

# HIERARCHICAL CLUSTERING

An iterative improvement procedure  
for hierarchical clustering

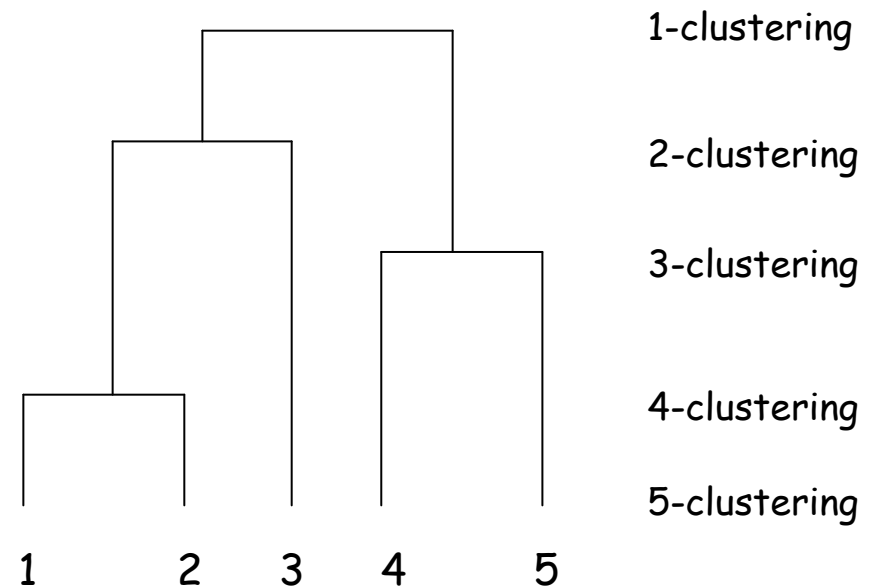
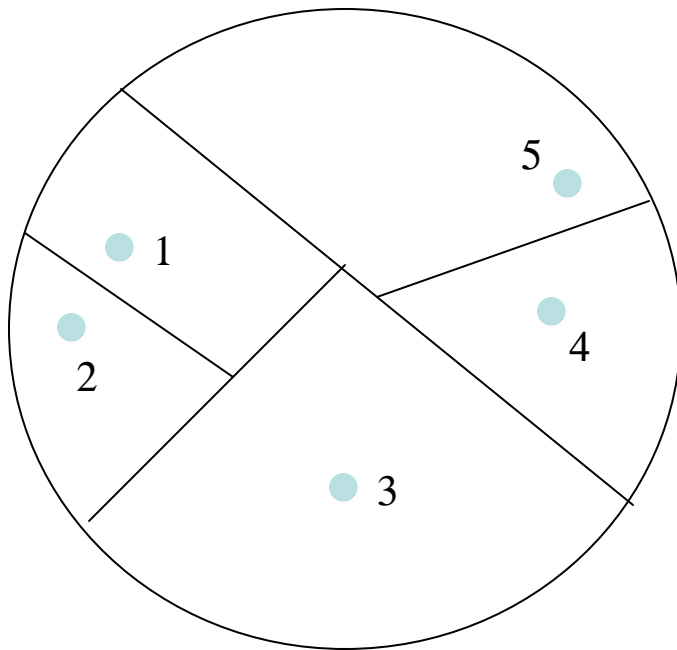
Andrew Rabinovich  
CSE 252C 10/12/04

# Why Hierarchical Clustering and Not Usual Clustering(K-means)?

- Obtain a hierarchy instead of an amorphous collection of groups
- No need to specify number of clusters
- Can view data at many levels of granularity, all at the same time
- Simple heuristics for constructing hierarchical clusterings

# Hierarchical clustering

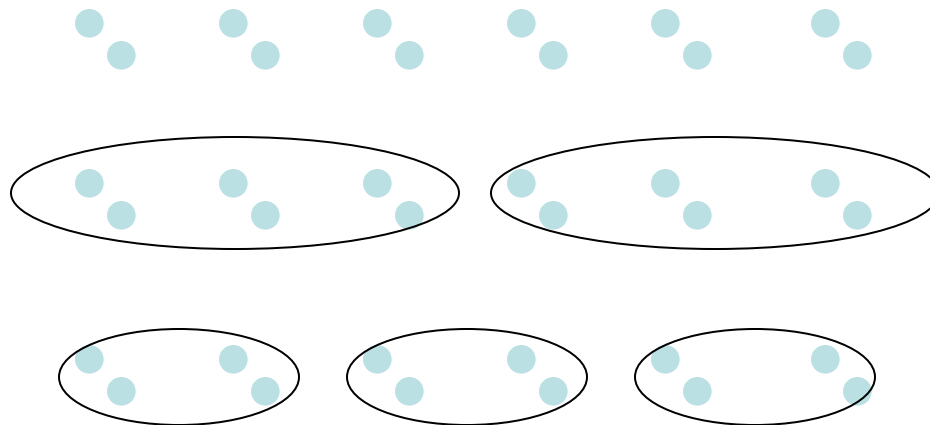
Recursive partitioning of a data set



# Problem

The whole enterprise of hierarchical clustering could use some more justification.

e.g.



# Conventional Hierarchical Techniques

- Agglomerative
- Divisive

# Standard agglomerative heuristics

1. Initially each point is its own cluster.
2. Repeatedly merge the two "closest" clusters.

Need to define distance between clusters...

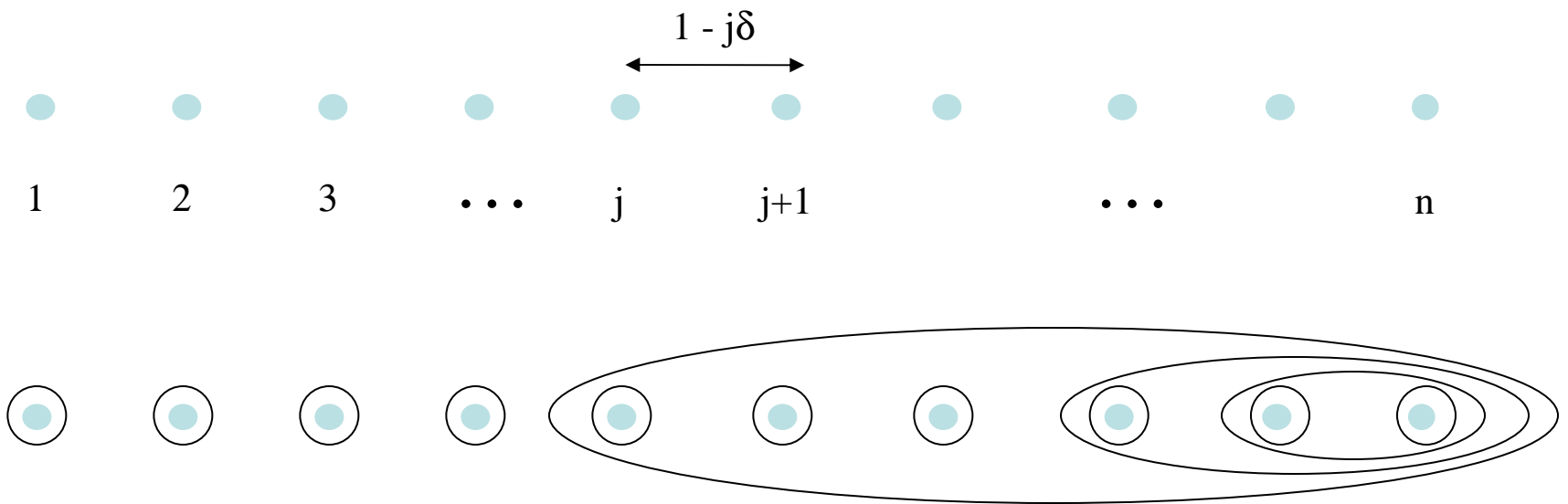
**Single-linkage:** distance between closest pair of points

**Average-linkage:** distance between centroids

**Complete-linkage:** distance between farthest pair

# Single-linkage clustering

Chaining effect.



