Multi-Dimensional Regression Analysis of Time-Series Data Streams

Yixin Chen, Guozhu Dong, Jiawei Han,
Benjamin W. Wah, Jianyong Wang

University of Illinois at Urbana-Champaign
Wright State University

Outline

- Characteristics of stream data
- Why on-line analytical processing and mining of stream data?
- Linearly compressed representation of stream data
- A stream cube architecture
- Stream cube computation
- Discussion
- Conclusions

Characteristics of Stream Data

- Huge volumes of data, possibly infinite
- Fast changing and requires fast response
- Data stream is more suited to our data processing needs of today
- Single linear scan algorithm: can only have one look
  - random access is expensive
- Store only the summary of the data seen thus far
- Most stream data reside at pretty low-level or multi-dimensional in nature—needs ML (multi-level) / MD (multi-dimensional) processing

Stream Data Applications

- Telecommunication calling records
- Business: credit card transaction flows
- Network monitoring and traffic engineering
- Financial market: stock exchange
- Engineering & industrial processes: power supply & manufacturing
- Sensor, monitoring & surveillance: video streams
- Security monitoring
- Web logs and Web page click streams
- Massive data sets (even saved but random access is too expensive)

Projects on DSMS (Data Stream Management System)

- STREAM (Stanford): A general-purpose DSMS
- Cougar (Cornell): sensors
- Aurora (Brown/MIT): sensor monitoring, dataflow
- Hancock (AT&T): telecom streams
- Niagara (OGI/Wisconsin): Internet XML databases
- OpenCQ (Georgia Tech): triggers, incr. view maintenance
- Tapestry (Xerox): pub/sub content-based filtering
- Telegraph (Berkeley): adaptive engine for sensors
- Tradebot (www.tradebot.com): stock tickers & streams
- Tribeca (Bellcore): network monitoring

Previous Work: Towards OLAP and Mining Data Streams

- Stream data model
  - Data Stream Management System (DSMS)
- Stream query model
  - Continuous Queries
  - Sliding windows
- Stream data mining
  - Clustering & summarization (Guha, Motwani, et al.)
  - Correlation of data streams (Gehrke, et al.)
  - Classification of stream data (Domingos, et al.)
  - Mining frequent sets in streams (Motwani, et al., VLDB'02)

Why Stream Cube and Stream OLAP?

- Most stream data are at pretty low-level or multi-dimensional in nature: needs ML/MD processing
- Analysis requirements
  - Multi-dimensional trends and unusual patterns
  - Capturing important changes at multi-dimensions/levels
  - Fast, real-time detection and response
  - Comparing with data cube: Similarity and differences
- Stream (data) cube or stream OLAP
  - Is it feasible?   How to implement it efficiently?

Multi-Dimensional Stream Analysis: Examples

- Analysis of Web click streams
  - Raw data at low levels: seconds, web page addresses, user IP addresses, …
  - Analysts want: changes, trends, unusual patterns, at reasonable levels of details
  - E.g., *Average clicking traffic in North America on sports in the last 15 minutes is 40% higher than that in the last 24 hours.”*
- Analysis of power consumption streams
  - Raw data: power consumption flow for every household, every minute
  - Patterns one may find: *average hourly power consumption surges up 30% for manufacturing companies in Chicago in the last 2 hours today than that of the same day a week ago*

Motivations for Stream Data Compression

- Challenges of OLAPing stream data
  - Raw data cannot be stored
  - Simple aggregates not powerful enough
  - History shape and patterns at different levels are desirable: multi-dimensional regression analysis
- Proposal
  - A scalable multi-dimensional stream data warehouse that can aggregate regression model of stream data efficiently without accessing the raw data
- Stream data compression
  - Compress the stream data to support memory- and time-efficient multi-dimensional regression analysis

Basics of General Linear Regression

- n tuples in one cell: ($\boldsymbol{x_i}$, y_i), $i = 1..n$, where y_i is the measure attribute to be analyzed
- For sample i, a vector of k user-defined predictors $\boldsymbol{u}_i$:

$$\mathbf{u}_i = \begin{pmatrix} u_0 \\ u_1(\mathbf{x}_i) \\ \ldots \\ u_{k-1}(\mathbf{x}_i) \end{pmatrix} = \begin{pmatrix} 1 \\ u_{i1} \\ \ldots \\ u_{i,k-1} \end{pmatrix}$$

- The linear regression model:

$$E(y_i \mid \mathbf{u}_i) = h^T \mathbf{u}_i = h_0 + h_1 u_{i1} + \ldots + h_{k-1} u_{i,k-1}$$

where ? is a $k \times 1$ vector of <u>regression parameters</u>

Theory of General Linear Regression

- Collect u_i into the $n \times k$ model matrix **U**

$$\mathbf{U} = \begin{pmatrix} 1 & u_{11} & u_{12} & \ldots & u_{1,k-1} \\ 1 & u_{21} & u_{22} & \ldots & u_{2,k-1} \\ . & . & . & . & . \\ . & . & . & . & . \\ 1 & u_{n1} & u_{n2} & \ldots & u_{n,k-1} \end{pmatrix}$$

- The *ordinary least square* (OLS) estimate $\hat{h}$ of h is the argument that minimize the residue sum of squares function

$$RSS(h) = (\mathbf{y} - \mathbf{U}?)^T(\mathbf{y} - \mathbf{U}?)$$

- Main theorem to determine the OLS regression parameters

$$\frac{\partial}{\partial h} RSS(h) = 0 \Rightarrow \hat{h} = (\mathbf{U}^T\mathbf{U})^{-1}\mathbf{U}^T\mathbf{y}$$

Linearly Compressed Representation (LCR)

- Stream data compression for multi-dimensional regression analysis
- Define, for *i, j* = 0, …, *k*-1:

$$q_{ij} = \sum_{h=1}^{n} u_{hi} u_{hj}$$

- The <u>linearly compressed representation</u> (LCR) of one cell:

$$\{\hat{h}_i \mid i = 0,\ldots,k-1\} \cup \{q_{ij} \mid i, j = 0,\ldots,k-1, i \le j\}$$

- Size of LCR of one cell: $k + \frac{k(k+1)}{2} = \frac{k^2 + 3k}{2}$,

  quadratic in *k*, independent of the number of tuples *n* in one cell

Matrix Form of LCR

- LCR consists of $\hat{h}$ and $\mathbf{T}$, where $\hat{h}^T = (\hat{h}_0, \hat{h}_1, \ldots, \hat{h}_{k-1})$

  and

$$\mathbf{T} = \begin{pmatrix} q_{00} & q_{01} & q_{02} & \cdots & q_{0,k-1} \\ q_{10} & q_{11} & q_{12} & \cdots & q_{1,k-1} \\ . & . & . & \cdots & . \\ . & . & . & \cdots & . \\ q_{k-1,0} & q_{k-1,1} & q_{k-1,2} & \cdots & q_{k-1,k-1} \end{pmatrix}$$

where

$\hat{h}$ provides OLS regression parameters essential for regression analysis

$\mathbf{T}$ is an auxiliary matrix that facilitates aggregations of LCR in standard and regression dimensions in a data cube environment

$\mathbf{T}^T = \mathbf{T} \Rightarrow$ LCR only stores the upper triangle of $\mathbf{T}$

Aggregation in Standard Dimensions

- Given LCR of *m* cells that differ in one standard dimension, what is the LCR of the cell aggregated in that dimension?
  - for *m* base cells

$$LCR_1 = (\hat{h}_1, \mathbf{T}_1), LCR_2 = (\hat{h}_2, \mathbf{T}_2), \ldots, LCR_m = (\hat{h}_m, \mathbf{T}_m)$$

