SCALING GEOMETRIC MONITORING OVER DISTRIBUTED STREAMS

by

Alexandros D. Keros

A thesis submitted in partial fulfillment
of the requirements for the degree

of

UNDERGRADUATE

in

Electronic and Computer Engineering

Approved:

Dr. Vasilis Samoladas
Major Professor

first reader
Committee Member

second reader
Committee Member

dean

Technical University of Crete
Chania, Crete, Greece

2015

Copyright © Alexandros D. Keros 2015

All Rights Reserved

Scaling Geometric Monitoring over Distributed Streams

by

Alexandros D. Keros, Undergraduate

Technical University of Crete, 2015

Abstract

BLAH BLAH

Thesis Supervisor: Dr. Vasilis Samoladas
Department: Electronic and Computer Engineering

(45 pages)

Public Abstract

BLAH BLAH

Acknowledgments

my mum

Contents

List of Tables

List of Figures

Part I

INTRODUCTION AND PRELIMINARIES

Chapter 1

Introduction

1.1 Overview

A multitude of recent emergent applications require real-time handling of rapidly incoming data, that may as well be great in size and distributed in nature. Such applications that follow a continuous distributed monitoring model are classified as *Data Stream Systems* [1]. Notable examples include, among others, distributed sensors, ISP network traffic monitoring, telecommunication system management, event monitoring, and real-time analysis tools for financial data.

These systems differ from the traditional Database Management Systems (DBMS), in the sense that they are following a *pull paradigm*, where large scale event monitoring is required or continuous queries are issued, instead of the *push paradigm*, where one-shot queries normally take place. Data Stream Systems are required to efficiently process, in real-time, data streams that are of high volume, continuous, size unbound, and most likely violative, in the sense that it would be inefficient to store them in memory. Additionally, the distributed nature of some applications incur an additional challenge to such systems, for they are required to communicate via a bandwidth-limited and possibly delay-inducing network in order to synchronize, reorganize, and provide a real-time overview of the results.

Consequently, intelligent algorithms must be devised that are able to guarantee high accuracy standards while limiting the communication overhead induced to the distributed setting.

application examples: threshold monitoring [2] value monitoring (which can be reduced to threshold monitoring [error threshold monitoring] - paper: sketch-based geometric monitoring of distributed stream queries - garofalakis, keren, samoladas [3])

complexity: monitoring value or threshold monitoring over the whole set of observations, in real time monitoring an arbitrary function (non linear function example), arbitrary number of features

possible solutions:

1.centralize

- suffers from network overload, storage overload

2.poll

- not real time, update frequency-accuracy trade-off

3.GM monitoring

-apply convex opt theory in order to reduce communication while retaining accuracy bounds

1.2 Motivation

A lot of work towards this direction, based on GM. We believe that there is still room for improvements regarding the way the method handles and represents data streams

1.3 Contributions

1.4 Thesis Outline

Chapter 2

Theoretical Background

The present chapter contains the background knowledge required throughout the length of this thesis. Section 2.1 describes the framework of the *Geometric Monitoring method*. Section 2.2 presents *multi-objective optimization* and dives into the algorithms used in our implementation. Section 2.4 discusses *graph maximum weight matching*, and, finally, in Section 2.3 we explain the *Savitzky-Golay filtering* used for smoothing, velocity and acceleration approximation.

2.1 Geometric Monitoring of Distributed Streams

The *Geometric Monitoring* method [2] was devised as a way to monitor threshold crossings of arbitrary functions over distributed data streams, i.e. be able to determine whether an arbitrary *monitoring function* $f(\cdot)$ over the data streams violated a predetermined threshold ($f(\cdot) > T$ or $f(\cdot) < T$). By mapping data streams to a feature space defined by the dimensionality of each data stream update and monitoring the convex hull surrounding the value of the monitoring function, Sharfman et al. were able to decompose the monitoring task into local constraints and apply distributed threshold monitoring, while reducing the communication costs required by central data processing.

In the current section a detailed presentation of this method is taking place. In Subsection 2.1.1 two system architectures are shown, a decentralized scenario and a centralized one, where Geometric Monitoring can be applied. Subsection 2.1.2 explains the computational model, followed by the method's geometric interpretation in Subsection 2.1.3. Finally, in Subsection 2.1.4 the protocol implementing the Geometric Monitoring method is described.