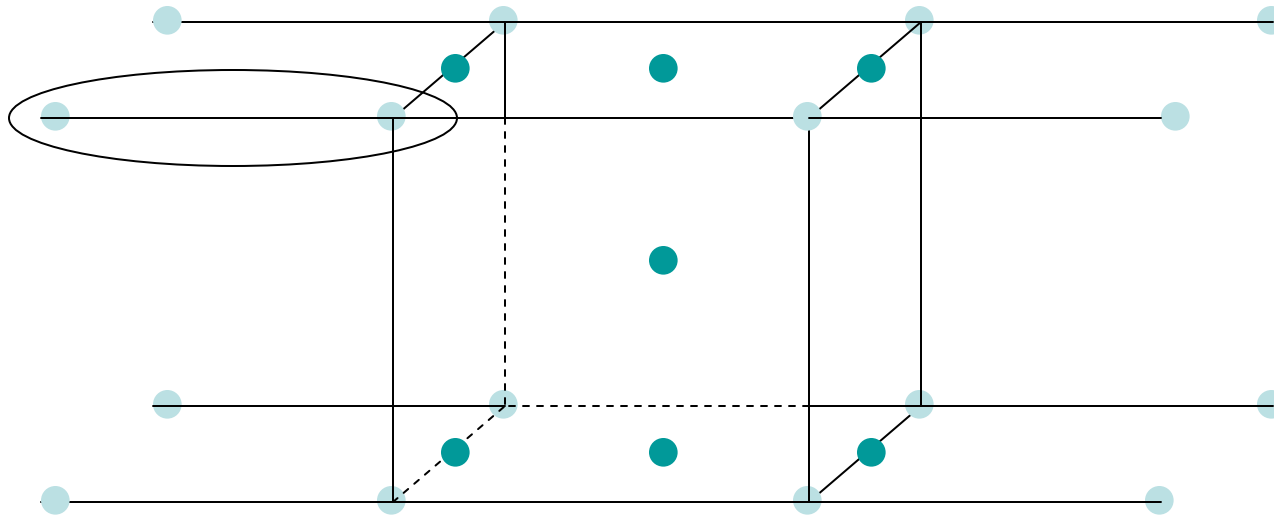
The  $k$ -clustering will have diameter about  $n-k$ , instead of  $n/k$ .

# Complete-linkage clustering

Can similarly construct a bad case...

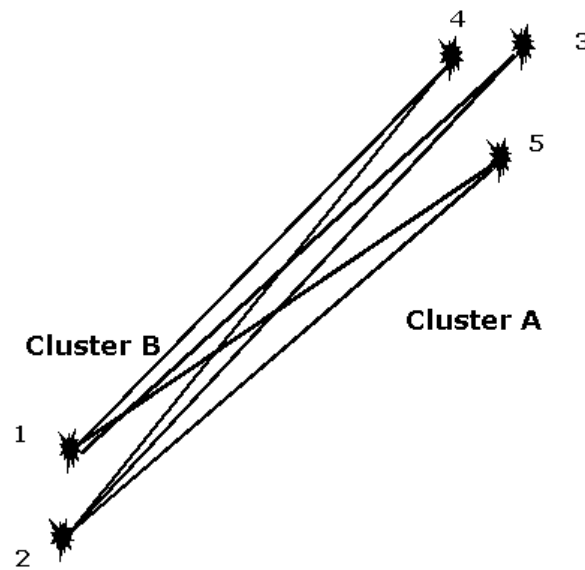
# Average-linkage clustering

Points in  $d$ -dimensional space,  $d = \log_2 k$ , under an  $l_1$  metric.



Final radius should be 1, instead is  $d$ .  
Therefore: **off by a factor of  $\log_2 k$ .**

# Average Linkage



The average linkage method avoids the extremes of either large clusters or tight compact clusters.

# Average Linkage Cost Functions

- Sokal-Michener  $\| \mu(S) - \mu(T) \|^2$
- $\frac{1}{|S| \cdot |T|} \sum_{x \in S, y \in T} \|x - y\|^2$
- Ward's Method:  $\frac{|S| \cdot |T|}{|S| + |T|} \| \mu(S) - \mu(T) \|^2$

# Divisive Methods

- Normalized Cut

A binary partition is performed recursively increasing the similarity within cluster and dissimilarity between clusters

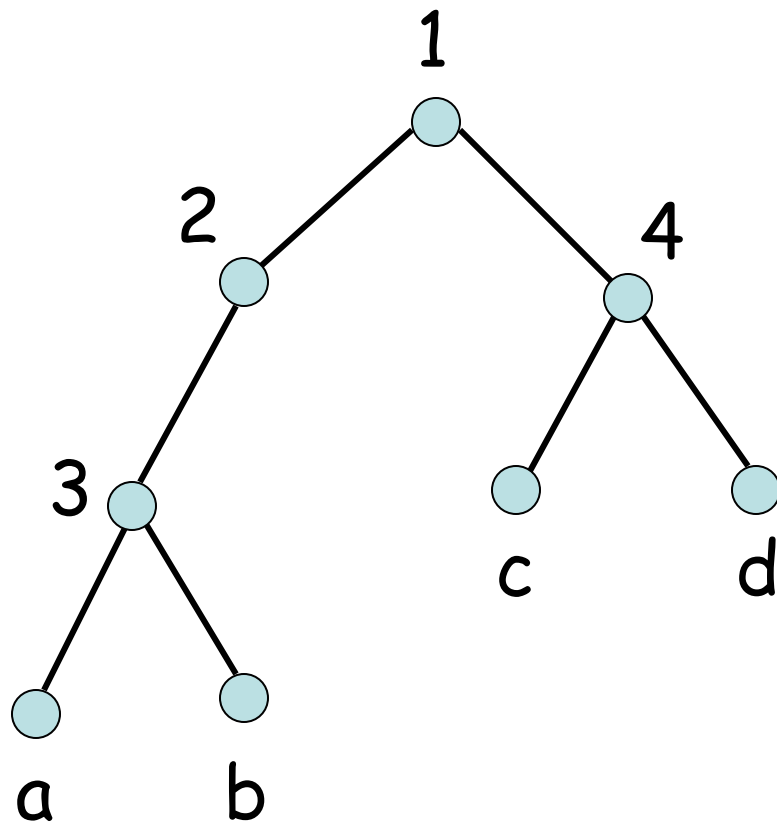
# Evaluation of all agglomerative methods

- Can't trace back
- Complexity at least  $O(n^2)$

# Goal:

- Local improvement algorithm for hierarchical clustering
- Hierarchical analog of k-means
- Key challenge: efficiency