  - for an aggregated cell

$$LCR_a = (\hat{h}_a, \mathbf{T}_a)$$

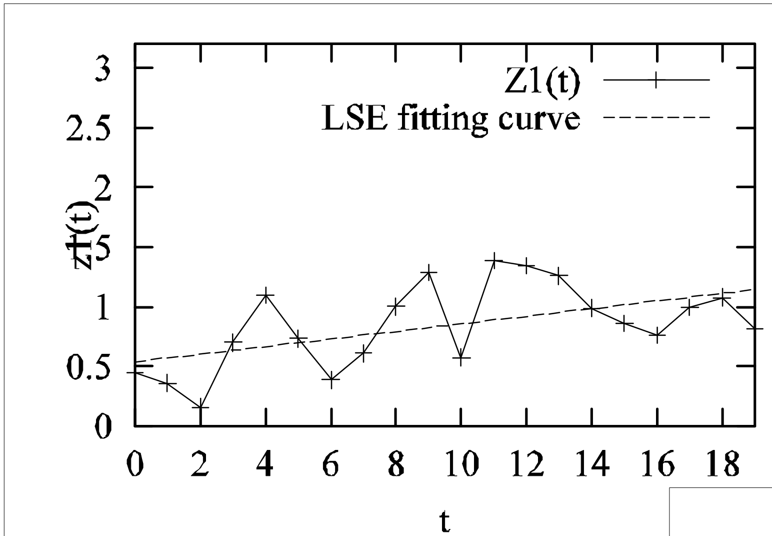
- The lossless aggregation formula

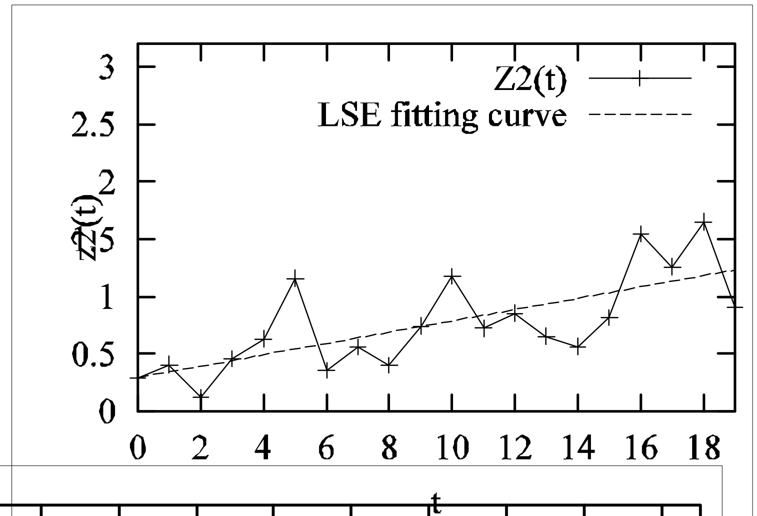
$$\hat{h}_a = \sum_{i=1}^{m} \hat{h}_i$$

$$\mathbf{T}_a = \mathbf{T}_1$$

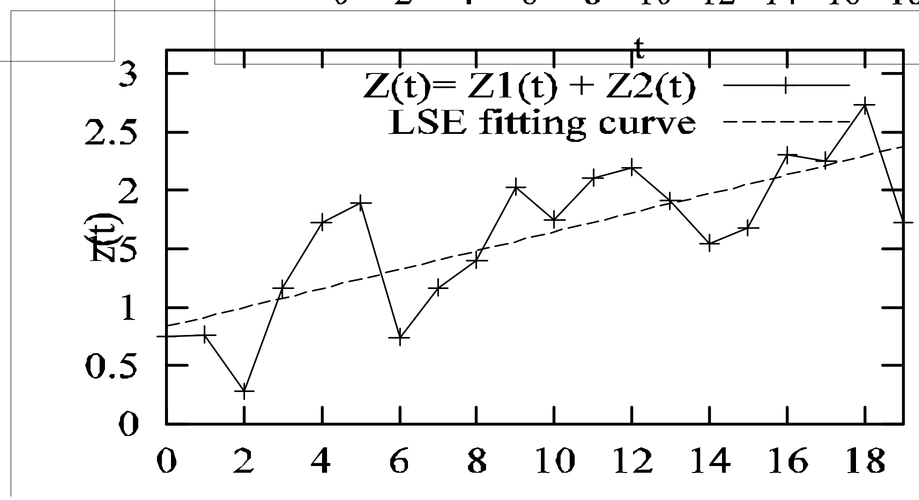
Stock Price Example—Aggregation in Standard Dimensions

- Simple linear regression on time series data
  - Cells of two companies





  - After aggregation:



Aggregation in Regression Dimensions

- Given LCR of *m* cells that differ in one regression dimension, what is the LCR of the cell aggregated in that dimension?

$$LCR_1 = (\hat{h}_1, \mathbf{T}_1), LCR_2 = (\hat{h}_2, \mathbf{T}_2), \ldots, LCR_m = (\hat{h}_m, \mathbf{T}_m)$$ for *m* base cells

$$LCR_a = (\hat{h}_a, \mathbf{T}_a)$$ for the aggregated cell

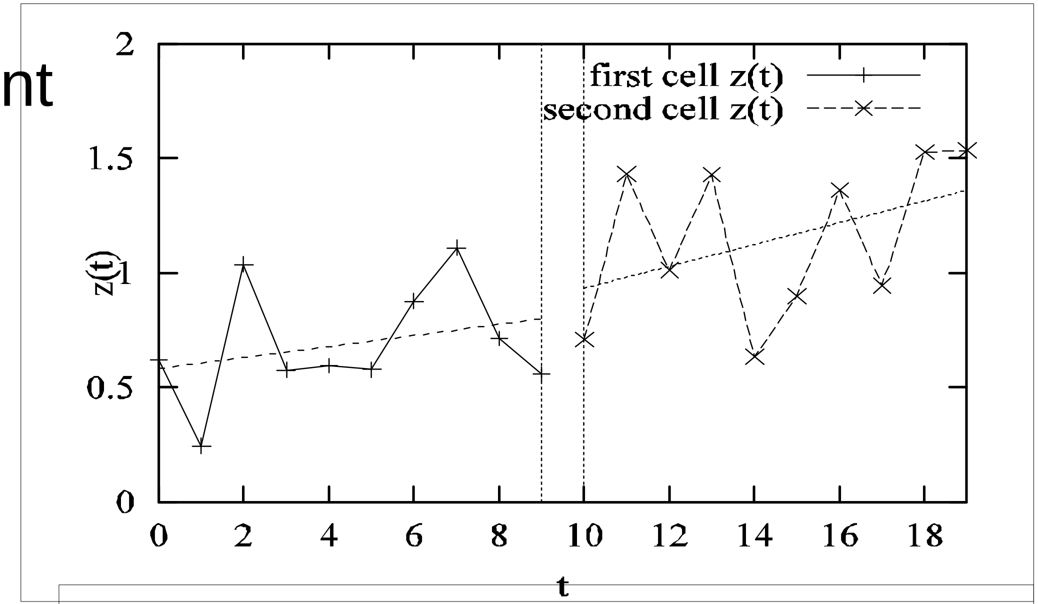
- The lossless aggregation formula

$$\hat{h}_a = \left(\sum_{i=1}^{m} \mathbf{T}_i \right)^{-1} \left(\sum_{i=1}^{m} \mathbf{T}_i \hat{h}_i \right)$$

