BIRCH:
An Efficient Data Clustering Method For Very Large Databases

Tian Zhang, Raghu Ramakrishnan, Miron Livny

CPSC 504
Presenter: Kendric Wang
Discussion Leader: Sophia (Xueyao) Liang

HelenJr. "Birches". Online Image. Flickr. 07 Nov 2009.

Outline

- What is Data Clustering?
- Advantages of BIRCH Algorithm
- Clustering Feature (CF) and CF Tree
- BIRCH Clustering Algorithm
- Applications of BIRCH
- Conclusion

What is Data Clustering?

- Given a large set of multi-dimensional data points
  - data space is usually not uniformly occupied
- Can group closely related points into a "cluster"
  - points are similar according to some distance-based measurement f'n
  - choose desired number of clusters, K
- discover distribution patterns in the dataset
- help visualize data and guide data analysis

What is Data Clustering?

- Popular data mining problem in
  - Machine Learning -- probability-based
  - Statistics -- distance-based
  - Database -- limited memory

- Define problem as:
  - Partitioning dataset to minimize "size" of cluster
  - Data set size may be larger than memory
  - Minimize I/O costs

Advantages of BIRCH vs. Other Clustering Algorithms

- BIRCH is “local”
  - clusters a point without having to check against all other data points or clusters
- Can remove outliers (“noise”)
- Produce good clusters with a single scan of dataset
- Linearly scalable
  - minimizes running time
  - adjusts quality of result with regard to available memory

Clustering Feature (CF)

- Compact - no need to store individual pts belonging to a cluster
- Three parts: $\mathbf{CF_i = (N_i, LS_i, SS_i)}$, i = 1,2,...,M (no. of clusters)

  N → number of data pts in cluster

  LS → linear sum of N data pts

  SS → square sum of N data pts
- Sufficient to compute distance between two clusters
- When merging two clusters, add the CFs