# Representation



Ordered binary tree

Large search space:  
given  $n$  data points,

$$\frac{n((n-1)!)^2}{2^{n-1}}$$

hierarchical  
clusterings

# Cost of hierarchical clustering

- Let  $C_k$  be the set of  $k$  clusters  $\{S_1, S_2, \dots, S_k\}$  defined by a flat clustering. The  $k$ -means cost is:

$$\text{cost}(C_k) = \sum_{j=1}^k \sum_{x \in S_j} \|x - \mu(S_j)\|^2$$

- We define the cost of a hierarchical clustering as the weighted sum of the  $n$  clusterings defined by the hierarchy

$$\text{hcost} = \sum_{i=1}^n w_k \text{cost}(C_k)$$


# Features of the cost function


- Based on  $k$ -means cost function
- Inherently emphasizes smaller  $k$
- Weights can be set to emphasize particular  $k$
- Using properties of Euclidean distance:  $O(n)$  time to calculate

# General Algorithm Idea

Use state space search methods to optimize criterion function

Two local moves:

 Keep the ordering fixed and restructure tree optimally

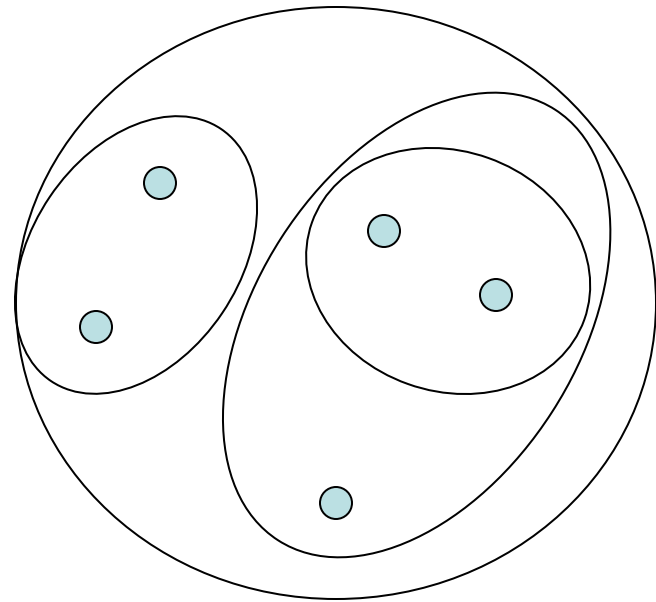
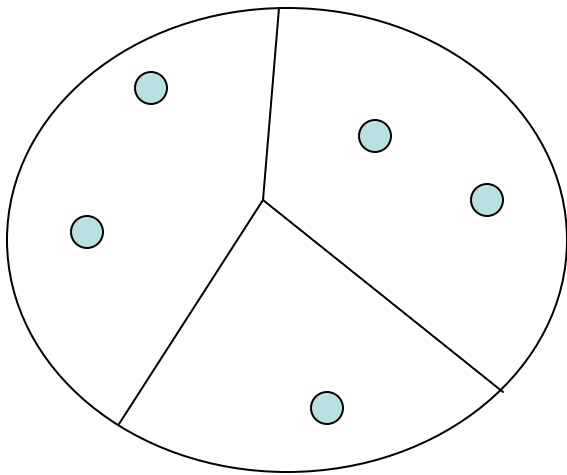
 Keep structure fixed and reorder split order of internal nodes optimally

*NP-Hard?*

*scheduling algorithm*

# Analogy with $k$ -means

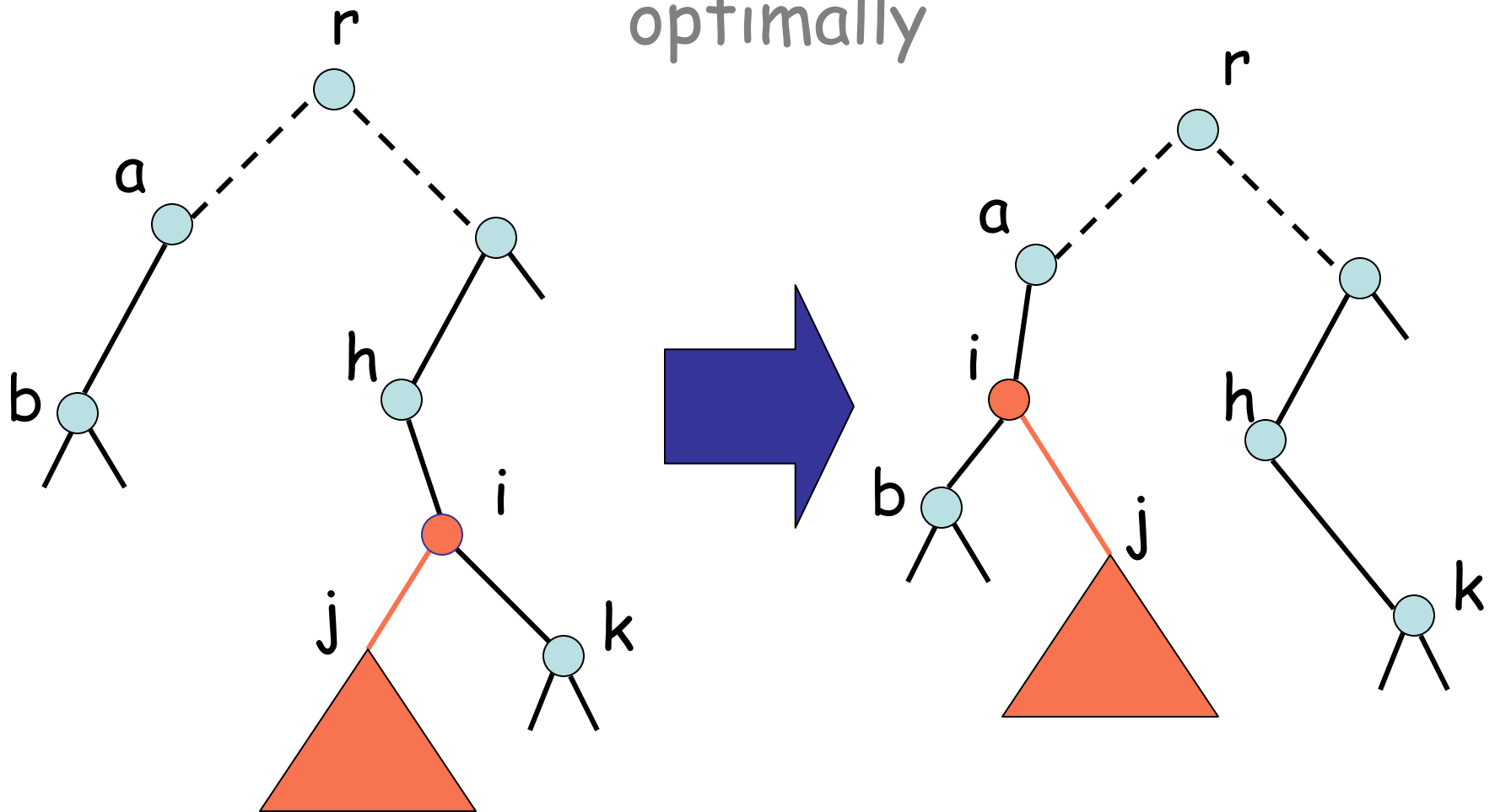
- Keep the means fixed and optimize the partitioning
- Keep the partition fixed and recompute means