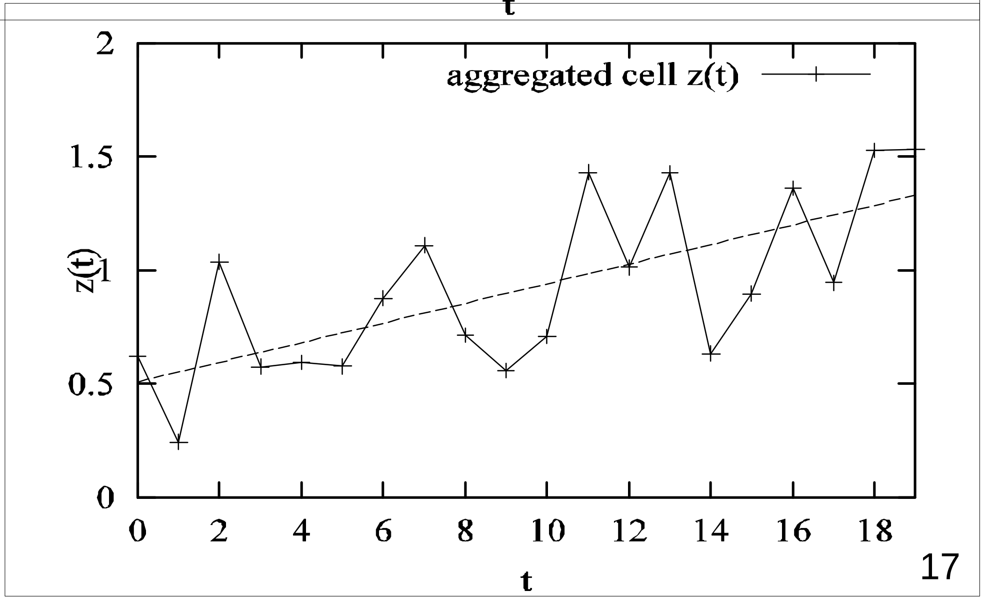
$$\mathbf{T}_a = \sum_{i=1}^{m} \mathbf{T}_i$$

Stock Price Example—Aggregation in Time Dimension

- Cells of two adjacent time intervals:



- After aggregation



Feasibility of Stream Regression Analysis

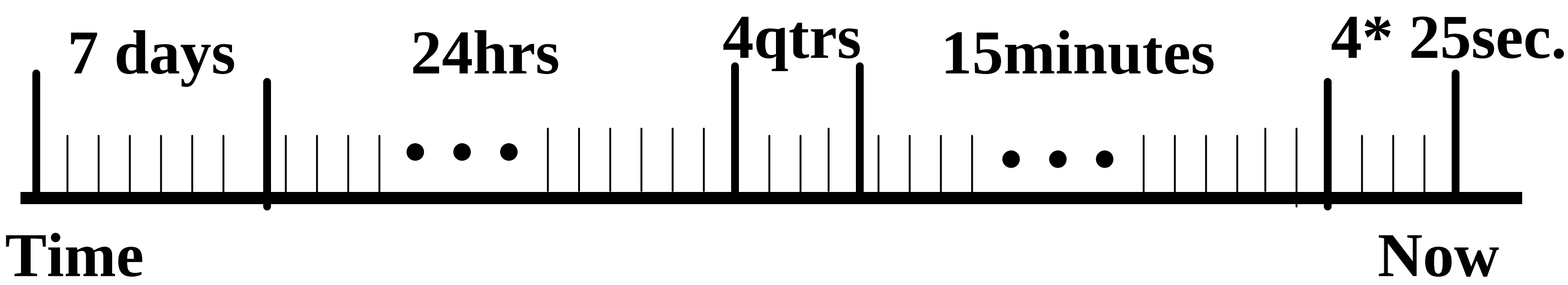
- Efficient storage and scalable (independent of the number of tuples in data cells)
- Lossless aggregation without accessing the raw data
- Fast aggregation: computationally efficient
- Regression models of data cells at all levels
- General results: covered a large and the most popular class of regression
  - Including quadratic, polynomial, and nonlinear models

A Stream Cube Architecture

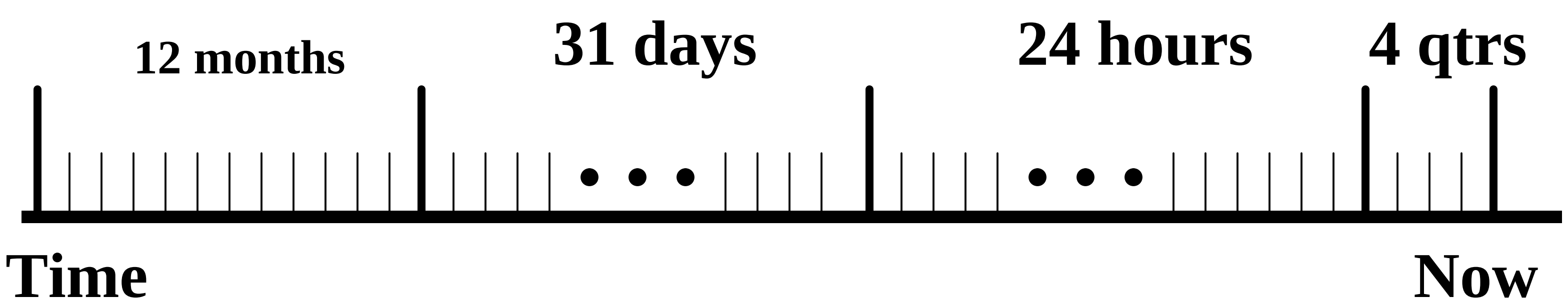
- A tilt time frame
  - Different time granularities
    - second, minute, quarter, hour, day, week, …
- Critical layers
  - Minimum interest layer (m-layer)
  - Observation layer (o-layer)
  - User: watches at o-layer and occasionally needs to drill-down down to m-layer
- Partial materialization of stream cubes
  - Full materialization: too space and time consuming
  - No materialization: slow response at query time
  - Partial materialization: what do we mean "partial"?