CF Tree

Branching Factor

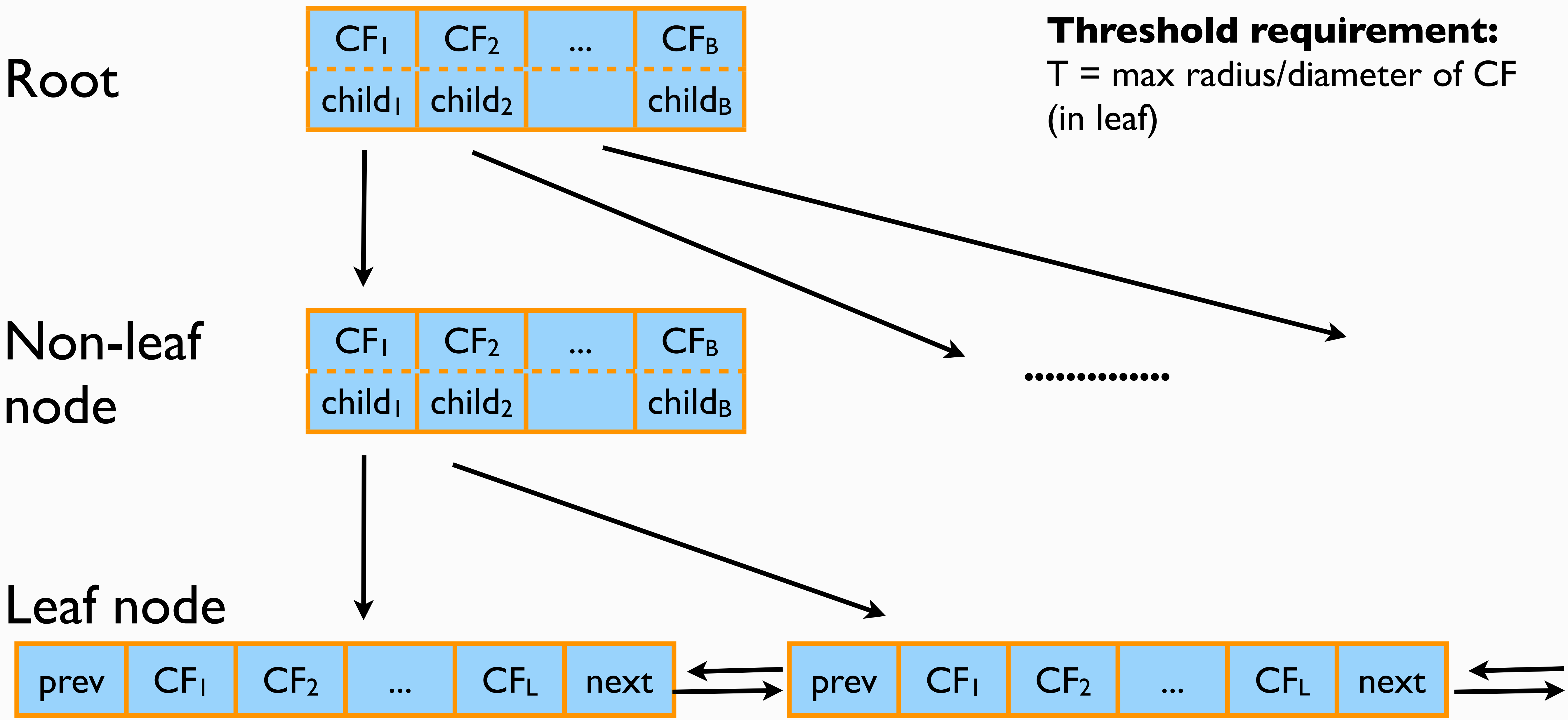
B = max no. of CF in non-leaf node

L = max no. of CF in leaf node

Threshold requirement:

T = max radius/diameter of CF (in leaf)

Entry: [CF, child]



CF Tree

- Tree size is a function of T
  - larger T, more points in each cluster, smaller tree
- good choice reduces number of rebuilds
  - if T too low, can be increased dynamically
  - if T too high, less detailed CF tree
- heuristic approach used to estimate next threshold value
- CF tree built dynamically as data is scanned and inserted

