ENEE698A

Subspace Clustering for High Dimensional Data

Jonghyun Choi

18th Oct. 2011

Purpose of this talk

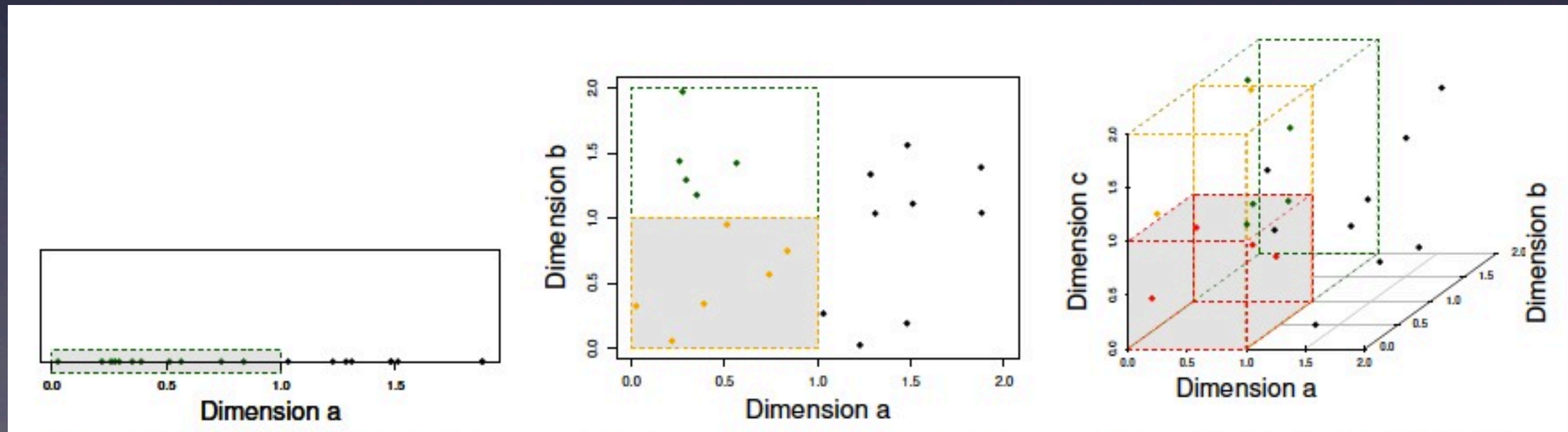
- Introducing subspace clustering
- Get familiar with the names of subspace clustering algorithms

Why high dimensional data are tricky?

- Features are correlated
  - Meaningful dimensions are few
- Curse of dimensionality
  - Data are equi-distant with high probability

Why high dimensional data are tricky?

- Features are correlated
  - Meaningful dimensions are few
- Curse of dimensionality
  - Data are equi-distant with high probability



(a) 11 Objects in One Unit Bin (b) 6 Objects in One Unit Bin (c) 4 Objects in One Unit Bin

Typical Remedy

- Dimension reduction
  - Feature transform
  - Feature selection

Typical Remedy

- Dimension reduction
  - Feature transform ➡ PCA
  - Feature selection

Typical Remedy

- Dimension reduction
  - Feature transform
  - Feature selection

Typical Remedy

- Dimension reduction
  - Feature transform
  - Feature selection ➡ Choosing subset of dimensions

Typical Remedy

- Dimension reduction
  - Feature transform
  - Feature selection

Typical Remedy

- Dimension reduction
  - Feature transform
  - Feature selection
- When does it work?
  - Generally works fine
- When doesn't it work?
  - When many features are meaningless (noisy)

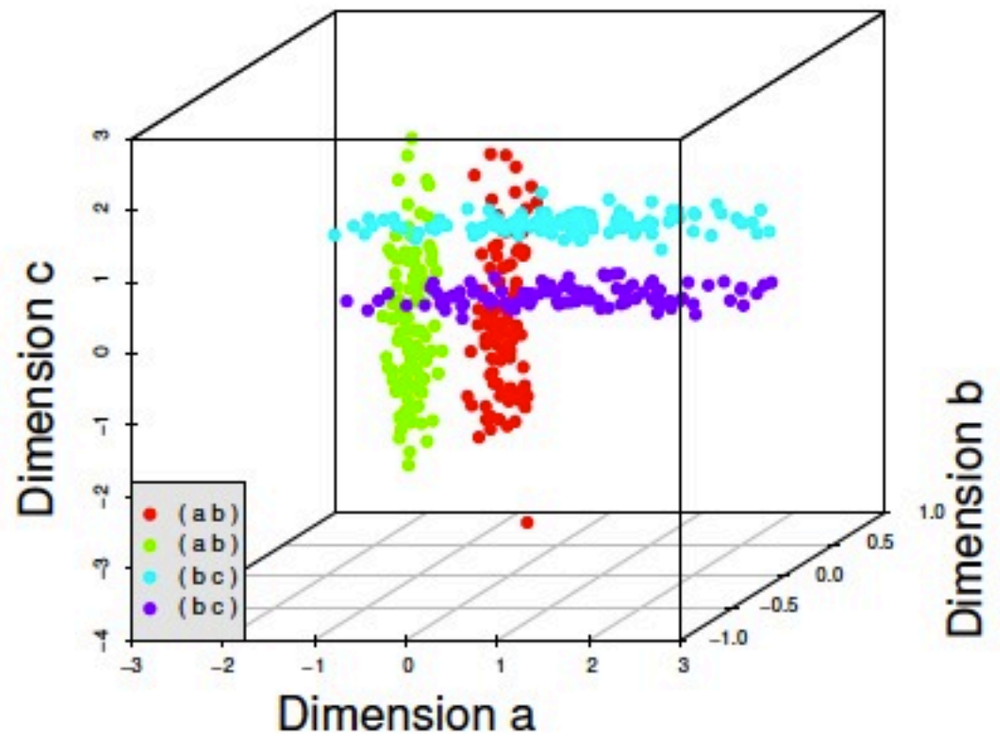
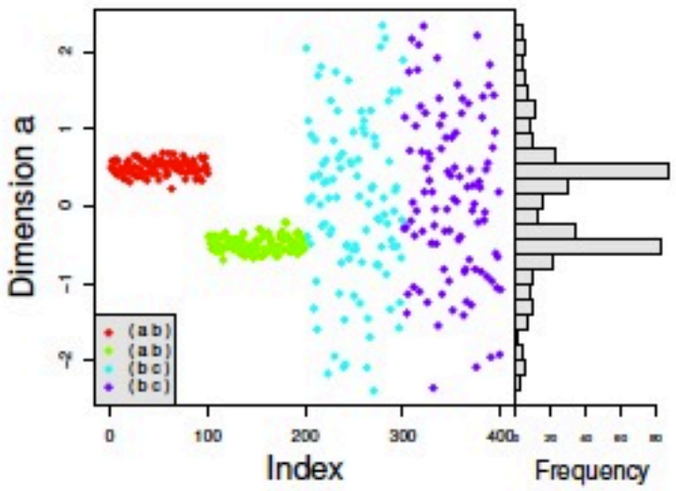
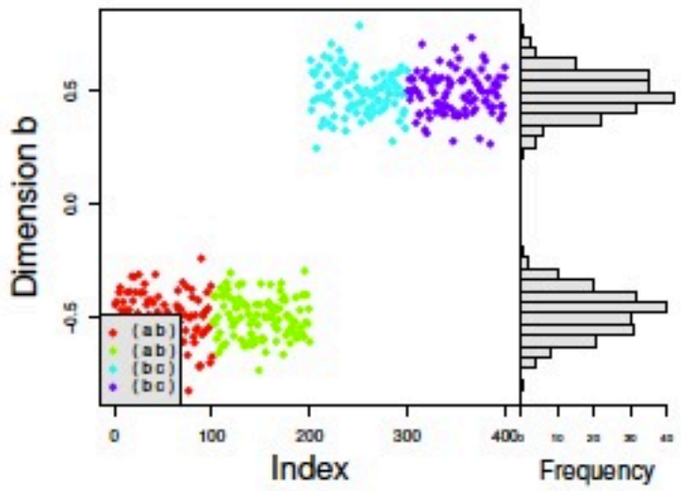


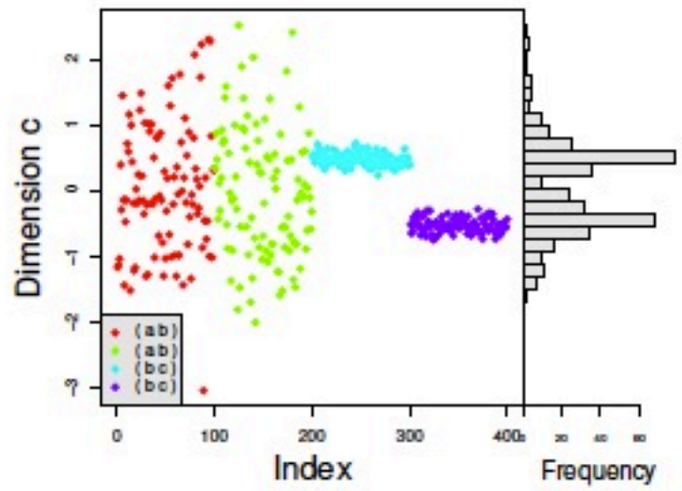
Figure 2: Sample dataset with four clusters, each in two dimensions with the third dimension being noise. Points from two clusters can be very close together, confusing many traditional clustering algorithms.



(a) Dimension a



(b) Dimension b

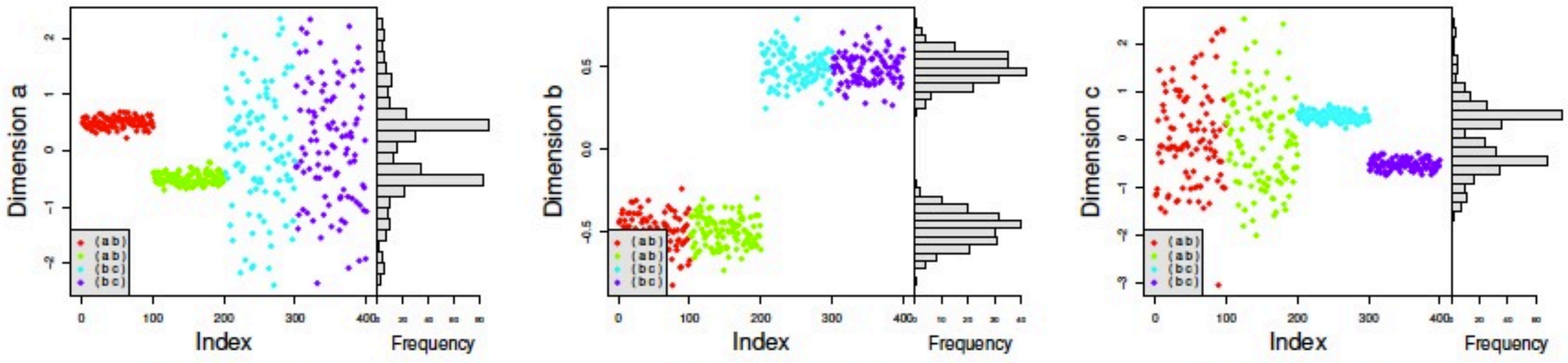


(c) Dimension c

Figure 3: Sample data plotted in one dimension, with histogram. While some clustering can be seen, points from multiple clusters are grouped together in each of the three dimensions.

None of the selected dimension or transformed dimension can separate them effectively

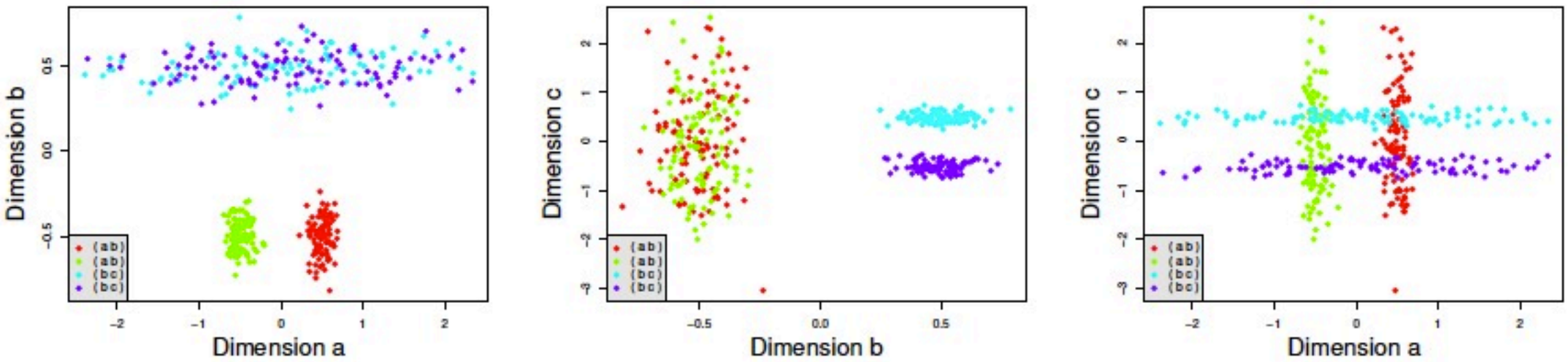
Subspace Clustering



(a) Dimension a (b) Dimension b (c) Dimension c

Figure 3: Sample data plotted in one dimension, with histogram. While some clustering can be seen, points from multiple clusters are grouped together in each of the three dimensions.

Finding better subset of dimension for **each** clusters



(a) Dims a & b (b) Dims b & c (c) Dims a & c

Figure 4: Sample data plotted in each set of two dimensions. In both (a) and (b) we can see that two clusters are properly separated, but the remaining two are mixed together. In (c) the four clusters are more visible, but still overlap each other are are impossible to completely separate.

Subspace Clustering

- What is it?
  - Extension of dimension selection
- Why is it better?
  - Better than universal dimension selection
  - No difficulties of finding sophisticated space transform
- Side product
  - Give clusters as well as their residing subspace

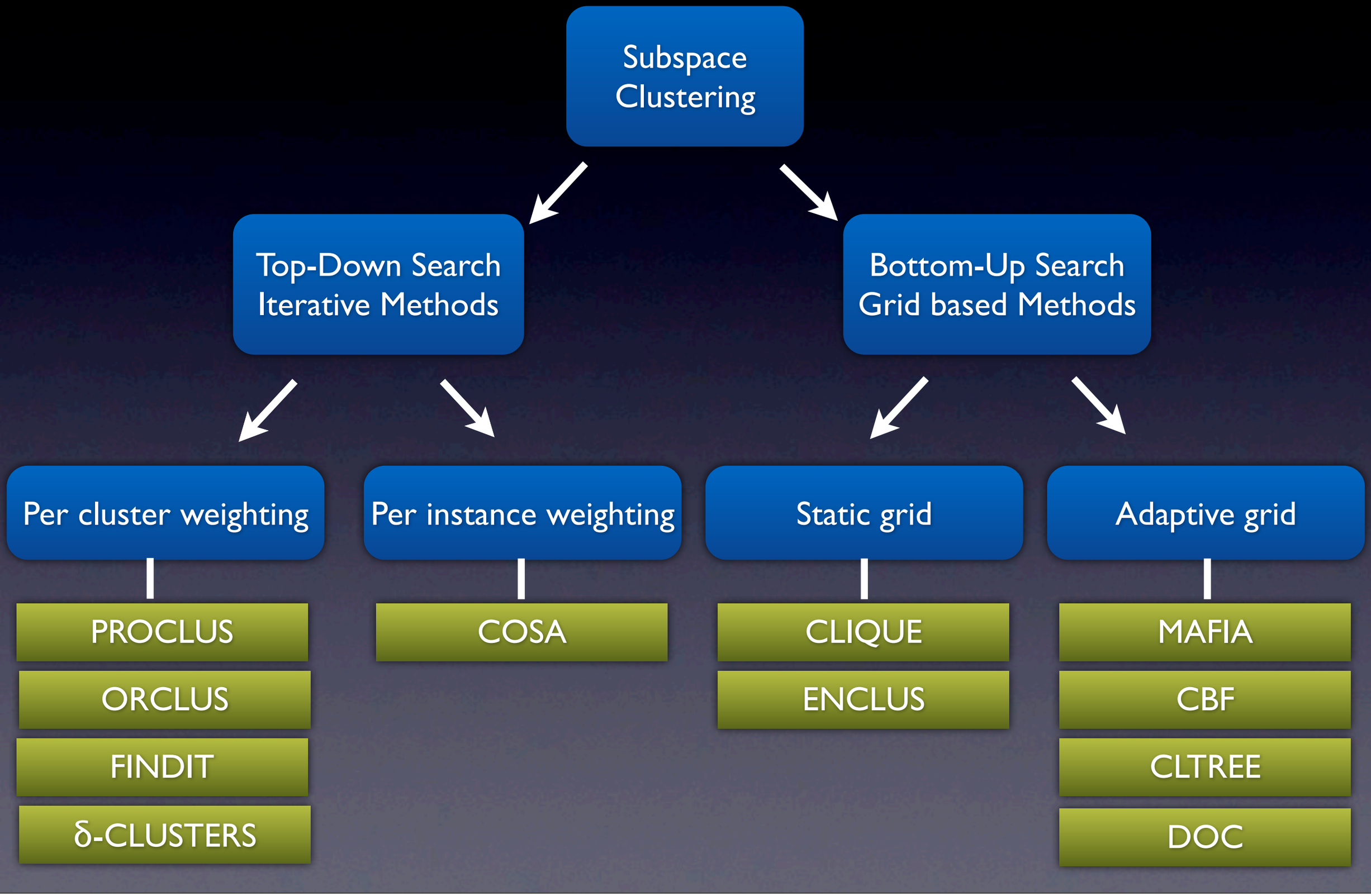
Issues in Finding Good Subspace

- How can I find the subset?
  - Most straightforward (stupid) way
    - Searching all combinations of subspaces
      - ✓ Not feasible
  - Better way
    - ✓ algorithms with efficient search methods
- Quality measure of good clustering that algorithm relies on

