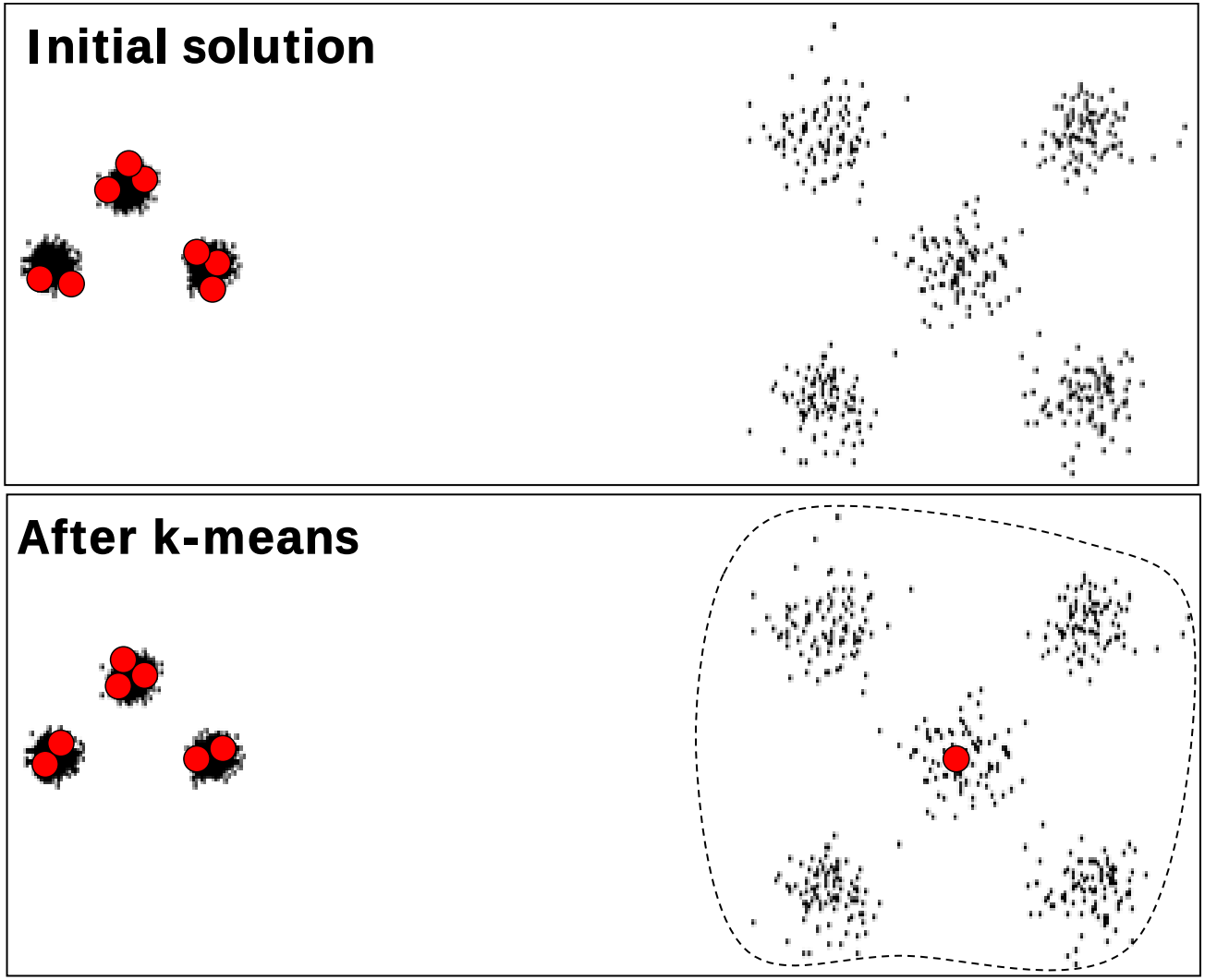


Density-based heuristics

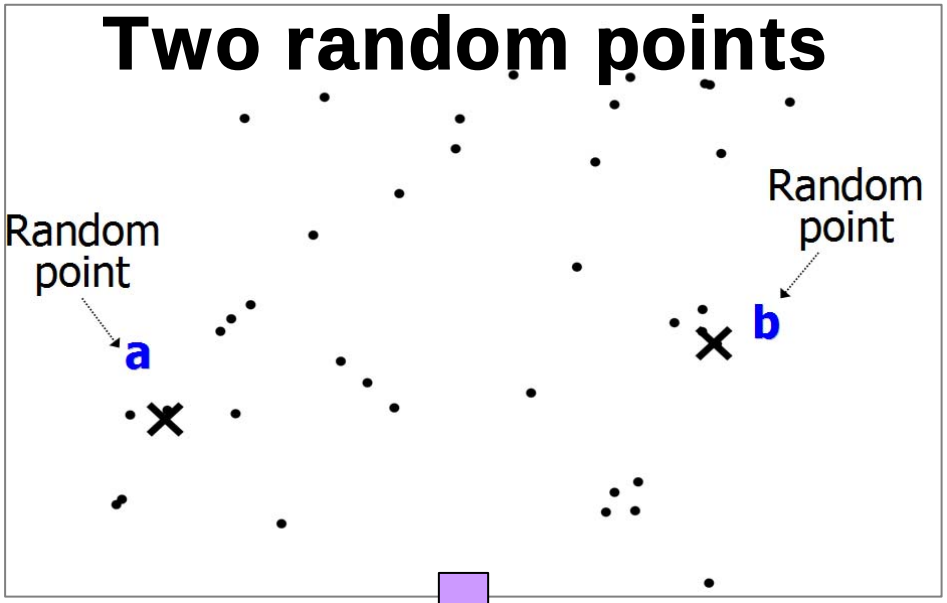
Luxburg

Luxburg's technique:

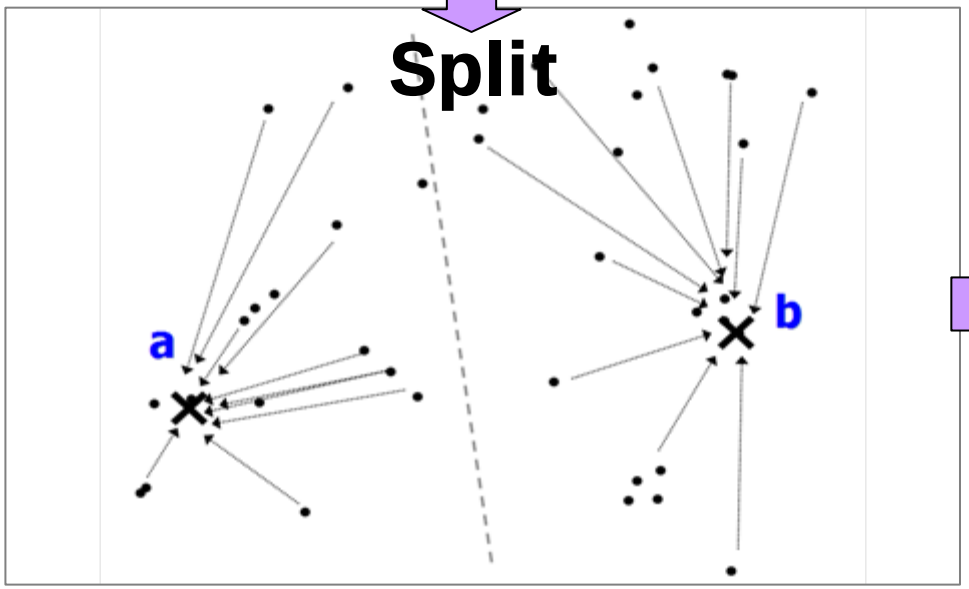
- Selects $k \log(k)$ preliminary clusters using k-means
- Eliminate the smallest.
- Furthest point heuristic to select k centroids.

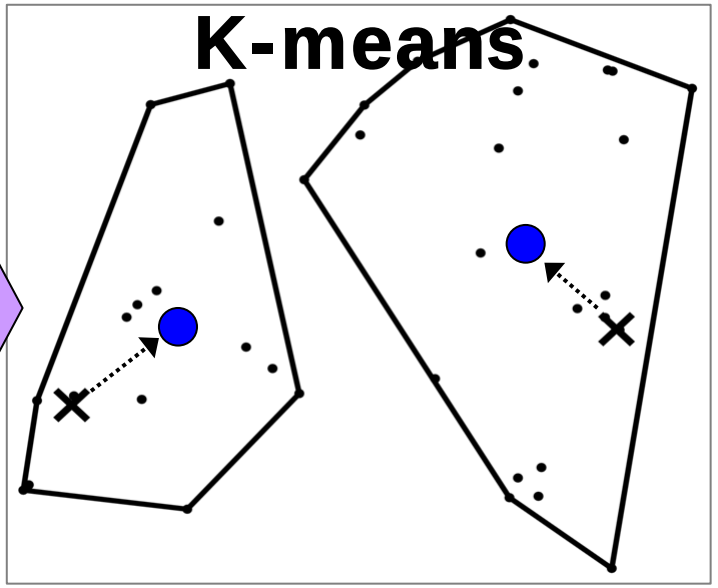


Splitting algorithm



- Select biggest cluster
- Select two random points
- Re-allocate points
- Tune by k-means locally





Results

Success rates

K-means (without repeats)

Average success rate

Method	s1	s2	s3	s4	a1	a2	a3	unb	b1	b2	dim32	Aver.	Fails
Rand-P	0%	47%	5%	63%	0%	0%	0%	0%	0%	0%	0%	**10%**	**8**
Rand-C	3%	11%	12%	26%	1%	0%	0%	0%	0%	0%	0%	**5%**	**6**
Maxmin	37%	16%	36%	9%	15%	1%	0%	22%	0%	0%	100%	**22%**	**3**
kmeans++	21%	24%	18%	30%	7%	0%	0%	51%	0%	0%	88%	**22%**	**4**
Bradley	21%	46%	49%	64%	7%	0%	0%	0%	0%	0%	2%	**17%**	**5**
Sorting	12%	20%	22%	36%	10%	0%	0%	0%	0%	12%	15%	**12%**	**4**
Projection	16%	29%	30%	42%	18%	0%	0%	0%	0%	92%	34%	**24%**	**4**
Luxburg	52%	60%	45%	61%	45%	33%	31%	0%	0%	17%	95%	**40%**	**2**
Split	78%	75%	62%	64%	51%	17%	5%	0%	0%	10%	99%	**42%**	**2**

