Processing Aggregate Queries over Continuous Data Streams

Alin Dobra
Computer Science Department
Cornell University

April 15, 2003
Relational Database Systems

<table>
<thead>
<tr>
<th>did</th>
<th>dname</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>Legal</td>
</tr>
<tr>
<td>17</td>
<td>Marketing</td>
</tr>
<tr>
<td>3</td>
<td>Development</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>did</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Join operator:

- Subset of cross product that satisfies constraint
- Relations contain no redundant information. Joins put information back together
Relational vs Streaming Data

Relational Database Systems

- Relations reside on disk
  - Access to data under complete control
- Queries refer to current state and posed once

New Applications: network monitoring, sensor networks, telecom call records

- Data arrives in the form of continuous stream
  - Order and rate of arrival not under the control of the system
- Queries are long running
- Most assumptions in relational database systems break

Computation over Data Streams

- Frequency moments [AGM96], distinct values [FM96], norms [I00]
- Stream processing engines: NIAGARA, STREAM, Aurora
• **Challenge:** deal with large amounts of streaming data using *small* memory
  – approximate answers often suffice (e.g., trend/pattern analysis)

• **Other applications:** one pass queries, sensor networks
Streaming Computational Model

- **Computational model:**
  - *Single Pass:* each record examined at most once
  - *Fixed Order:* no assumption about the arrival order can be made
  - *Small Space:* log or poly-log in data stream size – main memory algorithms
Class of Queries: Aggregates over Joins

- Equi-join $J$ specified by relations $R_1, \ldots, R_r$ and join constraints $R_1.a_1 = R_2.a_2, \ldots$

\[
J = R_1 \Join \cdots \Join R_r = \left\{ t \in R_1 \times \cdots \times R_r \mid t.R_1.a_1 = t.R_2.a_2, \ldots \right\}
\]

- Queries specified by a join and an aggregate: COUNT, SUM
- $\text{COUNT}(R_1 \Join \cdots \Join R_r)$ is the number of tuples of the join
- Self join size: $\text{SJ}(R) = \text{COUNT}(R \Join R)$
Example: Two-Way Join COUNT Query

Problem: Estimate result of query \( \text{COUNT}(F \bowtie_a G) \)

Example:

- Stream \( F: \) \begin{bmatrix} a & 1 & 1 & 2 & 3 & 1 & 3 \end{bmatrix} \), frequency vector \( f: \)
  \begin{bmatrix} i & 1 & 2 & 3 \\ f_i & 3 & 1 & 2 \end{bmatrix}

- Stream \( G: \) \begin{bmatrix} a & 3 & 1 & 3 & 1 & 1 \end{bmatrix} \), frequency vector \( g: \)
  \begin{bmatrix} i & 1 & 2 & 3 \\ g_i & 3 & 0 & 2 \end{bmatrix}

\[
\text{COUNT}(F \bowtie_a G) = fg^T
\]

\[
= \begin{bmatrix} 3 & 1 & 2 \end{bmatrix} \begin{bmatrix} 3 \\ 0 \\ 2 \end{bmatrix}
\]

\[
= 3 \cdot 3 + 1 \cdot 0 + 2 \cdot 2 = 13
\]

- Space requirement proportional with size of \( f \) and \( g \)

- Multi dimensional data space can be prohibitive
  E.g. Three attributes, each with domains of size 1000 \( \Rightarrow \) \( 10^9 \) words
Stream Data Synopses

- Frequency table maintenance over streams requires too much space
  \( \Rightarrow \) Summarization required

- Conventional data summaries fall short
  - Quantiles and 1-d histograms [MSL98], [GK 01], [GKMS02]
    * Cannot capture attribute correlations
    * Little support for approximation guarantees
  - Samples (e.g., using Reservoir Sampling)
    * Perform poorly for non foreign key joins [AGMS99]
    * Cannot handle deletions of records
  - Multi-dimensional histograms/wavelets
    * Construction requires multiple passes over the data
    * Concurrent work [TGIK02]

