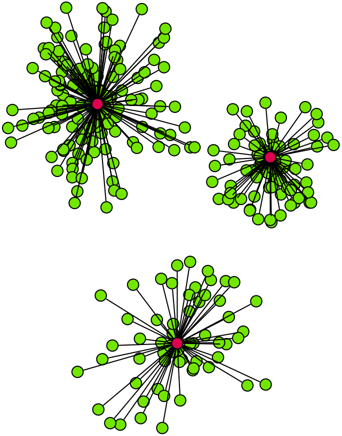
BICO: BIRCH meets Coresets for *k*-means

Hendrik Fichtenberger, Marc Gillé, Melanie Schmidt, Chris Schwiegelshohn, Christian Sohler

Efficient Algorithms and Complexity Theory, TU Dortmund, Germany

04.09.2013

The k-means Problem



- Given a point set $P \subseteq \mathbb{R}^d$,
- compute a set $C \subseteq \mathbb{R}^d$ with $|C| = k$ centers
- which minimizes

$$\text{cost}(P, C) = \sum_{p \in P} \min_{c \in C} ||c - p||^2,$$

the sum of the squared distances.

- Optimal 1-means center: centroid

$$\mu(P) = \frac{1}{|P|} \sum_{p \in P} p$$

Related work

Popular k-means algorithms

- *Lloyd (1982)*: Lloyd's algorithm
- *Arthur, Vassilvitskii (2007)*: k-means++

Streaming algorithms for Big Data (one-pass, limited memory)

- *MacQueen (1967)*: MacQueen's k-means algorithm
- *Zhang, Ramakrishnan, Livny (1997)*: BIRCH
- *O'Callaghan, Meyerson, Motwani, Mishra, Guha (2002)*: StreamLS
- *Ackermann, Lammersen, Märtens, Raupach, Sohler, Swierkot (2010)*: StreamKM++

Our contribution

Streaming algorithm for k-means which

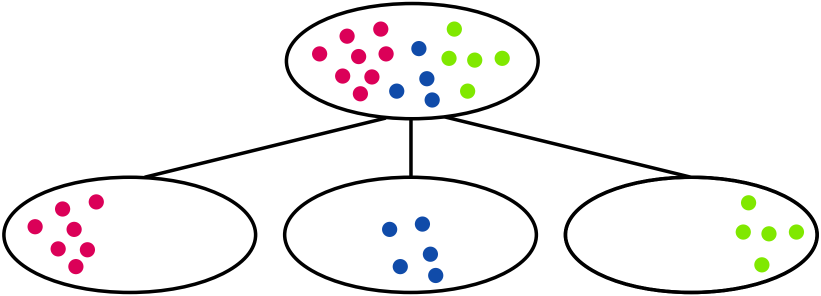
- is fast
- computes high quality solutions
- is easy to implement

Idea

1. Have a look at BIRCH
   - Very fast
   - Solution quality varies
2. Analyze its shortcomings
   - Construction yields no error bound
3. Improve it by drawing on theoretical observations
   - Bound error by utilizing coresets

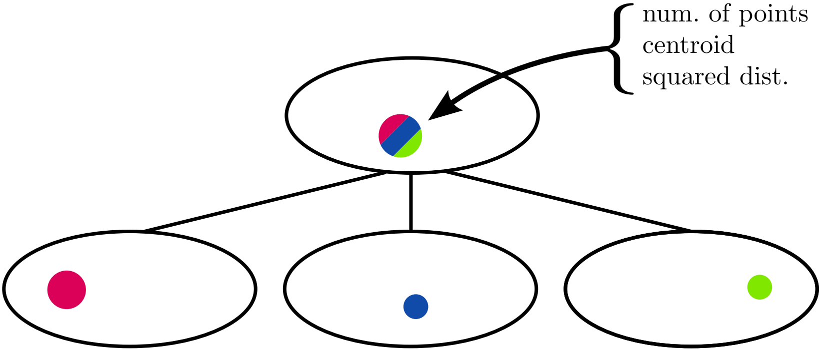
A bit about BIRCH

- Stores points in a tree
- Tree is updated point by point
- Each node represents a subset of the input point set