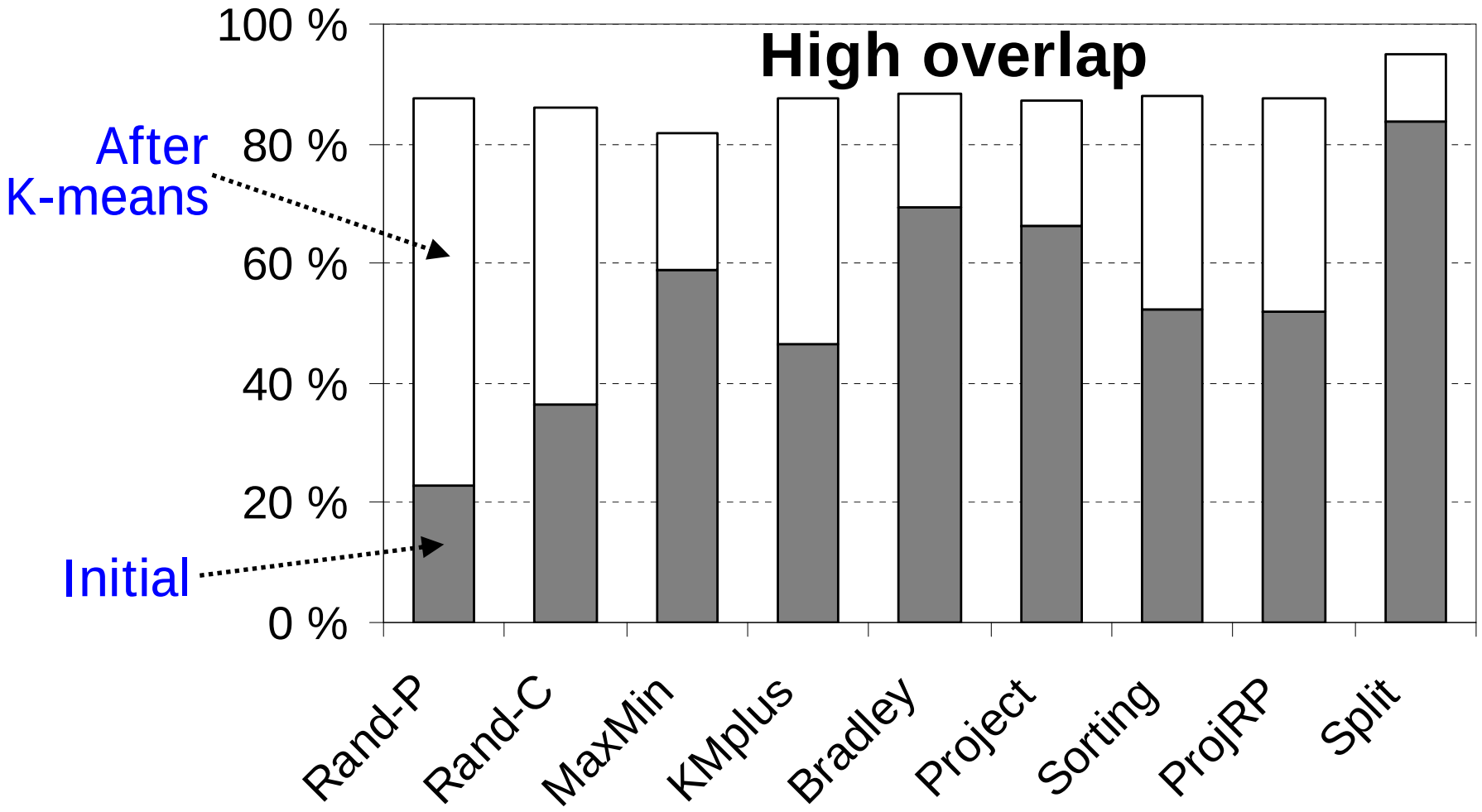
Most problems:
a2, a3, unbalance, Birch1, Birch2