- Our approach: use \textit{sketches} [AMS96],[AGMS99]
Outline of the Talk

• Background

• Introduction to sketches
  – Log space in the size of stream and size of domain of attributes
  – Provable *probabilistic guarantees*
  – Insertions and deletions possible

• Sketch partitioning

• Sketch sharing

• Future work
Recall Example

Problem: Estimate result of query $\text{COUNT}(F \Join_a G)$

Example:

- Stream $F$: $\begin{array}{|c|}
\hline
a & 1 & 1 & 2 & 3 & 1 & 3 \\
\hline
\end{array}$, frequency vector $f$: $\begin{array}{|c|}
\hline
i & 3 & 1 & 2 \\
\hline
f_i & 1 & 2 & 3 \\
\hline
\end{array}$

- Stream $G$: $\begin{array}{|c|}
\hline
a & 3 & 1 & 3 & 1 & 1 \\
\hline
\end{array}$, frequency vector $g$: $\begin{array}{|c|}
\hline
i & 1 & 2 & 3 \\
\hline
g_i & 3 & 0 & 2 \\
\hline
\end{array}$

$\text{COUNT}(F \Join_a G) = fg^T$

$= \begin{bmatrix} 3 & 1 & 2 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} 3 \\ 3 \end{bmatrix}$

$= 3 \cdot 3 + 1 \cdot 0 + 2 \cdot 2$

$= 13$
Basic Sketching Technique
(Alon, Gibbons, Matias, Szegedy 1999)

Main Idea:

- Summarize information in the stream with a single number ⇒ sketch
- Use sketches to recover query result

Sketches:

- Choose random \( \xi_1, \xi_2, \xi_3 \in \{-1, 1\} \)
- Stream \( F \):
  \[
  \begin{array}{c|ccccccc}
  a & 1 & 1 & 2 & 3 & 1 & 3 \\
  \end{array}
  \]
  \[
  X_F = \xi_1 + \xi_1 + \xi_2 + \xi_3 + \xi_1 + \xi_3
  \]
- Stream \( G \):
  \[
  \begin{array}{c|ccccccc}
  a & 3 & 1 & 3 & 1 & 1 \\
  \end{array}
  \]
  \[
  X_G = \xi_3 + \xi_1 + \xi_3 + \xi_1 + \xi_1
  \]
- Claim \( X = X_F X_G \) estimates \( \text{COUNT}(F \bowtie_a G) \)
Analysis of Basic Sketching Technique

Random vectors:
- $\xi = [\xi_1 \ldots \xi_n]$ random vector of $\pm 1$ values, called projection vector

Sketches:
- Sketch of $F$, $X_F = \sum_{t \in F} \xi_{t,a} = \sum_i f_i \xi_i = f\xi^T$
- Sketch of $G$, $X_G = \sum_{t \in G} \xi_{t,a} = \sum_i g_i \xi_i = g\xi^T$
- $X = X_F X_G$ estimates $\text{COUNT}(F \bowtie_a G)$ since
  \[E[X] = E[f\xi^T g] = fE[\xi^T]g^T = fg^T\]
  if $E[\xi^T] = I \iff \forall i_1 \neq i_2, E[\xi_{i_1} \xi_{i_2}] = 0$
- $X$ is not precise
  \[\text{Var}(X) \leq 2ff^Tgg^T = 2 \text{SJ}(F) \text{ SJ}(G)\]
  if $\forall i_1 \neq i_2 \neq i_3 \neq i_4, E[\xi_{i_1} \xi_{i_2} \xi_{i_3} \xi_{i_4}] = 0$
- $\xi$ need not have elements fully independent. 4-wise independence suffices
  \[\Rightarrow \xi \text{ can be efficiently generated from small seeds } [\text{ABI86}]\]
  \[\Rightarrow \xi \text{ vector is not stored. Required elements generated on the fly from seeds}\]
Sketch Error Reduction – Standard Solution