# The Graft:

Keep the ordering fixed and restructure tree optimally





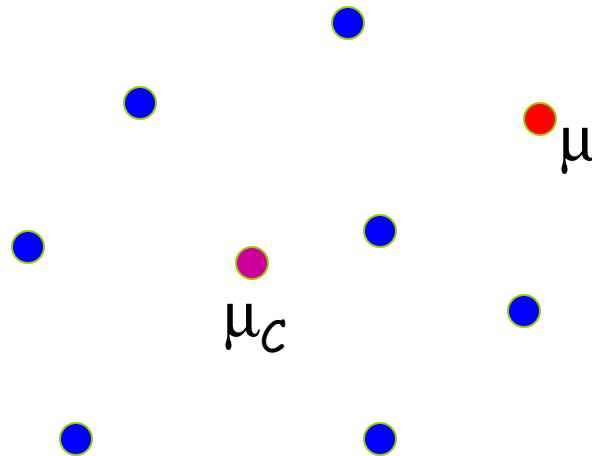
# Making the graft efficient

**Key idea:** Rewrite the hierarchical clustering cost *locally*, as a sum of contributions from individual tree edges.

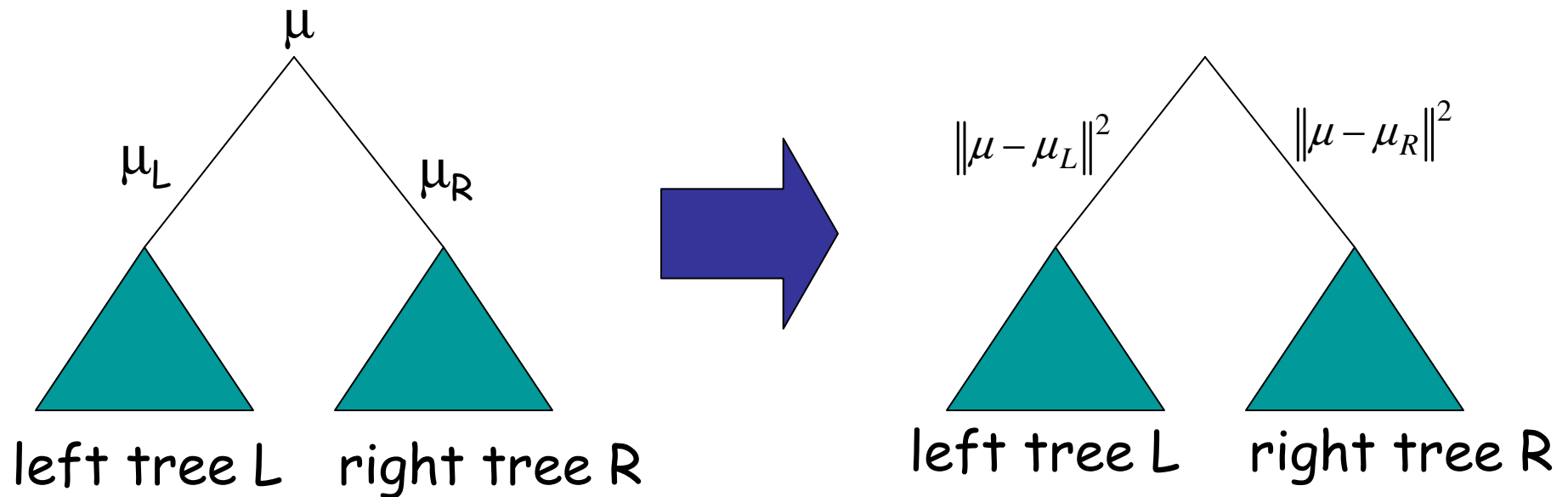
This can be done by exploiting special properties of squared Euclidean distance.

**Lemma:** Suppose cluster  $C$  has mean  $\mu_C$ ;  
then for any other point  $\mu$ ,

$$\sum_{x \in C} \|x - \mu\|^2 = \sum_{x \in C} \|x - \mu_C\|^2 + |C| \cdot \|\mu - \mu_C\|^2$$

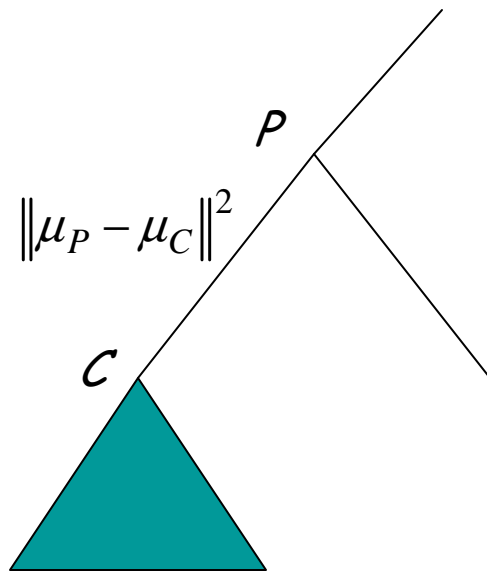


# Calculating $k$ -means cost by counting tree edges



$$\begin{aligned}\text{cost}(L \cup R) &= |L| * \|\mu - \mu_L\|^2 + \sum_{x \in L} \|x - \mu_L\|^2 + |R| * \|\mu - \mu_R\|^2 + \sum_{x \in R} \|x - \mu_R\|^2 \\ &= |L| * \|\mu - \mu_L\|^2 + \text{cost}(L) + |R| * \|\mu - \mu_R\|^2 + \text{cost}(R)\end{aligned}$$

# Counting edges for criterion function:



$$\text{Recall: } \text{hcost} = \sum_{i=1}^n w_k \text{cost}(C_k)$$

This is in fact just a linear combination of lengths of tree edges. The coefficient for the edge between  $P$  and  $C$  is:

$$|T_C| * \sum_{i=1}^{\sigma(p)-1} w_i$$

$O(n)$  computation time



# Optimal location for a sub-tree

Pick any subtree. What's the best place to move it (using a "graft"), while respecting split numbers?

Can be determined in just  $O(n)$  time:

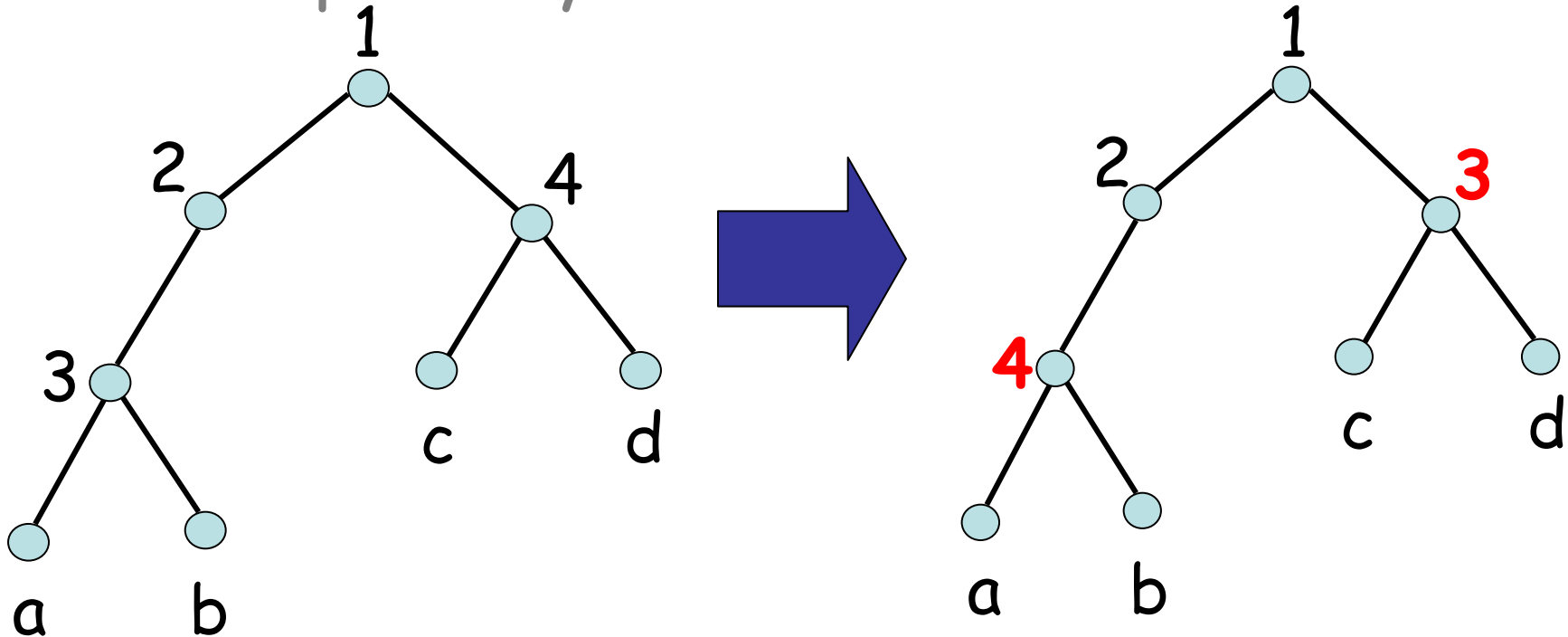
*Upward pass:* compute change in cost due to removing subtree