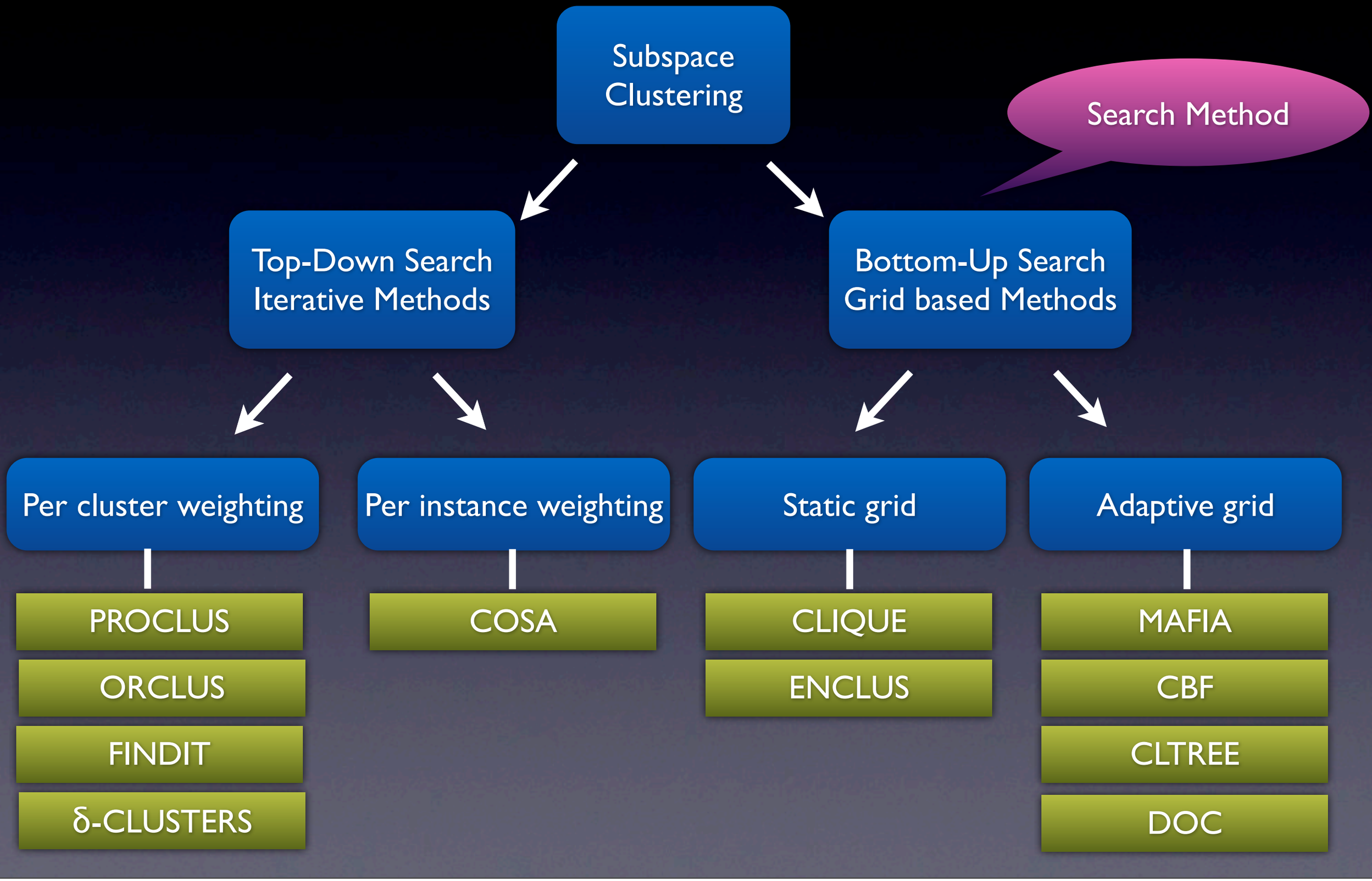
Categorizing Subspace Clustering Algorithms

- Efficient searching methods
  - Top-down / Bottom-up
- Measure of locality (Quality of clustering)