- Estimation of \( \text{COUNT}(F \rtimes_a G) \) from single sketches of \( F \) and \( G \) is too noisy
  \[
  E^2[X] \leq \text{Var}(X)
  \]

Solution:

- Average \( \frac{8}{\varepsilon^2} \frac{\text{Var}(X)}{E^2[X]} \) independent copies of \( X \) to reduce error to \( \varepsilon \)
- Compute median of \( 2 \log 1/\delta \) such averages to increase confidence to \( 1 - \delta \)

- Memory required weakly dependent on the size of the stream: \( \log N + 1 \) bits/sketch
Properties of Sketches

Advantages:

• Space requirement logarithmic in the size of the stream and of the domain of the attributes

• Provable probabilistic approximation guarantees

• Both insertion and deletion possible
  – For deletion tuples simply decrement the sketch instead of incrementing

• Simplicity

Disadvantages:

• They apply only to the query \( \text{COUNT}(F \bowtie a G) \)

• When \( \frac{\text{Var}(X)}{E^2[X]} \gg 1 \) a large amount of space required
Estimation of \( \text{COUNT}(R_1 \bowtie \cdots \bowtie R_r) \)  
(Dobra, Garofalakis, Gehrke, Rastogi, 2002)

\[
X_l = \sum_{t \in R_L} a_1 \xi_1^{(1)} \cdots a_m \xi_1^{(m)}
\]

- With \( X = \prod_{l=1}^r X_l \) can show:
  \[
  E[X] = \text{COUNT}(R_1 \bowtie \cdots \bowtie R_r)
  \]
  \[
  \text{Var}(X) \approx C \prod_{l=1}^r \text{SJ}(R_l)
  \]

  - Assign a distinct projection vector \( \xi \) to each join constraint
Our Contributions

• Unless distributions in $F$ and $G$ are similar $\frac{\text{Var}(X)}{E^2[X]} \gg 1$
  $\Rightarrow$ Need to produce a large number of copies of $X$

Contributions: Two novel methods to reduce the error of estimates

• Sketch partitioning (Dobra, Garofkalakis, Gehrke, Rastogi, 2002)
  – Use extra information to intelligently partition the problem
  – The partitioning results in error reduction

• Sketch sharing (Dobra, Garofkalakis, Gehrke, Rastogi, 2003)
  – Applicable when multiple queries are simultaneously answered
  – Share sketches between queries thus saving space
  – Multi query optimization problem
Outline of the Talk

• Background

• Introduction to sketches

• Sketch partitioning
  – Use statistical information to reduce error

• Sketch sharing

• Future work
Sketch Partitioning (DGGR02)

- Large variance means loose estimation guarantees
  ⇒ Building estimate with smaller variance reduces error

- Consider query: \( \text{COUNT}(F \bowtie_a G) \)

<table>
<thead>
<tr>
<th></th>
<th>( f_i )</th>
<th>( g_i )</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>20</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>10</td>
<td></td>
</tr>
</tbody>
</table>

\[ E[X] = 165, \quad E^2[X] = 27225 \]
\[ \text{Var}(X) \approx 2 \text{SJ}(F) \text{SJ}(G) \]
\[ = 2(20^2 + 5^2 + 10^2 + 2^2)(2^2 + 15^2 + 3^2 + 10^2) \]
\[ = 357604 \]

Idea: Split \( I = \{1, 2, 3, 4\} \) into \( I_1 = \{1, 3\} \) and \( I_2 = \{2, 4\} \)

\[ E[X'] = \text{COUNT}(F_1 \bowtie G_1) + \text{COUNT}(F_2 \bowtie G_2) = \text{COUNT}(F \bowtie G) \]
Sketch Partitioning (cont.)