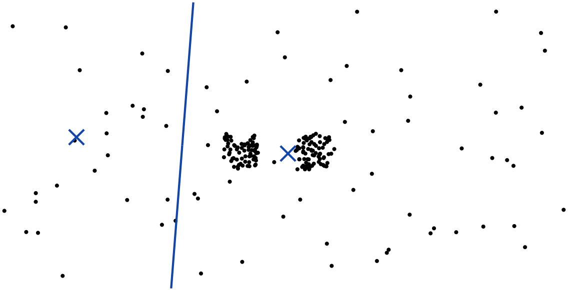
A bit about BIRCH

- Stores points in a tree
- Tree is updated point by point
- Each node represents a subset of the input point set
- Subset is summarized by number of points, the centroid of the set and the sum of the squared distances to the centroid



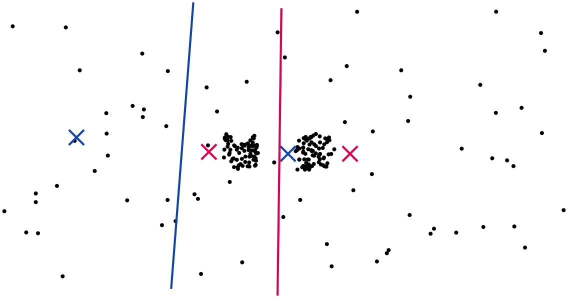
BIRCH: Insertion of a point

BIRCH: Insertion of a point



Problem BIRCH bases insertion on normalized cost and cannot distinguish between the two point clouds

BIRCH: Insertion of a point



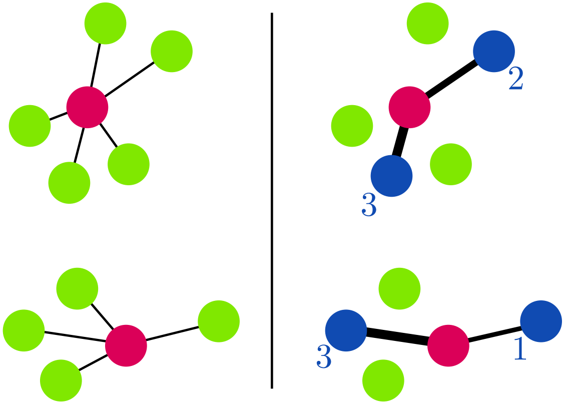
Problem BIRCH bases insertion on normalized cost and cannot distinguish between the two point clouds

Solution New condition for insertion based on coreset theory

Coresets

Given a set of points ■, a weighted subset ■ ⊂ ■ is a (k, ϵ)-coreset iff for all sets ■ of k centers it holds that

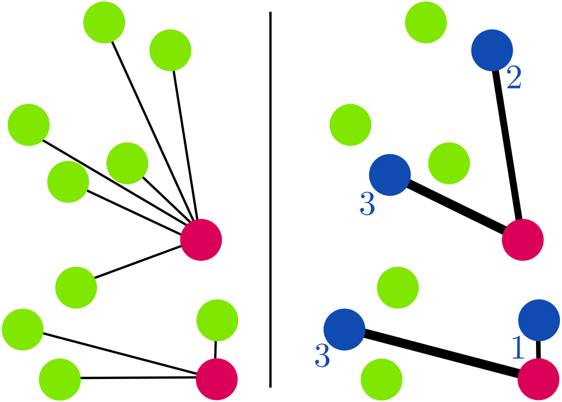
$$|\text{cost}(■, ■) - \text{cost}_{\text{weighted}}(■, ■)| \leq \epsilon \, \text{cost}(■, ■).$$



Coresets

Given a set of points ■, a weighted subset ■ ⊂ ■ is a (k, ϵ)-coreset iff for all sets ■ of k centers it holds that

$$|\text{cost}(■, ■) - \text{cost}_{\text{weighted}}(■, ■)| \leq \epsilon \, \text{cost}(■, ■).$$



Quality guarantee

New insertion decision rule yields the following guarantee:

Theorem

The union of all centroids weighted by the number of points in the corresponding node

- is a $(1 + \varepsilon)$-coreset
- has size $\mathcal{O}(k \cdot \log n \cdot \varepsilon^{-(d+2)})$ for constant d.

Quality guarantee

New insertion decision rule yields the following guarantee:

Theorem

The union of all centroids weighted by the number of points in the corresponding node

- is a $(1+\varepsilon)$-coreset
- has size $\mathcal{O}(k \cdot \log n \cdot \varepsilon^{-(d+2)})$ for constant d.