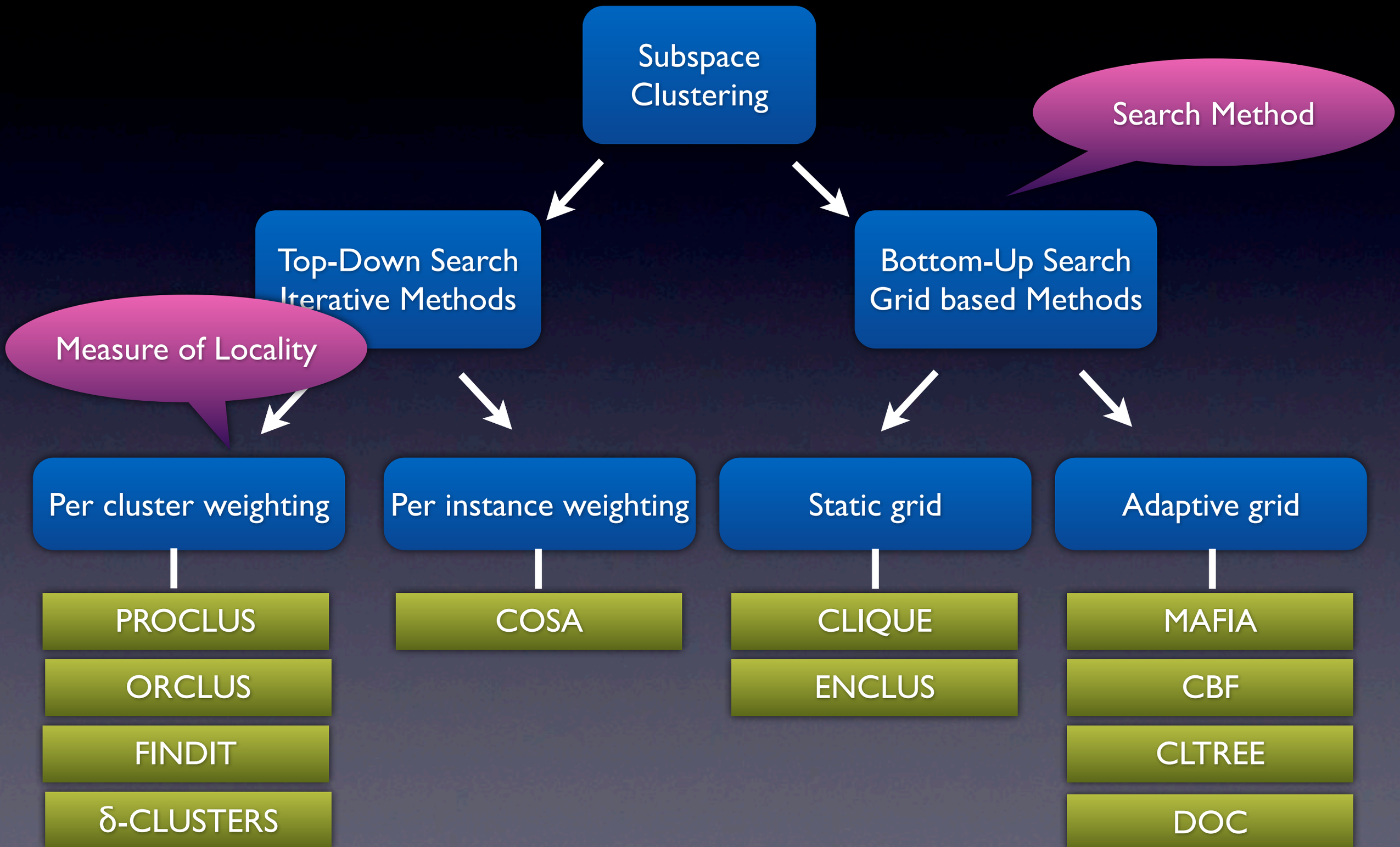
Family Tree of Subspace Clustering



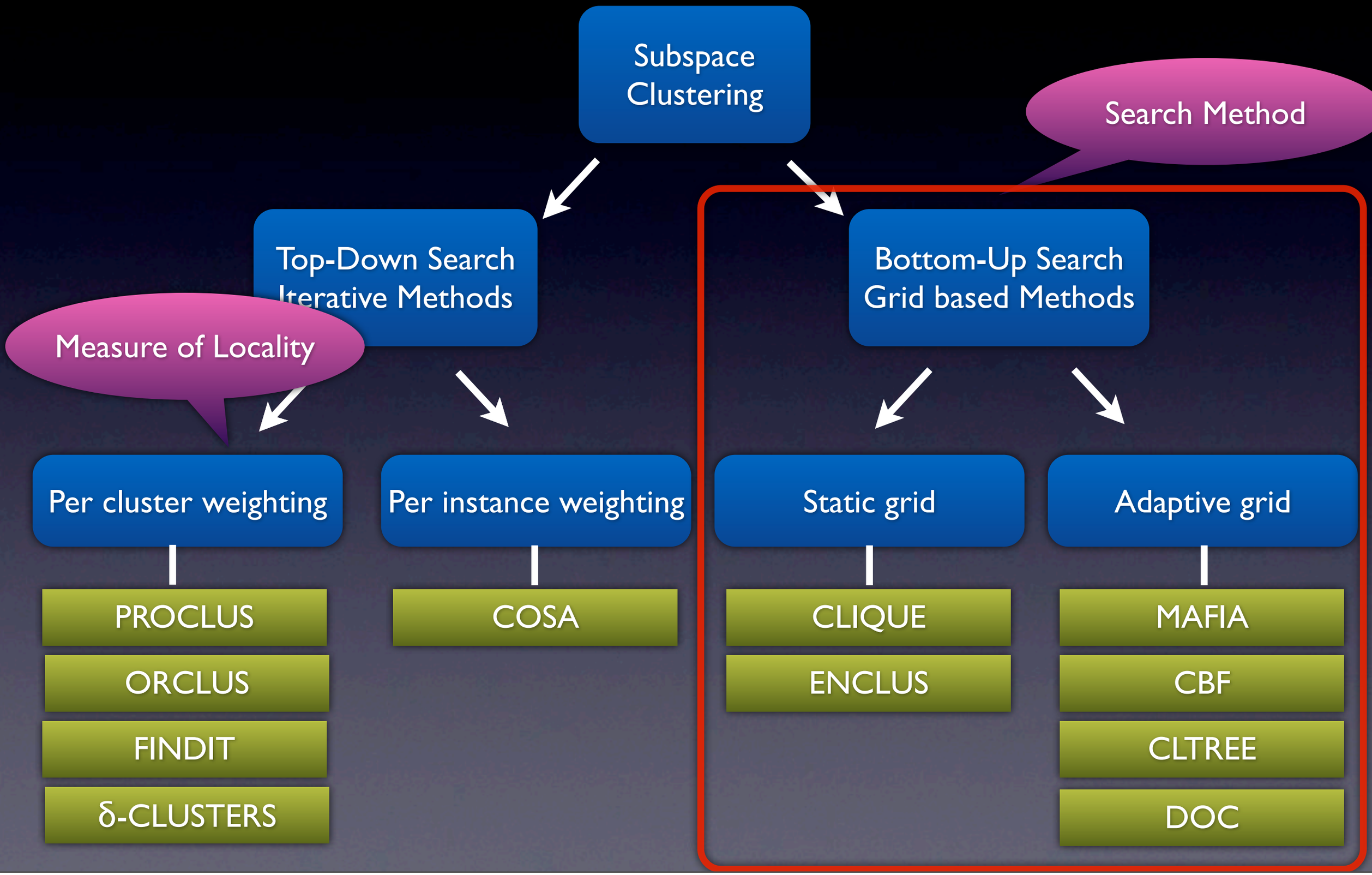
Family Tree of Subspace Clustering



Family Tree of Subspace Clustering

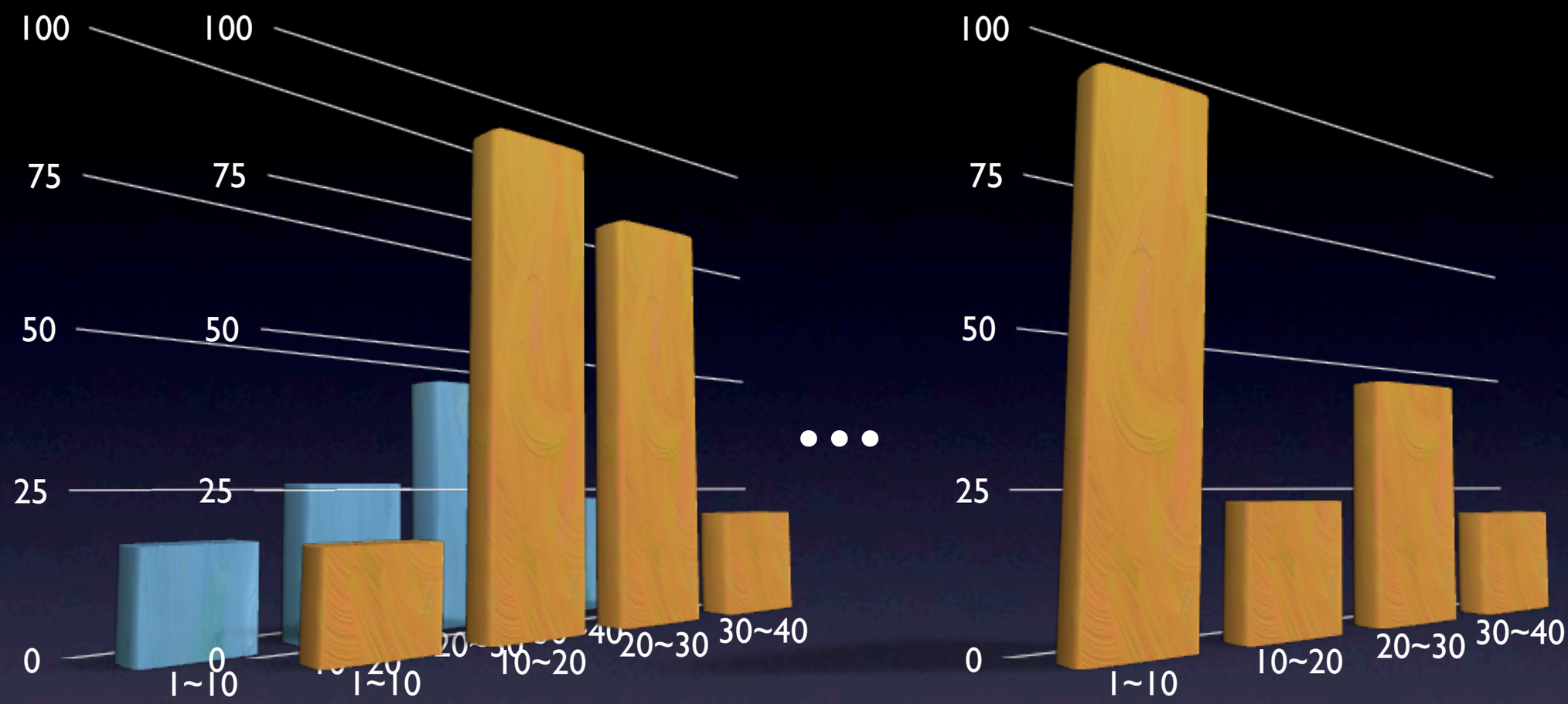


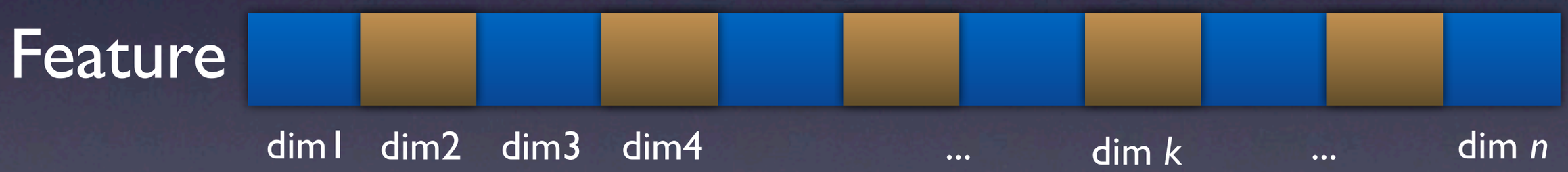
Family Tree of Subspace Clustering



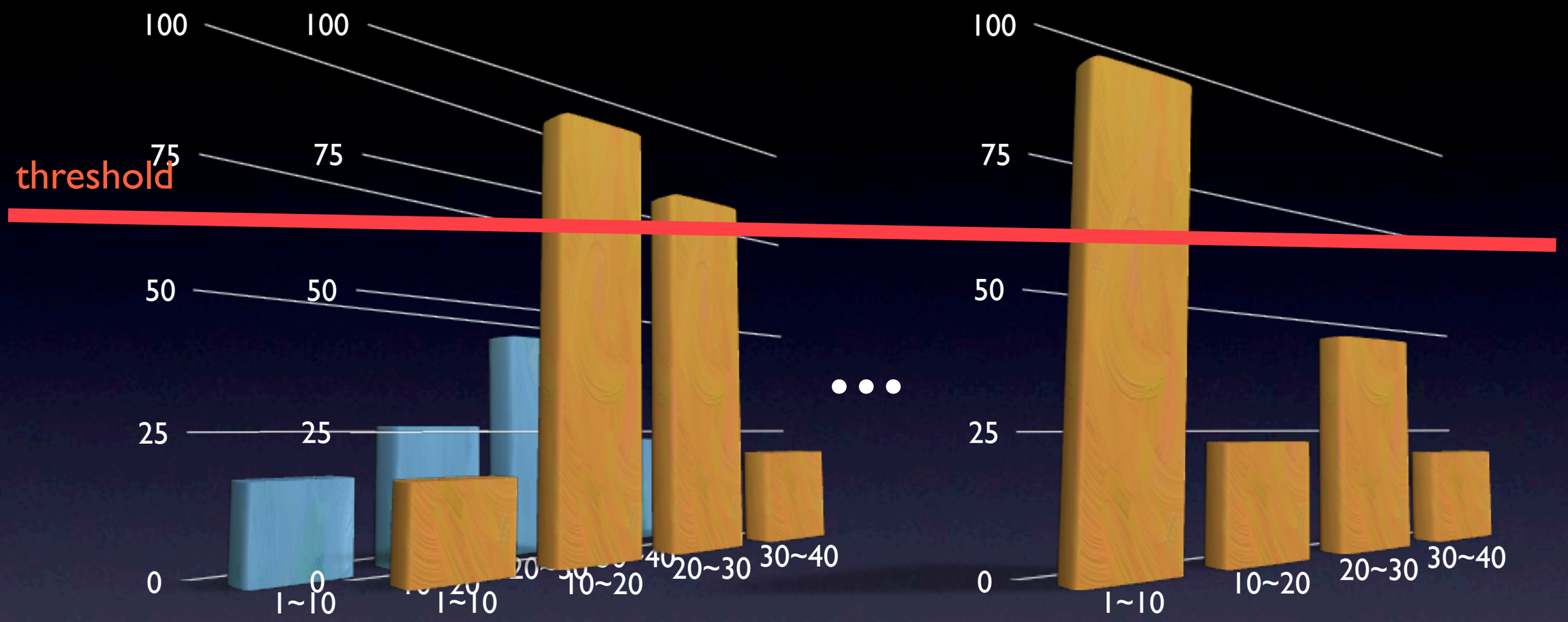
Bottom-Up Methods

Bottom-Up Methods





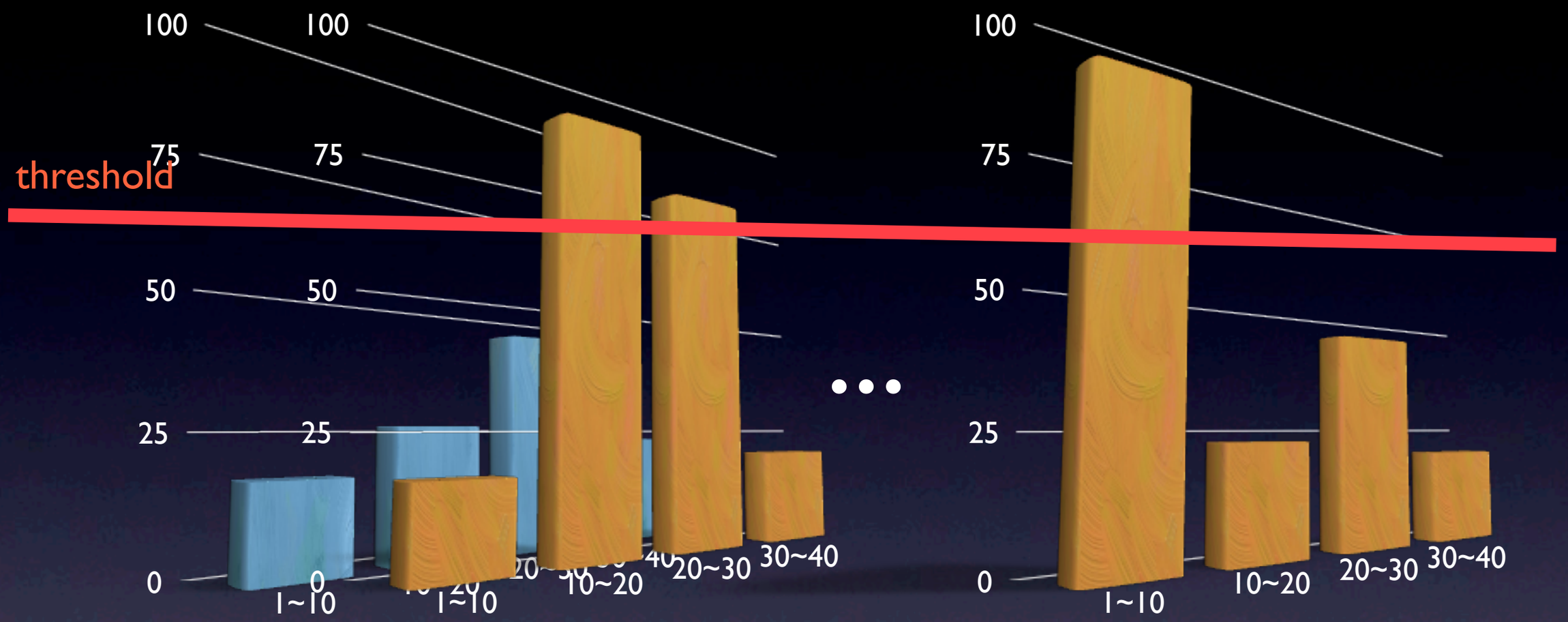
Bottom-Up Methods



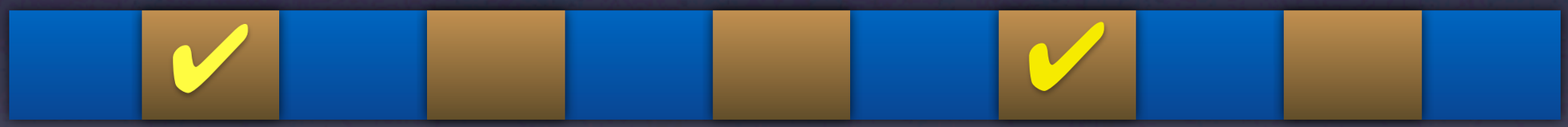
Feature

dim1 dim2 dim3 dim4 ... dim *k* ... dim *n*

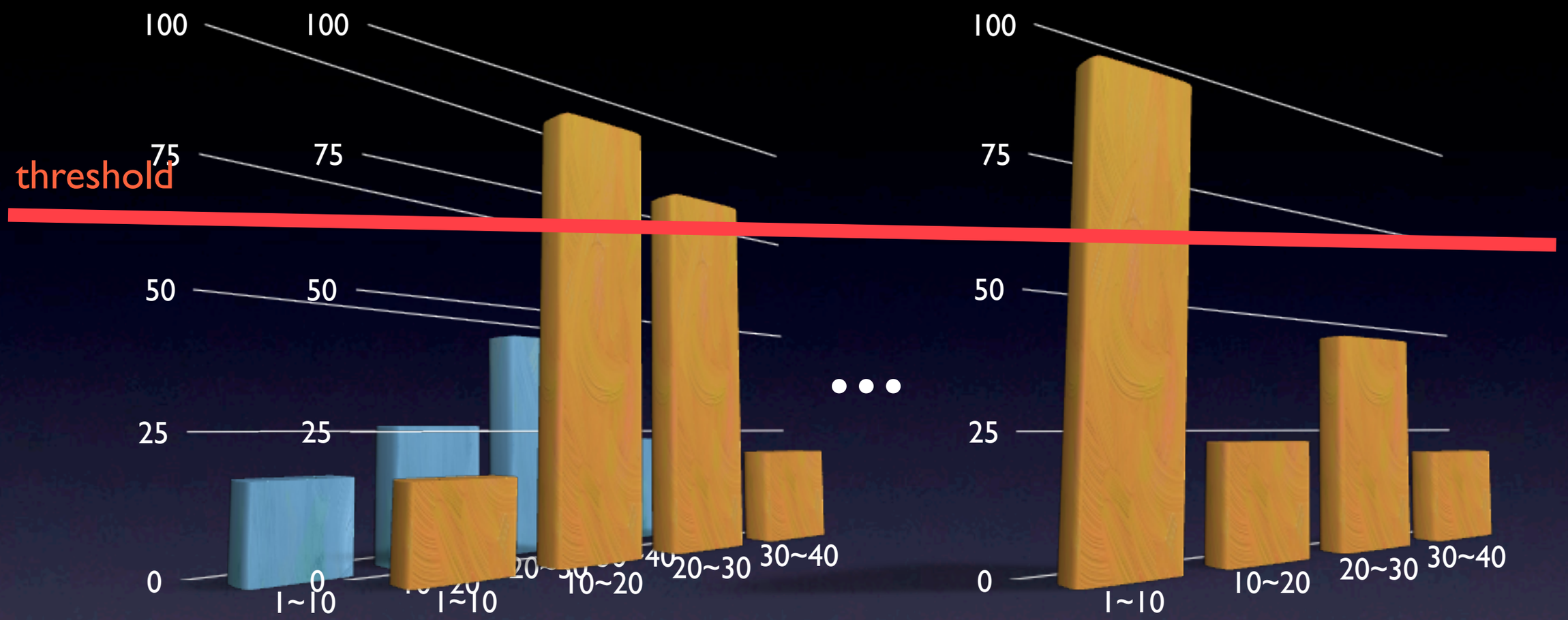
Bottom-Up Methods

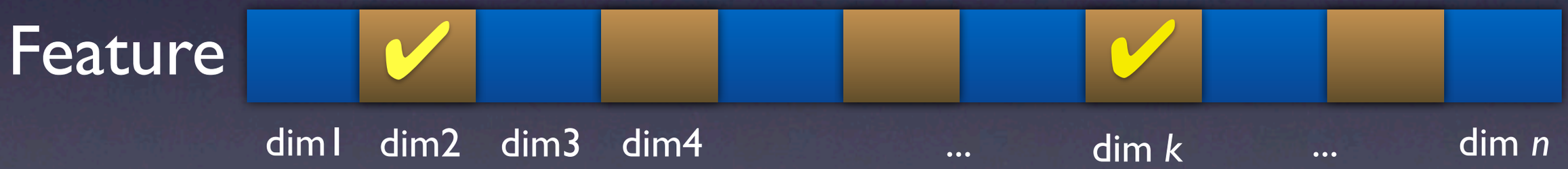


Feature



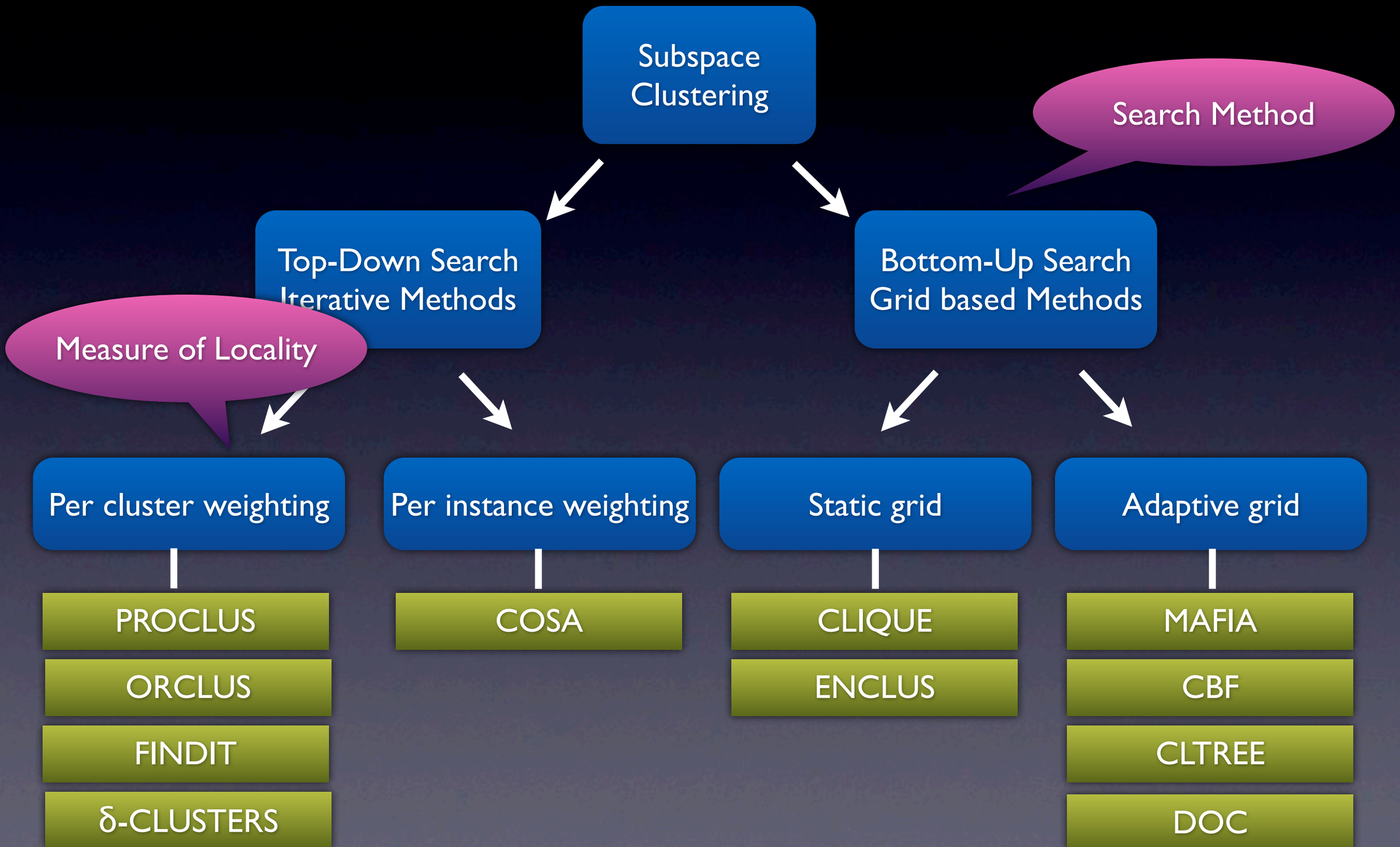
Bottom-Up Methods



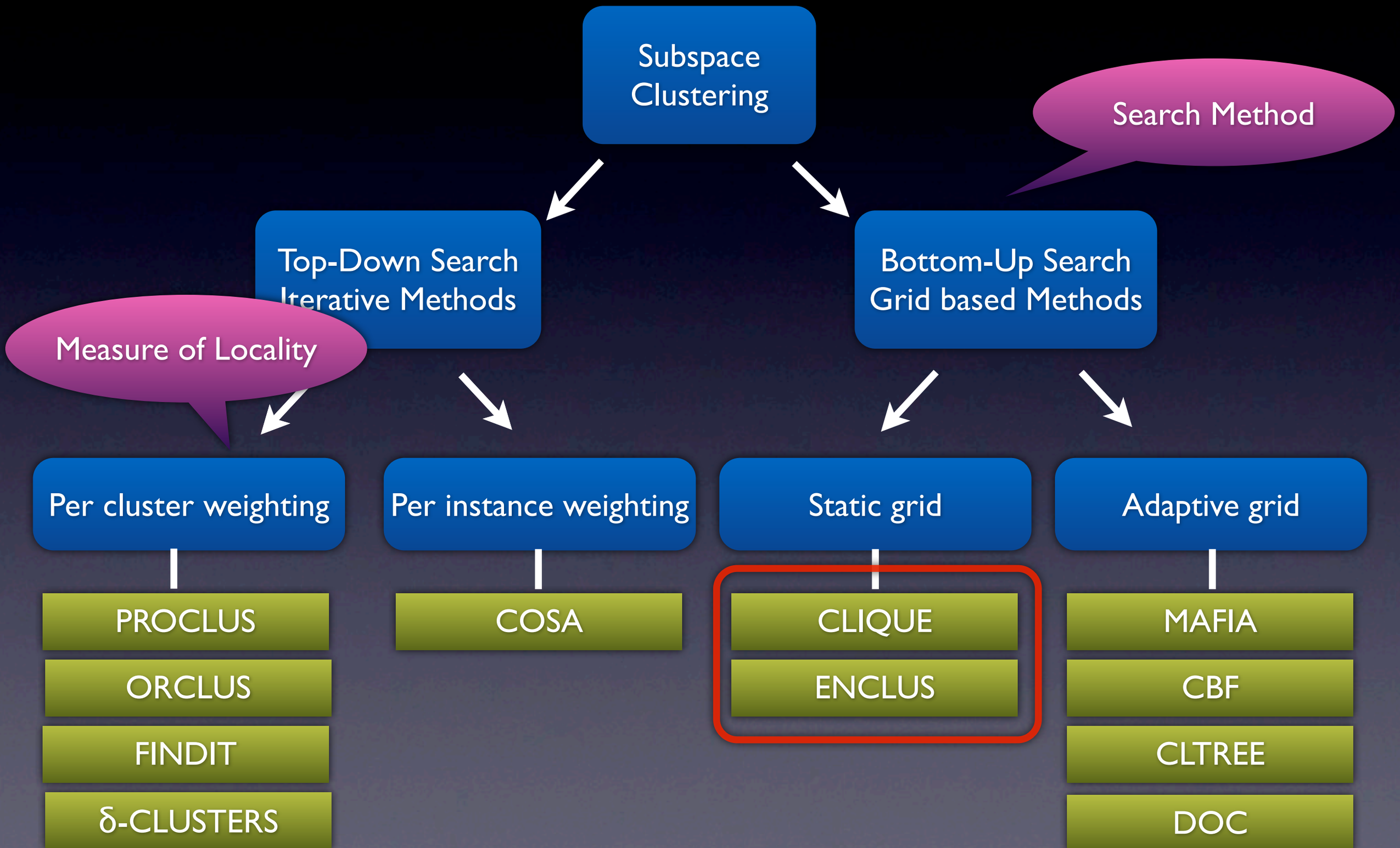


- Choose the dimensions that has dense bin (over the threshold)
- Form 2-dimensional spaces using chosen dimensions then cluster
- Allowed overlapping clusters

Family Tree of Subspace Clustering



Family Tree of Subspace Clustering



CLIQUE

- First subspace clustering method
- Procedure
  - Given subspaces of dense units
  - Sort them by *coverage* (= # pts in the dense unit /# of pts)
  - Pick the subspaces whose coverage is the largest
  - Merge adjacent dense grids in selected subspaces until finding a maximal region
- Drawback: possibly miss small clusters (might be important)

ENCLUS

Clustering accuracy

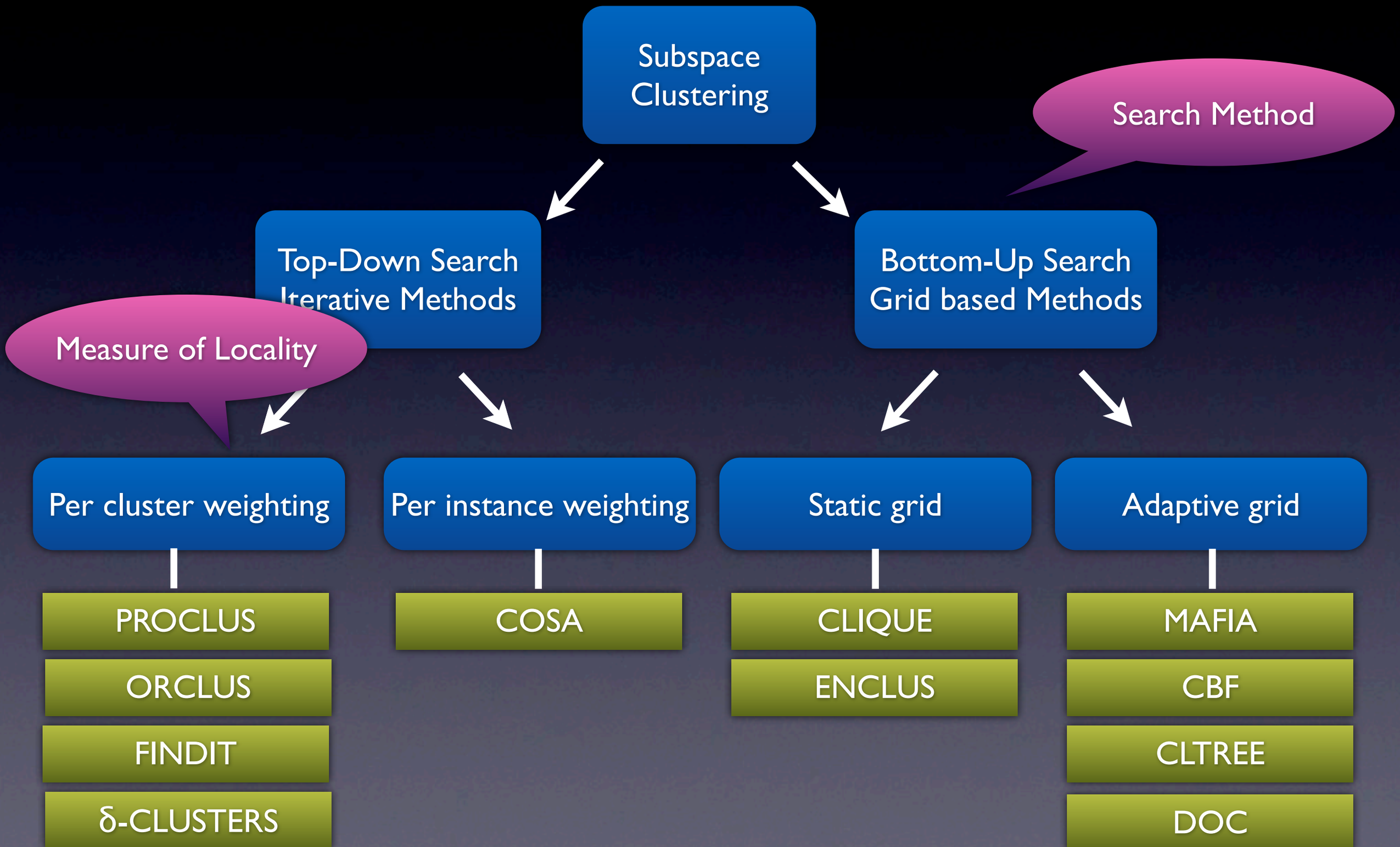