- Estimation of $\text{COUNT}(F_1 \Join G_1)$

<table>
<thead>
<tr>
<th>$i$</th>
<th>$f_i$</th>
<th>$g_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>20</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>3</td>
</tr>
</tbody>
</table>

$\text{Var}(X_1) \approx 2 \ SJ(F_1) \ SJ(G_1) \\
= 2(20^2 + 10^2)(2^2 + 3^2) \\
= 13000$

- Estimation of $\text{COUNT}(F_2 \Join G_2)$

<table>
<thead>
<tr>
<th>$i$</th>
<th>$f_i$</th>
<th>$g_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>5</td>
<td>15</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>10</td>
</tr>
</tbody>
</table>

$\text{Var}(X_2) \approx 2 \ SJ(F_2) \ SJ(G_2) \\
= 2(5^2 + 2^2)(15^2 + 10^2) \\
= 18850$

- $\text{Var}(X') = \text{Var}(X_1) + \text{Var}(X_2) = 31850$

- Improvement

$$\frac{\text{Var}(X)/m}{\text{Var}(X')/(m/2)} = \frac{357604}{231850} \approx 5.6 \Rightarrow \varepsilon \downarrow 2.4$$

- Intelligently splitting the domain of the join attribute $\Rightarrow$ reduction in error

- **Question:** Assuming we know $f$ and $g$, what is the best way to split?
Binary Sketch Partitioning

Optimization Problem: Find

- partition $I = I_1 \cup I_2$
- space allocation $m_1$ for $X_1$, $m_2$ for $X_2$ s.t. $m_1 + m_2 = m$

that minimizes

$$\frac{\text{Var}(X_1)}{m_1} + \frac{\text{Var}(X_2)}{m_2},$$

where

$$\text{Var}(X_k) \approx 2 \sum_{i \in I_k} f_i^2 \sum_{i \in I_k} g_i^2.$$

Solution:

- Allocate space proportional to $\sqrt{\text{Var}(X_k)}$
- Transformed optimization criterion:

$$\Psi(I_1) = \sqrt{\sum_{i \in I_1} f_i^2 \sum_{i \in I_1} g_i^2} + \sqrt{\sum_{i \in I_2} f_i^2 \sum_{i \in I_2} g_i^2}.$$

- Naive solution: check all $2^{|I|}$ ways to partition
Binary Sketch Partitioning

**Breiman’s Theorem**[BFOS84]: For $q_i > 0, r_i > 0$ and $\Phi(x)$ concave, an optimum of the problem

$$\arg\min_{I_1, I_2} \sum_{i \in I_1} q_i \Phi \left( \frac{\sum_{i \in I_1} q_i r_i}{\sum_{i \in I_1} q_i} \right) + \sum_{i \in I_2} q_i \Phi \left( \frac{\sum_{i \in I_2} q_i r_i}{\sum_{i \in I_2} q_i} \right)$$

has the property that:

$$\forall i \in I_1, \forall j \in I_2, r_i < r_j$$

**Corollary:** The solution of the optimization problem can be found in $O(|I| \log |I|)$ by sorting $i \in I$ in increasing order of $r_i$ and considering splits only in this order.

**Application 1:** Finding the split point for discrete attribute in classification tree construction. Taking $\Phi(x) = 2x(1-x), q_i = P[X = x_i], r_i = P[C = c_0 | X = x_i]$ the criterion is gini index (impurity) after split.

**Application 2:** Find the optimal binary partitioning:

$$\Phi(x) = \sqrt{x}, \quad q_i = \frac{f_i^2}{\sum_{i \in I} f_i^2}, \quad r_i = \frac{g_i^2}{f_i^2}, \quad \text{criterion becomes} \quad \frac{\Psi(I_1)}{\sum_{i \in I} f_i^2}$$
Binary Sketch Partitioning Algorithm

Algorithm:

- Order elements of $I$ in increasing order of $\frac{g_i}{f_i}$
- Choose the partitioning in this order that minimizes $\Psi(I_1)$
- For the optimal partitioning take memory allocation proportional to $\sqrt{\text{Var}(X_k)}$