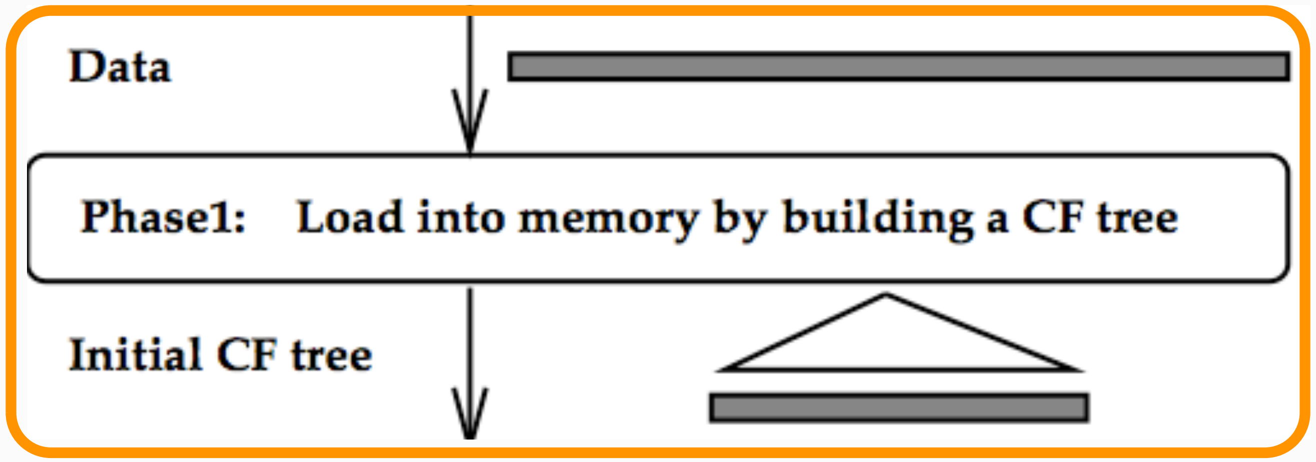
CF Tree Insertion

- Identify the appropriate leaf:
  - Start with CF list at root node, find the closest cluster (by using CF values)
  - Look at all the children of the cluster, find the closest
  - And so on, until you reach a leaf node
- Modify the leaf:
  - Find closest leaf entry and test whether it can absorb new entry without violating threshold condition
  - If not, add new entry to leaf
  - Leaves have a max size; may need to be split
- Modifying the path:
  - Once the point has been added, must update the CF of all ancestors

BIRCH Clustering Algorithm

- Phase 1: Load data into memory by building a CF tree
- Phase 2 (optional): Condense into desirable range by building smaller CF trees
- Phase 3: Global Clustering
- Phase 4 (optional): Cluster Refining

