Scaling Geometric Monitoring Over Distributed Streams

Alexandros D. Keros

June 23, 2016

Supervised by: Prof. V.Samoladas

Table of contents

Data Stream Systems[1]

- **Data streams**: Continuous, high volume, size unbound, violative, probably distributed
- *Pull paradigm*
- Centralizing and/or polling $\rightarrow$ prohibitive in terms of communication overhead
- Examples: telecommunication, sensor networks

The Geometric Monitoring Method[2]

- Threshold monitoring
- Nodes communicate when needed
  - Local constraints
  - Violation resolution (*false alarms*)
- Arbitrary function monitoring
- Tight accuracy bounds
- A promising framework for *distributed data stream monitoring*

Motivation

Problems:

- increasing node population
- data volume
- data dimensionality
- arbitrary functions
- **communication - accuracy tradeoff**

Need for:

- scalability warranties
- tight accuracy bounds
- incremental/real-time operation
- **Minimize communication while retaining accuracy bounds**

Motivation

Problems:

- increasing node population
- data volume
- data dimensionality
- arbitrary functions
- **communication - accuracy tradeoff**

Need for:

- scalability warranties
- tight accuracy bounds
- incremental/real-time operation
- **Minimize communication while retaining accuracy bounds**

Motivation

Problems:

- increasing node population
- data volume
- data dimensionality
- arbitrary functions
- **communication - accuracy tradeoff**

Need for:

- scalability warranties
- tight accuracy bounds
- incremental/real-time operation
- **Minimize communication while retaining accuracy bounds**

Motivation

Problems:

- increasing node population
- data volume
- data dimensionality
- arbitrary functions
- **communication - accuracy tradeoff**

Need for:

- scalability warranties
- tight accuracy bounds
- incremental/real-time operation
- **Minimize communication while retaining accuracy bounds**

Contributions

Expand the *geometric monitoring method*:

- heuristic method for violation resolution
- distance-based hierarchical node clustering[3]
- throughout method evaluation on synthetic and real-world datasets

Contributions

Expand the *geometric monitoring method*:

- heuristic method for violation resolution
- distance-based hierarchical node clustering[3]
- throughout method evaluation on synthetic and real-world datasets

Contributions

Expand the *geometric monitoring method*:

- heuristic method for violation resolution
- distance-based hierarchical node clustering[3]
- throughout method evaluation on synthetic and real-world datasets

Geometric Threshold Monitoring

- Izchak Sharfman, Assaf Schuster, and Daniel Keren (2006). "A Geometric Approach to Monitoring Threshold Functions over Distributed Data Streams". In: *2006 ACM SIGMOD ICMD*. SIGMOD '06
- **Threshold monitoring**: arbitrary function $f(\cdot)$, threshold T

$$f(\cdot) < T \text{ or } f(\cdot) > T$$

- **Idea**: decompose into local constraints at the nodes

System Architecture

Centralized Scenario

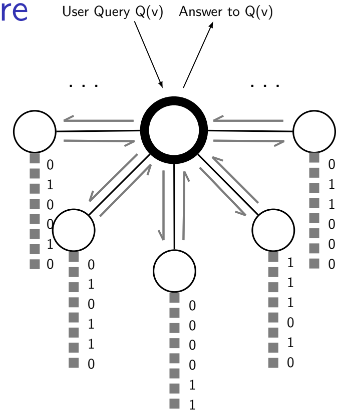


Figure: **Star-like network topology** example of the centralized scenario.The bold node represents the coordinator node. Dashed lines represent data streams and half arrows represent message exchanges.

Computational Model

Statistics vectors

- the *monitoring function* $f : \mathbb{R}^d \to \mathbb{R}$
- the *threshold* $T \in \mathbb{R}$
- the *monitoring node set* : $P = \{p_1, \ldots, p_n\}$
  with *weights* $w_1, \ldots, w_n$
- the *data streams* : $S = \{s_1, \ldots, s_n\}$
- the d-dimensional *local statistics vectors* : $\vec{v_1}(t), \ldots, \vec{v_n}(t)$
  represent each node's data stream at time t

Global statistics vector

$$\vec{v}(t) = \frac{\sum_{i=1}^{n} w_i \vec{v_i}(t)}{\sum_{i=1}^{n} w_i} \tag{1}$$

Computational Model

Estimate vector

Infrequent communication between nodes/nodes-coordinator:

Estimate vector

$$\vec{e}(t) = \frac{\sum_{i=1}^{n} w_i \vec{v_i}'}{\sum_{i=1}^{n} w_i} \tag{2}$$

- the last communicated *local statistics vector* of node p_i : $\vec{v_i}'$
- *Local statistics* divergence: $\Delta\vec{v_i}(t) = \vec{v_i}(t) - \vec{v_i}', i = 1, \ldots, n$

Centralized drift vector

$$\vec{u_i}(t) = \vec{e}(t) + \Delta\vec{v_i}(t) + \frac{\vec{\delta_i}}{w_i} \tag{3}$$

Computational Model

Balancing Process

Centralized scenario

Purpose: resolve possible false alarms

Balancing vector

$$\vec{b} = \frac{\sum_{p_i \in P'} w_i \vec{u_i}(t)}{\sum_{p_i \in P'} w_i} \tag{4}$$

- the *balancing set* P': a subset of nodes
- the *slack vector* at the nodes $\vec{\delta_i} = \vec{\delta_i}' + \Delta\vec{\delta_i}$, $\sum_{p_i \in P'} \Delta\vec{\delta_i} = \vec{0}$:

$$\Delta\vec{\delta_i} = w_i \vec{b} - w_i \vec{u_i}(t) \ \forall\ p_i \in P' \tag{5}$$

, readjusts the *drift vectors* (3).