Example:

- Optimal partition

<table>
<thead>
<tr>
<th>$i$</th>
<th>$\frac{g_i}{f_i}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>.1</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>.3</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
</tr>
</tbody>
</table>

- Ordering within $I$: $1, 3, 2, 4$
- Optimal partition: $I_1 = \{1, 3\}, I_2 = \{2, 4\}$
- Considering splits only in this order matches the intuition

- Optimal memory allocation 5 : 6
Sketch Partitioning: Extensions

Using Imperfect Information:

- Can get good partitioning without knowledge of perfect frequency tables
- Use rough unidimensional histograms instead
  - time and space dependency on number of buckets instead of $|I|$
  - provable approximation quality

Generalization:

- K-ary split partitioning
  - Needs generalization of Breiman’s Theorem
  - Dynamic programming algorithm
- Multi-way joins
  - NP-hard
  - With independence assumption becomes tractable
    * decomposes into independent unidimensional problems
Experimental Study

Datasets:

- Synthetic data sets [Vitter & Wang 99]:
  - Rectangular regions in multi-dimensional attribute space uniformly distributed. Domain size 1024
  - Each region has a Zipfian distribution
  - Number of tuples in each region Zipf distributed
  - Total number of tuples: 10M
  - To simulate correlations we perturbed the placement of regions

- Census data set (www.bls.census.gov) – reported in [DGGR02]

Comparison: estimation using basic method

Query load: JOIN-COUNT queries

Error metric: relative error $= 100 \frac{\text{actual} - \text{approx}}{\text{actual}} \%$
Join of two independent unidimensional relations on a common attribute

- Space for sketches 4000 words. 45% - 65% improvement
- Partitioning in two good enough; finer histograms do not help much

Alin Dobra – Processing Aggregate Queries over Continuous Data Streams
Chain join of three bidimensional relations on two common attributes

- Space for sketches 9000 words. 30% - 50% improvement
- Same trends

Alin Dobra – Processing Aggregate Queries over Continuous Data Streams
Outline of the Talk

- Background
- Introduction to sketches
- Sketch partitioning
  - Exploit commonality in the queries to reduce error
- Future work
Answering multiple queries simultaneously (DGGR, under review)

- Can apply the basic technique for each join independently

\[
\text{COUNT}(R_1 \Join R_2) = \sum_{t \in R_1} \xi_{t.a}^{(1)} \cdot a \sum_{t \in R_2} \xi_{t.a}^{(1)}
\]

\[
\text{COUNT}(R_1 \Join R_3 \Join R_2) = \sum_{t \in R_1} \xi_{t.a}^{(2)} \cdot a \sum_{t \in R_3} \xi_{t.a}^{(2)} \xi_{t.b}^{(3)} \cdot a \sum_{t \in R_2} \xi_{t.a}^{(3)}
\]

- If sketches are reused fewer sketches have to be maintained
  - The total available space is divided between fewer sketches

**Question:** How and when can sketches be reused?
**Sketch Sharing**

![Diagram of sketch sharing](image)

- Sharing the sketch of $R_1$ for both joins
  - Use $\xi^{(1)}$ to enforce both $<R_1.a, R_2.a>$ and $<R_1.a, R_3.a>$ constraints
- Number of sketches *reduced* from 5 to 4
  - Each sketch gets 25% more space

COUNT($R_1 \bowtie R_2$)
COUNT($R_1 \bowtie R_3 \bowtie R_2$)

Alin Dobra – Processing Aggregate Queries over Continuous Data Streams
Sketch Sharing

- Sketch sharing for $R_1$
  $\Rightarrow \xi^{(1)}$ on both $<R_1.a, R_2.a>$ and $<R_3.b, R_2.a>$ edges
  $\Rightarrow \xi^{(1)}$ used on all edges

- Number of sketches reduced from 5 to 3
  $\Rightarrow$ Each sketch gets 66% more space