- Consider coverage, density, and correlation
- Similar to CLIQUE
- Different by using *entropy* instead of using *coverage* to find subspace

➡ *Entropy*: coverage, density and correlation

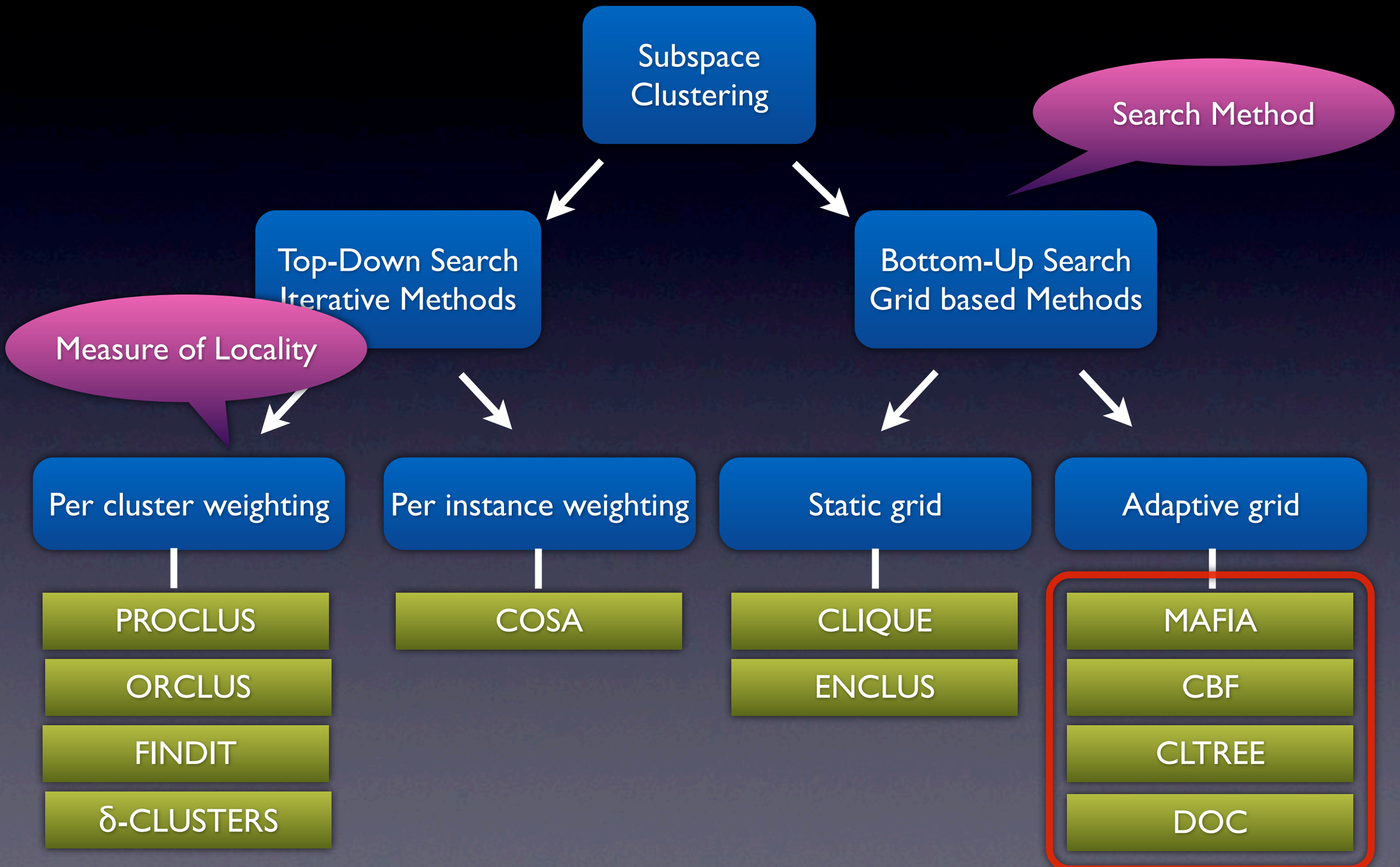
✓ Low entropy means picked distribution (low entropy ≈ high coverage)

✓ *Interest*: measure of correlation, difference bet'n sum of *entropy* measurements

Family Tree of Subspace Clustering



Family Tree of Subspace Clustering



Scalability

MAFIA: Merging of Adaptive Finite Intervals

Clustering accuracy

- Another extension of CLIQUE
  - Smaller number of bins
    - ➡ By merging the adjacent bins
  - Parallelizing the procedure
- Procedure
  - Make initial histogram
  - Merge adjacent bins of similar density
    - ➡ Bin boundary accurately represents cluster boundary

Scalability

MAFIA: Merging of Adaptive Finite Intervals

Clustering accuracy

- Another extension of CLIQUE

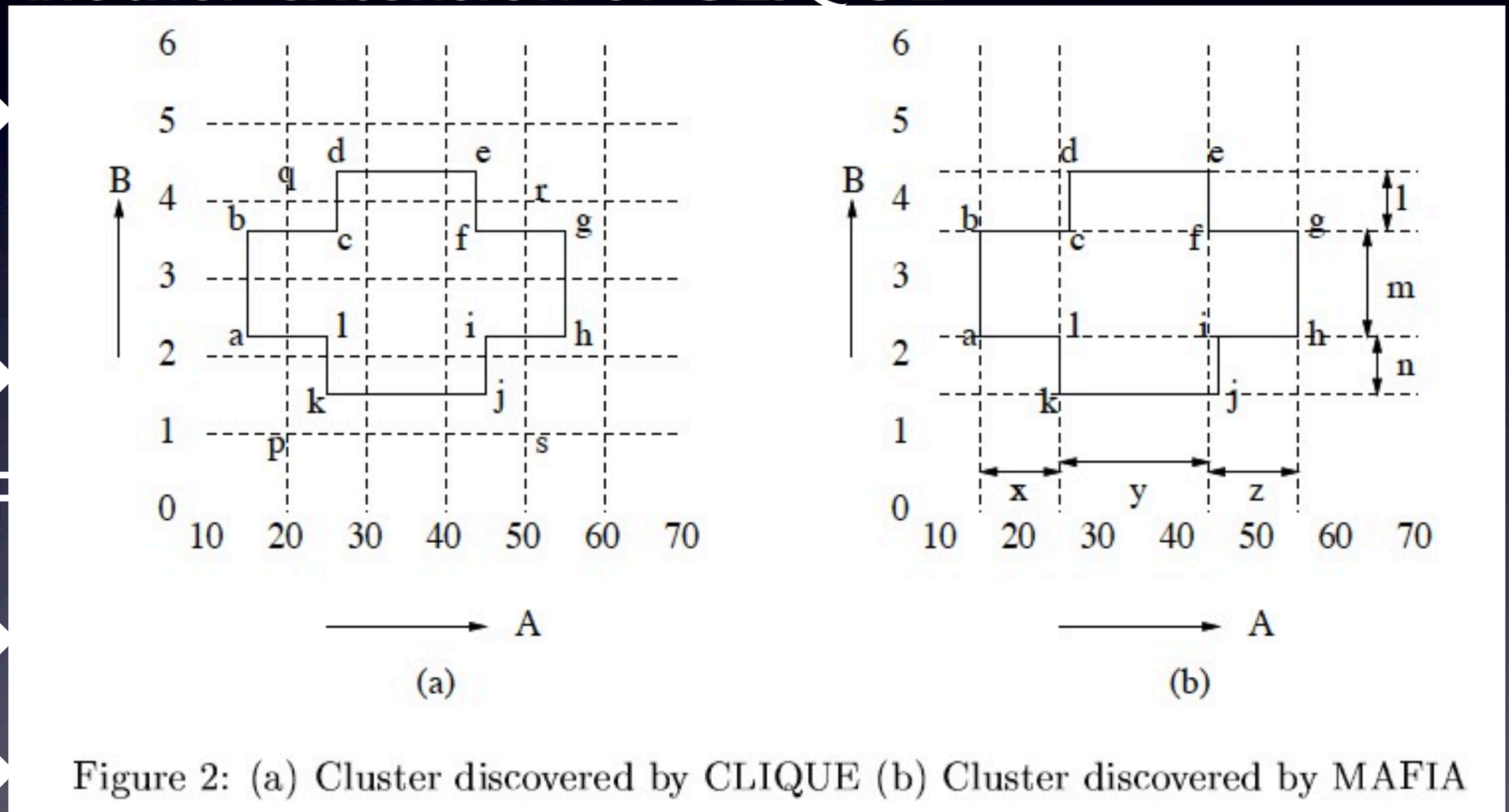


Figure 2: (a) Cluster discovered by CLIQUE (b) Cluster discovered by MAFIA

➡ Bin boundary accurately represents cluster boundary

Scalability

CBF: Cell Based Clustering

- Scales to dimension
  - As dimension grows, number of bins grows exponentially
  - Use a cell creation algorithm: find optimal partition of bins
- Scales to size of dataset
  - Stores bins in an efficient filtering based index structure
- Slightly less accurate than CLIQUE