A Tilt Time-Frame Model

Up to 7 days



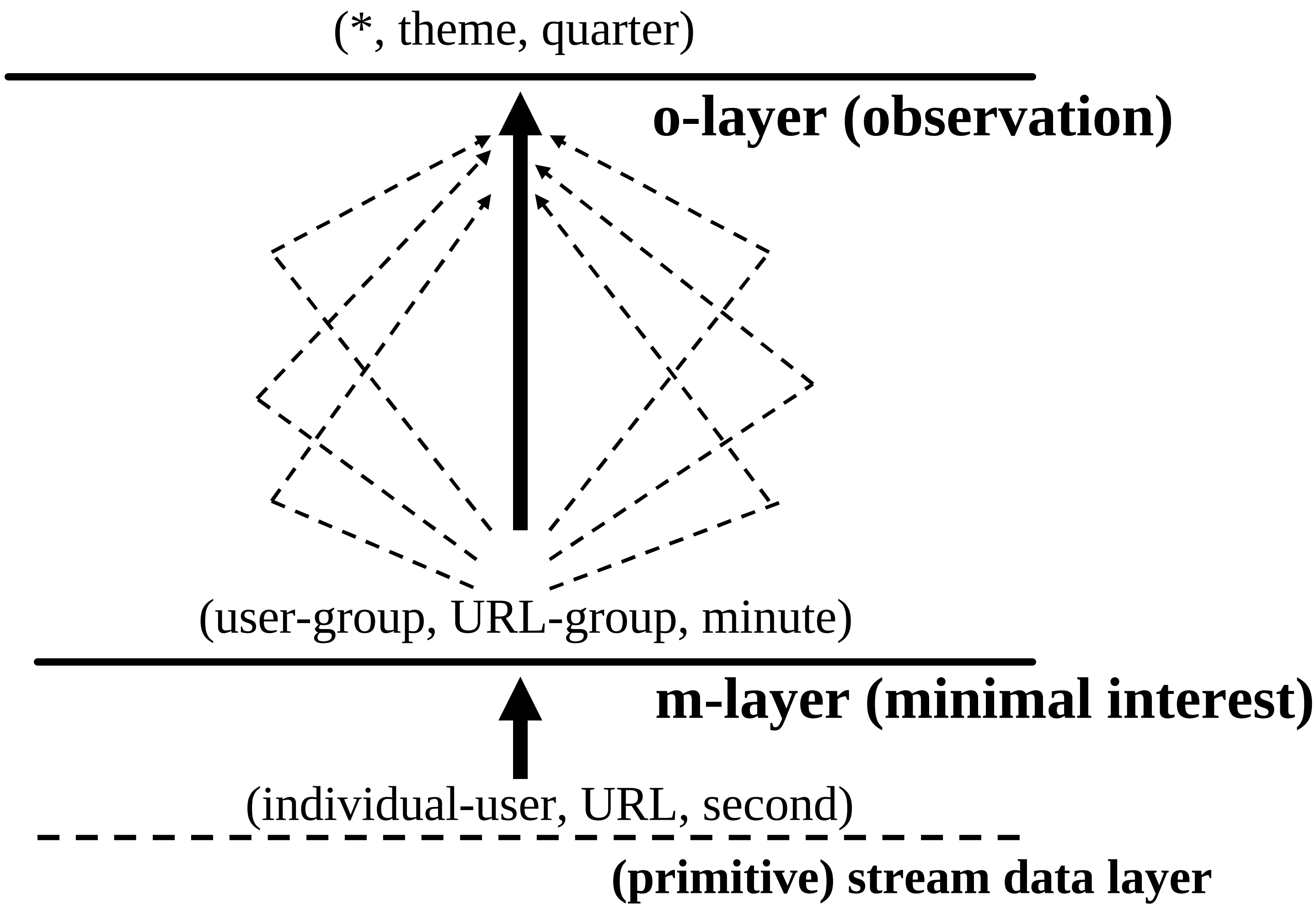
Up to a year



Benefits of Tilt Time-Frame Model

- Each cell stores the measures according to tilt-time-frame
  - Limited memory space: Impossible to store the history in full scale
- Emphasis more on recent data
  - Most applications emphasize on recent data (slide window)
- Natural partition on different time granularities
  - Putting different weights on remote data
  - Useful even for uniform weight
- Tilt time-frame forms a new time dimension
  - for mining changes and evolutions
- Essential for mining unusual patterns or outliers
  - Finding those with dramatic changes
  - E.g., exceptional stocks—not following the trends

Two Critical Layers in the Stream Cube



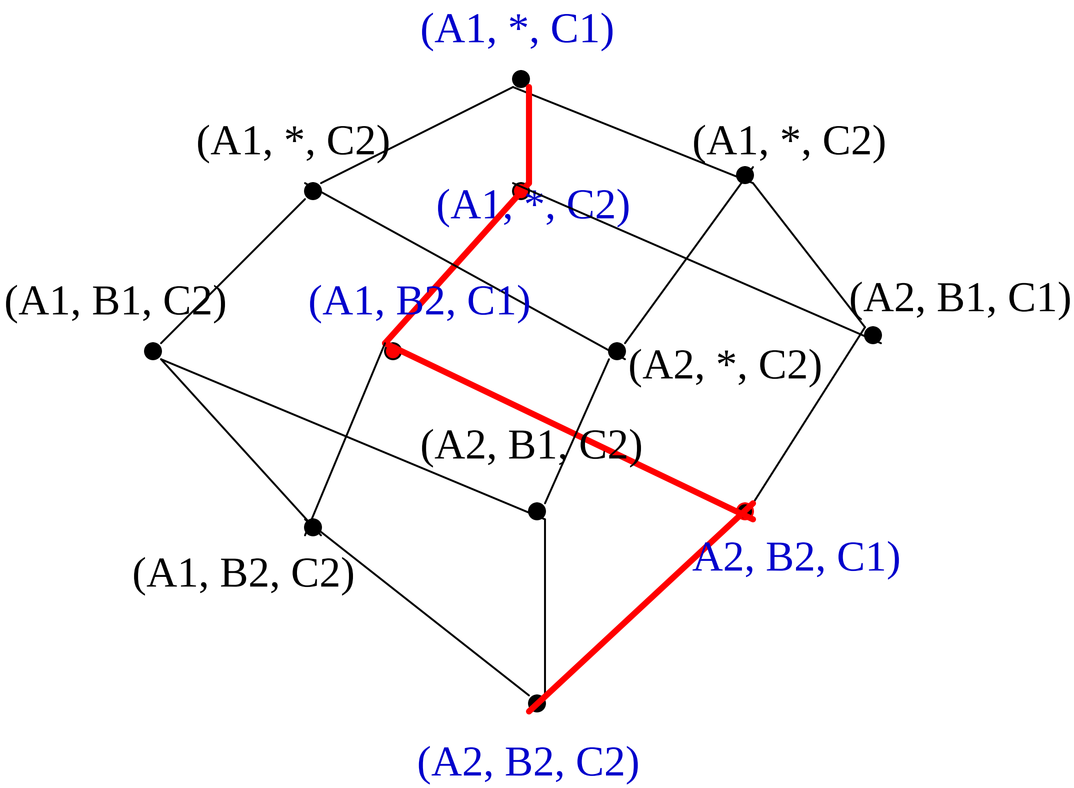
What Are the Issues?

- Materialization problem
  - Only materialize cuboids of the critical layers?
  - Popular path approach vs. exception cell approach
- Computation problem
  - How to compute and store stream cubes efficiently?
  - How to discover unusual cells and patterns between the critical layer?

On-Line Materialization vs. On-Line Computation

- On-line materialization
  - Materialization takes precious resources and time
    - Only incremental materialization (with slide window)
  - Only materialize "cuboids" of the critical layers?
    - Some intermediate cells that should be materialized
  - Popular path approach vs. exception cell approach
    - Materialize intermediate cells along the popular paths
    - Exception cells: how to set up exception thresholds?
    - Notice exceptions do not have monotonic behavior
- Computation problem
  - How to compute and store stream cubes efficiently?
  - How to discover unusual cells between the critical layer?