2.1.1 System Architecture

In [2] two different scenarios of Geometric Monitoring corresponding to different network topologies are examined. The *decentralized scenario* refers to a topology where nodes are allowed to communicate with each other and a central node is absent. The *centralized scenario* models a star network topology, where a coordinator node communicating with all other nodes is existent.

Decentralized Scenario

The topology examined is that of a partially or fully connected mesh network where a coordinator node is absent and nodes are allowed to broadcast to the network or communicate with each other according to the links existent between them. Data stream update vectors arrive continuously at each of the monitoring nodes and nodes must always be synchronized, i.e. all nodes must be aware of the monitoring task's state at all times. An example is depicted in Figure 2.1.

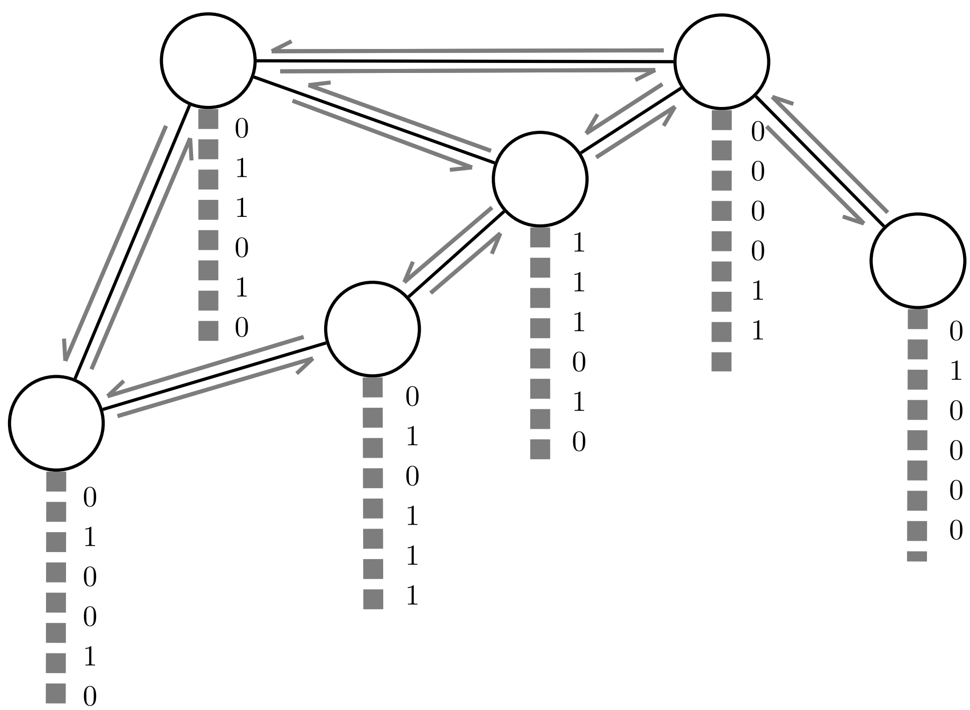


Figure 2.1: Network topology example of the decentralized scenario. Dashed lines represent data streams and half arrows represent message exchanges.

Centralized Scenario

The *centralized*, or *coordinator-based* scenario is built upon a star network topology, where all monitoring nodes communicate with a central node, the *coordinator node*. Nodes receive data stream update vectors continuously, and must communicate their state information to the coordinator node when needed. The coordinator receives data stream updates as well, which can be modelled by an additional monitoring node responsible for the cooridinator node's data stream. Communication between monitoring nodes is not allowed, thus, only the communicator can, and must, be aware of the state of the monitoring task at all times. An example is depicted in Figure 2.2.