CLTree

Scalability

- Different binning method
  - Use decision tree to partition dim. to bin
- Scale well to size of dataset and # of subspace dim.
- No need to specify # and size of cluster

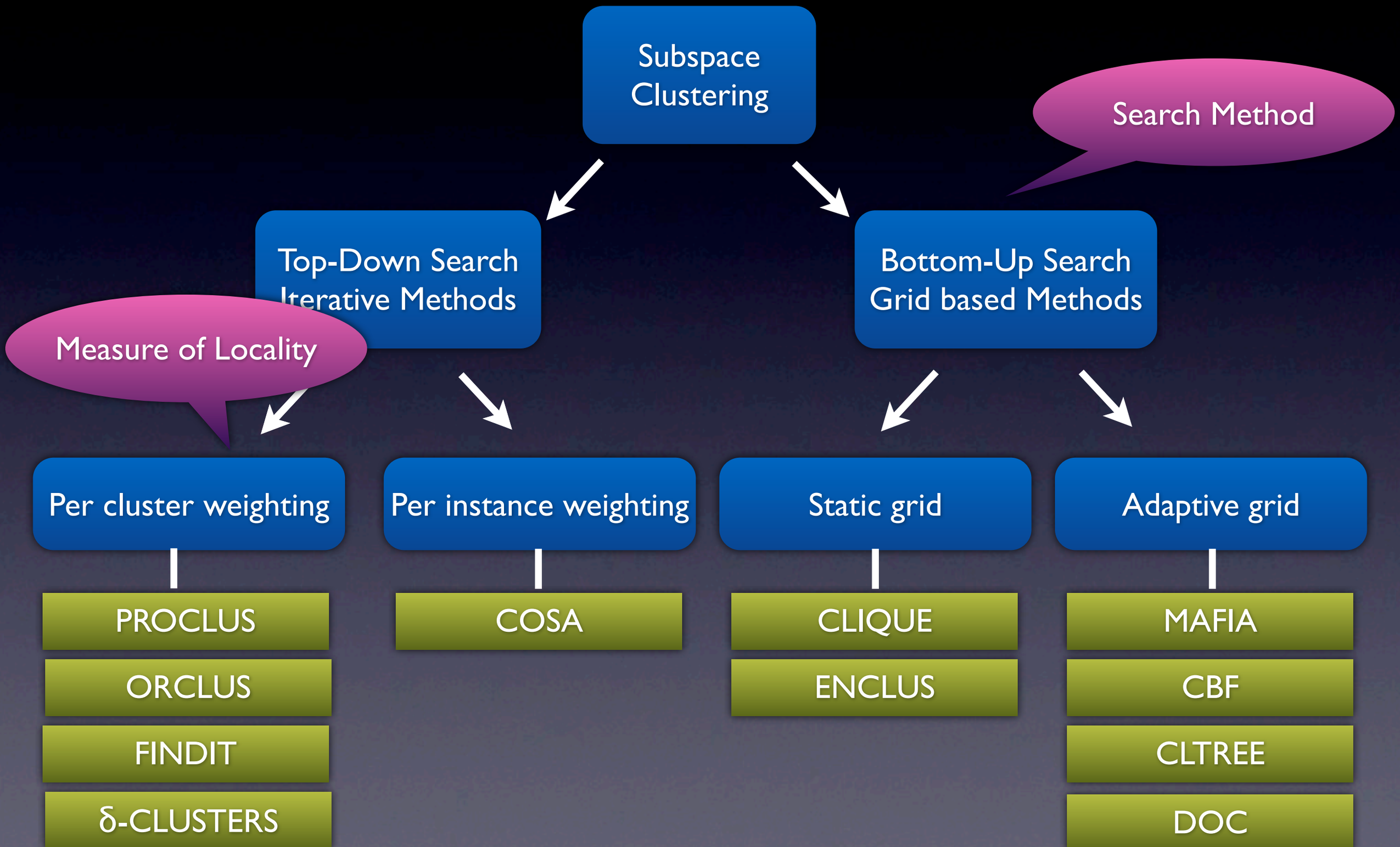
DOC: Density based Optimal projective Clustering

Hybrid
method of bottom-up and top-down

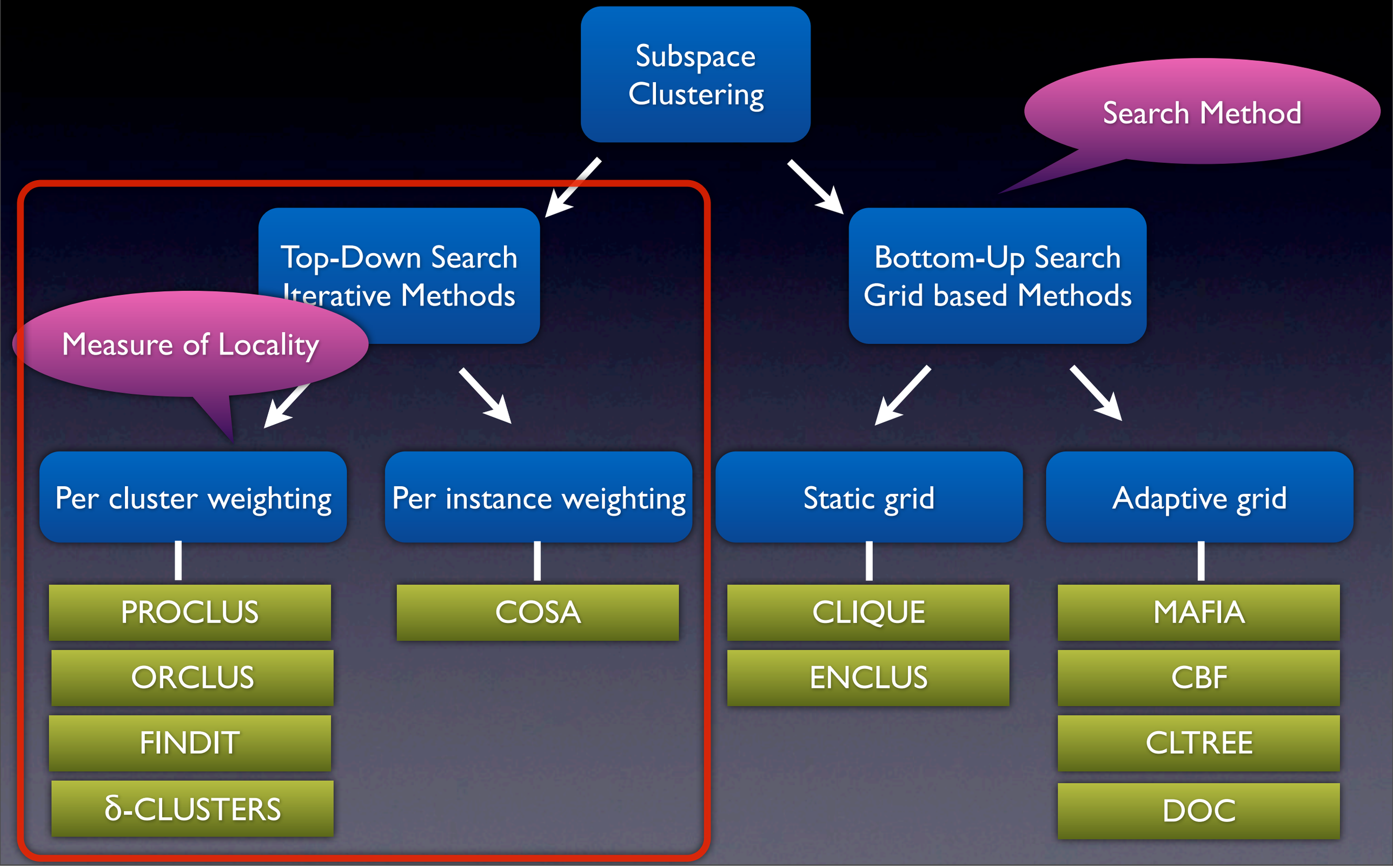
- Define "projective cluster": (C,D)
  - C: subset of data, D: subset of dimension
- Goal: find a (C,D) pair that C shows best clustering tendency in D

- Procedure
  - Define discriminating set X (random subset of data): discriminating relevant dimension to irrelevant dimension
  - Random sample p from C, All samples q from X
  - Must hold: $|p(i)-q(i)| \leq w$, i is dimension in D
  - Iterate with random re-sampling of p and X, until the best results

Family Tree of Subspace Clustering



Family Tree of Subspace Clustering

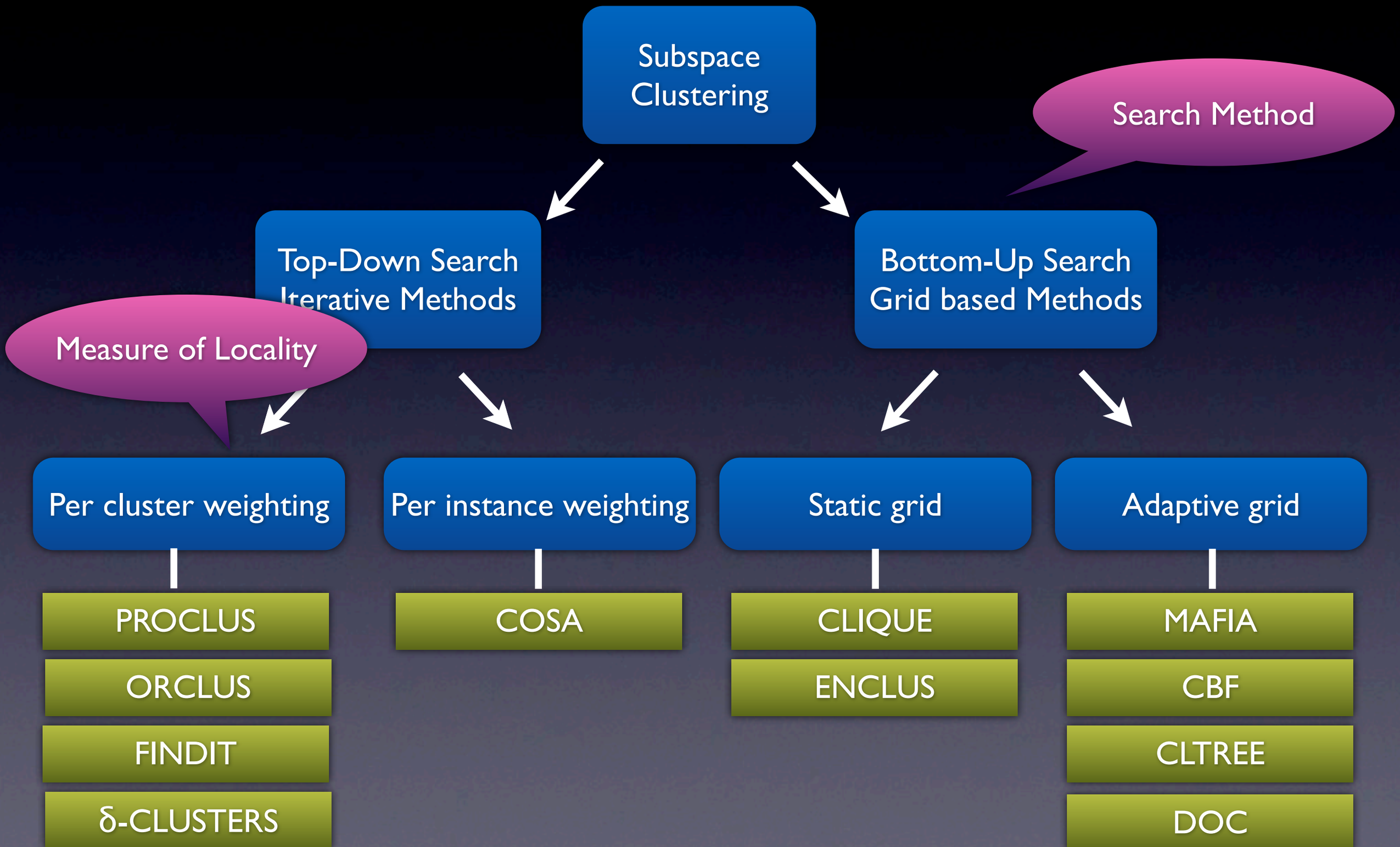


Top-Down Methods

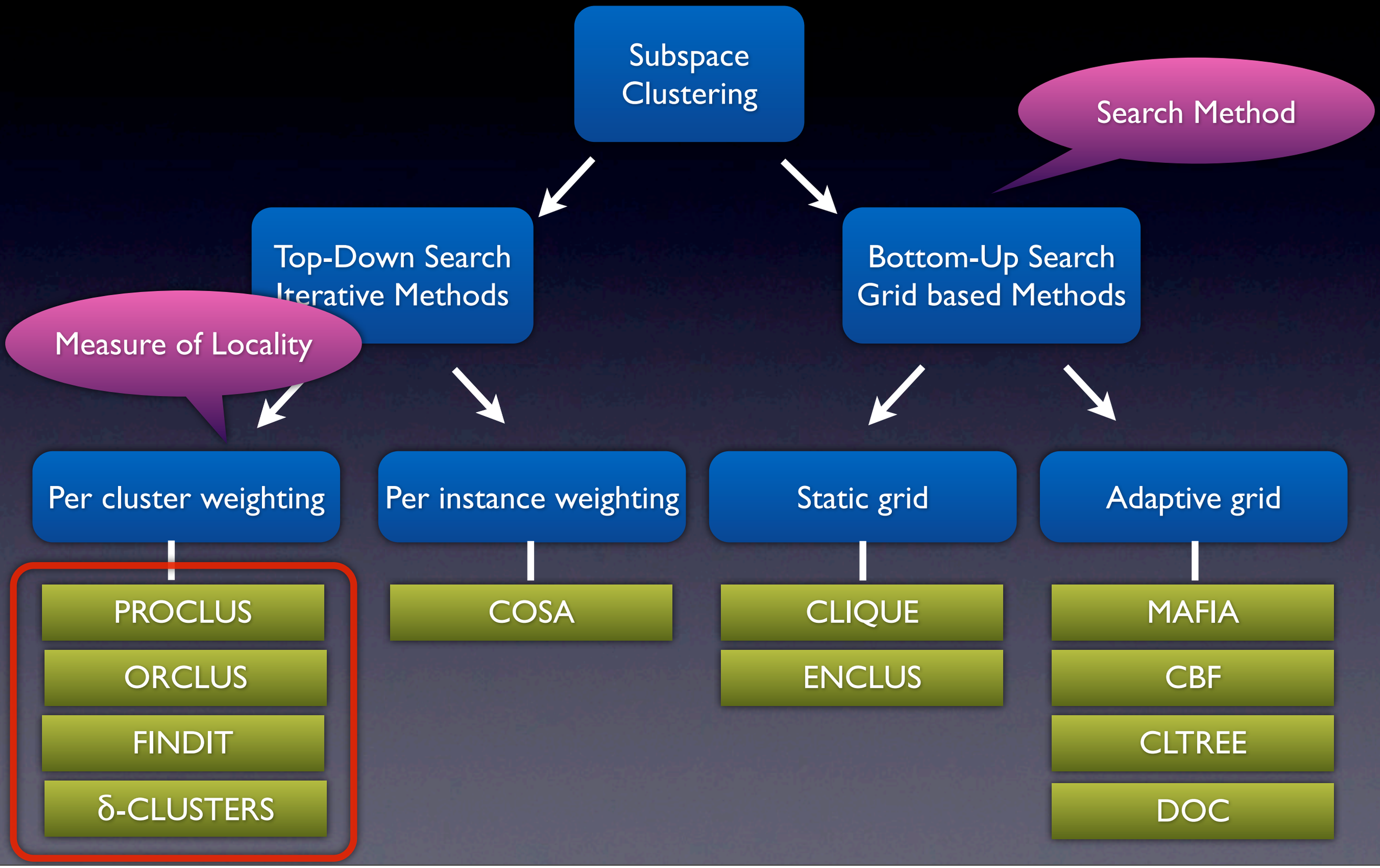
- Procedure
  - Cluster with initial weight in full dimension
  - Update weight on dimension for each cluster
    - Re-do cluster with the updated weight
- Expensive full dim. clustering is in each iteration
- Not allowing overlapped clustering
- But allowing outliers