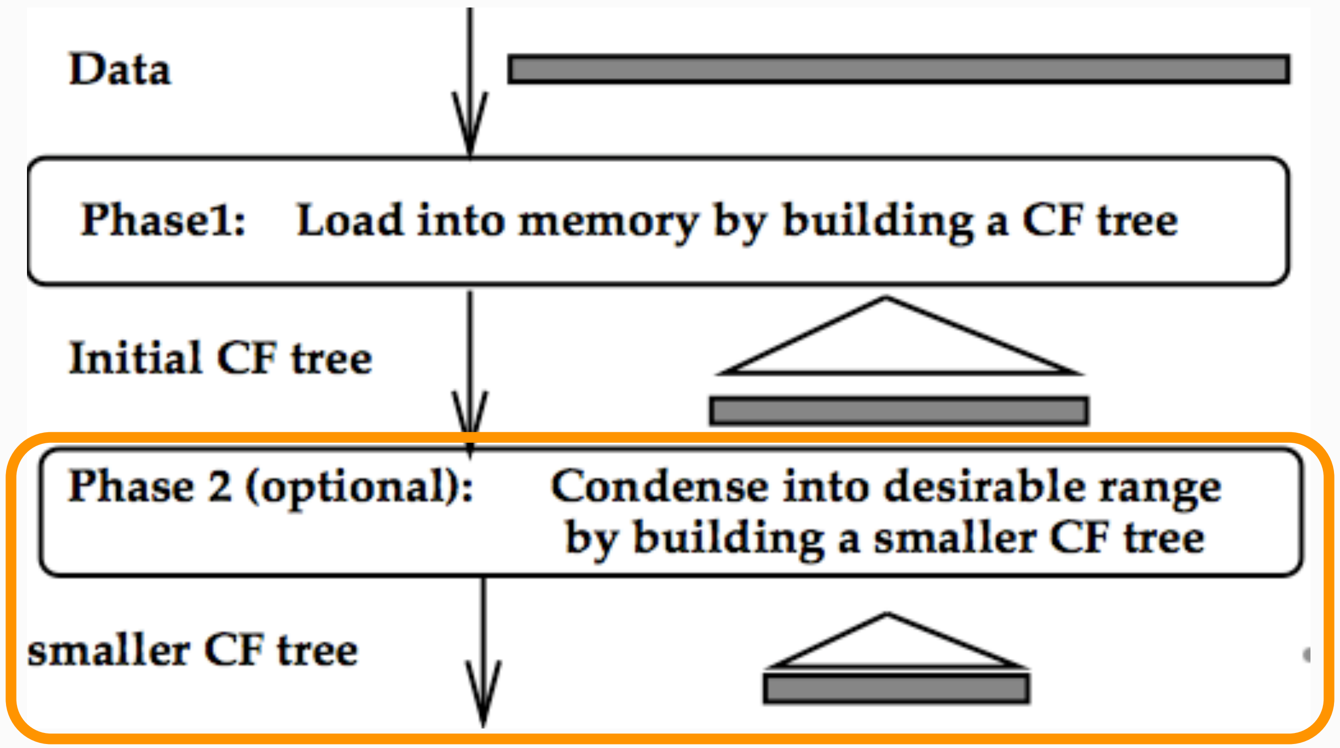
Phase 1 BIRCH



- Start with initial threshold T and insert points into tree
- If we run out of memory, increase T and rebuild
  - Re-insert leaf entries from old tree into new tree
  - remove outliers
- Methods for initializing and adjusting T are adhoc
- After phase 1:
  - data "reduced" to fit in memory
  - subsequent processing occurs entirely in memory (no I/O)