No. of datasets never solved

Cluster overlap

G2

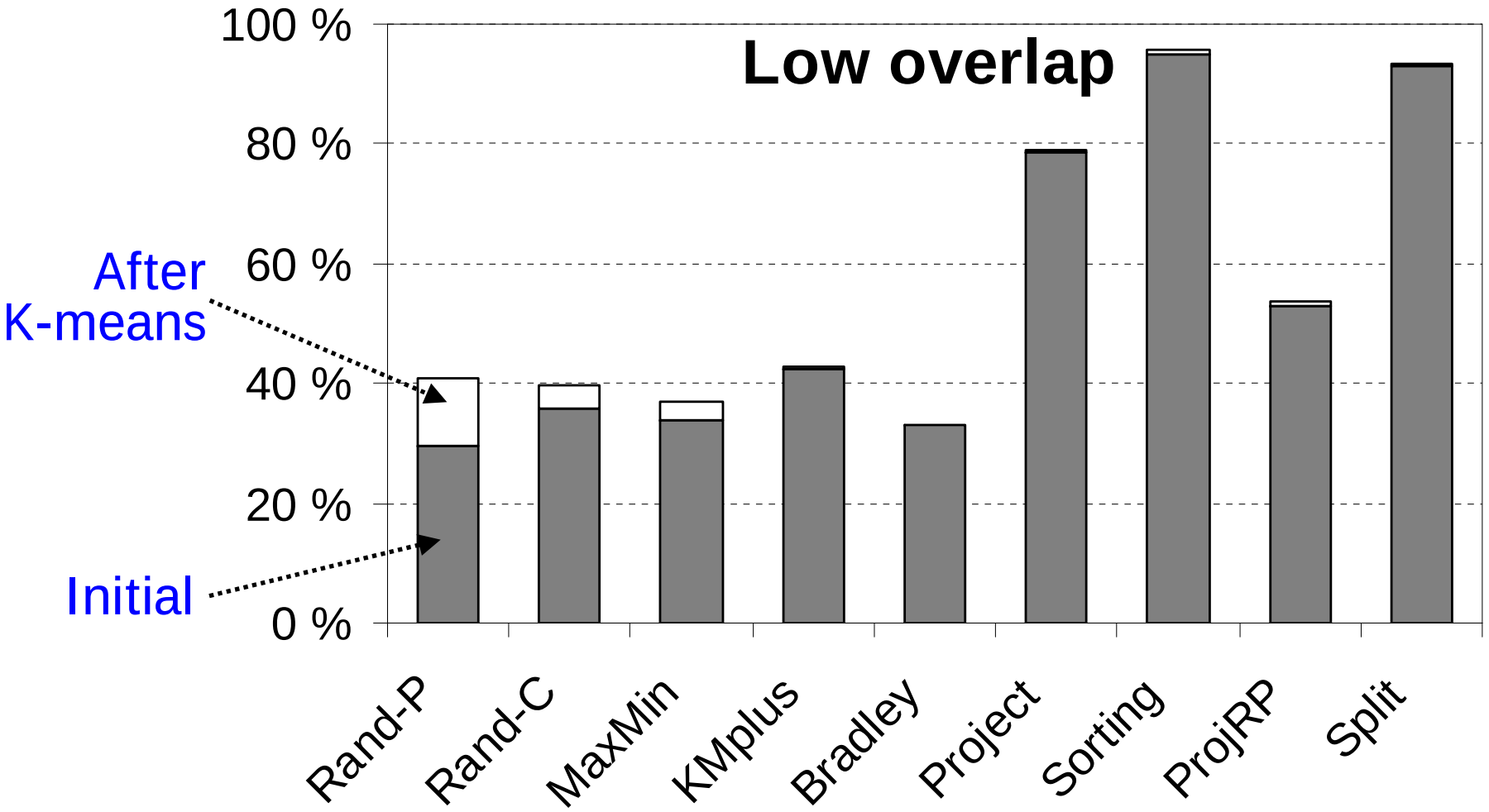
High cluster overlap



Cluster overlap

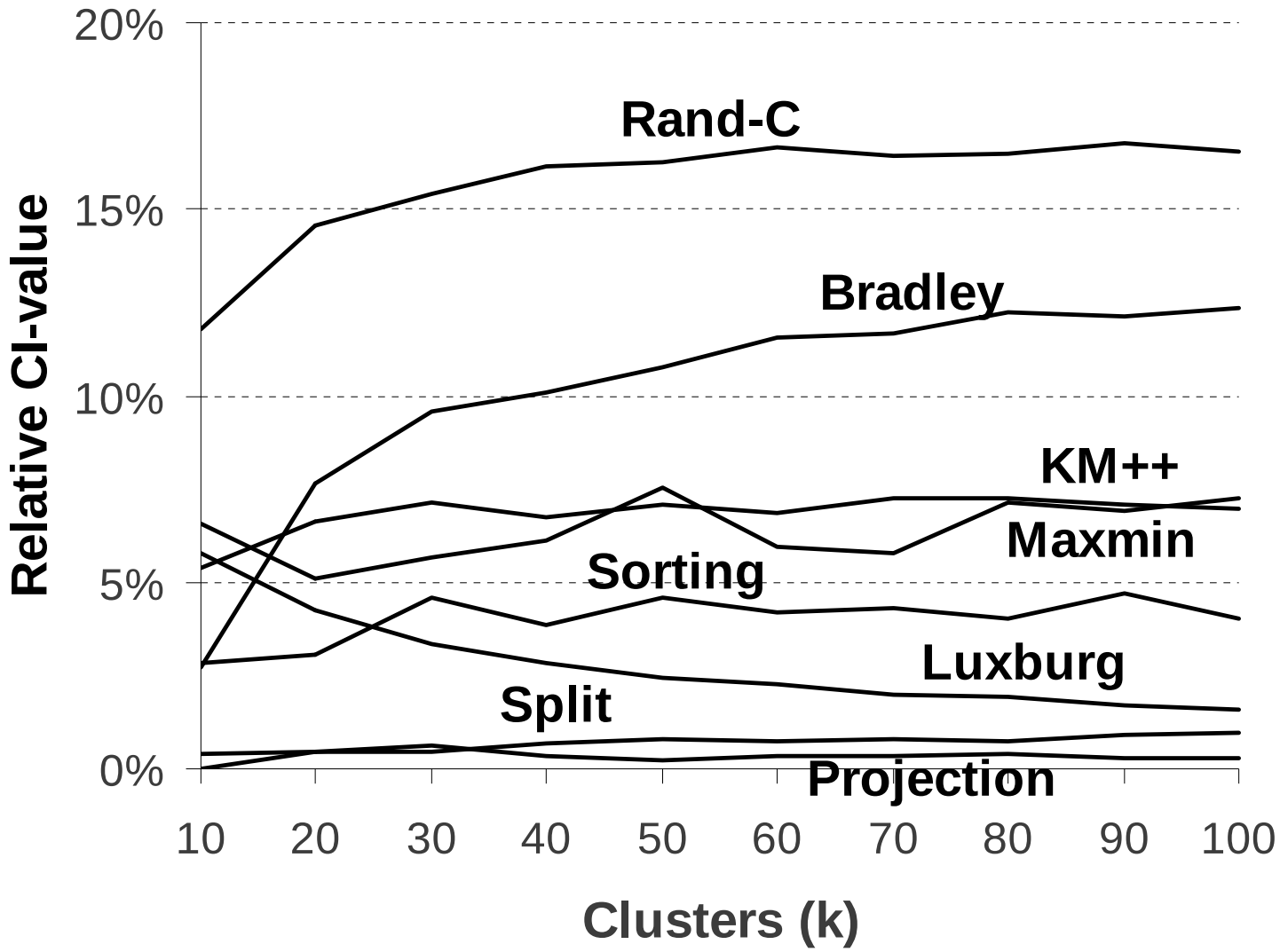
G2

Low cluster overlap

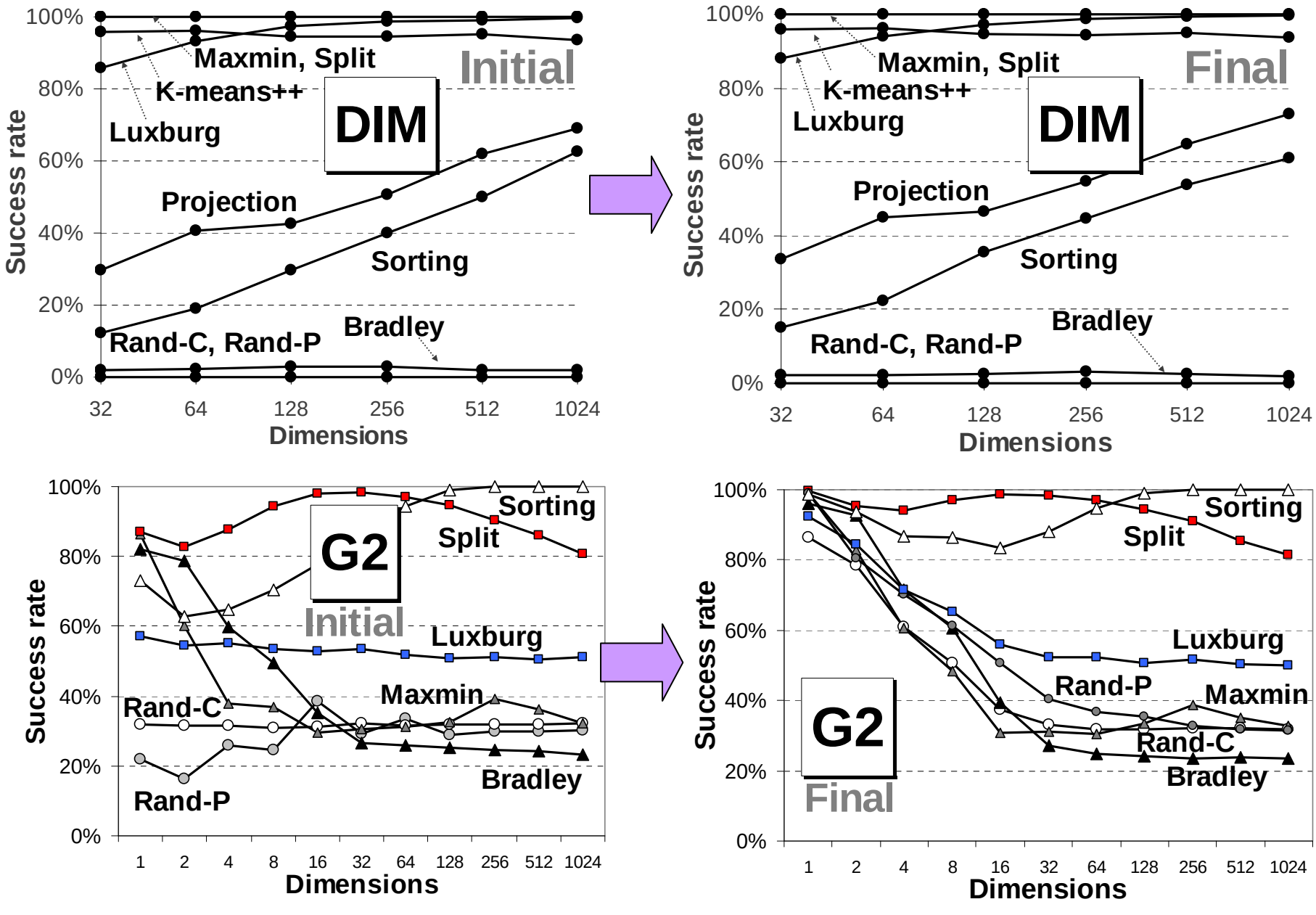


Number of clusters

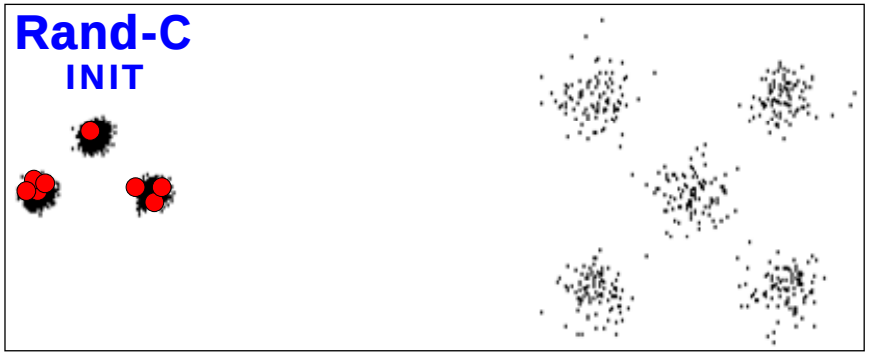
Birch2 subsets

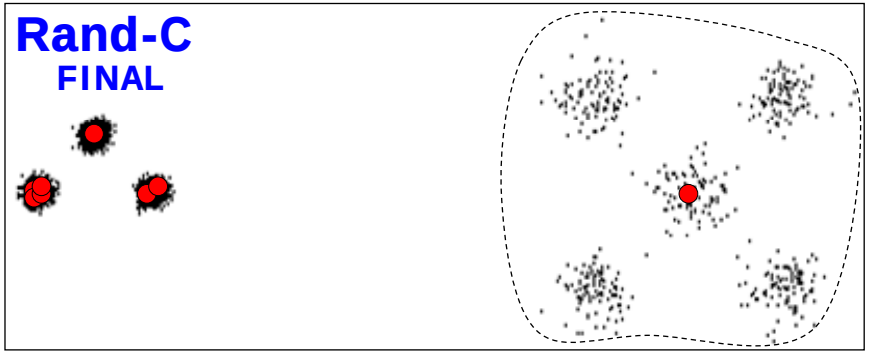


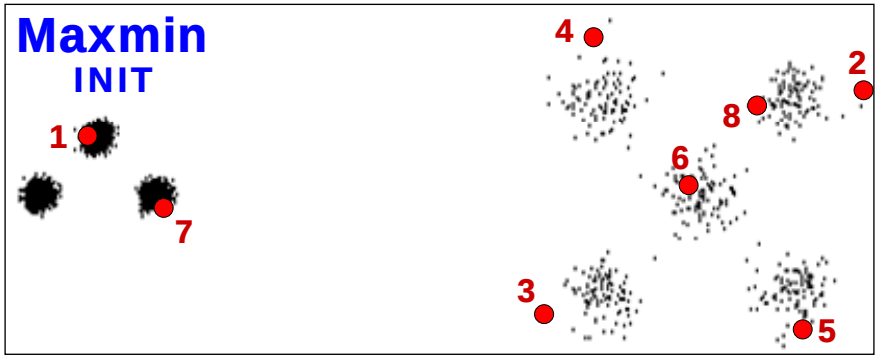
Dimensions

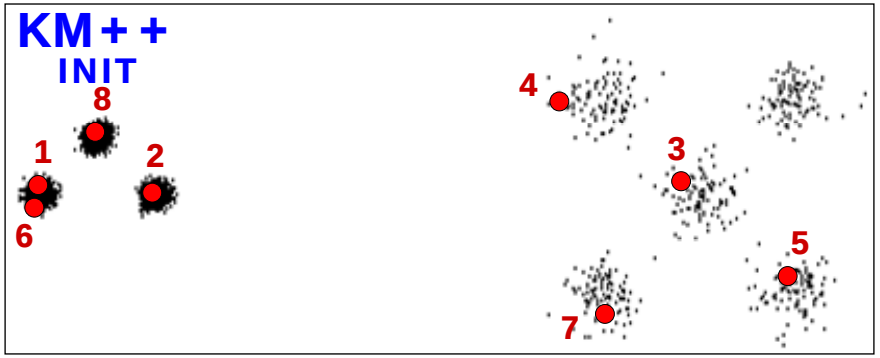

Initial
DIM
Maxmin, Split
K-means++
Luxburg
Projection
Sorting
Bradley
Rand-C, Rand-P
Success rate
Dimensions
100%
80%
60%
40%
20%
0%
32
64
128
256
512
1024
Final
DIM
Maxmin, Split
K-means++
Luxburg