Binary

Family Tree of Subspace Clustering



Family Tree of Subspace Clustering



PROCLUS

- First algorithm of top-down method, randomized algorithm
- Procedure
  - init: randomly choose samples from data
  - iteration: choose potential medoids far apart from each other using a greedy algorithm
  - cluster refinement: select random set of k medoids and replace bad medoids, check whether clustering improved with the new set of medoids
- For each medoid, choose a subset of dimensions whose *avg distance* < *statistical expectation*

ORCLUS

Clustering accuracy

- Extension of PROCLUS using Eigen-analysis
- Procedure
  - assign cluster: distance bet'n pts and cluster center is defined by new subspace vectors E by Eigen-analysis (in next step)
  - subspace determination (compute E): choos… orthonormal bases with least spread (by covariance matrix)

    Slow
  - merge: clusters near and having similar directions of least spread are merged
- To speed up, random sampling can be used

FINDIT: Fast and Intelligent Subspace Clustering Using Dimension Voting

Clustering accuracy

- Similar to PROCLUS
- Use new distance measure: DOD (Dimension Oriented Distance)
- Procedure
  - sampling: select two small sets by randomly choose samples from dataset, find initial medoids
  - cluster forming: find correlated dimensions using DOD measure for each medoid. Merging nearby points until the quality stabilized
  - data assigning: assign to a medoid based on subspace found

Varying sized subspace per cluster

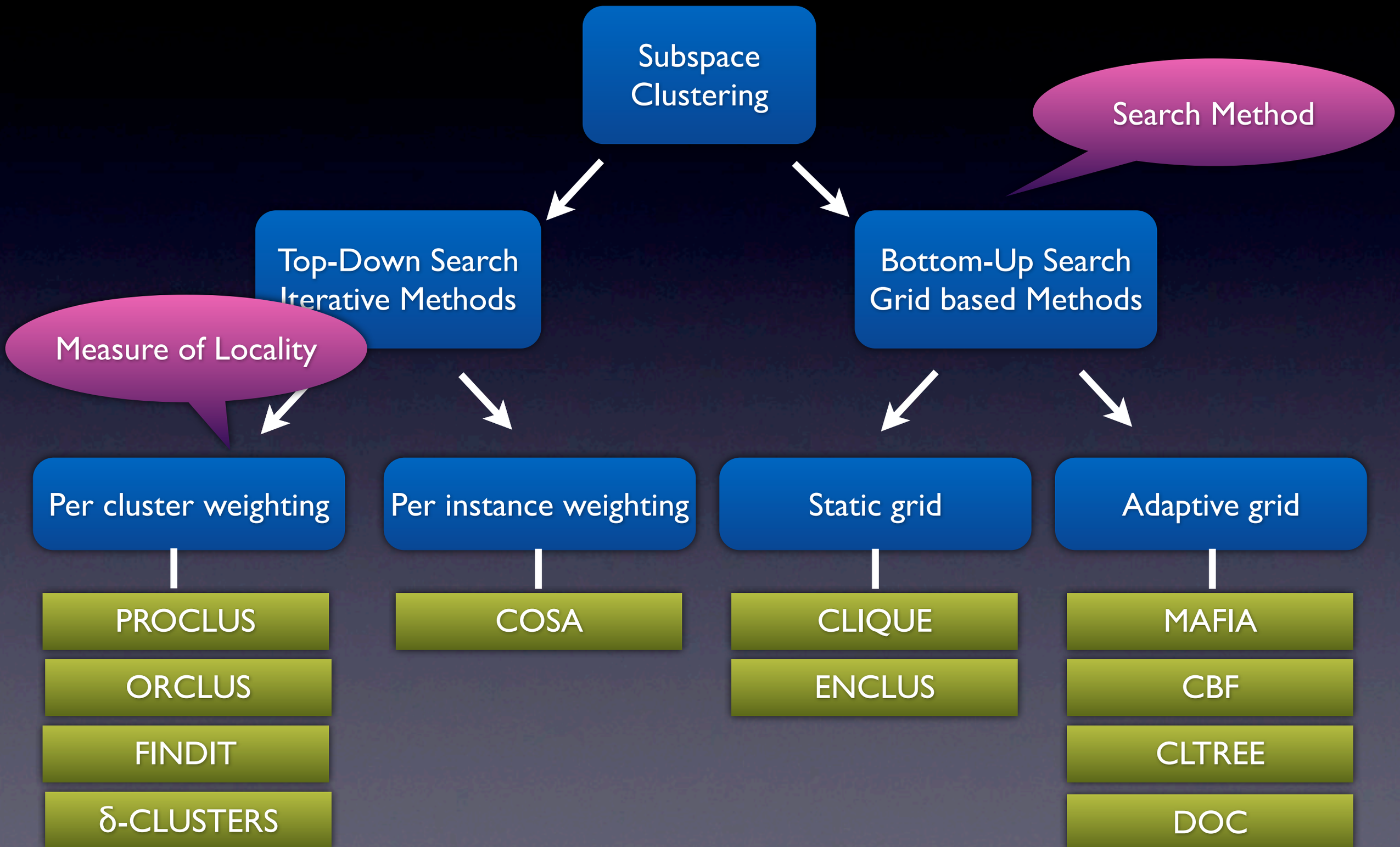
- Parameters: min. # of pts in a cluster, min. distance bet'n two clusters

δ-Clusters

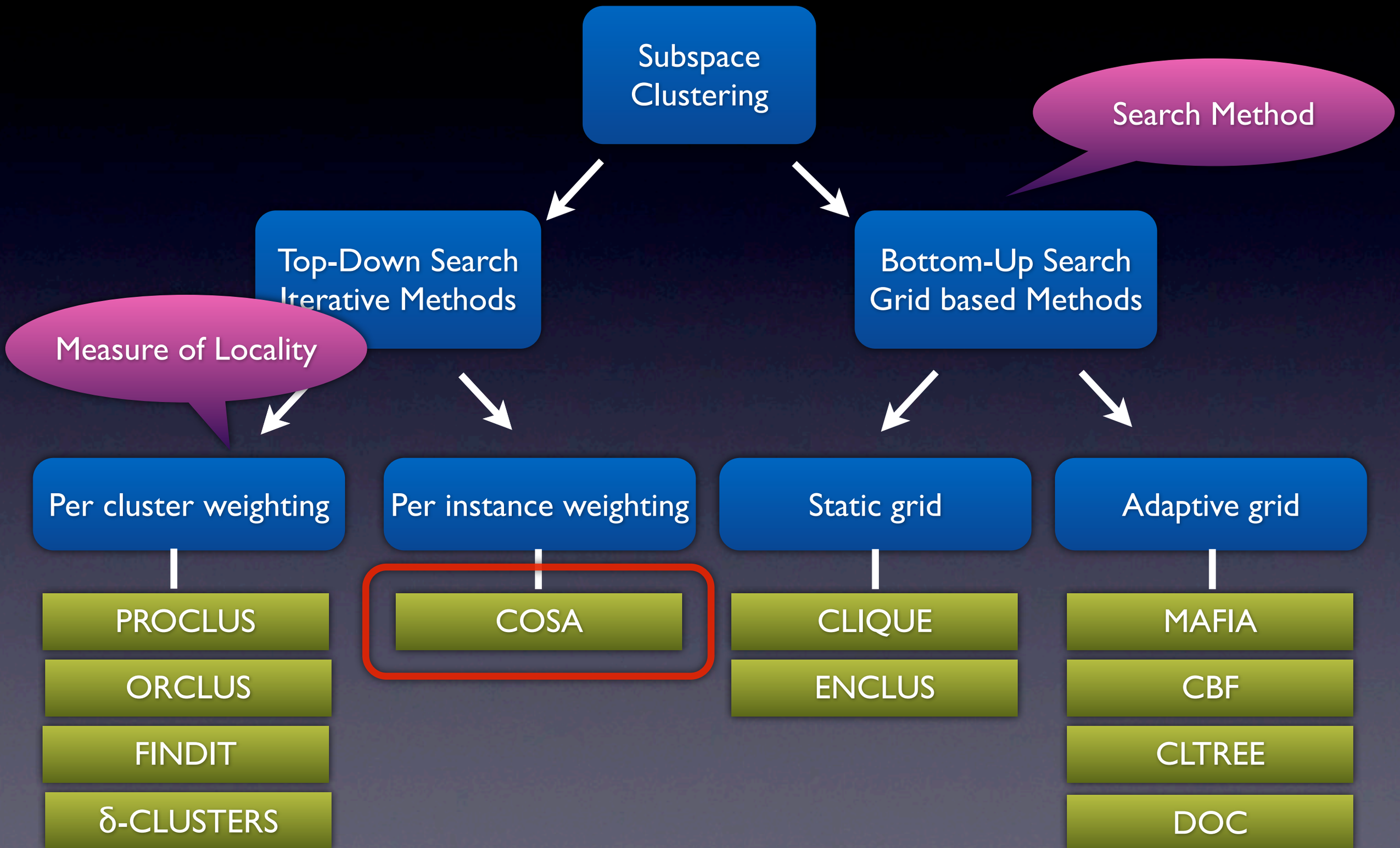
Different distance measure

- New distance measure: *coherence*
- Procedure
  - Begin with initial seeds
    - randomly swapping dimensions/data points to improve clustering quality
    - With quality measure of *Residue* (decrease in coherence), iterates the previous steps or stop
- Parameters: # of clusters, individual cluster size

Family Tree of Subspace Clustering



Family Tree of Subspace Clustering



COSA: Clustering On Subsets of Attributes

of subspace dimension need not be specified

- Assigning weights on dimension for <u>each data points</u> (not cluster-wise)
- Procedure
  - Init. with equal weights on all dimensions
  - Find *knn* and give high weights on dimensions with a small dispersion
  - Using weighted dimension to refine the distance measure and find new *knn*
  - iterate until the *knn* stabilized
  - Use final distance measure to get a distance based clustering method
- Stable over various k value in *knn*

Empirical Comparison

- Bottom-Up representative
  - MAFIA
- Top-Down representative
  - FINDIT

- Expected Results
  - Scalability
    - Top-Down scales well to dataset size
    - Bottom-Up scales well to large dimension
  - Clustering Accuracy
    - ?

Empirical Comparison

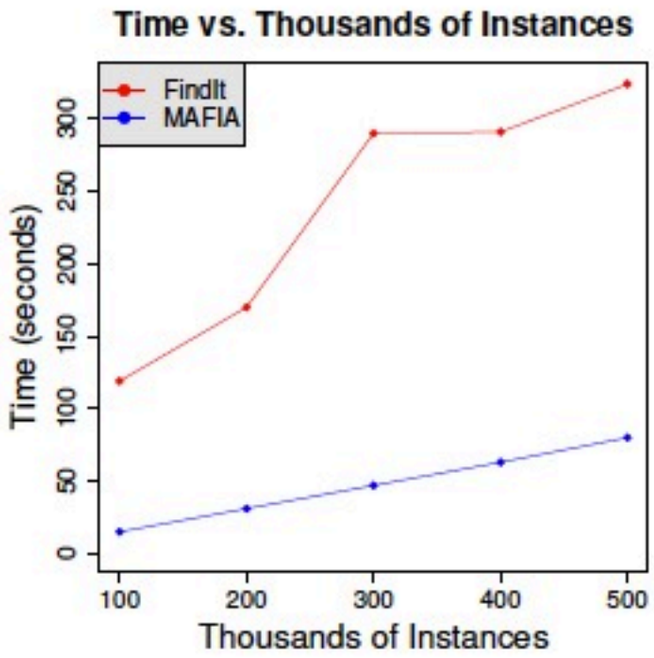
- Bottom-Up representative
  - MAFIA
- Top-Down representative
  - FINDIT

- Expected Results
  - Scalability
    - Top-Down scales well to dataset size
    - Bottom-Up scales well to large dimension
  - Clustering Accuracy
    - ?