Geometric Interpretation

Convexity Property

Convexity Property

$$\vec{v}(t) = \frac{\sum_{i=1}^{n} w_i \vec{u_i}(t)}{\sum_{i=1}^{n} w_i} \tag{6}$$

Theorem ([a])

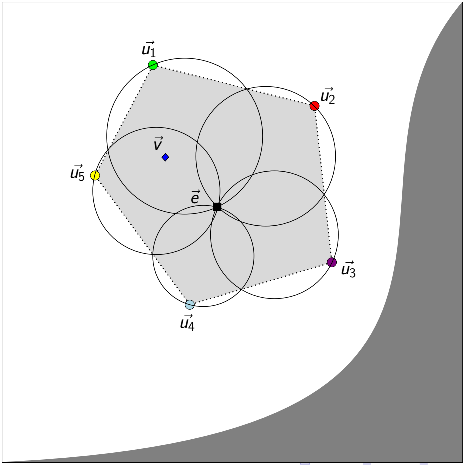
[a] Izchak Sharfman, Assaf Schuster, and Daniel Keren (2006). "A Geometric Approach to Monitoring Threshold Functions over Distributed Data Streams". In: *2006 ACM SIGMOD ICMD*. SIGMOD '06.

Let $\vec{x}, \vec{y_1}, \ldots, \vec{y_n} \in \mathbb{R}^d$ be a set of vectors in $\mathbb{R}^d$. Let $Conv(\vec{x}, \vec{y_1}, \ldots, \vec{y_n})$ be the convex hull of $\vec{x}, \vec{y_1}, \ldots, \vec{y_n}$. Let $B(\vec{x}, \vec{y_i})$ be a ball centered at $\frac{\vec{x}+\vec{y_i}}{2}$ and with radius of $\|\frac{\vec{x}+\vec{y_i}}{2}\|_2$ i.e., $B(\vec{x}, \vec{y_i}) = \{\vec{z} \mid \|\vec{z} - \frac{\vec{x}+\vec{y_i}}{2}\|_2 \leq \|\frac{\vec{x}+\vec{y_i}}{2}\|_2\}$, then

Geometric Interpretation

Convexity Property & Local Constraints

Figure: Example of a convex hull (light gray) defined by the drift vectors $\vec{u_i}, i = 1, 2, 3, 4, 5$. The hull is bounded by the spheres created from the estimate vector $\vec{e}$ and the drift vectors $\vec{u_i}, i = 1, 2, 3, 4, 5$. The global statistics vector $\vec{v}$ is guaranteed to be contained in the convex hull of the drift vectors.



Protocol

Centralized Algorithm

Algorithm 1: Centralized algorithm's coordinator node operation

```
begin
    Wait for $< INIT, \cdot >$ messages from all monitoring nodes;
    /* Initialization */
    Compute estimate vector $\vec{e}(0)$;
    if new $< REP, \vec{v_i}(t), \vec{u_i}(t) >$ message received then
    /* Monitoring operation */
        $P' = P' \cup \{< i, \vec{v_i}(t), \vec{u_i}(t) >\}$;
        Balance($P'$);
    end
end
Function Balance($P'$)                /* Balancing Process */
    Compute balancing vector $\vec{b}$;
    if $B(\vec{e}, \vec{b})$ is not monochromatic then
        if $P - P' \neq \emptyset$ then
            Send $< REQ >$ message to random node in $P - P'$ set;
        else
            Compute estimate vector $\vec{e}(t)$;
            Send $< NEW\text{-}EST, \vec{e}(t) >$ message to all nodes;
            return;
        end
    else
        foreach $p_i \in P'$ do
            Compute slack adjustment vector $\Delta\vec{\delta_i}$;
            Send $< ADJ\text{-}SLK, \Delta\vec{\delta_i} >$ message to node $p_i$;
            return;
        end
    end
end
```