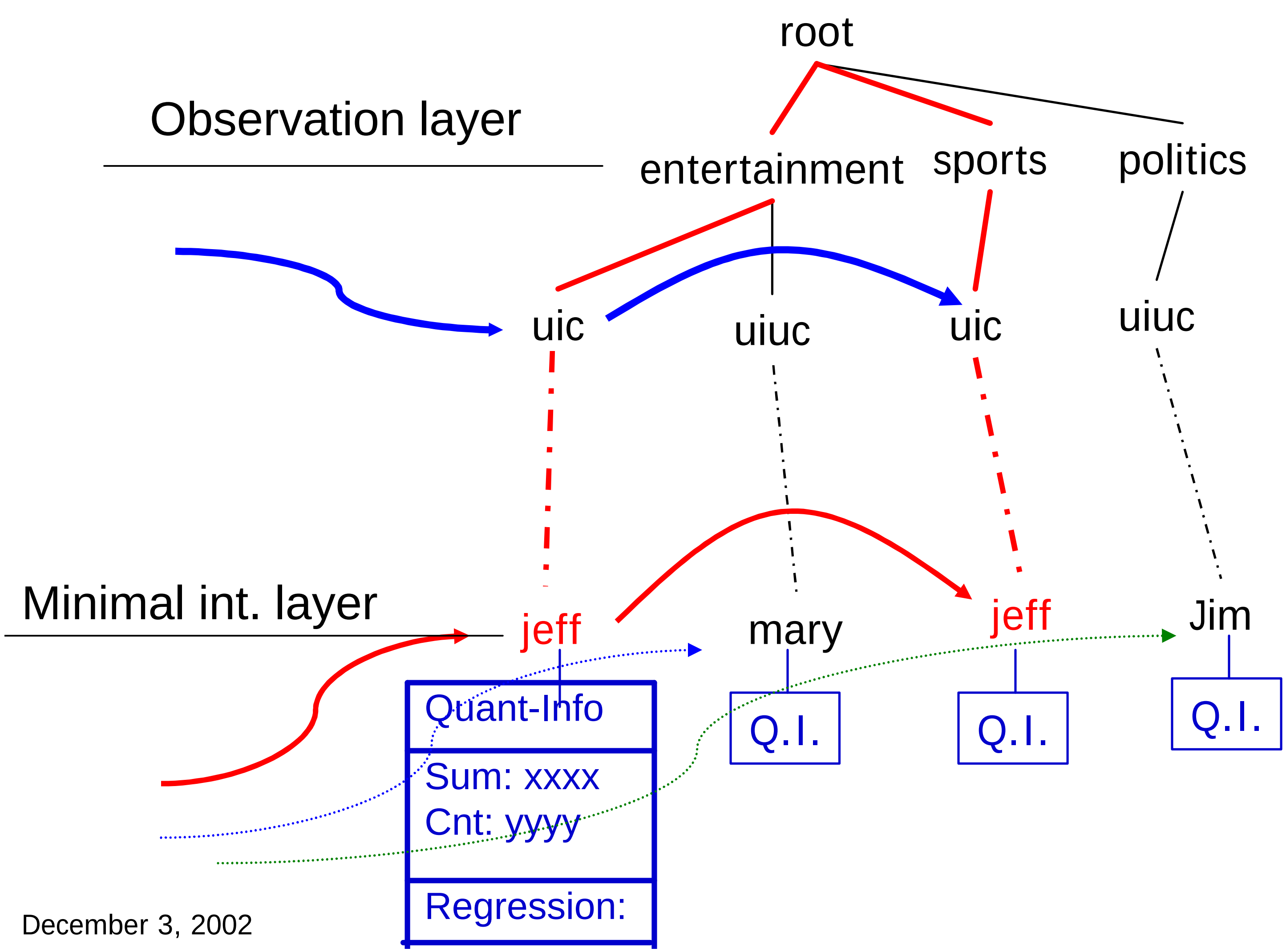
Stream Cube Structure: from m-layer to o-layer



Stream Cube Computation

- Cube structure from m-layer to o-layer
- Three approaches
  - All cuboids approach
    - Materializing all cells (too much in both space and time)
  - Exceptional cells approach
    - Materializing only exceptional cells (saves space but not time to compute and definition of exception is *not flexible*)
  - Popular path approach
    - Computing and materializing cells only along a popular path
    - Using H-tree structure to store computed cells (which form the *stream cube—a selectively materialized cube*)

An H-Tree Cubing Structure



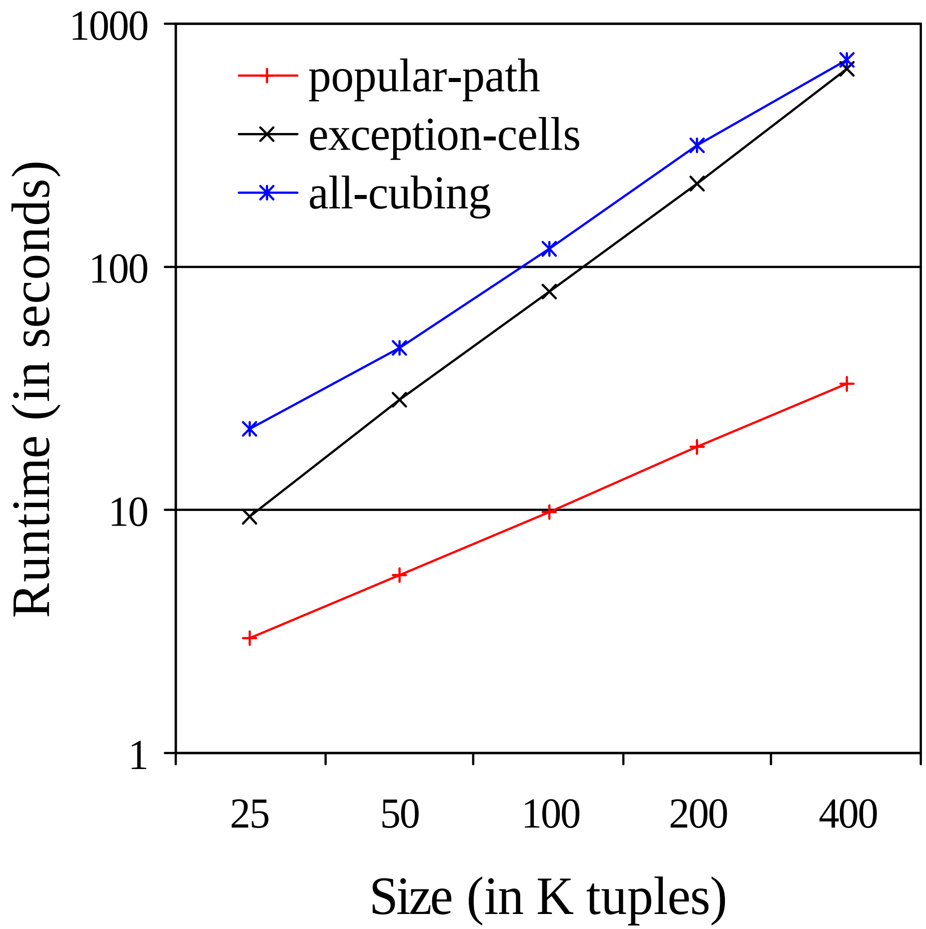
Benefits of H-Tree and H-Cubing

- H-tree and H-cubing
  - Developed for computing data cubes and ice-berg cubes
    - J. Han, J. Pei, G. Dong, and K. Wang, "*Efficient Computation of Iceberg Cubes with Complex Measures*", SIGMOD'01
  - Compressed database
  - Fast cubing
  - Space preserving in cube computation
- Using H-tree for stream cubing
  - Space preserving
    - Intermediate aggregates can be computed incrementally and saved in tree nodes
  - Facilitate computing other cells and multi-dimensional analysis
  - H-tree with computed cells can be viewed as *stream cube*