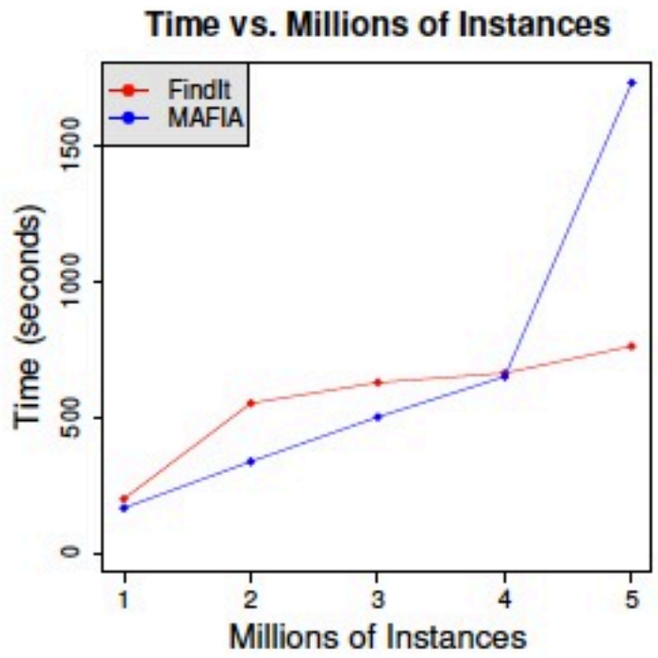
Random sampling

Adding nature

Scalability to Dataset size



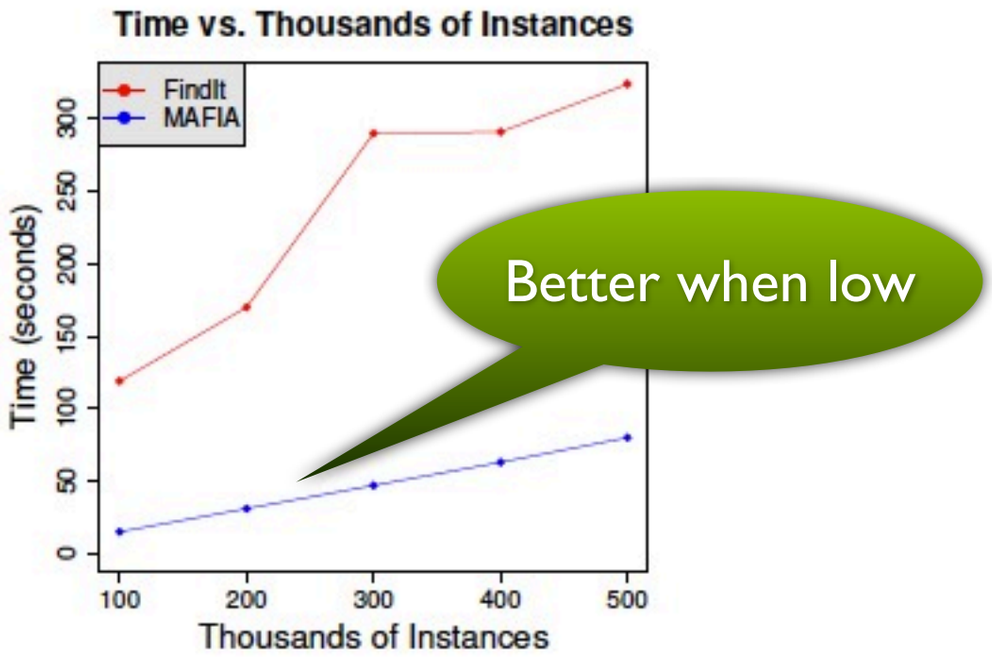
(a) Thousands



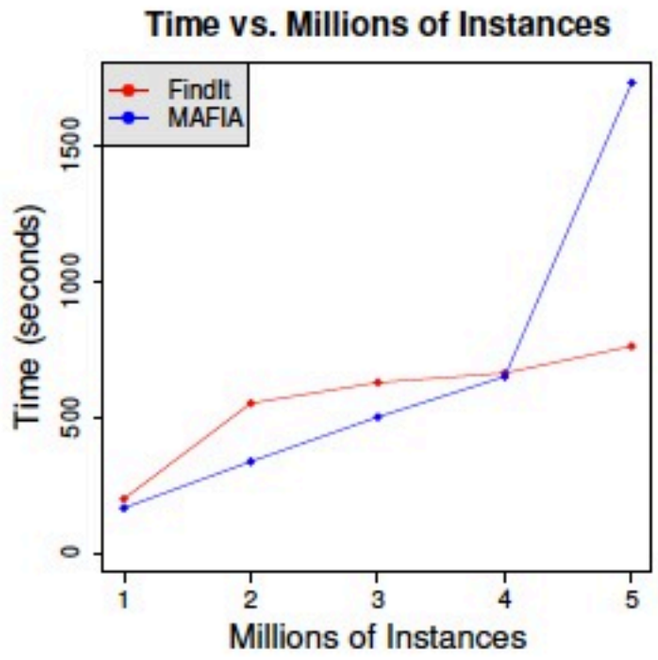
(b) Millions

Figure 6: Running time *vs.* number of instances for FINDIT (top-down) and MAFIA (bottom-up). MAFIA outperforms FINDIT in smaller datasets. However, due to the use of sampling techniques, FINDIT is able to maintain relatively high performance even with datasets containing five million records.

Scalability to Dataset size



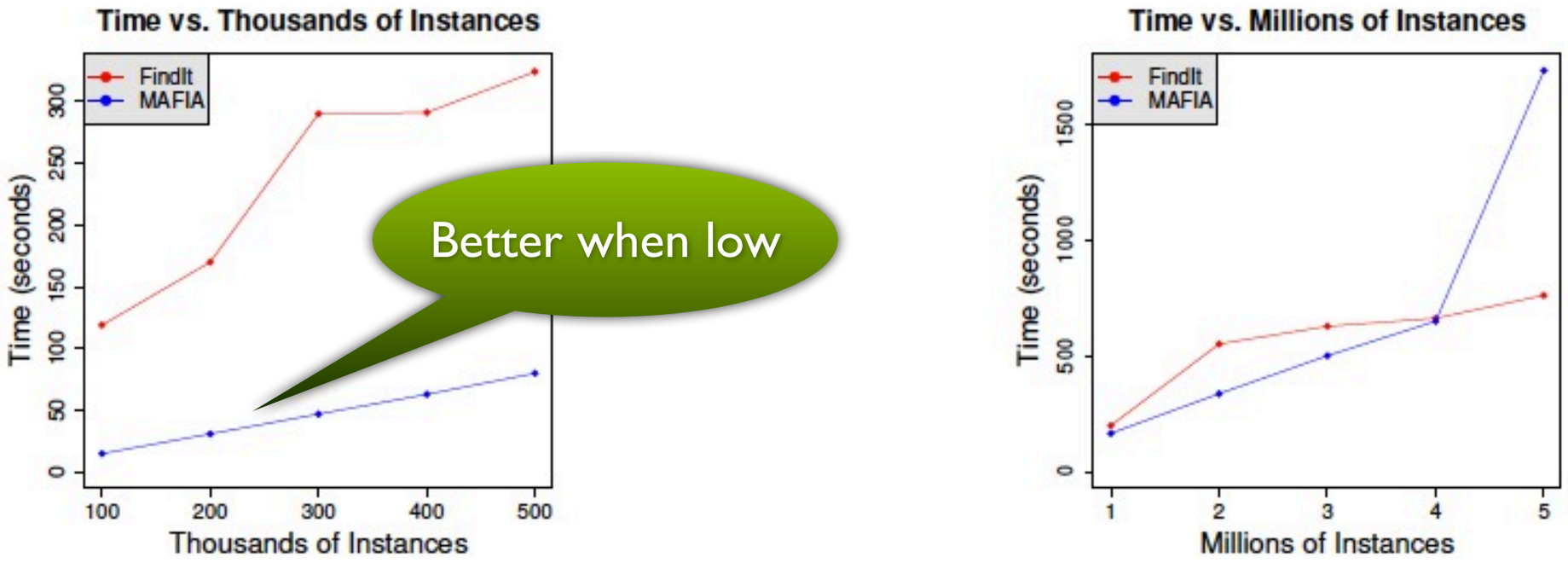
(a) Thousands



(b) Millions

Figure 6: Running time *vs.* number of instances for FINDIT (top-down) and MAFIA (bottom-up). MAFIA outperforms FINDIT in smaller datasets. However, due to the use of sampling techniques, FINDIT is able to maintain relatively high performance even with datasets containing five million records.

Scalability to Dataset size



(a) Thousands

(b) Millions

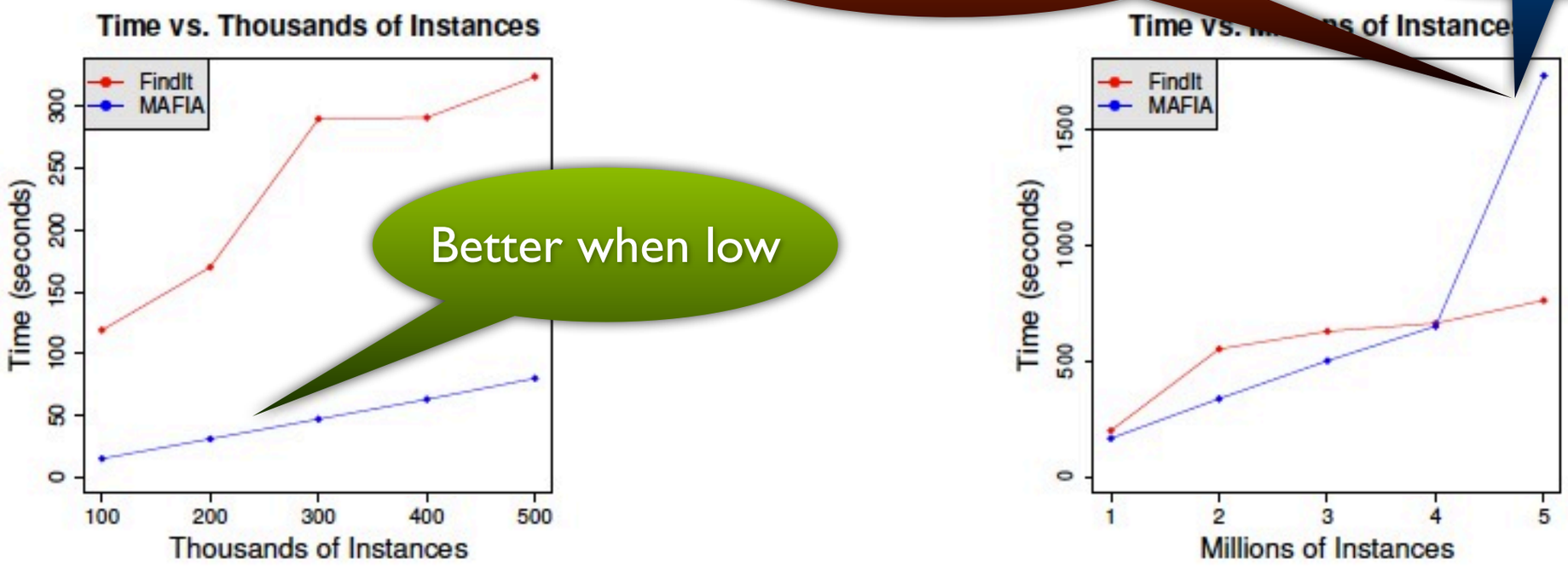
Figure 6: Running time *vs.* number of instances for FINDIT (top-down) and MAFIA (bottom-up). MAFIA outperforms FINDIT in smaller datasets. However, due to the use of sampling techniques, FINDIT is able to maintain relatively high performance even with datasets containing five million records.

MAFIA(bottom-up) wins?

Scalability to Dataset size

Might lose accuracy

Power of random sampling of data points



(a) Thousands

(b) Millions

Figure 6: Running time *vs.* number of instances for FINDIT (top-down) and MAFIA (bottom-up). MAFIA outperforms FINDIT in smaller datasets. However, due to the use of sampling techniques, FINDIT is able to maintain relatively high performance even with datasets containing five million records.

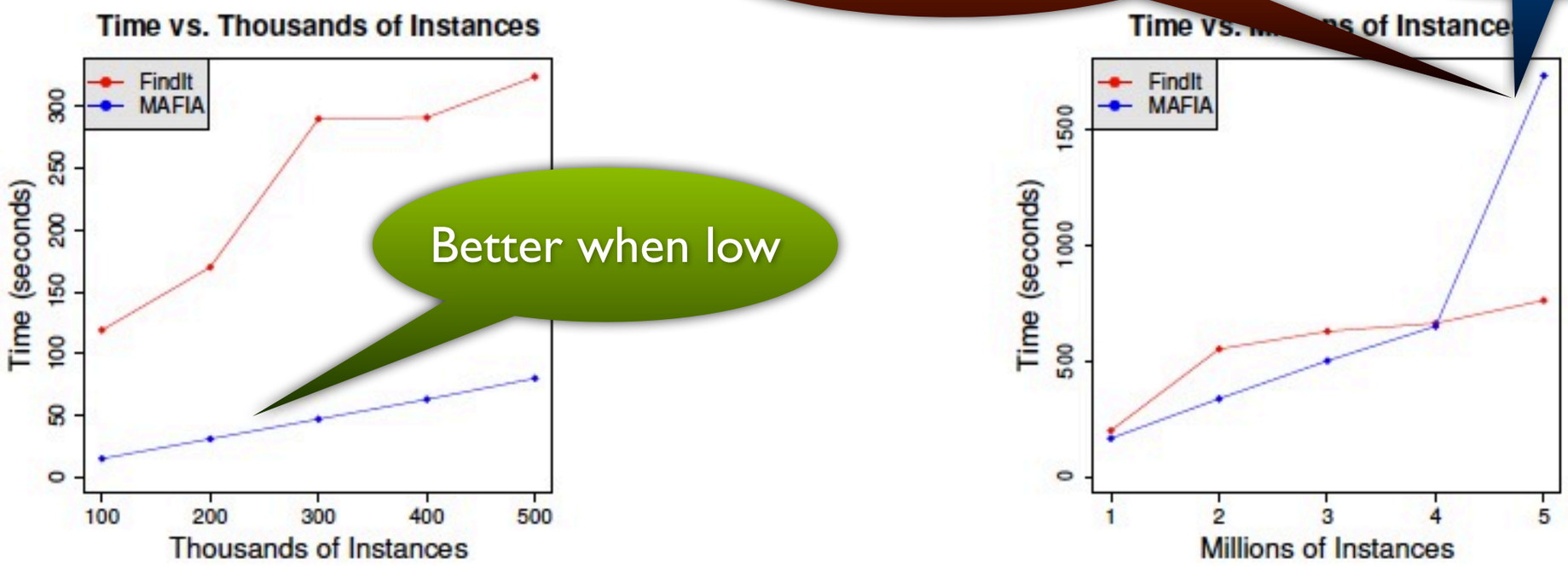
Better when low

MAFIA(bottom-up) wins?

Scalability to Dataset size

Might lose accuracy

Power of random sampling of data points



(a) Thousands

(b) Millions

Figure 6: Running time *vs.* number of instances for FINDIT (top-down) and MAFIA (bottom-up). MAFIA outperforms FINDIT in smaller datasets. However, due to the use… performance even with datasets containing five million…

Better when low

MAFIA(bottom-up) wins?

FINDIT(top-down) wins!

Scalability to Dimension

- Since bottom-Up method only add interesting dimension, it doesn't bother with dimension increase

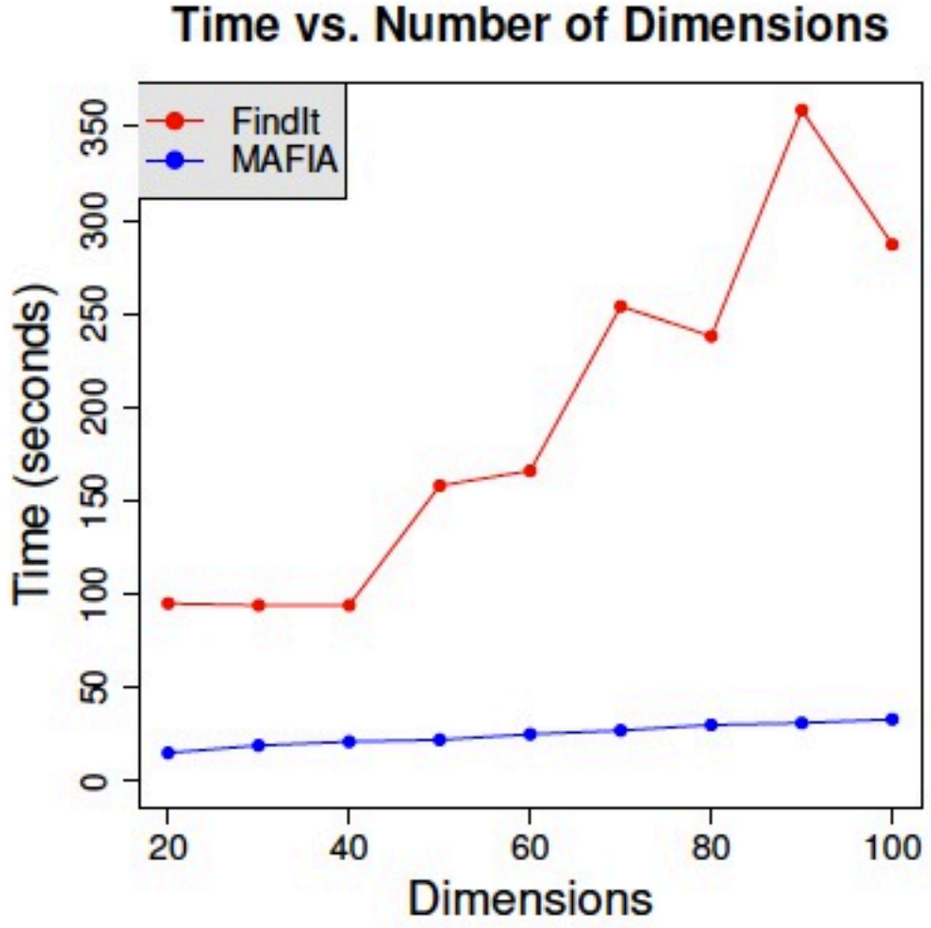


Figure 7: Running time *vs.* number of dimensions for FINDIT (top-down) and MAFIA (bottom-up). MAFIA consistently outperforms FINDIT and scales better as the number of dimensions increases.