Algorithm 2: Centralized algorithm's monitoring node operation

```
begin
    foreach node $p_i$ do                /* Node initialization */
        Send $< INIT, \vec{v_i}(0) >$ message to coordinator;
        $\vec{v_i}' = \vec{v_i}(0)$;
        $\vec{\delta_i} = \vec{0}$;
        Wait message from coordinator;
        if $< NEW\text{-}EST, \vec{e} >$ message received then
            Set $\vec{e}(t) = \vec{e}$;
        end
    end
    foreach node $p_i$ do                /* Main monitoring task */
        foreach new $s_i$ stream update $\vec{v_i}(t)$ do
            Recalculate drift vector $\vec{u_i}(t)$;
            if $B(\vec{e}, \vec{u_i}(t))$ is not monochromatic then
                Send $< REP, \vec{v_i}(t), \vec{u_i}(t) >$ message to coordinator;
                Wait for $< NEW\text{-}EST, \cdot >$ or $< ADJ\text{-}SLK, \cdot >$
                message from coordinator;
            end
            if new message $< REQ >$ received then
                Send $< REP, \vec{v_i}(t), \vec{u_i}(t) >$ message to coordinator;
                Wait for $< NEW\text{-}EST, \cdot >$ or $< ADJ\text{-}SLK, \cdot >$
                message from coordinator;
            end
            if new $< NEW\text{-}EST, \vec{e} >$ message received then
                Set $\vec{e}(t) = \vec{e}$;
                $\vec{v_i}' = \vec{v_i}(t)$;
                $\vec{\delta_i} = \vec{0}$;
            end
            if new $< ADJ\text{-}SLK, \Delta\vec{\delta_i} >$ message received then
                Recompute delta vector $\vec{\delta_i}$;
            end
        end
    end
end
```

Multi-objective Optimization

- **Multiple**, possibly **conflicting** objectives to be *simultaneously* optimized
- *Pareto optimality*(*non-dominated solutions*): optimal solutions where none of the objective functions can be optimized without the simultaneous degradation of other objective functions' values.
- Let vector of m objectives $F(x) = [F_1(x), F_2(x), \ldots, F_m(x)]$:

$$\begin{aligned} &\min_{x \in \mathbb{R}^n} F(x) \\ &\text{s.t. } l \leq x \leq u \\ &\qquad G_i = 0, i = 1, \ldots, k_e \\ &\qquad G_j \leq 0, j = k_e + 1, \ldots, k \end{aligned}$$

- Finding *Pareto optimal* solutions is generally **NP-hard**.

Non-linear Constraint Optimization

Primal Descent

Algorithm 3: Generic primal descent

begin
    Choose initial point $x_0 \in X$ and set $t = 0$; `/* Initialization */`
    while *maximum iteration limit OR convergence* **do** `/* Search */`
        $t = t + 1$;
        Determine search direction d_t;
        Determine step length s_t, so that $f(x_t + s_t d_t) < f(x_t)$;
        Update;
    end
end

SQP

- The *Lagrangian function*: $\mathcal{L}(x, \lambda) = F(x) + \sum_{i=1}^{k} \lambda_i G_i(x)$
- *Quadratic programming subproblems*:

$$\min_{d \in \mathbb{R}^n} \frac{1}{2} d^T H_t d + \nabla F(x_t)^T d$$
$$\nabla G_i(x_t)^T d + G_i(x_t) = 0, i = 1, \ldots, k_e$$
$$\nabla G_i(x_t)^T d + G_i(x_t) \leq 0, i = k_e + 1, \ldots k$$

,where:

H_t:Hessian of the Lagrangian function at iteration t

d:search direction

The Savitzky-Golay Filter

- Low-pass smoothing filter
- *Moving window averaging* paradigm: $g_i = \sum_{n=-n_L}^{n_R} c_n f_{i+n}$
- *Least-squares fit* of polynomial $y_i(x)$ over window $n_L + n_R + 1$:

$$\sum_{j=i-n_L}^{i+n_R} (y_i(x_j) - f_j)^2 = \min$$

,where:

$$y_i(x) = a_0 + a_1 \frac{x - x_i}{\Delta x} + a_2 (\frac{x - x_i}{\Delta x})^2 + \cdots + a_M (\frac{x - x_i}{\Delta x})^M$$

- Set g_i to the value of the fitted point x_i

Maximum Weight Matching

Let $G = (V, E)$ a graph:

- *maximum weight matching* $M \subseteq E$: a subset of edges where
  - no two edges share a common vertex
  - largest possible number of edges
  - maximizes the sum of weights

Maximum Weight Matching

The Primal-Dual Method

Primal-Dual Method

$$\text{Constraints in Primal} \Longleftrightarrow \text{Variables in Dual}$$
$$\text{Constraints in Dual} \Longleftrightarrow \text{Variables in Primal}$$

The primal:

$$\begin{aligned} \max \quad & \sum_{(u,v)\in E} x_{u,v} w_{u,v} & \\ \text{s.t.} \quad & x_{u,v} \geq 0 & , (u,v) \in E \\ & \sum_{u\in e : e\in E} x_e \leq 1 & , u \in V \end{aligned}$$

The dual:

$$\begin{aligned} \min \quad & \sum_{u\in V} y_u & \\ \text{s.t.} \quad & y_u \geq 0 & , u \in V \\ & y_u + y_v \geq w_{u,v} & , (u,v) \in E \end{aligned}$$

Related Work

- *Safe Zones*: optimal local constraints fitted to nodes' data distributions[4]
- Ellipsoidal bounding regions, decouplement of estimate vector from bounding ball construction[5]
- Simple shapes as local constraints, hierarchical clustering of nodes for participation to the balancing operation[6]
- Prediction models based on velocity and acceleration[7]

Problem Formulation

Reduce the communication burden of the *Geometric Monitoring* method by:

- Optimally position *drift vectors* during the *balancing process*
- Appropriate node selection for inclusion in the *balancing set*

,in order to **increase scalability** in terms of:

- node popullation
- stream dimensionality.