Phase 2

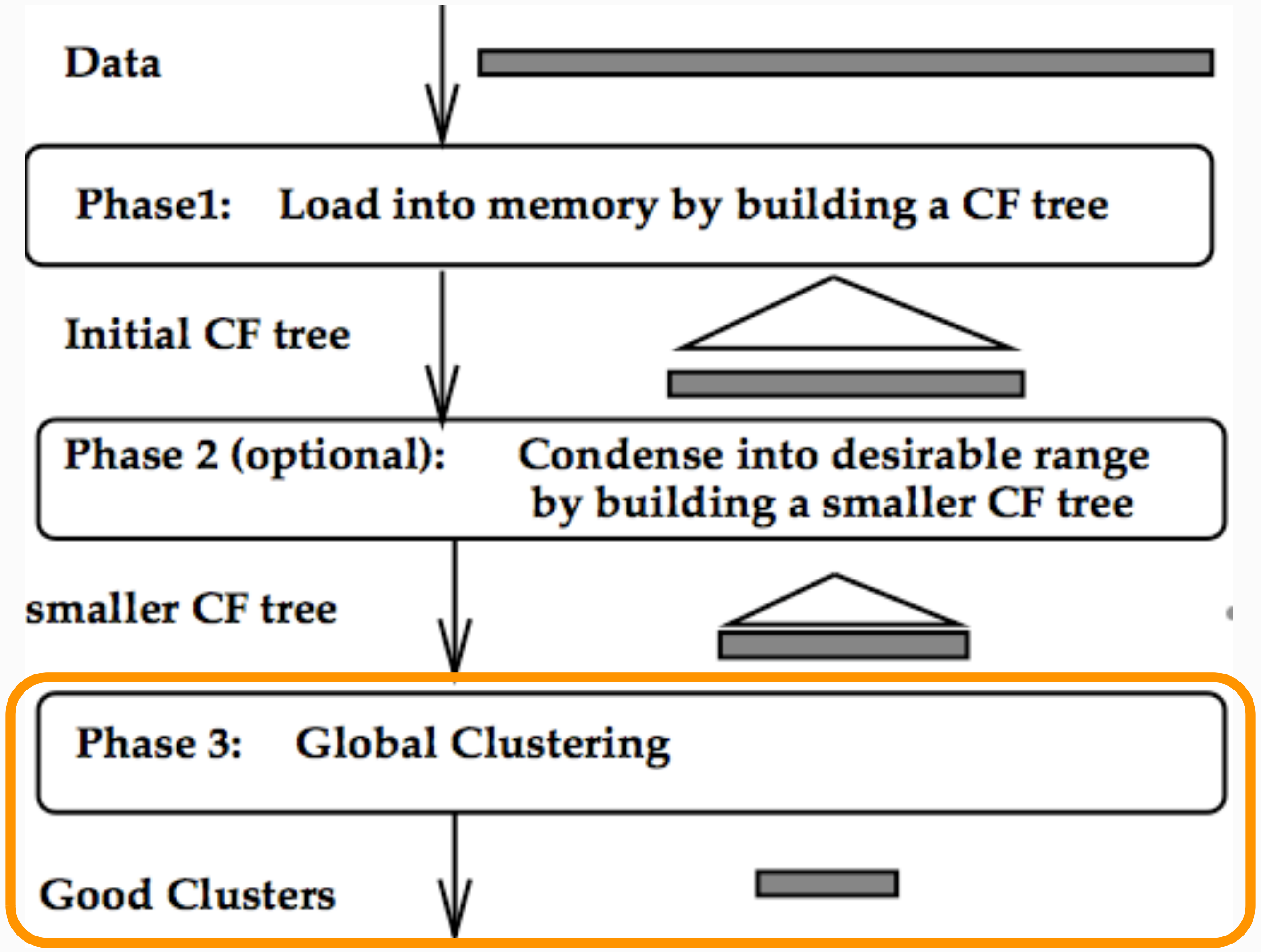
BIRCH



- Optional
- No. of clusters produced in Phase 1 may be not be suitable for algorithms used in Phase 3
- Shrink tree as necessary
  - remove more outliers
  - crowded subclusters are merged