Projection
Sorting
Bradley
Rand-C, Rand-P
Success rate
Dimensions
G2
Initial
Sorting
Split
Luxburg
Maxmin
Rand-C
Bradley
Rand-P
Success rate
Dimensions
1
2
4
8
16
32
64
128
256
512
1024
G2
Final
Sorting
Split
Luxburg
Rand-P
Maxmin
Rand-C
Bradley
Success rate
Dimensions

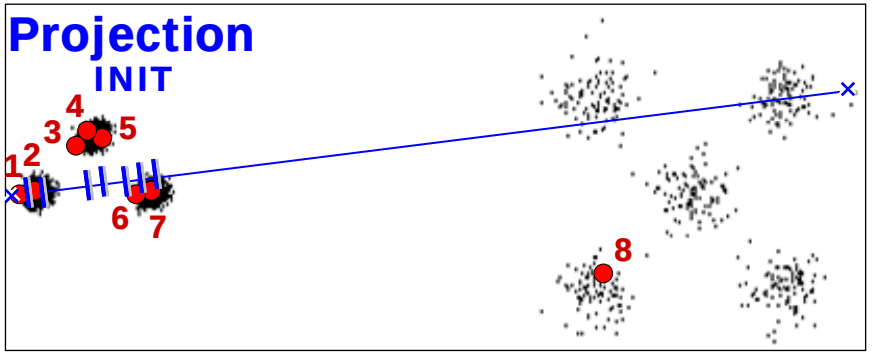
Unbalance


Rand-C
INIT


Rand-C
FINAL


Maxmin
INIT
1
7
4
2
8
6
3
5


KM++
INIT
8
1
2
6
4
3
5
7


Projection
INIT
4
3
5
1
2
6
7
8

Success rates

Repeated k-means

Furthest point approaches solve unbalance

Average success rate

Method	s1	s2	s3	s4	a1	a2	a3	unb	b1	b2	dim32	Aver.	Fails
Rand-P	0%	100%	100%	100%	0%	0%	0%	0%	0%	0%	0%	**27%**	**8**
Rand-C	96%	100%	100%	100%	56%	2%	0%	0%	0%	0%	2%	**41%**	**4**
Maxmin	100%	100%	100%	100%	100%	58%	36%	100%	0%	0%	100%	**72%**	**2**
kmeans++	100%	100%	100%	100%	98%	20%	0%	100%	0%	0%	100%	**65%**	**3**
Bradley	100%	100%	100%	100%	100%	4%	4%	4%	0%	0%	84%	**54%**	**2**
Sorting	100%	100%	100%	100%	100%	24%	0%	0%	2%	100%	100%	**66%**	**2**
Projection	100%	100%	100%	100%	100%	18%	0%	0%	0%	100%	100%	**65%**	**3**
Luxburg	100%	100%	100%	100%	100%	100%	100%	0%	46%	100%	100%	**86%**	**1**
Split	100%	100%	100%	100%	100%	100%	100%	0%	36%	100%	100%	**85%**	**1**

Still problems:
Birch1, Birch2

No. of datasets never solved

How many repeats needed?