The Geometric Monitoring Framework

Assumptions

- Coordinator-based scenarion
- Instantaneous, loss-less, reliable communication
- Iterative operation based on time-steps
- System pause during violation resolution
- Coordinator node does not monitor a stream

Algorithm 4: Iterative network operation

Data: *monitoringNodes*: a list of Monitoring nodes, *coordinator*: the Coordinator node

```
begin
    initialization;
    repeat
        foreach node ∈ monitoringNodes do
            node.DataVectorUpdate();
            node.ComputeDriftVector();
        end
        foreach node ∈ monitoringNodes do
            node.CheckForViolation();
            if localViolation then
                node.Report();
                coordinator.Balance();
            end
        end
    until globalViolation;
end
```

The Distance-based Hierarchical Clustering

The Idea

Node selection for *violation resolution*:

- Elevate *randomness* of the initial *geometric monitoring* method[8]
- Hierarchical node clustering scheme[9]
- Decouple matching from the data distribution at the nodes[10]
- Accurately follow *global statistics vector*
- Node "cancel each other out" during the *balancing process*

The Distance-based Hierarchical Clustering

The Weight Function

Weight function

$$w_{i,j} = \sum_{t=t_0}^{t_{end}} [(f(\vec{v}_{global}(t)) - f(\frac{\vec{v}_i(t) + \vec{v}_j(t)}{2})) + (|\vec{v}_i(t) - \vec{v}_j(t)|)]$$

The Distance-based Hierarchical Clustering

The Algorithm

Algorithm 5: Recursively create Monitoring node pairs and hierarchy

Function `DistancePairer`($nodes$, i)

  Data: $nodes = [(n_1, [\vec{v_1}(t_0), ..., \vec{v_1}(t_{end})]), ..., (n_k, [\vec{v_k}(t_0), ..., \vec{v_k}(t_{end})])]$: list of nodes with their respective data vectors, i: pair type, initial=1

  Result: $nodeHierarchy$: dictionary of *Type-k* pairs

  if $length(nodes) = 1$ **then** `// recursion stopping condition`

    return $nodeHierarchy$;

  end

  $g = CreateCompleteGraph(nodes)$; `// complete graph with` *nodes* `as vertices`

  foreach $(n_i, n_j) \in g.Edges()$ **do** `// assign weights to edges`

    $w_{i,j} = \sum_{t=t_0}^{t_{end}} [(f(\vec{v}_{global}(t)) - f(\frac{\vec{v_i}(t)+\vec{v_j}(t)}{2})) + (|\vec{v_i}(t) - \vec{v_j}(t)|)]$;

    $g.edge(n_i, n_j).weight = w_{i,j}$;

  end

  $nodeHierarchy$(Type-i) $= g.maximalWeightMatching()$; `// node pairs of` *Type-i*

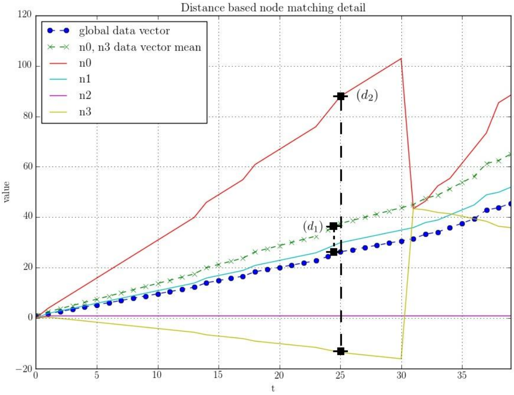
  `DistancePairer`($nodeHierarchy(Type\text{-}i), i * 2$);

end

The Distance-based Hierarchical Clustering

Example

Figure: Distance based node matching operating on 4 nodes ($\{n_0, n_1, n_2, n_3\}$). Distance d_1 : the distance of the data vector mean of the paired nodes n_0 and n_3 from the global mean (*global data vector*), distance d_2 : denotes the in-between distance of data vectors $\vec{v_0}(t)$ and $\vec{v_3}(t)$ of the node pair. Both distances are taking part in the edge weighting process.



The Heuristic Balancing

The Idea

In prior work:

- identical handling of nodes during the balancing process
- ignore stream idiosyncrasies and monitoring function peculiarities

The *heuristic balancing* method:

- optimally position *drift vectors in space*
- take into account stream behaviour (*velocity* and *acceleration*)

The Heuristic Balancing

The Optimizing Function

Weight function

$$\max \min \frac{(T - x_i) - accel_i(t_{lv}) * t^2}{vel_i(t_{lv})}, \forall n_i \in P'$$

where:

t : the variable to optimize

T : monitoring threshold

x_i : the maximum value of the monitoring function $f(\cdot)$ over the bounding ball $B(\vec{e}(t_{lv}), \vec{u_i}(t_{lv}))$

$vel_i(t_{lv})$: the estimated velocity of the maximum value of the monitoring function $f(\cdot)$