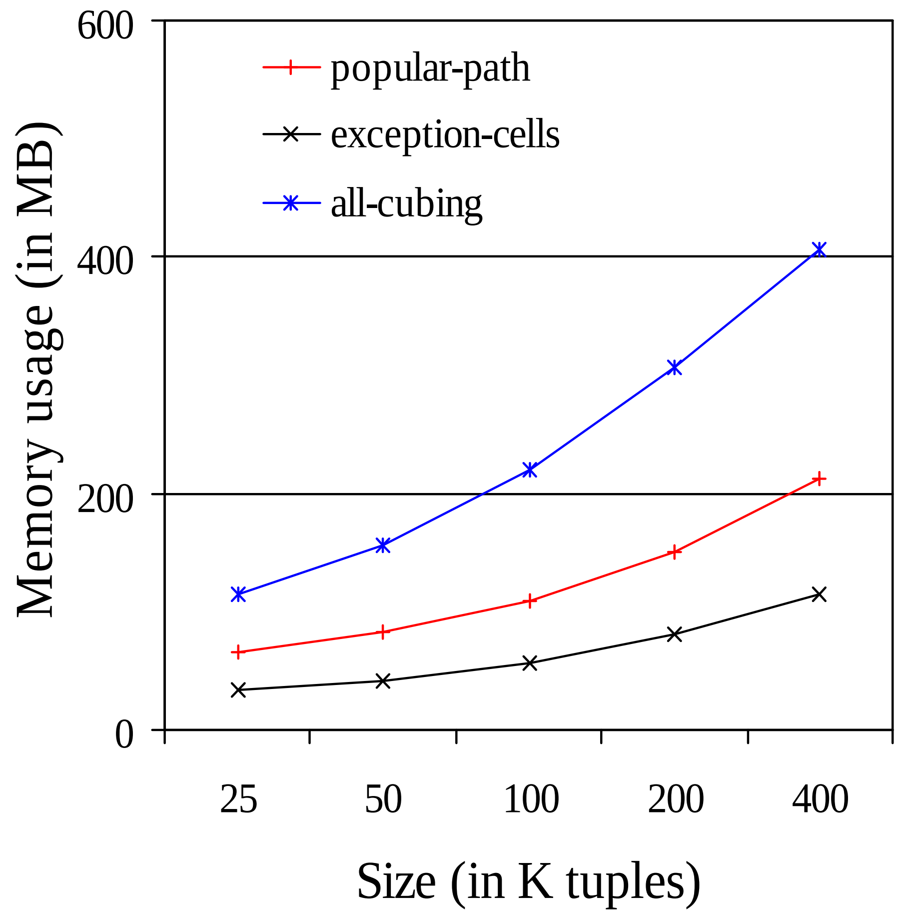
Feasibility Analysis

- Popular path
  - Computing layers along the popular path
  - Other planes/cells will be computed when requested
  - Using H-cube structure to store computed cells (which form the stream cube)
  - Tradeoff for time/space between cube materialization and online query computation
- Exception cells approach
  - How to set up an appropriate thresholds for all the applications?

Time and Space vs. Number of Tuples at the m-Layer (Dataset D3L3C10T400K)