*Downward pass:* calculate change in cost for each potential location



# Reordering internal nodes

Keep structure fixed and reorder  
split order of internal nodes  
optimally





# Optimal internal node ordering

**Key idea:** Each split in the tree reduces the clustering cost (i.e., finer approximation to the data).

We want to place the best splits early on... this is a *scheduling* problem! (board example)



# Optimal split ordering

- Let  $g(\cdot)$  be the reduction in cost of splitting cluster rooted at  $x$ ,

$$T_x: g(T_x) = \text{cost}(T_x) - (\text{cost}(T_L) + \text{cost}(T_R))$$

- Overall hierarchical clustering cost is a time-weighted sum of these:

$$\text{hcost} = \frac{1}{n} \sum_{x \in V} \sigma(x) g(x)$$

where  $\sigma(x)$  is the time (1 to  $n$ ) at which  $T_x$  is split.



# A scheduling algorithm

Choose order  $\sigma(\cdot)$  of splits.

Greedy choice based on  $g(\cdot)$  doesn't work - a split whose immediate benefit is small might nevertheless enable future good splits.

Horn [1972]: next subtree  $T$  to split is the one which maximizes  $\frac{1}{|T|} \sum_{z \in T} g(z)$

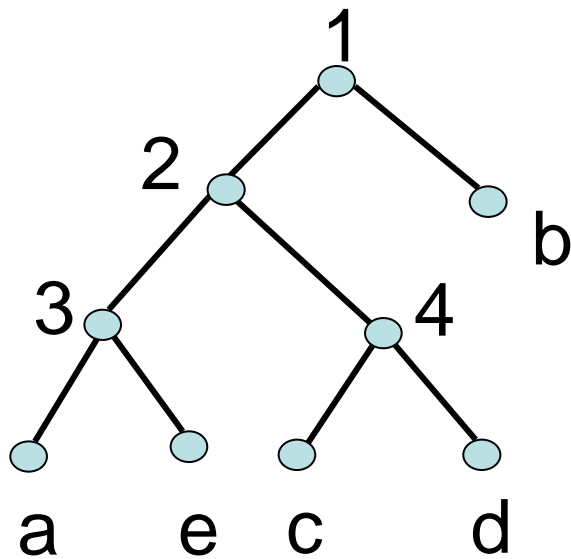
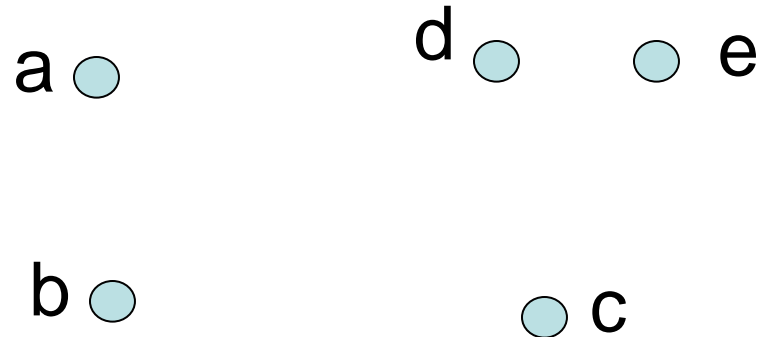
Can be implemented in  $O(n \log n)$  time.

# Final Algorithm

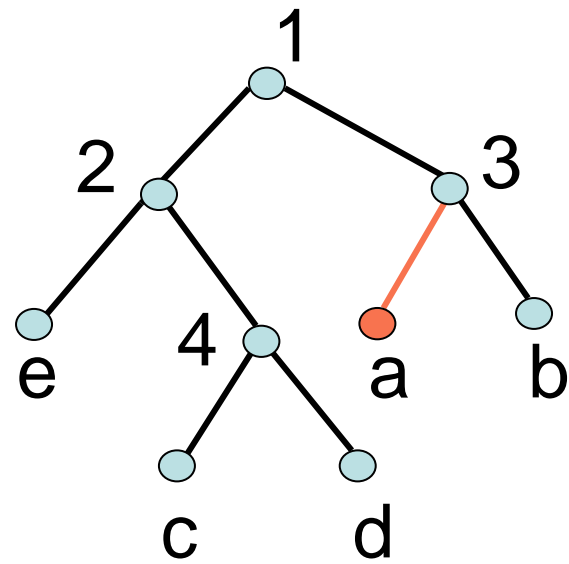
Iterate between two local moves

- Keep structure fixed and reorder split order of internal nodes optimally  $O(n \log n)$
- Optimally relocate  $r$  sub-trees:  
 $O(r n) \rightarrow O(n)$ , since  $r < n$