A3

Initialization	CI-value						
	6	5	4	3	2	1	0
Rand-P	-	-	-	-	-	-	-
Rand-C	2	4	11	54	428	11111	-
Maxmin				1	3	14	216
Kmeans++		1	2	3	14	138	8696
Bradley		1	2	8	58	1058	33333
Sorting	1	2	4	13	73	1143	-
Projection	1	2	3	9	46	581	18182
Luxburg						1	3
Split					1	2	9

How many repeats needed?

Unbalance

Initialization	CI-value						
	6	5	4	3	2	1	0
Rand-P			1	97	8333	-	-
Rand-C			1	16	69	1695	100k
Maxmin						1	4
Kmeans++						1	2
Bradley			1	3	6	70	1471
Sorting			1	-	-	-	-
Projection			1	935	16667	-	-
Luxburg			1	59	16667	-	-
Split			1	9524	-	-	-

Summary of results

CI-values

Method	s1	s2	s3	s4	a1	a2	a3	unb	b1	b2	dim32	KM	RKM
Rand-P	1.4	0.0	0.0	0.0	4.9	8.8	16.7	3.6	8.5	74.0	2.6	12.4	11.0
Rand-C	0.1	0.0	0.0	0.0	0.3	1.8	2.9	2.9	2.8	10.9	1.1	4.5	2.1
Maxmin	0.0	0.0	0.0	0.0	0.0	0.5	0.6	0.0	2.8	3.9	0.0	2.2	0.7
kmeans++	0.0	0.0	0.0	0.0	0.0	0.8	1.6	0.0	1.7	3.4	0.0	2.3	0.7
Bradley	0.0	0.0	0.0	0.0	0.0	0.9	2.1	1.2	2.0	8.5	0.0	3.1	1.3
Sorting	0.0	0.0	0.0	0.0	0.0	0.8	2.2	4.0	2.2	0.0	0.0	2.7	0.8
Projection	0.0	0.0	0.0	0.0	0.0	0.9	2.0	3.9	1.9	0.0	0.0	2.2	0.4
Luxburg	0.0	0.0	0.0	0.0	0.0	0.0	0.0	3.7	0.6	0.0	0.0	1.2	0.4
Split	0.0	0.0	0.0	0.0	0.0	0.0	0.0	4.0	0.6	0.0	0.0	1.2	0.4

- K-means: CI=4.5 15%
- Repeated K-means: CI=2.0 6%
- Maxmin initialization: CI=2.1 6%
- Both: **CI=0.7** 1%

- Most application: Good enough!
- Accuracy vital: Find better method! (random swap)

- Cluster overlap most important factor

Effect of different factors

Method	Overlap	Clusters	Dimension	Unbalance
Rand-P	No effect	Constant	No effect	Very bad
Rand-C	No effect	Constant	No effect	Very bad
Maxmin	Bad	Constant	No effect	A bit worse
kmeans++	A bit worse	Constant	No effect	A bit worse
Bradley	Good	Constant	No effect	Bad
Sorting	A bit worse	Constant	No effect	Very bad
Projection	A bit worse	Constant	No effect	Very bad
Luxburg	A bit worse	Minor effect	No effect	Very bad
Split	A bit worse	Constant	No effect	Very bad
KM iterations	Good	Constant	No effect	No effect

Conclusions

How effective:

- Repeats + Maxmin reduces error **4.5 → 0.7**

Is it enough:

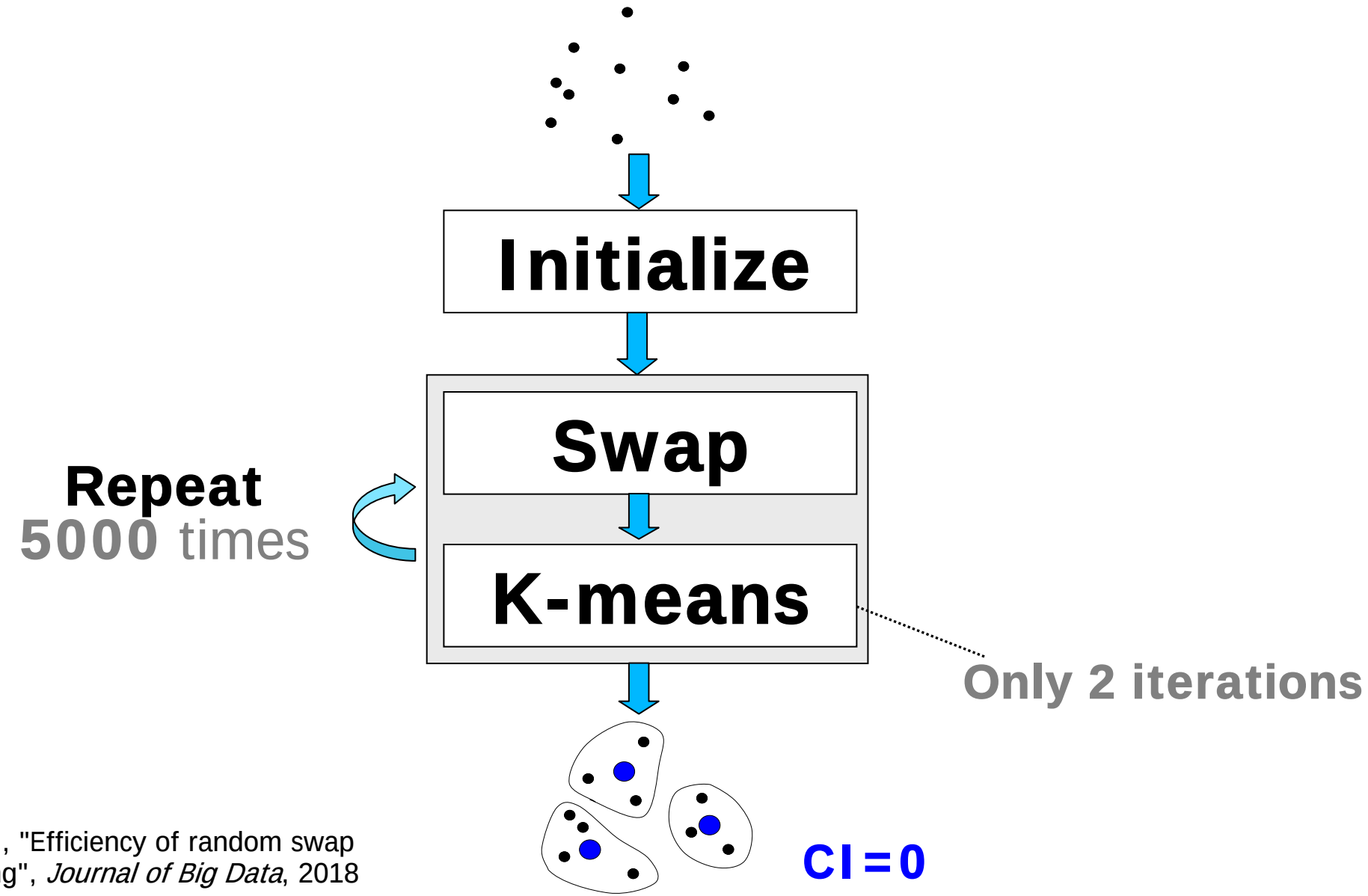
- For most applications: **YES**
- If accuracy important: **NO**