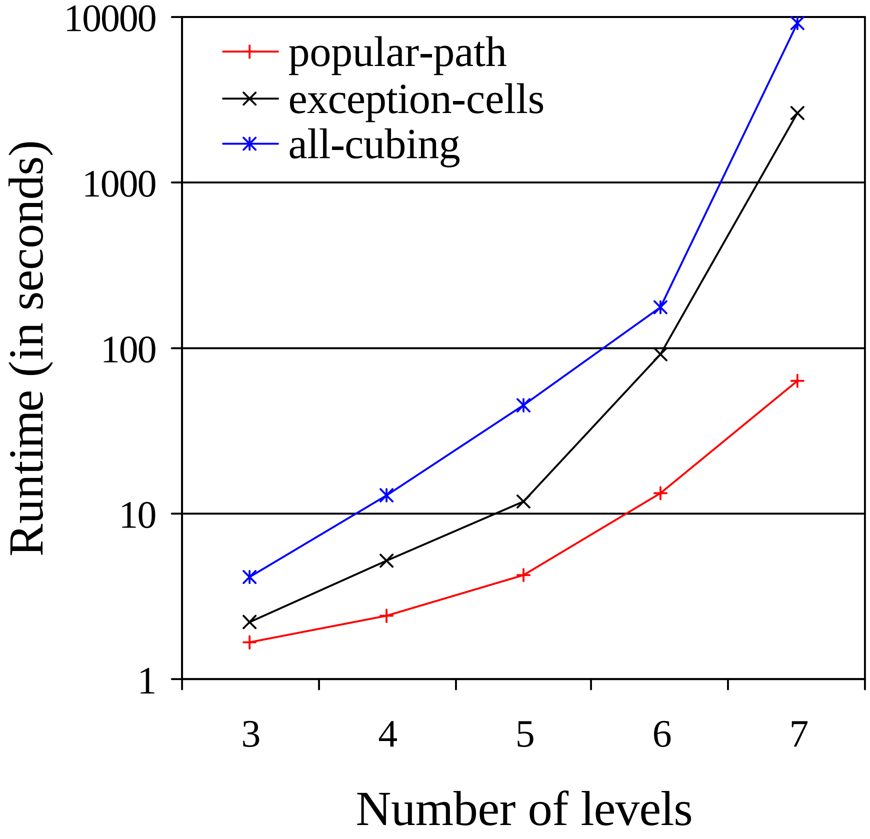


a) Time vs. m-layer size

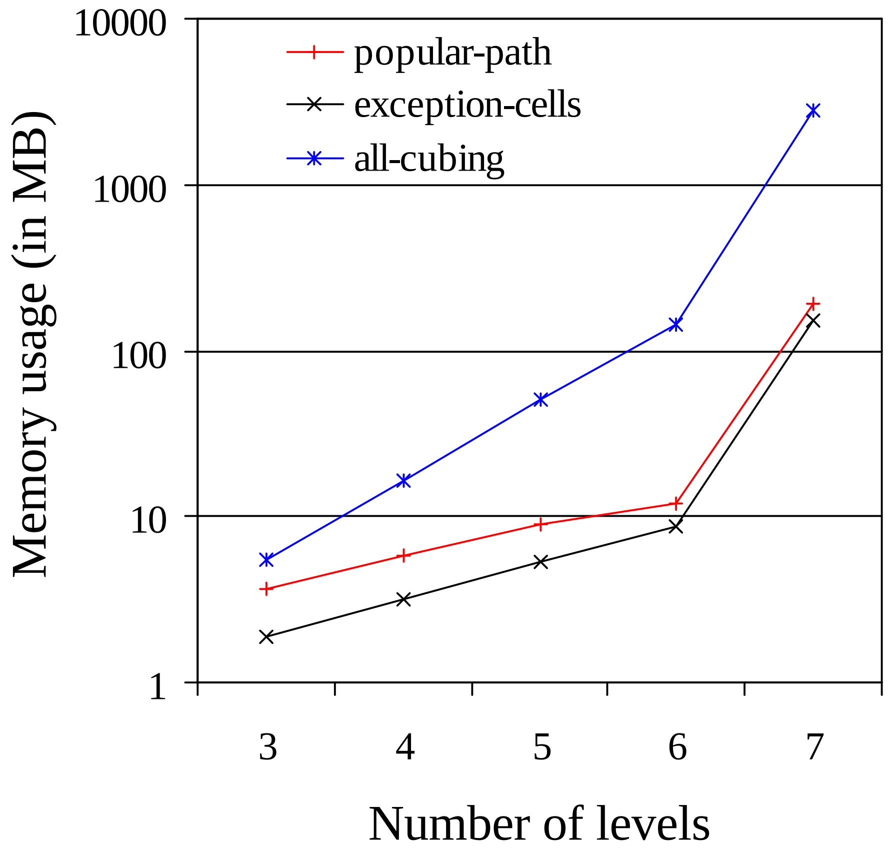


b) Space vs. m-layer size

Time and Space vs. the Number of Levels



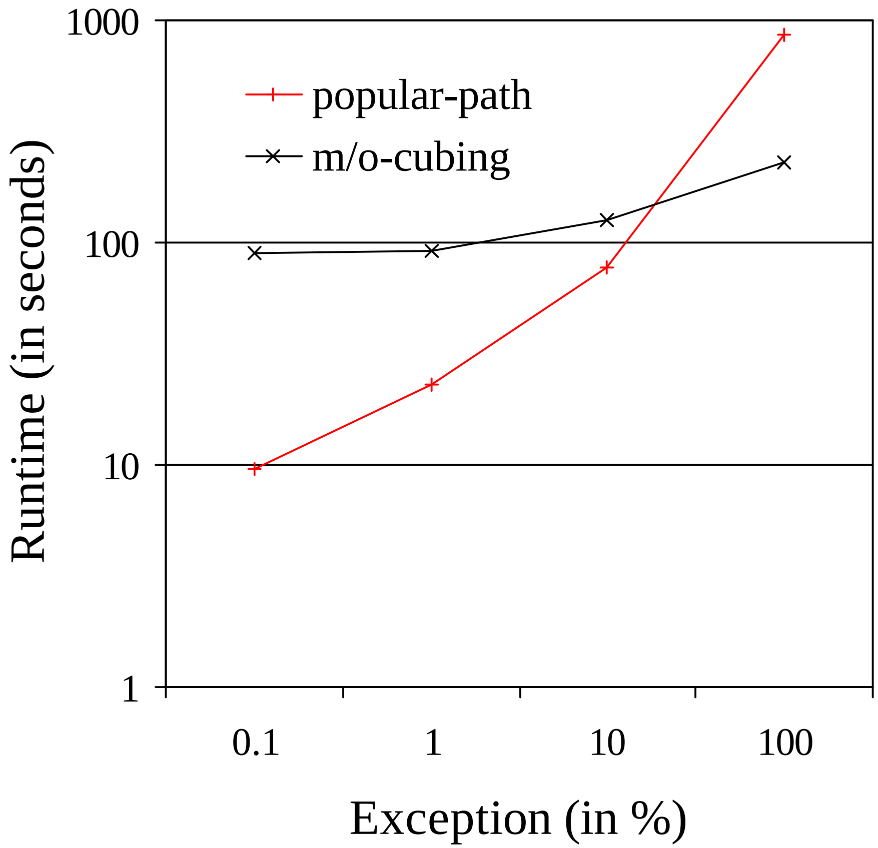
a) Time vs. # levels



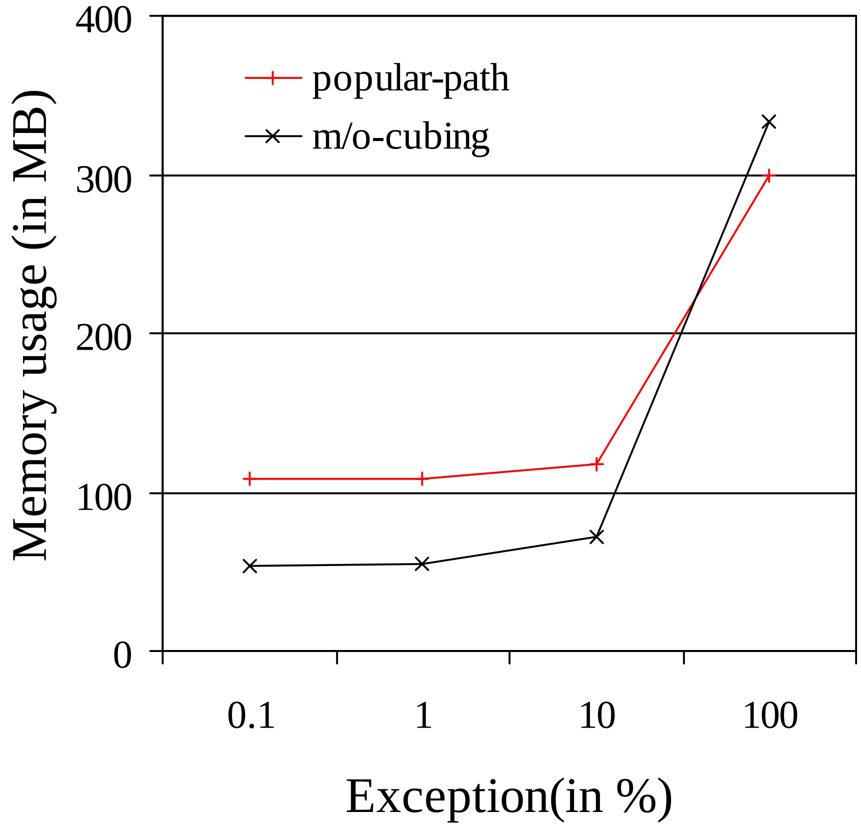
b) Space vs. # levels

Time and Space Usage vs. Percentage of Exception

(data: D3L3C10T100K)