# Example

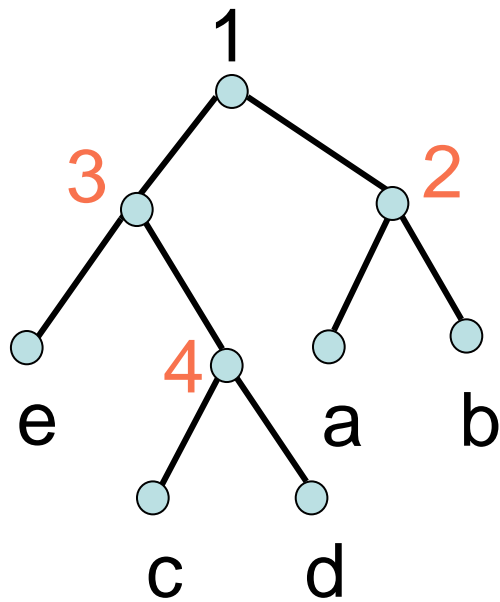


Initial tree: cost 62.25

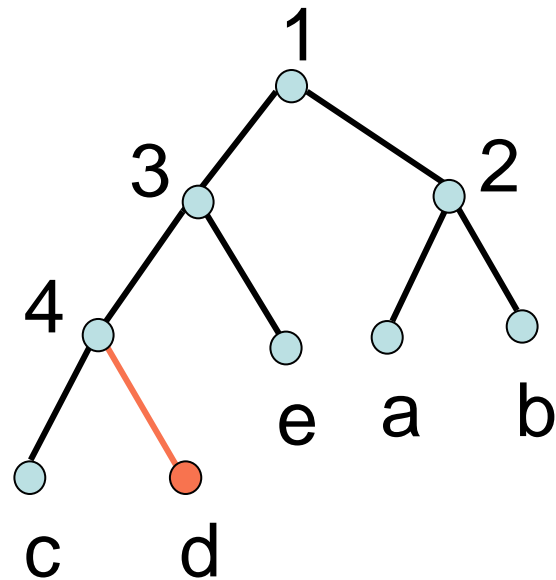


After graft: 27.0

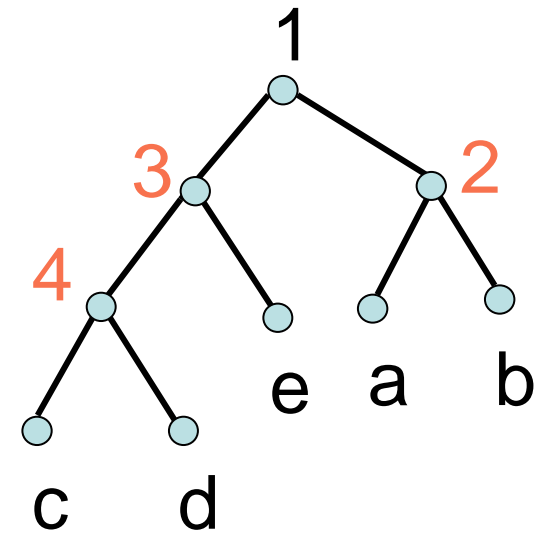
# Example, continued



After reordering:  
cost 25.5



Graft



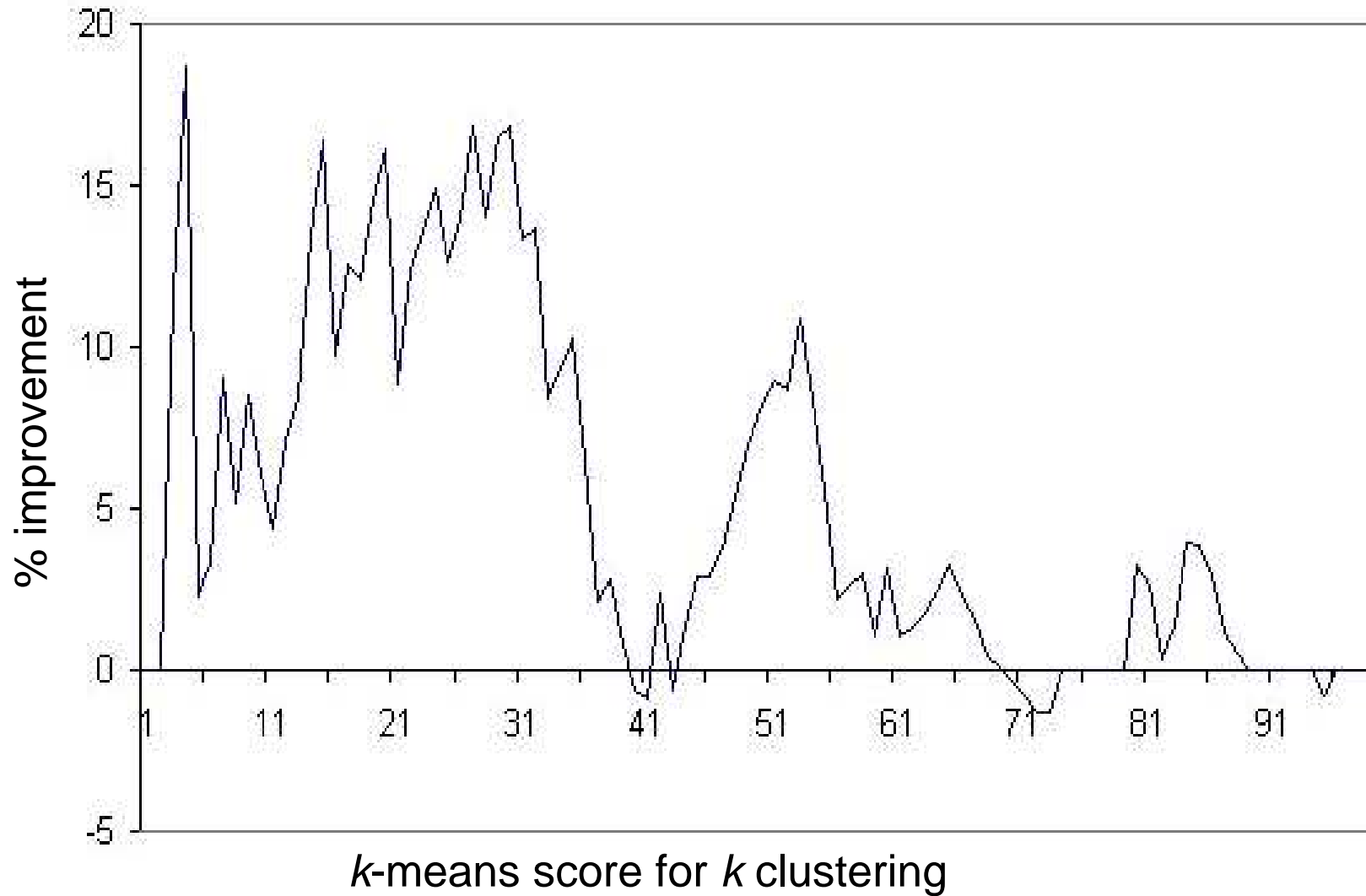
Final tree:  
cost 21.0

# Improvement over average linkage

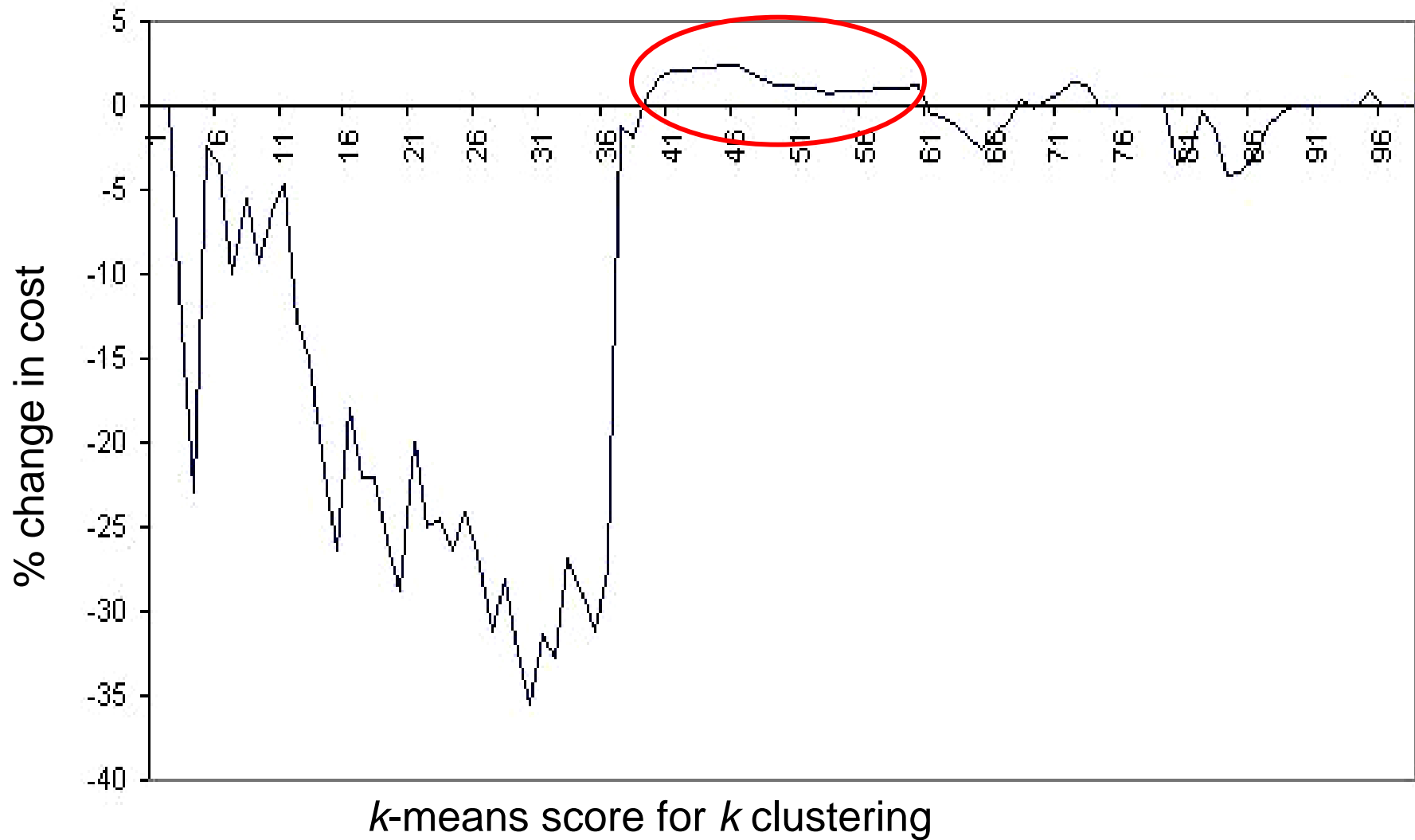
	Iterative algorithm, initialized with average linkage	Iterative algorithm, initialized randomly	Annealed algorithm, initialized with average linkage	Average linkage
20 points	21.59 8 iterations	Best: 20.88 Worst: 24.56	Best: 20.88	21.99
100 points	411.83 233 iterations	Best: 430.35 Worst: 449.38	NA	444.15

- Random and annealed methods find better optima