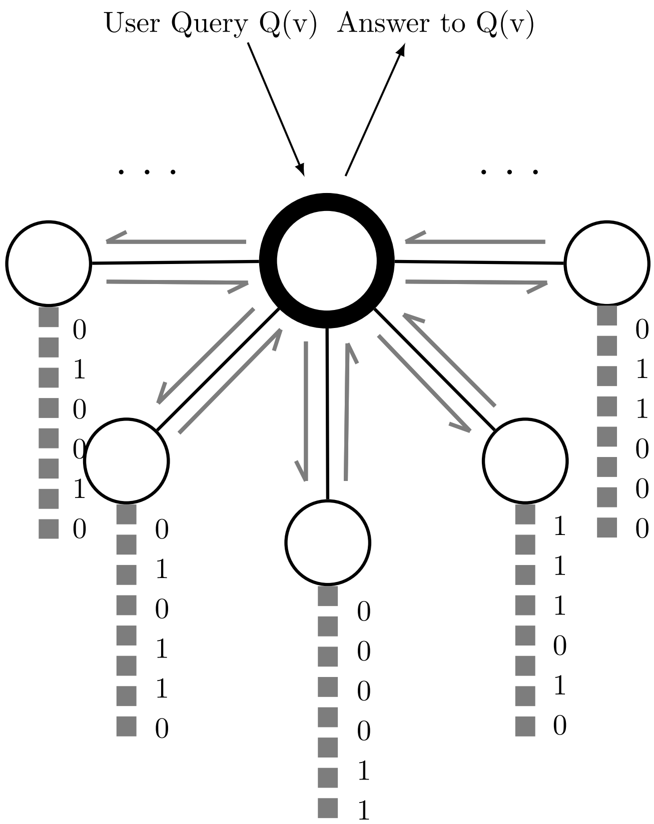


Figure 2.2: Network topology example of the centralized scenario.The bold node represents the coordinator node. Dashed lines represent data streams and half arrows represent message exchanges.

2.1.2 Computational Model

The main goal of the Geometric Monitoring method is to efficiently detect threshold crossings of an arbitrary function over distributed data streams. This is realized via vector projections of the data streams and convex local constraint assignments regarding said vectors at the nodes.

Let $f : \mathbb{R}^d \to \mathbb{R}$ be an arbitrary function, the *monitoring function*, whose value over the data streams needs to be monitored, so that if $f(\cdot) > T$ or $f(\cdot) < T$ an alarm is raised. For linear functions this problem is trivial, so that by letting, for example, x_1 and x_2 be data stream values at different nodes and requiring $f(\frac{x_1+x_2}{2}) > 10$ to be monitored, it holds that $f(\frac{x_1+x_2}{2}) = \frac{f(x_1)+f(x_2)}{2}$, and the problem can be decomposed to local constraints $f(x_i) < 10, i = 1,2$ at both nodes, i.e. a node remains silent until it violates its local constraint. Consider now the case of a non-linear function. By knowing the value of the function at the nodes nothing can be deduced about the function's value over the average of the monitoring streams and where it is positioned with respect to the threshold. Let $f(x) = 10x - x^2$, $x_1 = 0$ and $x_2 = 9$. Even thought $f(x_1) = 0 < 10$ and $f(x_2) = 9 < 10$, their average violates the specified threshold, $f(\frac{x_1+x_2}{2}) = f(4.5) = 24.75 > 10$.

In order to be able to effectively track non-linear functions, in the likes of the aforementioned example, a mapping of the streams to a vector space is taking place. Let $P = \{p_1, ..., p_n\}$ be the monitoring node set with weights $w_1, ..., w_n$, which can be either static or time varying. Their respective data streams $S = \{s_1, ..., s_n\}$ are represented by $\vec{v_1}(t), ..., \vec{v_n}(t)$, the d-dimensional *local statistics vectors* of the nodes at time t. The *global statistics vector* at time t is the weighted average of the local statistics vectors, as such:

$$\vec{v}(t) = \frac{\sum_{i=1}^{n} w_i \vec{v_i}(t)}{\sum_{i=1}^{n} w_i} \tag{2.1}$$

Infrequent communication between monitoring nodes, in the decentralized scenario, or between monitoring nodes and the coordinator, in the coordinator-based scheme, dictates the need to keep track of the value of the global statistics vector at the time the last global communication occurred, thus forming the *estimate vector*:

$$\vec{e}(t) = \frac{\sum_{i=1}^{n} w_i \vec{v_i}'}{\sum_{i=1}^{n} w_i} \tag{2.2}$$

,where $\vec{v_i}'$ is the last communicated statistics vector of node p_i.

At the monitoring nodes the difference between the current local statistics vector and the last communicated statistics vector is denoted by $\Delta\vec{v_i}(t) = \vec{v_i}(t) - \vec{v_i}', i = 1, ..., n$. The *drift vector* $\vec{u_i}(t), i = 1, ..., n$, also maintained at the monitoring nodes, represents the deviation of each node's data stream from the estimate vector and is defined differently in the two scenarios:

- In the **decentralized** setting the drift vector is regarded as the displacement of the local statistics vector from the estimate vector:

$$\vec{u_i}(t) = \vec{e}(t) + \Delta\vec{v_i}(t) \tag{2.3}$$

- In the **centralized** setting the monitoring nodes forward their state to the coordinator node, who has a global overview of the monitoring task at hand. This property allows the coordinator to counteract the effects a specific stream has on the partially observed monitoring task with an other, "opposite", stream belonging to a different monitoring node. This is taken care by the *balancing process* initiated every time a local violation occurs, which is responsible for computing and communicating the *slack vector* $\vec{\delta_i}$ to the nodes that contributed to the process, thus providing them with the necessary disposition of their drift vectors, as such:

$$\vec{u_i}(t) = \vec{e}(t) + \Delta\vec{v_i}(t) + \frac{\vec{\delta_i}}{w_i} \tag{2.4}$$

Balancing Process

The balancing process taking place in the **centralized scenario** is initiated by the coordinator node every time a threshold violation occurs, with the objective of resolving a possibly false alarm with minimal communication overhead. This task is executed by collecting a subset of monitoring nodes' data, the *balancing set* P', until the average of their drift vectors, the *balancing vector*, does not cause a threshold crossing. The balancing vector is formulated as follows:

$$\vec{b} = \frac{\sum_{p_i \in P'} w_i \vec{u_i}(t)}{\sum_{p_i \in P'} w_i} \tag{2.5}$$

After a successful balancing process has come to an end, $\Delta\vec{\delta_i}$ slack vector adjustments for all participants in the balancing set P' are computed and communicated to their respective sites, so that local drift vectors can be readjusted to reflect the balancing operation by computing $\vec{\delta_i} = \vec{\delta_i}' + \Delta\vec{\delta_i}$, where $\vec{\delta_i}'$ the previous slack vector (Equation 2.4). These adjustments are calculated as follows:

$$\Delta\vec{\delta_i} = w_i\vec{b} - w_i\vec{u_i}(t) \; \forall \; p_i \in P' \tag{2.6}$$

, where $\sum_{p_i \in P'} \Delta\vec{\delta_i} = \vec{0}$. Once the slack vector adjustments have been communicated to the respective monitoring nodes participating in P', their drift vectors are essentially set to the value of the newly computed balancing vector.

In case the balancing process proves unsuccessful all monitoring nodes are contained in the balancing set P' and a new estimate vector is computed with the data cumulated at the coordinator node. Subsequently, all drift vectors and slack vectors are set to $\vec{0}$.

2.1.3 Geometric Interpretation

The estimate vector, being the product of the system's previous global synchronization, is known to all monitoring nodes and denotes the last known position of the global statistics vector. That being said, the estimate vector is considered valid if it resides on the same side of the threshold as the unknown global statistics vector. In order to estimate the current position of the global statistics vector, since a mere observation of the monitoring function's value at each stream provides no information about its current location (as described in Section 2.1.2), it is vital that the task is decomposed into local constraints that will guarantee the timely detection of a violation of the aforementioned statement.

The *convexity property* of the drift vectors, along with Theorem 1 [2], are sufficient in provide a framework for decomposing the monitoring task into local constraints at the nodes. Both the convexity property and the relevant theorem are repeated below for completeness.

The convexity property dictates that the weighted average of the drift vectors equal the global statistics vector, as such:

$$\vec{v}(t) = \frac{\sum_{i=1}^{n} w_i \vec{u_i}(t)}{\sum_{i=1}^{n} w_i} \tag{2.7}$$

The geometric interpretation of the property guarantees that the global statistics vector $\vec{v}$ is always contained in the convex hull defined by the drift vectors $\vec{u_i}, i = 1, ..., n$.

Theorem 1 (Sharfman et al. [2])**.** *Let $\vec{x}, \vec{y_1}, ..., \vec{y_n} \in \mathbb{R}^d$ be a set of vectors in $\mathbb{R}^d$. Let $Conv(\vec{x}, \vec{y_1}, ..., \vec{y_n})$ be the convex hull of $\vec{x}, \vec{y_1}, ..., \vec{y_n}$. 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 $Conv(vecx, \vec{y_1}, ..., \vec{y_n}) \subset B(\vec{x}, \vec{y_i})$.*

Essentially, Theorem 1 states that n d-dimensional spheres defined by $n+1$ vectors can effectively bound the convex hull defined by said vectors, as such: $Conv(\vec{x}, \vec{y_1}, \vec{y_2}, ..., \vec{y_n}) \subset \cup B(\vec{x}, \vec{y_i}), i =$

$1, ..., n$, which finds direct application to the distributed monitoring task if $\vec{x} = \vec{e}$ and $\vec{y_i} = \vec{u_i}, i = 1, ..., n$. An example is depicted in Figure 2.3 .