- Simple sketch sharing algorithm

Alin Dobra – Processing Aggregate Queries over Continuous Data Streams
**Sketch Sharing**

**Problem:** Estimate $\text{COUNT}(R_1 \Join R_2 \Join R_3) = f_{R_1} f_{R_2}^T f_{R_3}^T$.

\[ \Phi = E \left[ \sum_{t \in R_1} \xi_{t,a} \sum_{t \in R_2} \xi_{t,a} \xi_{t,b} \sum_{t \in R_3} \xi_{t,a} \right] \]

\[ = E[f_{R_1} \xi^T \xi f_{R_2}^T \xi f_{R_3}^T] \]

\[ = f_{R_1} E[\xi^T \xi f_{R_2}^T \xi f_{R_3}^T] \]

\[ \neq f_{R_1} f_{R_2}^T f_{R_3}^T \]

since $E[\xi^T \xi f_{R_2}^T \xi f_{R_3}^T] \neq f_{R_2}^T$

- $\Phi \neq \text{COUNT}(R_1 \Join R_2 \Join R_3)$
  - $\Rightarrow$ sharing sketch for $R_2$ results in wrong estimation

- Apparent reason: $\xi^{(1)}$ assigned to both constraints of red join
Estimation of \( \text{COUNT}(R_1 \Join \cdots \Join R_r) \)

\[
X_l = \sum_{t \in R_L} \xi_{t.a_1} \cdots \xi_{t.a_m}
\]

- With \( X = \prod_{l=1}^r X_l \) can show:

\[
E[X] = \text{COUNT}(R_1 \Join \cdots \Join R_r)
\]

\[
\text{Var}(X) \approx C \prod_{l=1}^r \text{SJ}(R_l)
\]

- Assign a distinct projection vector \( \xi \) to each join constraint
Sketch Sharing: Conflicts in Sharing Schemes

Correctness: within each join a different $\xi$-family is assigned to each join constraint.

No conflict.

Conflict.
Sketch Sharing: Algorithm

**Problem:** Given a set of queries to be approximated compute, the sketch sharing scheme with no conflicts that has the smaller number of sketches.

**Characterization:** Problem is NP-hard.

**Heuristic Algorithm:**

1. Start with no sketch sharing
2. Find pair of sketches that can be shared (if any)
   - Avoid introducing conflicts
3. Repeat Step 2 until no sharing possible
Space Allocation Problem

Key Observation:

- Allocating identical space to each sketch might not optimize average error/max error
- Relative square error for $Q$
  \[ \varepsilon_Q^2 \sim \frac{\text{Var}(X)}{M_Q E^2[X]} = \frac{W_Q}{M_Q} \]
- $m_v$ memory allocation of sketch $v \in \mathcal{V}$, $M_Q$ memory allocation for query $Q \in \mathcal{Q}$

Space allocation problem: Find $m_v$, $v \in \mathcal{V}$ and $M_Q$, $Q \in \mathcal{Q}$ that minimizes either

- Average Error: $\sum_{Q \in \mathcal{Q}} \frac{W_Q}{M_Q}$, or
- Max Error: $\max_{Q \in \mathcal{Q}} \frac{W_Q}{M_Q}$

such that

\[ \forall v \in \mathcal{V}, \forall Q \in Q(v) : M_Q \leq m_v \quad \text{(sketch-query constraint)} \]
\[ \sum_{v \in \mathcal{V}} m_v = m \quad \text{(total memory constraint)} \]
Solving the Space Allocation Problem

- Can integrate space allocation problems into sketch sharing algorithm

**Heuristic Algorithm:**

1. Start with no sketch sharing
2. Find pair of sketches that can be shared (if any)
   - Avoid introducing conflicts
3. Repeat Step 2 until no sharing possible
   - In Step 2 of algorithm
     * solve the space allocation problem for each pair
     * pick among the candidate pairs the one that reduces error the most