Practical use

- Choose maximum number of nodes m ($=$ coreset size)
  - $m := 200k$ seems to be a good choice

Experimental Setup

Algorithms for comparison

- StreamKM++ and BIRCH (author's implementations)
- MacQueen's k–means algorithm (ESMERALDA)

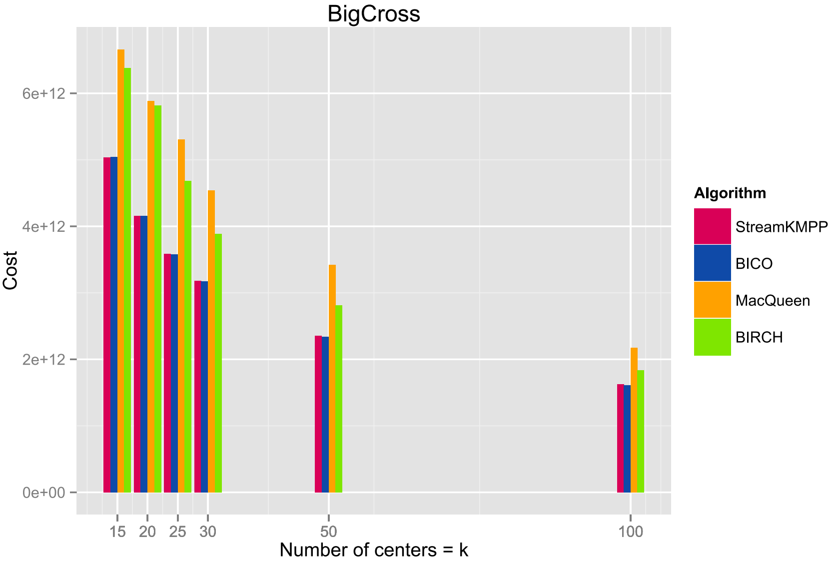
Data sets

	BigCross	CalTech128	Census	CoverType	Tower
n	$1 \cdot 10^7$	$3 \cdot 10^6$	$2 \cdot 10^6$	$6 \cdot 10^5$	$5 \cdot 10^6$
d	57	128	68	55	3
$n \cdot d$	$7 \cdot 10^8$	$4 \cdot 10^8$	$2 \cdot 10^8$	$3 \cdot 10^7$	$1 \cdot 10^7$

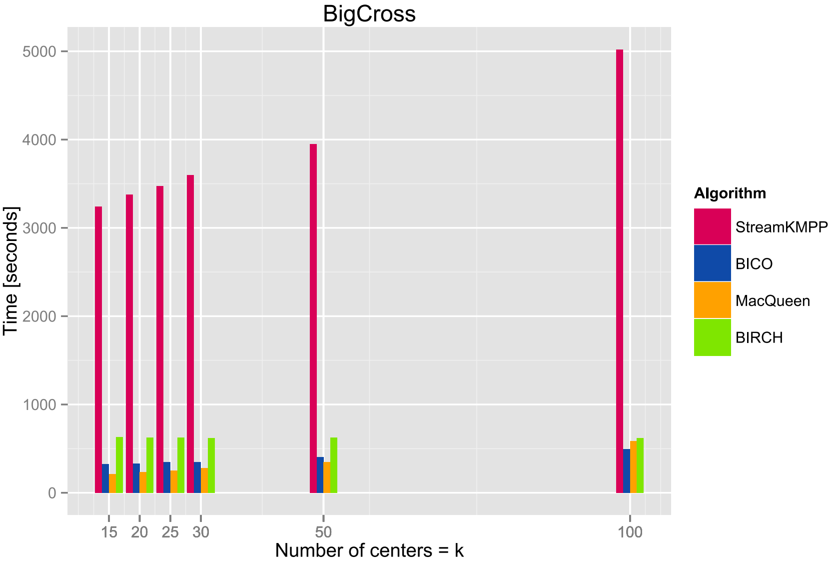
Diagrams

- 100 runs for every test instance
- Values shown in the diagrams are mean values