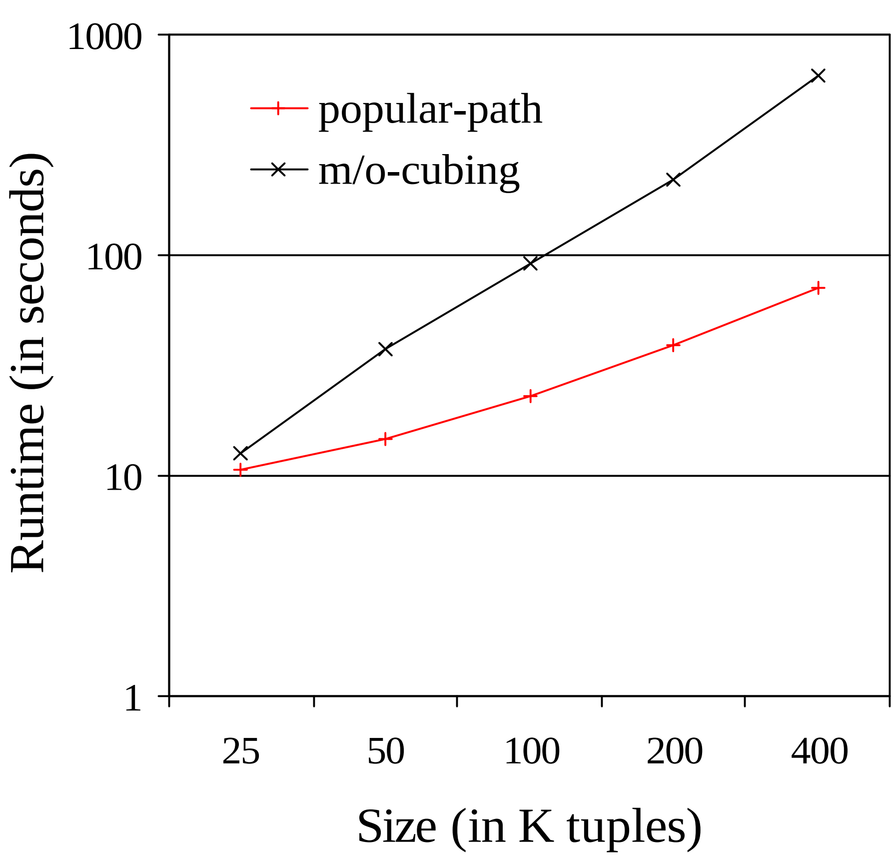
a) Time vs. exception



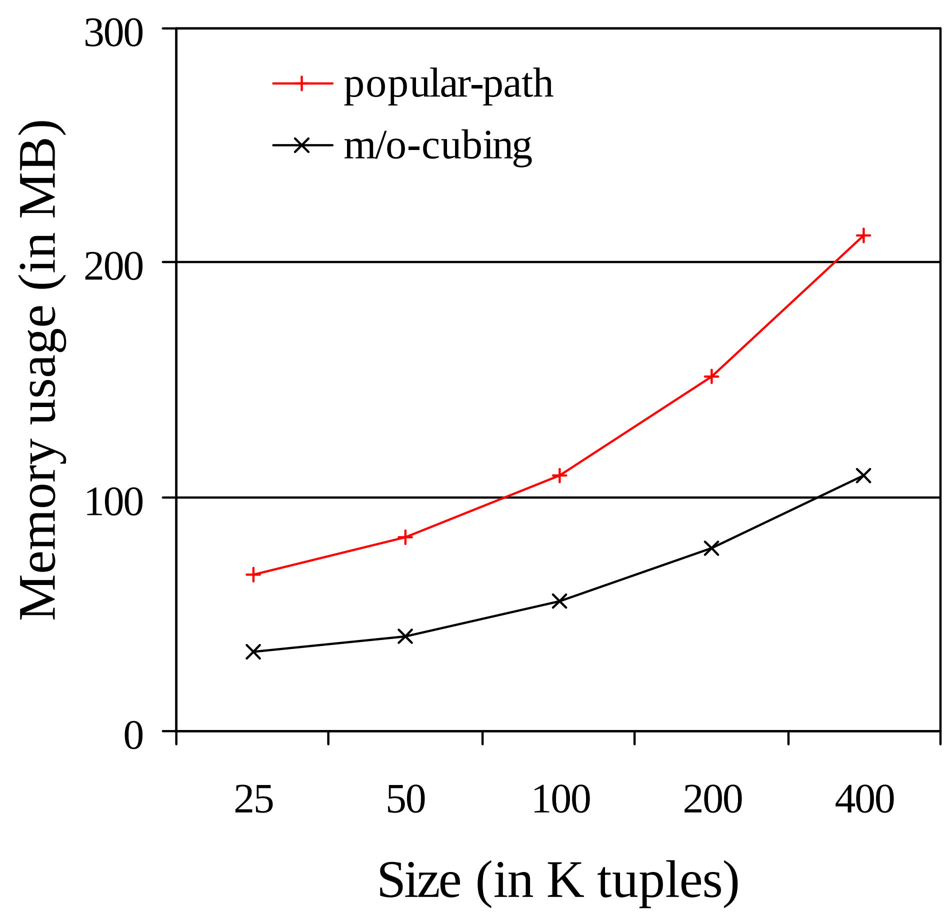
b) Space vs. exception

Time and Space Usage vs. Size of the m-Layer

(with cube structure of D3L3C10 and exception rate of 1%)



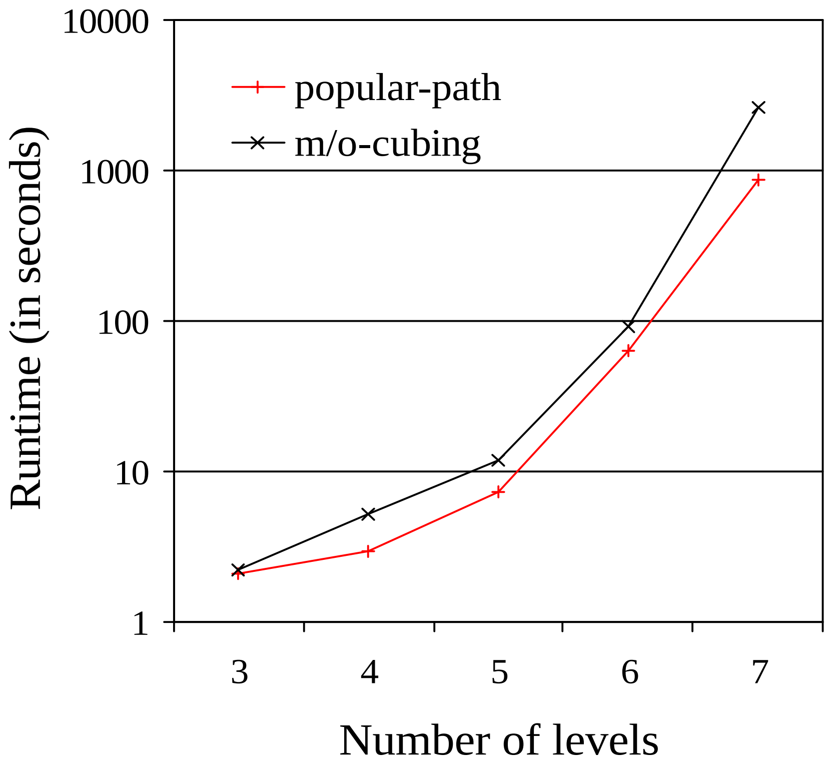
a) Time vs. m-layer size



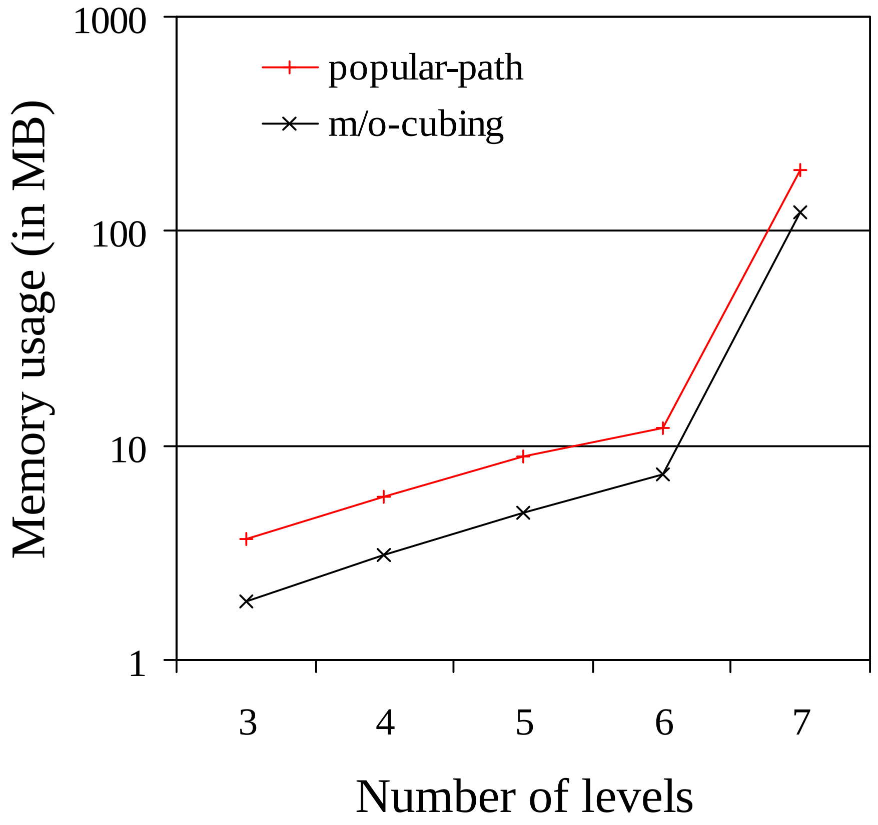
b) Space vs. m-layer size

Time and Space Usage vs. # of Levels from m- to o- Layers

(with cube structure of D2C10T10K and exception rate of 1%)



a) Time vs. # of levels



b) Space vs. # of levels

Discussion

- Important but missing link—Multi-level and multi-dimensional stream data analysis
- A multi-dimensional stream data analysis framework
  - Tilt time frame (weighted vs. uniform weights on time)
  - Critical layers
  - Popular path approach (partial materialization of stream cubes)
- Mining stream data at high-level, multiple-levels, or in multiple dimensions
  - Discovery of changes and evolutions in data streams

Conclusions

- Stream data analysis
  - Besides query and mining, stream cube and OLAP are powerful tools for finding general and unusual patterns
- A multi-dimensional stream cube framework
  - Tilt time frame
  - Critical layers
  - Popular path approach
- An important issue for further study
  - Mining stream data at high-level, multiple-levels, or in multiple dimensions