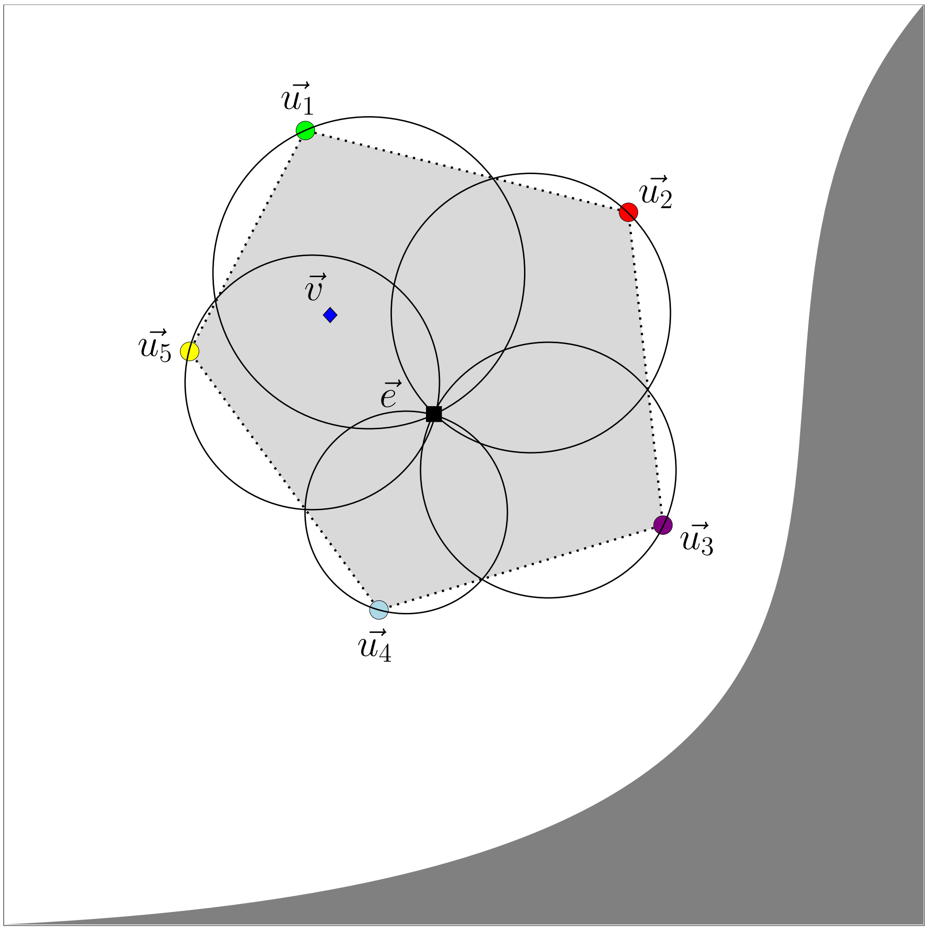


Figure 2.3: Example of a convex hull (light gray) defined by the drift vectors $\vec{u_i}, i = 1, 2, 3, 4, 5$. The hull is bounded by the spheres created from the estimate vector $\vec{e}$ and the drift vectors $\vec{u_i}, i = 1, 2, 3, 4, 5$. The global statistics vector $\vec{v}$ is guaranteed to be contained in the convex hull of the drift vectors

Local Constraints

monochromaticity of balls

balls monochromatic means threshold upheld

2.1.4 Protocol

decentralized algorithm (in short, for completeness) centralized algorithm (in detail)

*we will focus on that

2.2 Multiobjective Optimization

what is mop

use examples

kinds:

a.numerical

b.evolutionary

2.2.1 SLSQP

2.2.2 Sohr's algorithm a.k.a. ralg

algorithm description

2.3 Savitzky-Golay Filtering

filtering generals

examples of uses of filters

filters:

Kalman

+,- Moving Average

+,- Savitzky-Golay a.k.a. ??? +,-

algorithm description

2.4 Maximum Weight Matching in Graphs

general graph theory (introductory)

what is max weight matching

algorithm description

Chapter 3

Related Work

cite papers working on the original metioned above

function specific stuff

bounding ellipsoids

reference vector change (estimate vector)

safe zones

prediction

matching

*no work on slack vector distribution during balancing, we do!