$accel_i(t_{lv})$: the estimated acceleration of the maximum value of the monitoring function $f(\cdot)$

t_{lv} : time of Local Violation occurrence

P' : the balancing set

The Heuristic Balancing

The Algorithm

Algorithm 6: Heuristic Balancing

Function `RepMessageReceived`($< n_i, v_i, u_i, vel_i, accel_i >$)
    add n_i to balancing set P';
    `Balance();`
end
Function `Balance`(P')
    if $length(P') = 1$ **then**
        `RequestNode();` `// request node based on respective gathering scheme`
    end
    $\vec{b} = \sum_{P'} \frac{w_i * \vec{u_i}}{w_i}$;
    if $B(\vec{e}, \vec{b})$ *is monochromatic* **then**
        `/* heuristic optimization procedure, */`
        `/* returns the optimal drift vector positions in set` O `*/`
        $O = DriftVectorOptimizationProblem()$;
        foreach $n_i \in P'$ **do**
            $\Delta\delta_i = w_i * \vec{u_i}' - w_i * \vec{u_i}$; `//` $\vec{u_i}'$ `denotes the optimal drift vector position`
            $Send(< ADJSLK, n_i, \Delta\delta_i >)$;
        end
    end
end

The Heuristic Balancing

Example

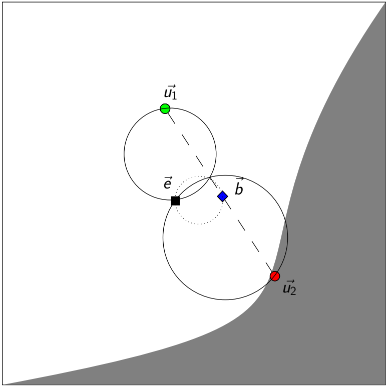


Figure: The classic balancing method.

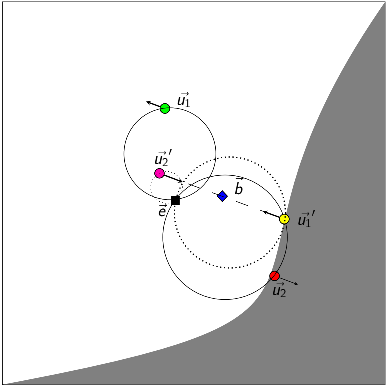


Figure: The heuristic balancing method.

Velocity and Acceleration Estimation via SG Filtering

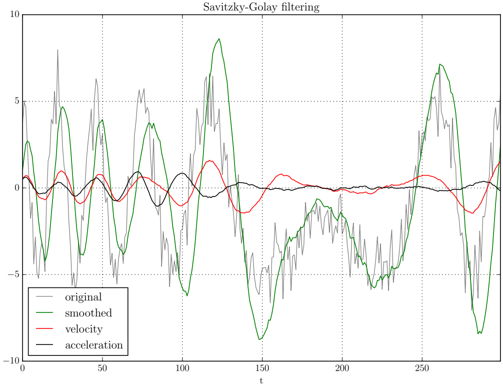


Figure: Savitzky-Golay filtering of a signal with added Gaussian noise. The smoothing window is 50 points, centered at the far right.

Acronyms for Implemented Methods

GM: the initial geometric monitoring method[11]

HM: the proposed heuristic balancing process

DIST: the proposed distance-based hierarchical node clustering

DISTR: the distribution-based hierarchical node clustering[12]

Synthetic Data

- $v_i(t_{k+1}) = v_i(t_k) + (1-\lambda)u_k + \lambda u_{k+1}$
- 1-dimensional
- Velocities sampled from user-specified Gaussian distribution
- λ smoothing parameter
- Additive Gaussian noise
- 3 sets of datasets: *LIN*, *INT*, *NOISE*

Synthetic Data

Examples

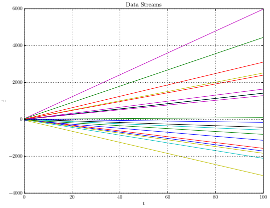


Figure: *LIN* local statistics streams of 20 nodes

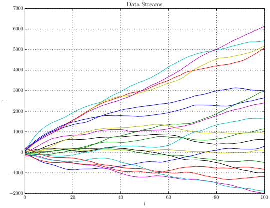


Figure: *INT* local statistics streams of 20 nodes

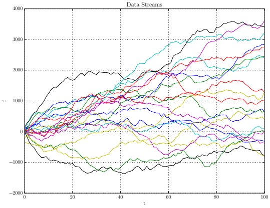


Figure: *NOISE* local statistics streams of 20 nodes

Real-world Data

- "European Environmental Agency - AQ e-Reporting" database[13]
- Hourly measurements of NO_2 and NO, in micro-grams per cubic meter, averaged over a window of five days for a whole year.
- Nodes correspond to randomly selected air quality measurement stations across Austria.

Real-world Data

Examples

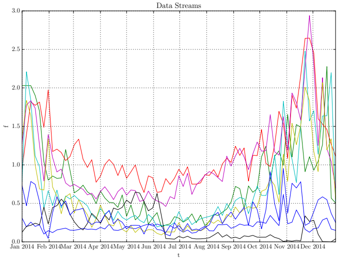


Figure: Streams of 8 nodes monitoring the ratio NO/NO_2.