- Minimize max error: reduces to classic optimization problem
  - Can be solved in $O(|\mathcal{V}| \log |\mathcal{V}|)$

- Minimize average error: complex convex optimization problem
  - Integer version NP-hard
  - Continuous version gives solution with good approximation
    * efficient custom solution

Alin Dobra – Processing Aggregate Queries over Continuous Data Streams
Continuous Average Error Problem

- Optimization problem:

\[
\min \sum_{Q \in \mathcal{Q}} W_Q \Phi(M_Q)
\]

\[\forall v \in \mathcal{V}, \forall Q \in \mathcal{Q}(v) : M_Q \leq m_v\]

\[\sum_{v \in \mathcal{V}} m_v = m\]

- Strictly convex optimization problem
  - Solvable in polynomial time with general convex solvers. Slow in practice

- Solution satisfies the Karush-Kuhn-Tucker conditions:

\[\forall Q \in \mathcal{Q} : W_Q \Phi'(M_Q) + \sum_{v \in \mathcal{V}(Q)} \mu_{v,Q} = 0\]

\[\forall v \in \mathcal{V} : \sum_{Q \in \mathcal{Q}(v)} \mu_{v,Q} = \lambda\]

\[\forall Q \in \mathcal{Q}, \forall v \in \mathcal{J} : \mu_{v,Q} \cdot (m_v - M_Q) = 0\]

\[\sum_{v \in \mathcal{J}} m_v = M\]
Continuous Average Error Problem

- KKT equations not directly solvable
- Key idea: define equivalence relation $\equiv$
  - $v \in \mathcal{V}$ and $Q \in Q(v)$ are equivalent if $m_v = M_Q$ in optimal solution + closure
  - Induces equivalence classes $C_1, C_2, \ldots$ over $\mathcal{V} \cup \mathcal{Q}$

- Using KKT conditions can show:
  - Is enough to find the equivalence classes $C_1, C_2, \ldots$ to get the solution
  - Can get conditions that equivalence classes satisfy
  - Can setup a Max-flow problem that finds divider between classes
Experiments – TPC-H

\[ \mathcal{W}_1 = Q_1 : Q_{12}, \quad \mathcal{W}_2 = Q_1 : Q_{29} \]

Alin Dobra – Processing Aggregate Queries over Continuous Data Streams
Relative error reduced by 35%

Number of sketches reduced from 34 to 16
• Relative error reduced by 45%. Larger reduction

• Number of sketches reduced from 82 to 25
Summary

• Basic sketch technique
  – Summarize streams by projecting them on random vectors
  – Log space, provable estimation guarantees, deletions
  – Require a lot of space when variance is large

Contributions:

• Sketch partitioning
  – Uses extra information to intelligently split domain of attributes
  – Results in better approximation scheme

• Sketch sharing
  – Exploits commonality in queries
  – Space and computation is shared among queries

• Both methods lead to interesting optimization problems
Future Work: Applications

- Aggregates over joins are building blocks for higher level applications
  - Useful for correlating information from multiple sources
- Integrate approximate query processing with data-mining and monitoring
Future Work: Distributed Computation

• Sketches are linear: $\text{SK}(D_1 \cup D_2) = \text{SK}(D_1) + \text{SK}(D_2)$
  $\Rightarrow$ Sketches are useful for distributed computation
    – No need to move data. Send sketches instead
    – Computation of aggregates over joins as easy as computation of SUM

• Gossip-style decentralized algorithms for aggregate queries (with Kempe & Gehrke)

Alin Dobra – Processing Aggregate Queries over Continuous Data Streams
Future work

- Extensions
  - Combination with other synopses data-structures
  - Distinct value queries
  - Retroactive queries

- Improvements in the presence of extra knowledge
  - Types of extra knowledge
    * Schema information: foreign keys, size of relations
    * Statistical information
    * Workload information
  - Find novel ways to use extra knowledge

- Approximate stream query processing system
  - Integration into existing system or new system
Questions