Important factors:

- Cluster overlap critical for k-means
- Dimensions does not matter

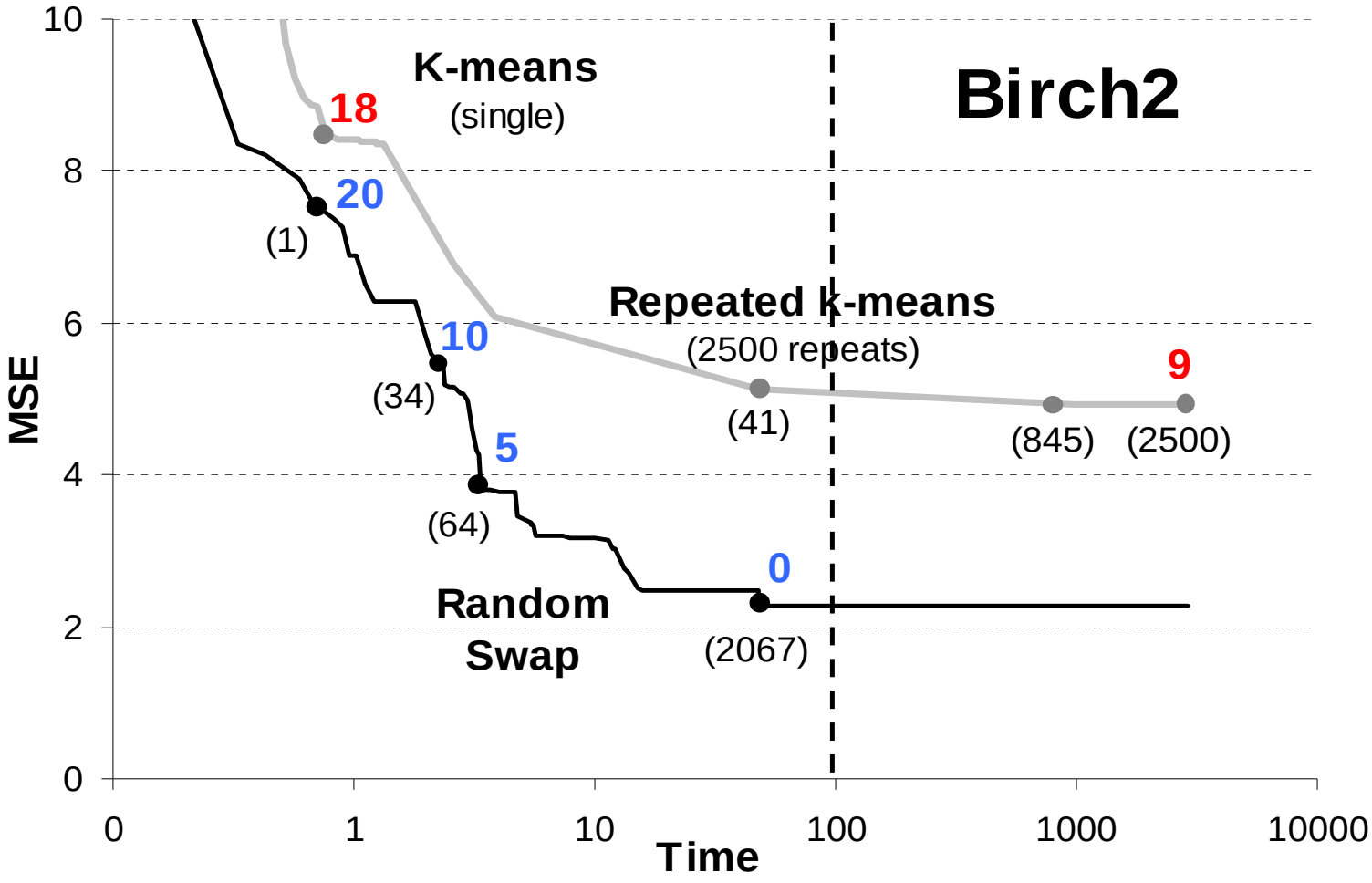
Random swap

Random swap (RS)



P. Fränti, "Efficiency of random swap clustering", *Journal of Big Data*, 2018

How many repeats?



P. Fränti, "Efficiency of random swap clustering", *Journal of Big Data*, 2018

The end