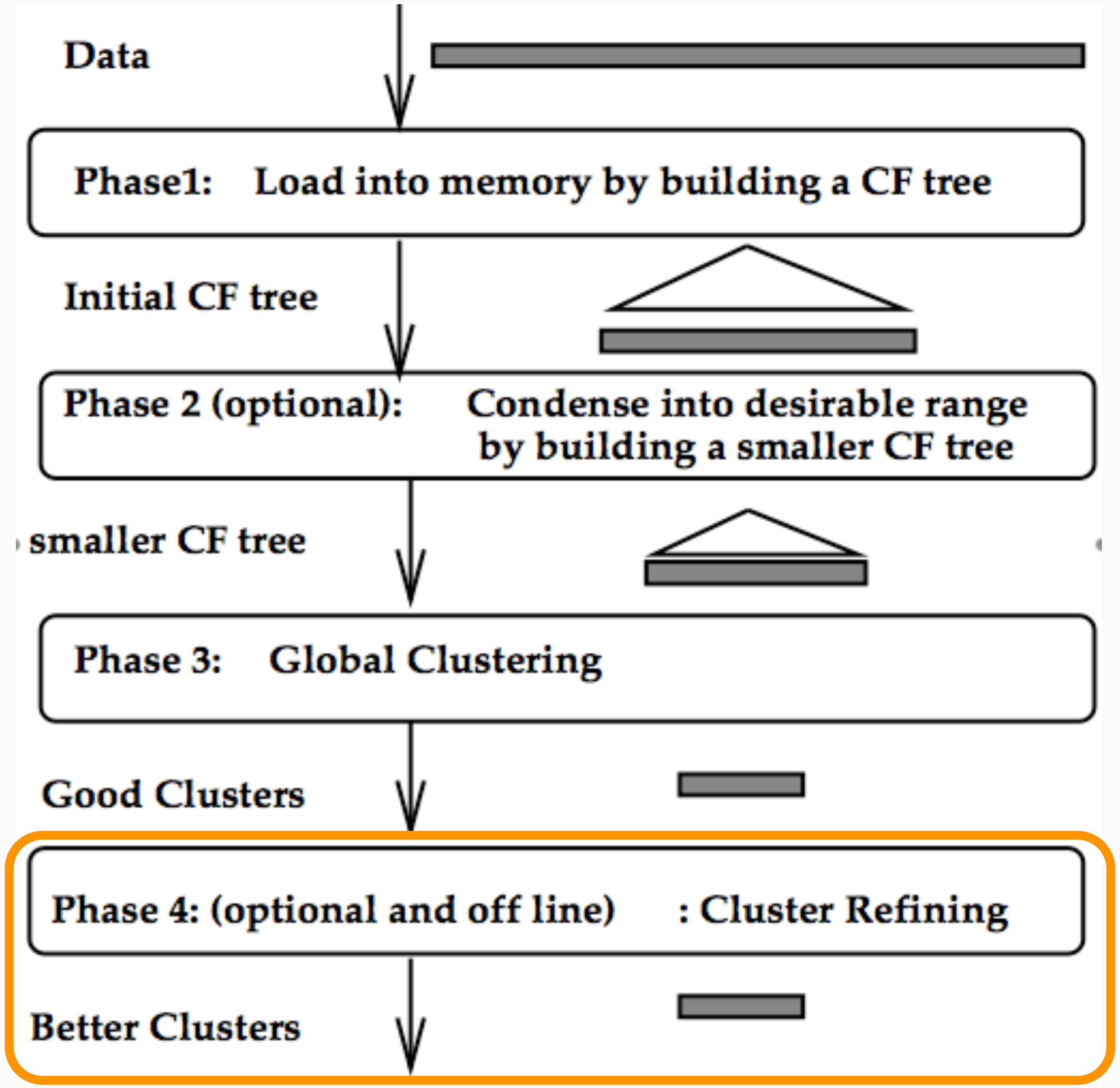
Phase 3

BIRCH



- Problems after Phase 1:
  - input order affect results
  - splitting triggered
- Use leaf nodes of CF tree as input to a standard ("global") clustering algorithm
  - KMeans, HC
- Phase 1 has reduced the size of the input dataset enough so that the standard algorithm can work entirely in memory