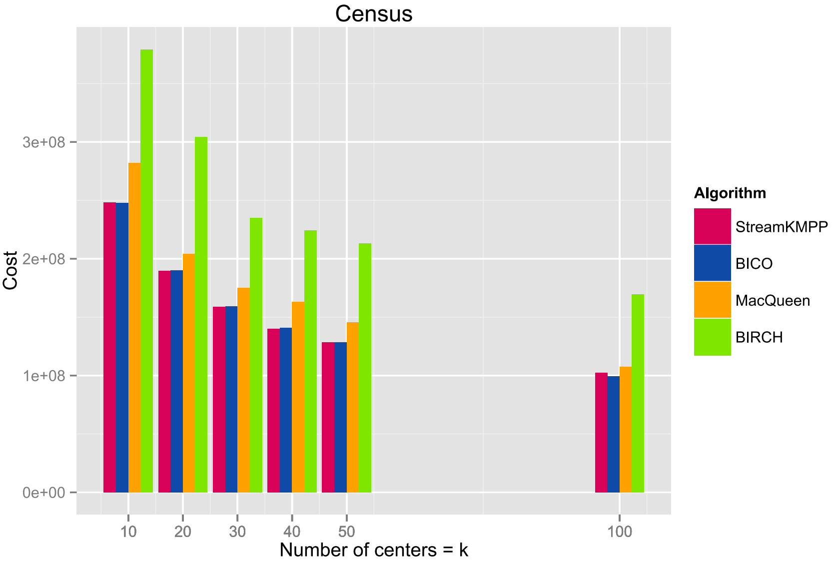
Experimental Results — BigCross: Costs



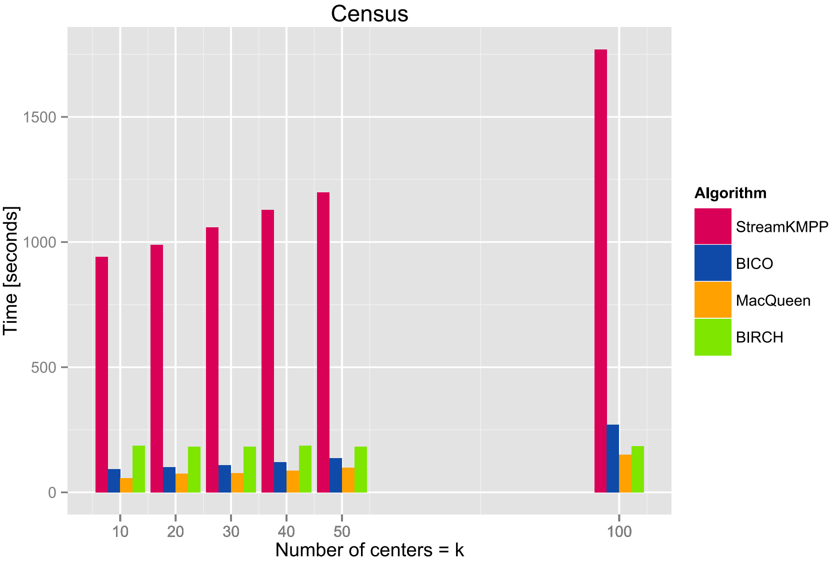
Experimental Results — BigCross: Time



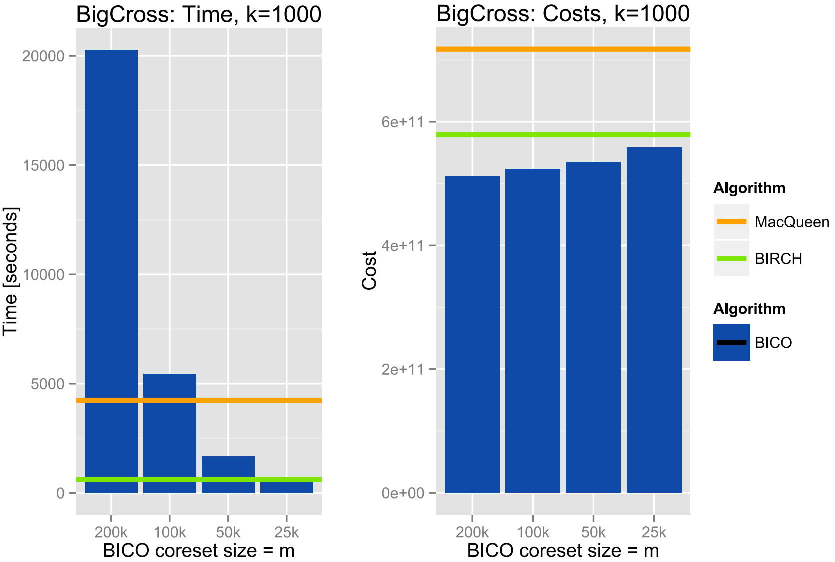
Experimental Results — Census: Costs



Experimental Results — Census: Time



Trade off quality against runtime



Thank you for your attention!