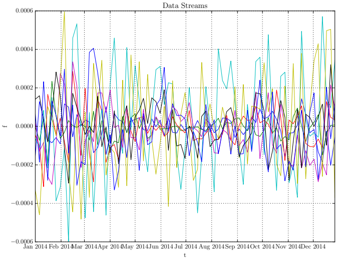


Figure: Streams of 8 nodes monitoring the variance of NO_2 air pollutant.

GM, DIST, DISTR Comparison

LIN dataset

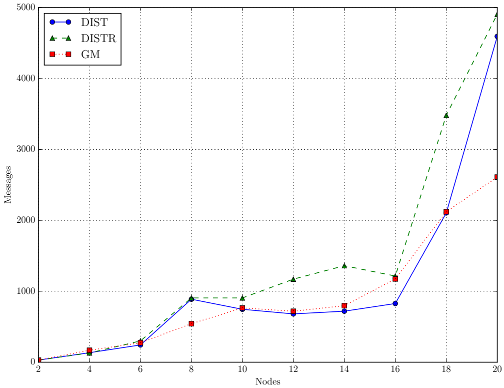


Figure: Communication costs of methods GM, DISTR and DIST for the *LIN* dataset.

GM, DIST, DISTR Comparison

INT dataset

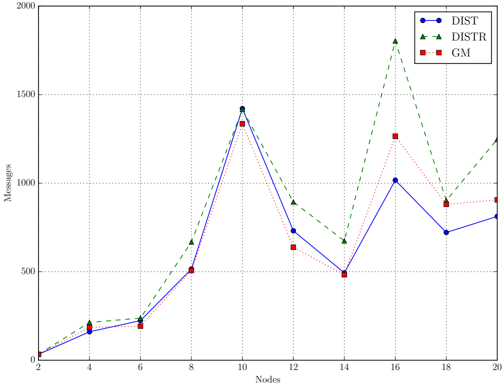


Figure: Communication costs of methods GM, DISTR and DIST for the *INT* dataset.

GM, DIST, DISTR Comparison

NOISE dataset

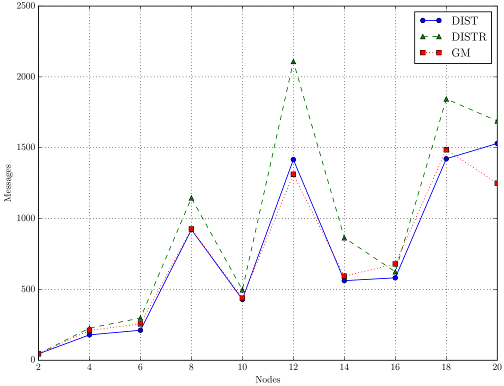


Figure: Communication costs of methods GM, DISTR and DIST for the *NOISE* dataset.

GM, DIST, DISTR Comparison

LIN dataset

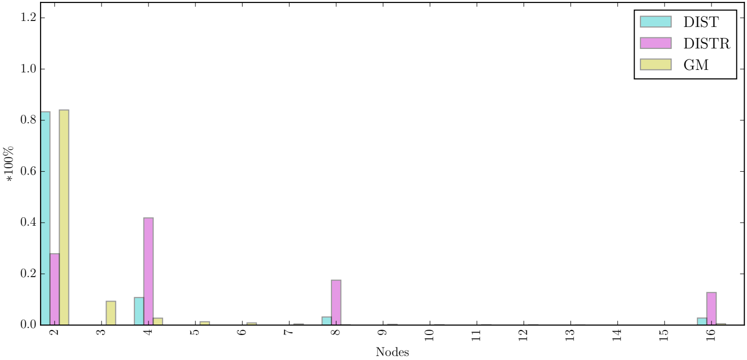


Figure: Number of nodes participating in violation resolutions as a fraction of total Local Violations, for the *LIN* dataset.

GM, DIST, DISTR Comparison

INT dataset

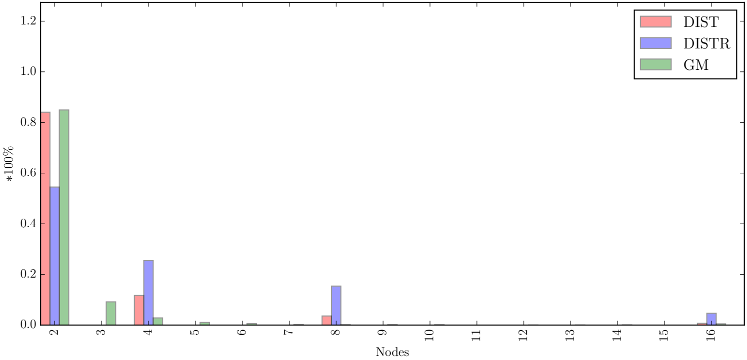


Figure: Number of nodes participating in violation resolutions as a fraction of total Local Violations, for the *INT* dataset.