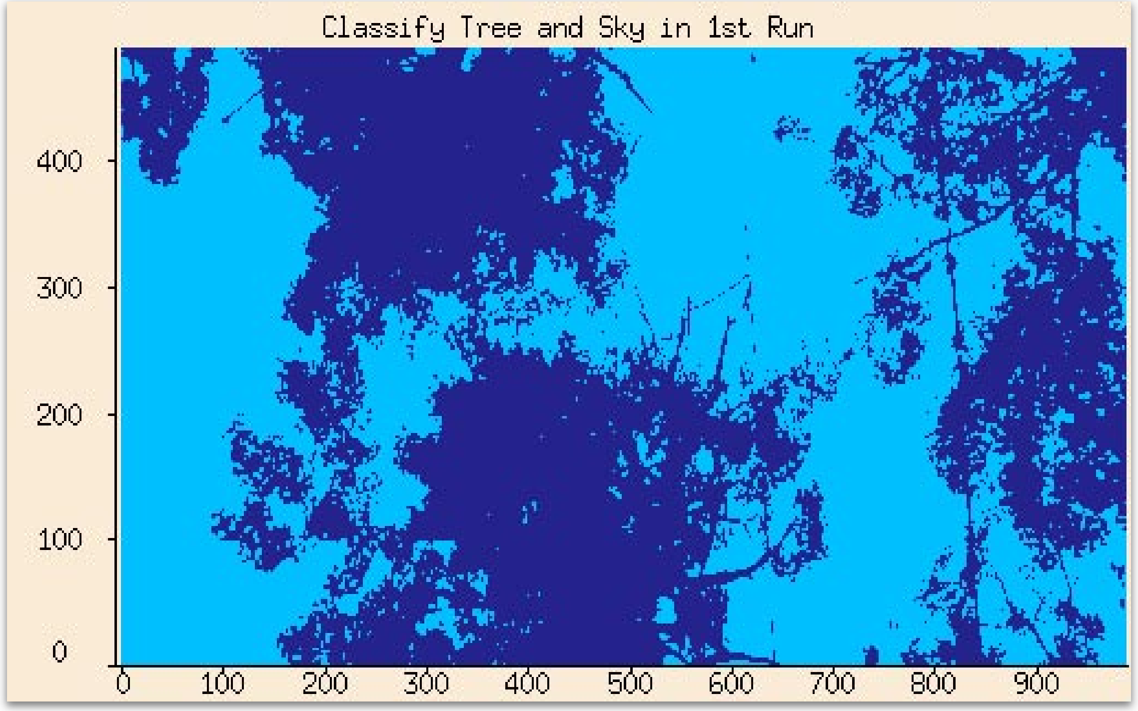
Phase 4 BIRCH

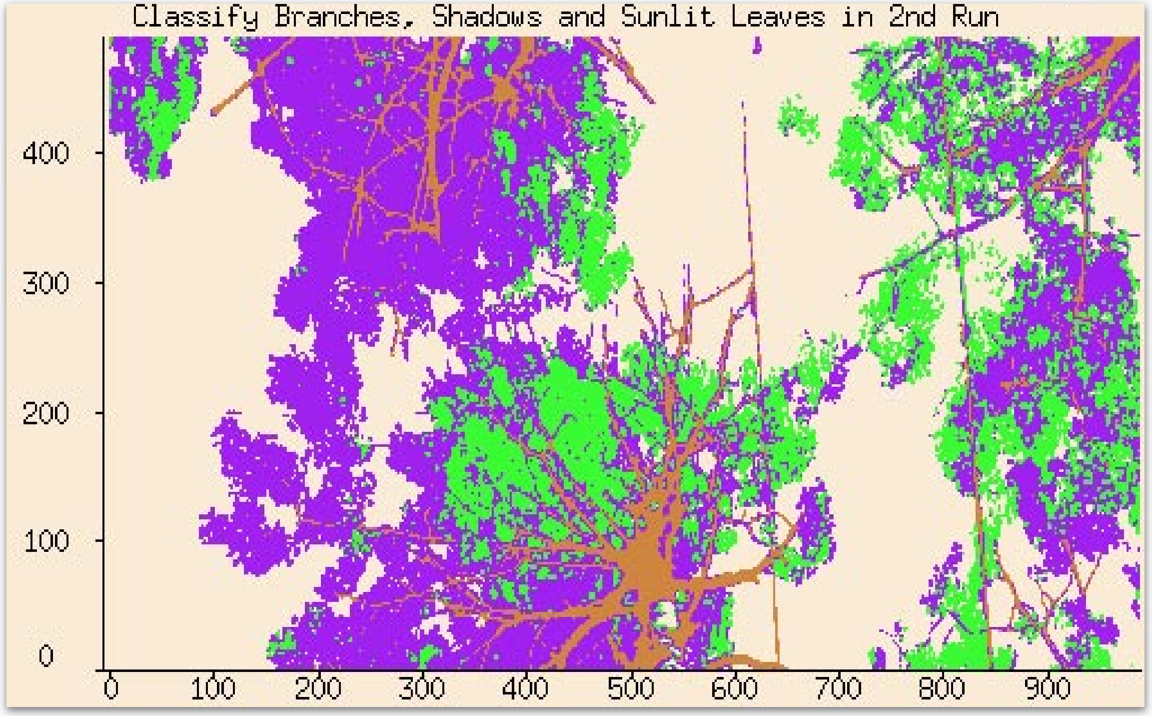


- Optional
- Scan through data again and assign each data point to a cluster
  - choose cluster whose centroid is closest
- This redistributes data points amongst clusters in more accurate fashion than original CF cluster
- Can be repeated for improved refinement of clusters