- Simulated annealing is much more consistent than local search with random initialization
- Number of iterations to converge grows with data size

# Improvement over average linkage

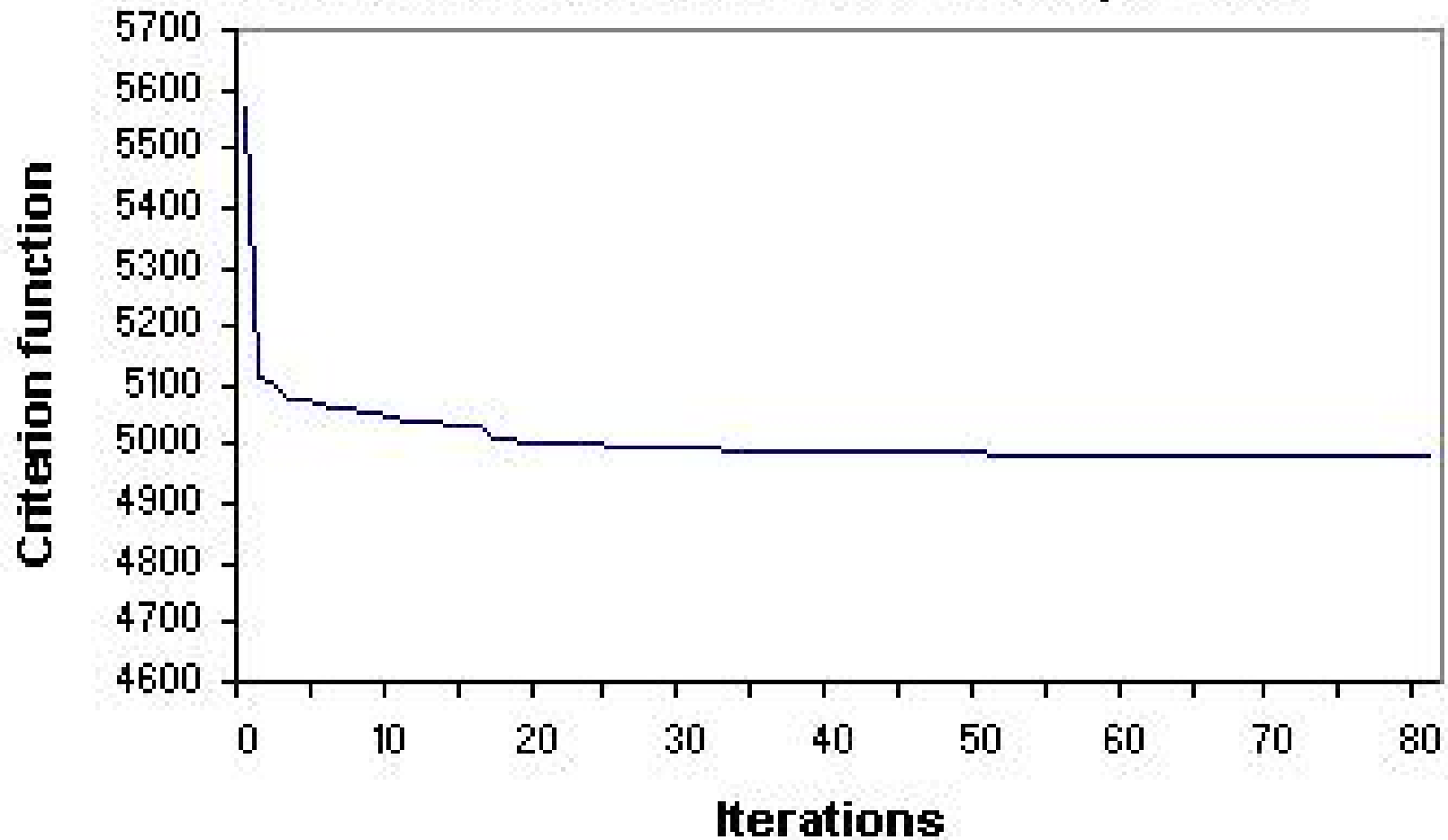


# Weighted (40 - 60 ) vs. uniform



# Convergence

Criterion function for 500 data points



# Applications

- Gene Expression Analysis
- Microarray Data
- Image Segmantation
- Desease Classification

# Summary

There is a basic existence question about hierarchical clustering which needs to be addressed:

must there always exist a hierarchical clustering in which, for each  $k$ , the induced  $k$ -clustering is close to optimal?

It turns out the answer is yes.

(see S. Dasgupta and P.M. Long)

# Summary

In fact, there is a simple, fast algorithm to construct such hierarchical clusterings.