Scalability to Dimension

- Since bottom-Up method only add interesting dimension, it doesn't bother with dimension increase

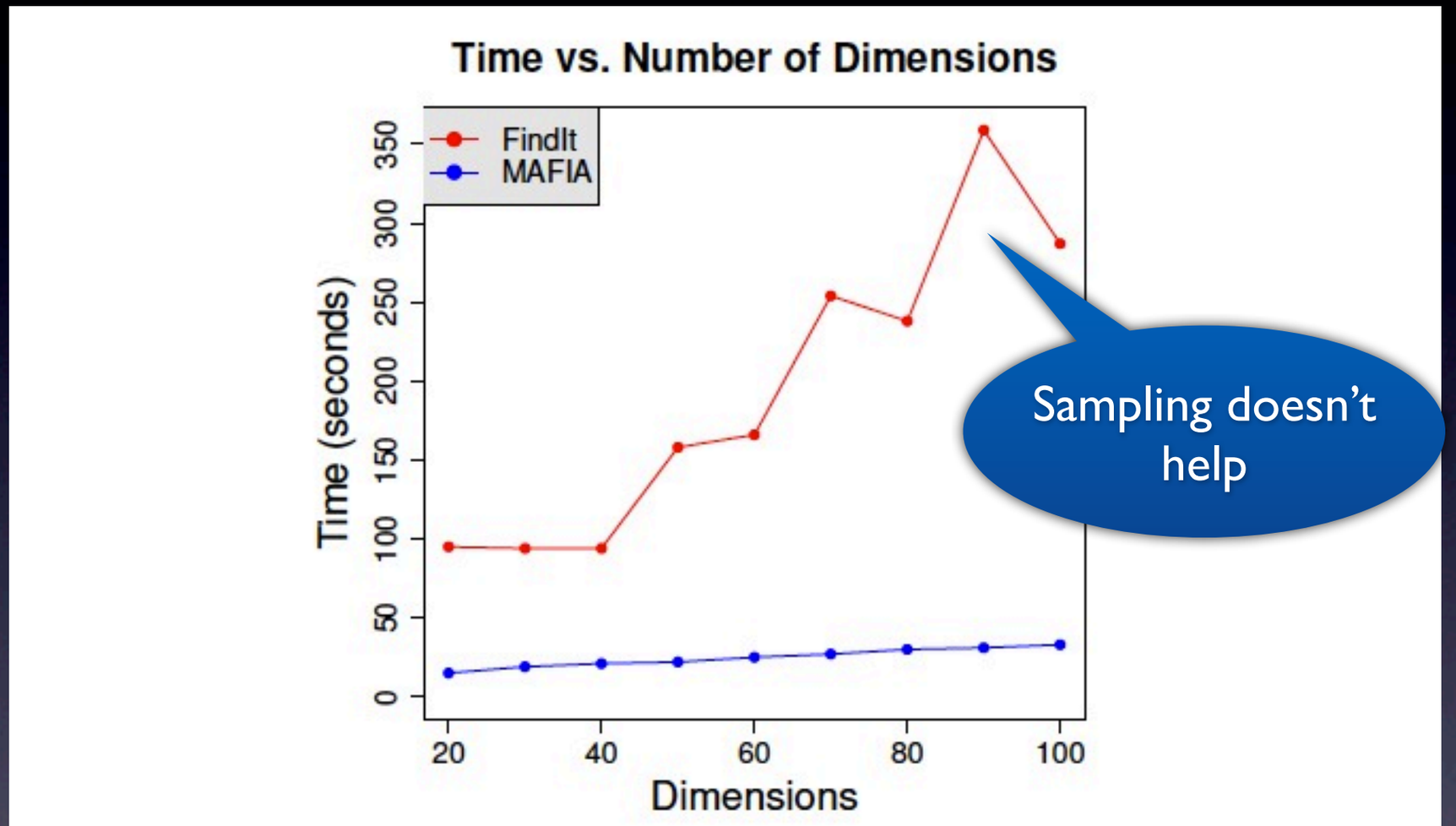
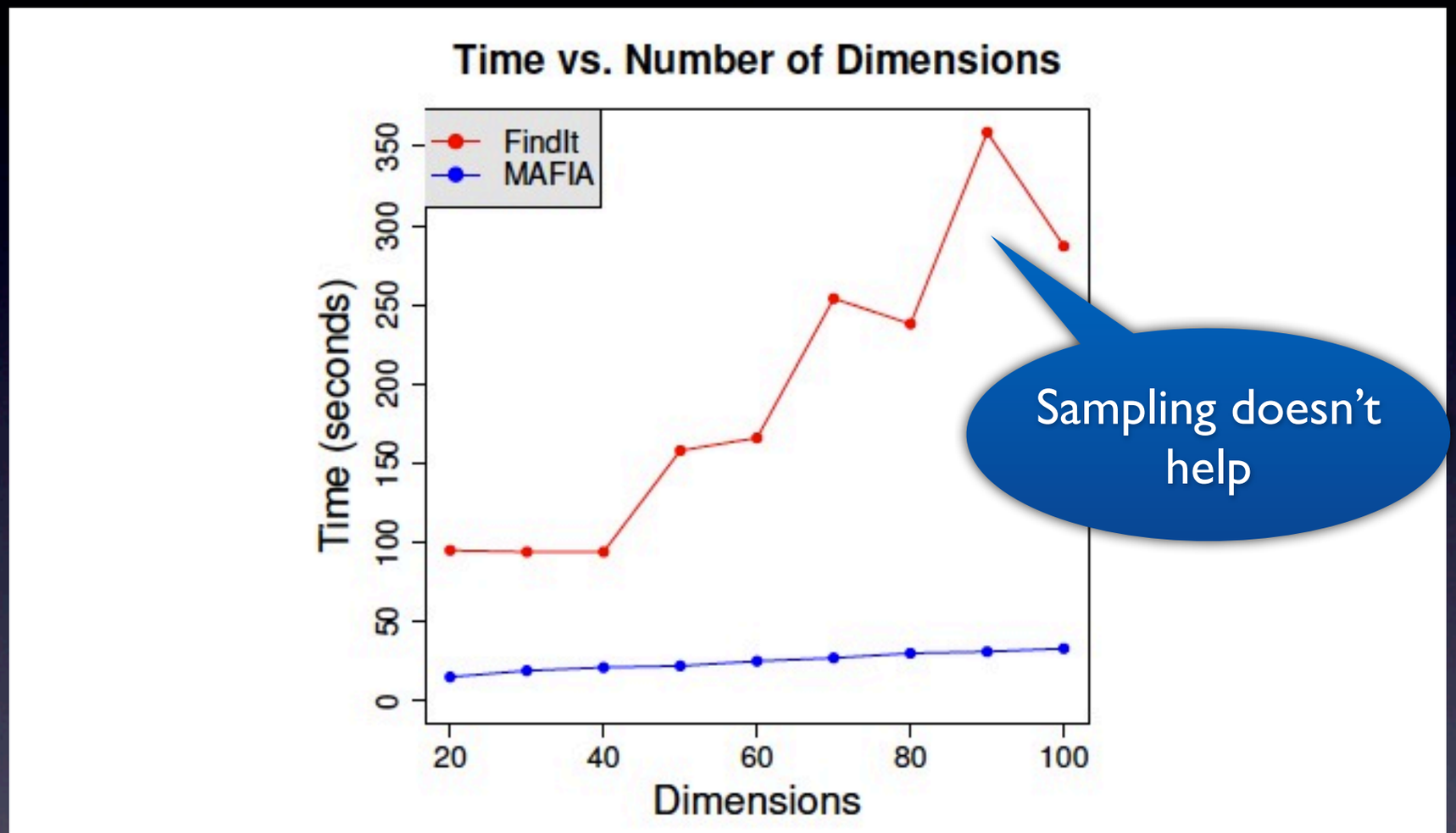


Figure 7: Running time *vs.* number of dimensions for FINDIT (top-down) and MAFIA (bottom-up). MAFIA consistently outperforms FINDIT and scales better as the number of dimensions increases.

Scalability to Dimension

- Since bottom-Up method only add interesting dimension, it doesn't bother with dimension



time *vs.* number of dimensions for and MAFIA (bottom-up). MAFIA ms FINDIT and scales better as the increases.

MAFIA(bottom-up) wins!

Clustering Accuracy

- Low dimensional data (D=20)

Cluster	1	2	3	4	5
Input	(4, 6, 12, 14, 17)	(1, 8, 9, 15, 18)	(1, 7, 9, 18, 20)	(1, 12, 15, 18, 19)	(5, 14, 16, 18, 19)
Output	(4, 6, 14, 17)	(1, 8, 9, 15, 18)	(7, 9, 18, 20)	(12, 15, 18, 19)	(5, 14, 18, 19)

Table 1: MAFIA misses one dimension in 4 out 5 clusters with $N = 100,000$ and $D = 20$.

Cluster	1	2	3	4	5
Input	(11, 16)	(9, 14, 16)	(8, 9, 16, 17)	(0, 7, 8, 10, 14, 16)	(8, 16)
Output	(11, 16)	(9, 14, 16)	(8, 9, 16, 17)	(0, 7, 8, 10, 14, 16)	(8, 16)

Table 2: FINDIT uncovers all of the clusters in the appropriate dimensions with $N = 100,000$ and $D = 20$.

Clustering Accuracy

- Low dimensional data (D=20)

Missing!

Missing!

Cluster	1	2	3	4	5
Input	(4, 6, 12, 14, 17)	(1, 8, 9, 15, 18)	(1, 7, 9, 18, 20)	(1, 12, 15, 18, 19)	(5, 14, 16, 18, 19)
Output	(4, 6, 14, 17)	(1, 8, 9, 15, 18)	(7, 9, 18, 20)	(12, 15, 18, 19)	(5, 14, 18, 19)

Table 1: MAFIA misses one dimension in 4 out 5 clusters with $N = 100,000$ and $D = 20$.

Cluster	1	2	3	4	5
Input	(11, 16)	(9, 14, 16)	(8, 9, 16, 17)	(0, 7, 8, 10, 14, 16)	(8, 16)
Output	(11, 16)	(9, 14, 16)	(8, 9, 16, 17)	(0, 7, 8, 10, 14, 16)	(8, 16)

Table 2: FINDIT uncovers all of the clusters in the appropriate dimensions with $N = 100,000$ and $D = 20$.

Clustering Accuracy

- Low dimensional data (D=20)

Missing!

Missing!

Cluster	1	2	3	4	5
Input	(4, 6, 12, 14, 17)	(1, 8, 9, 15, 18)	(1, 7, 9, 18, 20)	(1, 12, 15, 18, 19)	(5, 14, 16, 18, 19)
Output	(4, 6, 14, 17)	(1, 8, 9, 15, 18)	(7, 9, 18, 20)	(12, 15, 18, 19)	(5, 14, 18, 19)

Table 1: MAFIA misses one dimension in 4 out 5 clusters with $N = 100,000$ and $D = 20$.

Cluster	1	2	3	4	5
Input	(11, 16)	(9, 14, 16)	(8, 9, 16, 17)	(0, 7, 8, 10, 14, 16)	(8, 16)
Output	(11, 16)	(9, 14, 16)	(8, 9, 16, 17)	(0, 7, 8, 10, 14, 16)	(8, 16)

Table 2: FINDIT uncovers all of the clusters in the appropriate dimensions with $N = 100,000$ and $D = 20$.

FINDIT(top-down) wins!

Clustering Accuracy (cont'd)

- Higher dimensional data (D=100)

Cluster	1	2	3	4	5
Input	(4, 6, 12, 14, 17)	(1, 8, 9, 15, 18)	(1, 7, 9, 18, 20)	(1, 12, 15, 18, 19)	(5, 14, 16, 18, 19)
Output	(4, 6, 14, 17)	(8, 9, 15, 18) (1, 8, 9, 18) (1, 8, 9, 15)	(7, 9, 18, 20)	(12, 15, 18, 19)	(5, 14, 18, 19)

Table 4: MAFIA misses one dimension in four out of five clusters. All of the dimensions are uncovered for cluster number two, but it is split into three smaller clusters. $N = 100,000$ and $D = 100$.

Cluster	1	2	3	4	5
Input	(1, 5, 16, 20, 27, 58)	(1, 8, 46, 58)	(8, 17, 18, 37, 46, 58, 75)	(14, 17, 77)	(17, 26, 41, 77)
Output	(5, 16, 20, 27, 58, 81)	*None Found*	(8, 17, 18, 37, 46, 58, 75)	(17, 77)	(41)

Table 3: FINDIT misses many dimensions and and entire cluster at high dimensions with with $N = 100,000$ and $D = 100$.

Clustering Accuracy (cont'd)

- Higher dimensional data (D=100)

Cluster	1	2	3	4	5
Input	(4, 6, 12, 14, 17)	(1, 8, 9, 15, 18)	(1, 7, 9, 18, 20)	(1, 12, 15, 18, 19)	(5, 14, 16, 18, 19)
Output	(4, 6, 14, 17)	(8, 9, 15, 18) (1, 8, 9, 18) (1, 8, 9, 15)	(7, 9, 18, 20)	(12, 18, 19)	(5, 14, 18, 19)

Missing!

Missing!

Table 4: MAFIA misses one [illegible] in four out of [illegible] clusters. All of the dimensions are covered for cluster number two, but it is split into thre[illegible]sters. $N = 1$[illegible] $D = 100$.

Missing!

Missing!

Missing!

Missing!

Cluster	1	2	3	4	5
Input	(1, 5, 16, 20, 27, 58)	(1, 8, 46, 58)	(8, 17, 18, 37, 46, 58, 75)	(14, 17, 77)	17, 26, 41, 77)
Output	(5, 16, 20, 27, 58, 81)	*None Found*	(8, 17, 18, 37, 46, 58, 75)	(17, 77)	(41)

Table 3: FINDIT misses many dimensions and and entire cluster at high dimensions with with $N = 100,000$ and $D = 100$.

Clustering Accuracy (cont'd)

- Higher dimensional data (D=100)

Missing! Missing!

Cluster	1	2	3	4	5
Input	(4, 6, 12, 14, 17)	(1, 8, 9, 15, 18)	(1, 7, 9, 18, 20)	(1, 12, 15, 18, 19)	(5, 14, 16, 18, 19)
Output	(4, 6, 14, 17)	(8, 9, 15, 18) (1, 8, 9, 18) (1, 8, 9, 15)	(7, 9, 18, 20)	(12, 18, 19)	(5, 14, 18, 19)

Table 4: MAFIA misses one dimension in four out of five clusters. All of the dimensions are recovered for cluster number two, but it is split into three clusters. $N = 1$ $D = 100$.

Missing! Missing! Missing! Missing!

Cluster	1	2	3	4	5
Input	(1, 5, 16, 20, 27, 58)	(1, 8, 46, 58)	(8, 17, 18, 37, 46, 58, 75)	(14, 17, 77)	17, 26, 41, 77)
Output	(5, 16, 20, 27, 58, 81)	*None Found*	(8, 17, 18, 37, 46, 58, 75)	(17, 77)	(41)

Table 3: FINDIT misses many dimensions and and entire cluster at high dimensions with with $N = 100,000$ and $D = 100$.

MAFIA (bottom-up) wins!

Summary of Results

- Clustering Accuracy
  - Small dimension: Top-down > Bottom-up
  - Large dimension: Top-down < Bottom-up
- Scalability
  - Top-Down scales well to dataset size
  - Bottom-Up scales well to large dimension

Summary of the Talk

- Subspace clustering: Clustering in its residing subspaces by choosing dimension
- Two approaches
  - Top-down: iterative evaluation of clustering quality, selection, and clustering
    - ➡ Find non-overlapping clusters (includes outliers)
    - ➡ Slow
    - ➡ Require # of cluster as parameter
  - Bottom-Up: select a dimension at a time
    - ➡ Find overlapping clusters
    - ➡ Require density threshold as parameter

So What?

- Lots of features are high dimensional
  - Useful to cluster them
- Subspace clustering can be used as dimension reduction
- Cast your problem into clustering problems and solve with it


Question
Comment