Apps of Data Clustering

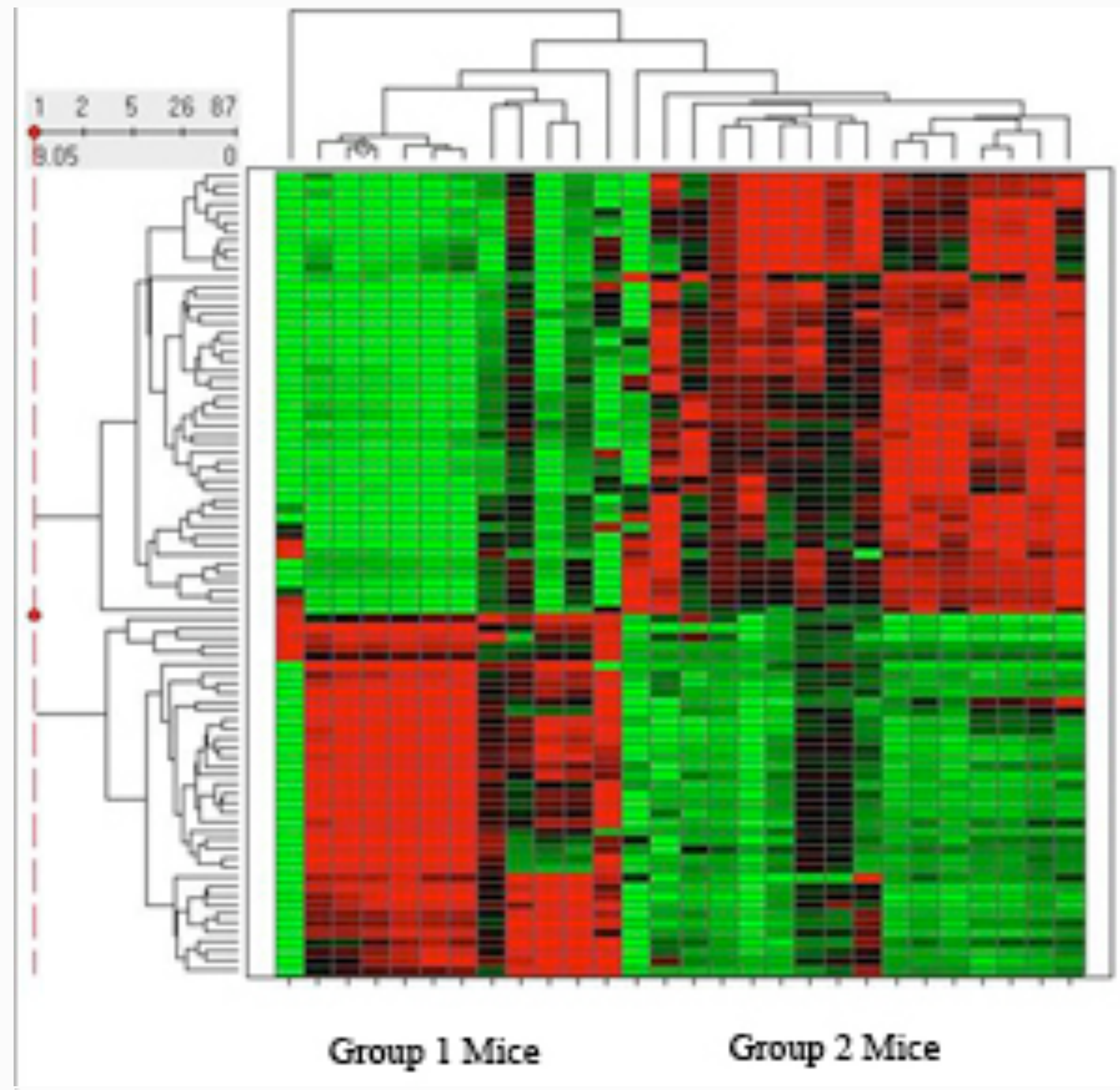
- Helps identify natural groupings that exist within a dataset
- Image processing
  - separate similar properties in an image





Apps of Data Clustering

- Bioinformatics
  - identifying genes that are regulated by common mechanisms
- Market analysis
  - distinguish groups of consumers with similar tastes



Conclusion

- BIRCH performs better than other existing algorithms on large datasets
  - reduces I/O
  - accounts for memory constraint
- Produces good clustering from only one scan of entire dataset: O(n)
- Handles outliers