GM, DIST, DISTR Comparison

NOISE dataset

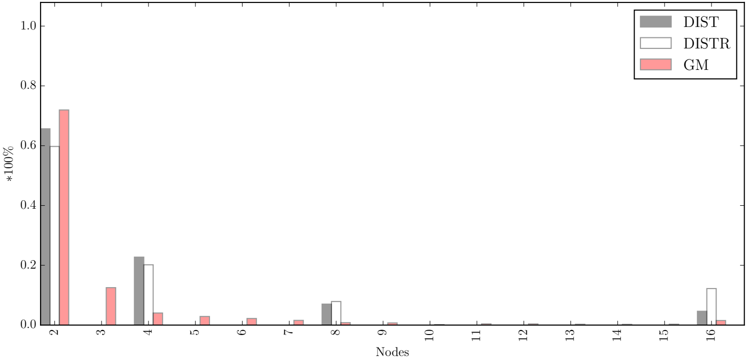


Figure: Number of nodes participating in violation resolutions as a fraction of total Local Violations, for the *NOISE* dataset.

GM, HM Comparison

Messages

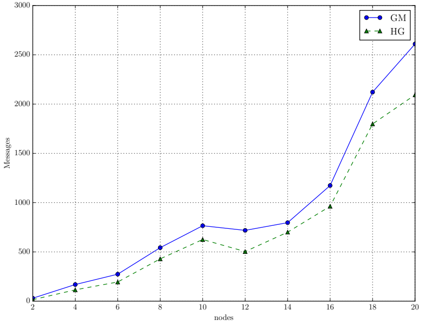


Figure: Communication cost of methods GM and HM for the *LIN* dataset.

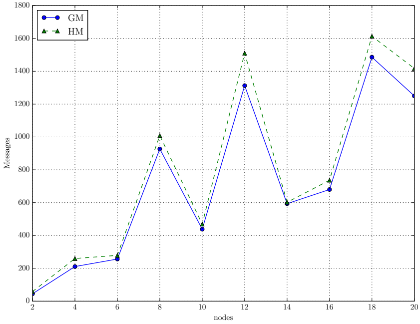


Figure: Communication cost of methods GM and HM for the *NOISE* dataset.

GM, HM Comparison

Drift vectors

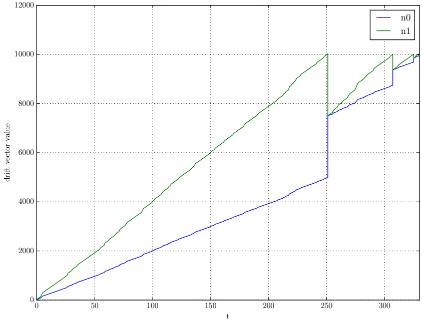


Figure: Drift vectors of 2 nodes, as formulated by the GM algorithm.

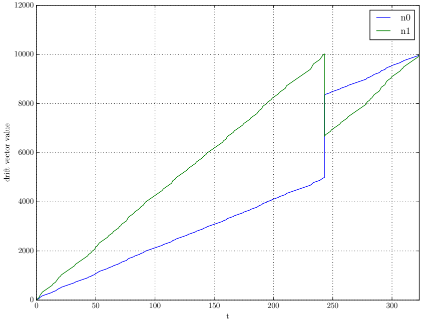


Figure: Drift vectors of 2 nodes, as formulated by the HM algorithm.

GM, HDM Comparison

LIN

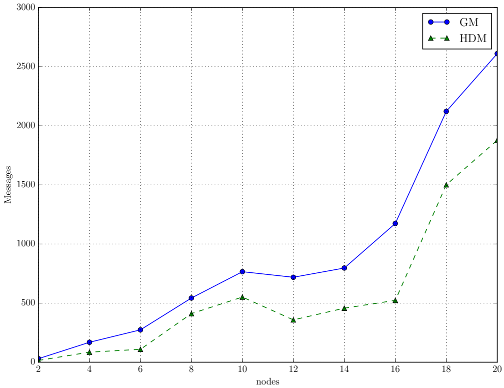


Figure: Communication cost of methods GM and HDM for the *LIN* dataset.

GM, HDM Comparison

INT

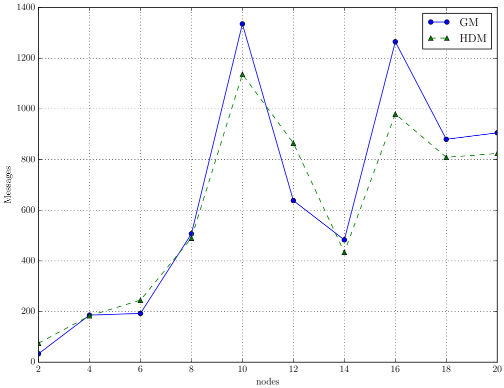


Figure: Communication cost of methods GM and HDM for the *INT* dataset.