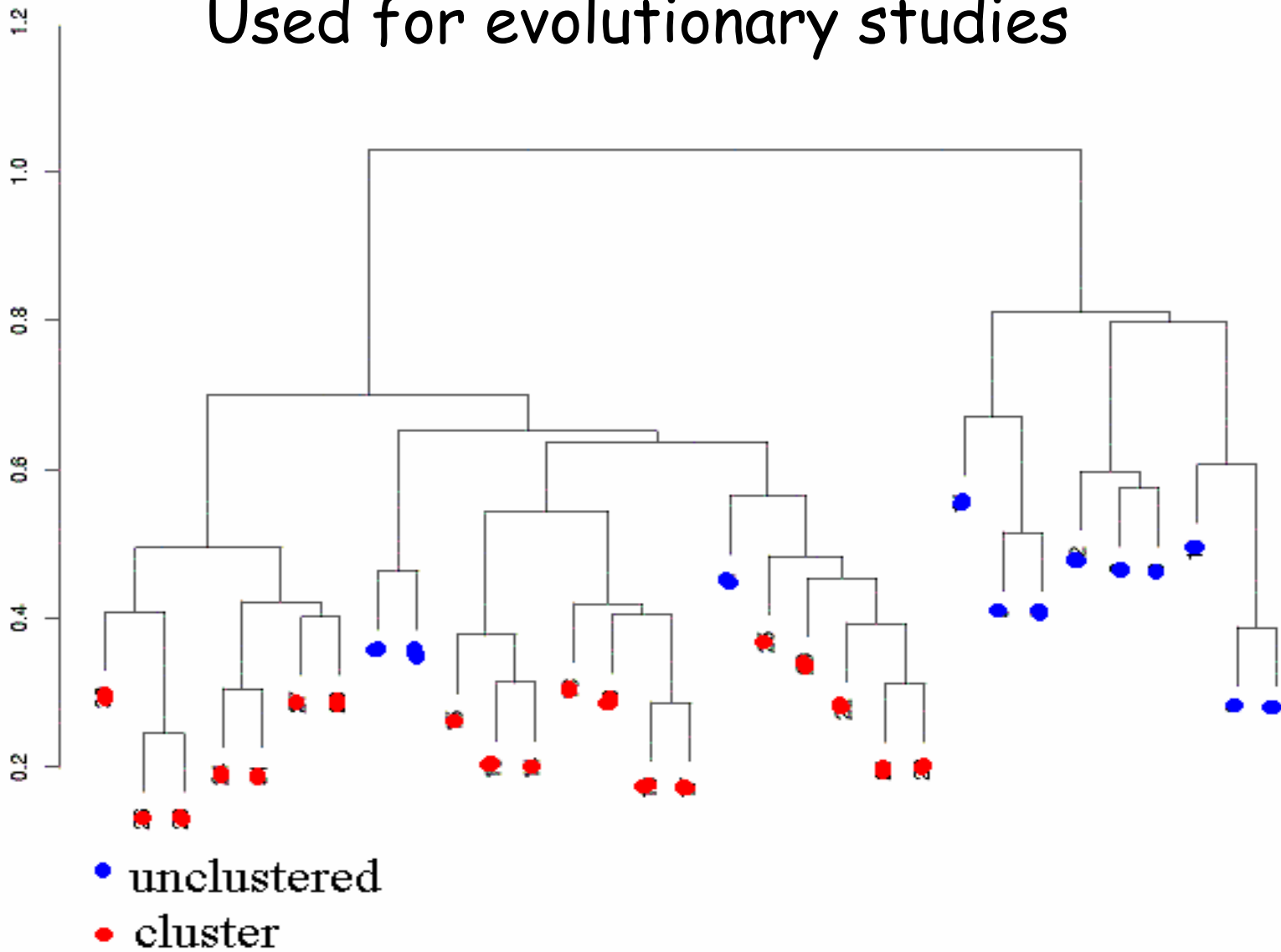
Meanwhile, the standard agglomerative heuristics do not always produce close-to-optimal clusterings.

# Discussion

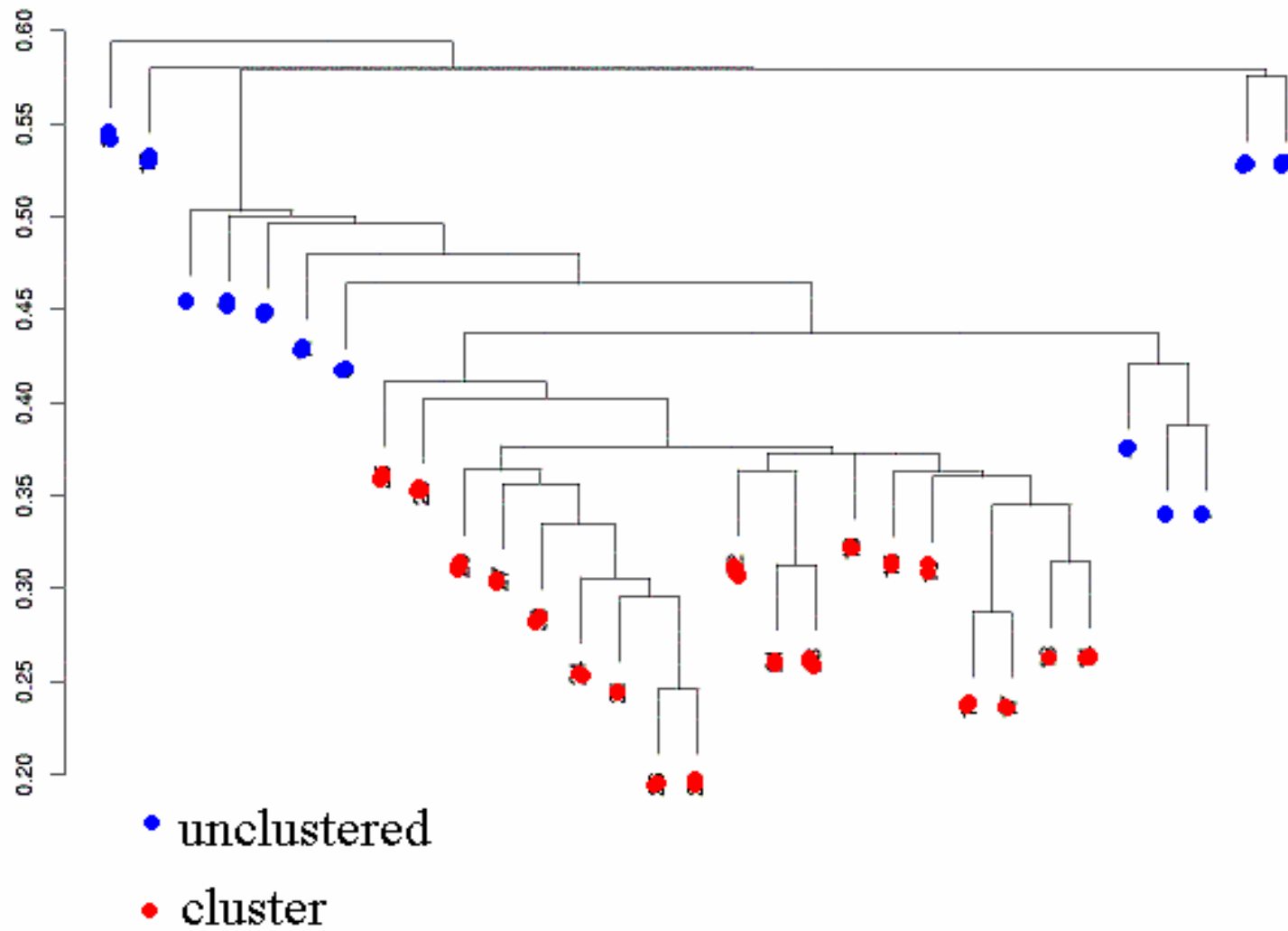
- Other cost functions for clustering.
- Local improvement procedures for hierarchical clustering?

# Single linkage (Nearest Neighbor)

Used for evolutionary studies



# Complete linkage (FN)



# Ward's method (information loss)