GM, HDM Comparison

NOISE

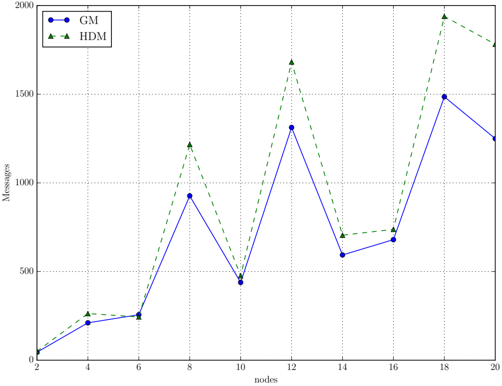


Figure: Communication cost of methods GM and HDM for the *NOISE* dataset.

GM, HDM Comparison

NOISE - window

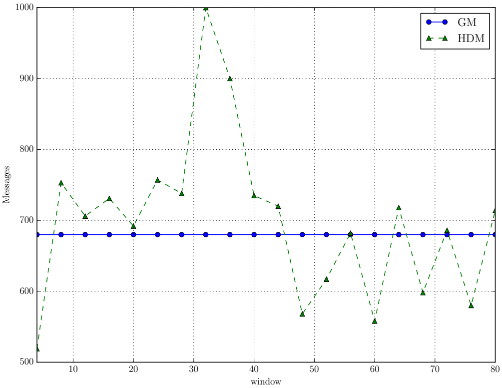


Figure: Communication cost of methods GM and HDM for the *NOISE* dataset. Approximation order is set to 1.

GM, HDM Comparison

NOISE - order

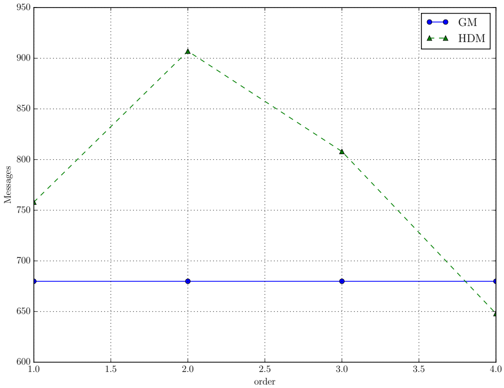


Figure: Communication cost of methods GM and HDM for the *NOISE* dataset. The Savitzky-Golay window size is set to 24.

GM, HDM Comparison

Dimensions - Quadratic Function

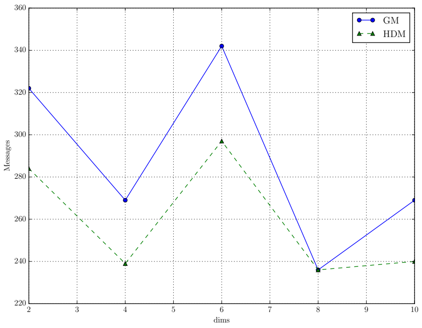


Figure: Communication cost of methods GM and HDM for the *LIN* dataset.

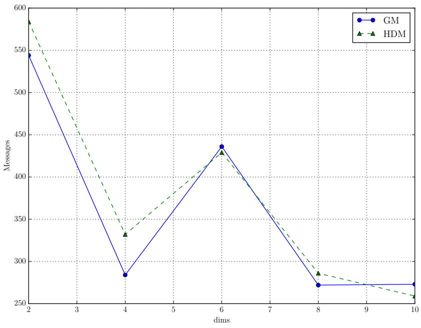


Figure: Communication cost of methods GM and HM for the *NOISE* dataset.

GM, HDM Comparison

Air Pollution Monitoring

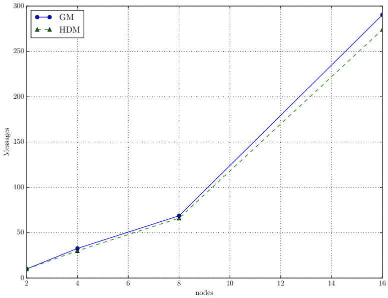


Figure: 4 to 16 nodes,variance monitoring of NO_2. The Savitzky-Golay window size is set to 6, the order is set to 2.

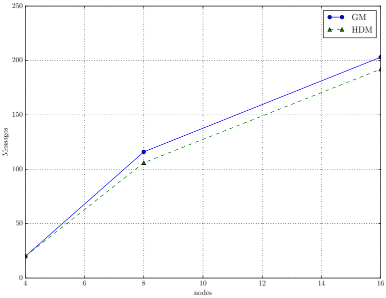


Figure: 4 to 16 nodes, when monitoring NO/NO_2. The Savitzky-Golay window size is set to 10, the order is set to 1.

Summary & Concluding Remarks

The *Geometric monitoring* method:

- An efficient framework for monitoring distributed data streams
- Scalability can be improved $\rightarrow$ reduce communication costs

Our contributions:

- Distance-based hierarchical node clustering
- Heuristic balancing method based on SQP and Savitzky-Golay Filtering
- Detailed evaluation of proposed methods

Comments:

+ Communication reduction of up to 60%
+ Methods fully compatible with the rest of the work
- Parameter tweaking for satisfactory results
- Multi-objective optimization can be computationally expensive

Future Work

- Multi-objective optimization solvers
- More elaborate optimizing fuctions
- Sophisticated prediction models (Gaussian processes)
- Parameter estimation techniques

The End

Thank you

Questions?