Part II

PROBLEM DEFINITION AND IMPLEMENTATION

Chapter 4

Problem Statement

papers in chapter 3 do not scale well

why?

where exactly?

we try our luck at it, how? (one liner)

Chapter 5

Implementation

This chapter provides a detailed description of the implemented system. In Section 5.1, the Geometric Monitoring method implementation is described, along with the necessary simplifying assumptions to aid experimentation. Following that, in Section 5.2 an algorithm for node matching is proposed, inspired by the violation recovery method found in [4]. In Section 5.3, the heuristic based balancing method for local violation resolution is presented, along with the necessary data stream tracking scheme. Finally, the main implementation challenges are discussed.

5.1 Geometric Monitoring Implementation

The initial Geometric Monitoring method [2], which is described in detail in Section 2.1, provides two algorithms for threshold monitoring of distributed data streams. These algorithms operate on different network structures and implement a somewhat different handling of threshold violations.

The decentralized algorithm operates on a coordinator-less environment, where nodes are allowed to communicate with each other, whereas the coordinator-based algorithm has a Star network topology, where the coordinator node is the central node (the *hub*) and the Monitoring nodes reside on the edges of the network. The algorithm operating on the decentralized setting does not provide a balancing process for local violation resolution. On the other hand, the coordinator based algorithm implements a violation resolution operation every time a local violation occurs, which aims to minimize the communication overhead induced by false violation reports.

Our focus is centered towards a simplified **coordinator-based algorithm** (Algorithm **??**), described in Section 2.1, as it provides a framework for the heuristic balancing process, as well as the node matching operation presented in detail in Sections 5.3 and 5.2 respectively.

To aid method formulation and experimentation, the following simplifying assumptions have been made regarding the coordinator-based algorithm:

- Communication between nodes is considered instantaneous. There is no delay when passing messages through the network. The problem of message handling in a real-world Geometric Monitoring method implementation, where message delays are induced by the underlying network, has been studied in detail in [5].

- Communication between nodes is considered loss-less and reliable. In case network reliability can not be guaranteed appropriate methods should be considered.

- The system operates in an iterative fashion, as described in Algorithm 1. This simplification of the real-time distributed monitoring process to an iterative process provides a more manageable setting for experimentation without distorting the results of the proposed methods, which can be applied directly to the original real-time distributed setting.

- The system pauses at each violation, until the violation is resolved. During violation resolution Monitoring nodes do not receive updates from their respective data streams.

- The Coordinator node does not participate in the monitoring operation. The Coordinator node does not receive updates from a data stream, it only receives messages from the Monitoring nodes in case of threshold violation. This assumption can easily be elevated by considering an additional monitoring node responsible for handling the coordinator's data stream monitoring operation.

Algorithm 1: 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
```

5.2 Distance Based Node Matching

The balancing method of the coordinator-based algorithm, as described in Section 2.1 [2, 6], aims at resolving local violations that do not result in a global violation (*false alarms*) by balancing the violating node's drift vector with the respective vectors of *randomly* chosen nodes. Consider the violating node n_i with weight $w_i = 1$, so that the bounding ball $B(\vec{e}(t), \vec{u_i}(t))$ is not monochromatic, and the randomly requested node n_j with weight $w_j = 1$, so that the newly formed bounding ball is $B(\vec{e}(t), \frac{\vec{u_i}(t)+\vec{u_j}(t)}{2})$, where $\vec{e}(t)$ the estimate vector at time t and $\vec{u_i}(t)$, $\vec{u_j}(t)$ the drift vectors of nodes n_i, n_j at time t, respectively. If the resulting bounding ball is monochromatic the violation is resolved, otherwise another node is *randomly* requested for balancing.

As observed in [4], the original balancing method's node choosing scheme can be inefficient, so a more efficient and deterministic approach should be adopted. Optimal pairing of nodes and the construction of a hierarchical structure (Figure 5.1) reduces the communication overhead of false alarms, with the vast majority of violation resolutions requiring only the assigned node pair to be successful. The criterion by which nodes are paired attempts to maximize the probability of a successful balance by maximizing *"the percentage of pairs of data vectors from both nodes whose sum is in the Minkowski sum of the nodes' safe-zones"* [4], or, in this case, whose resulting bounding ball is monochromatic.

Here, the same node pairing scheme is followed, but with a different, distance based, criterion for grouping nodes into disjoint pairs and creating the hierarchical structure depicted in Figure 5.1. The method proceeds as follows (Algorithm 2):

1. Monitoring nodes are visualized as the nodes of a complete graph $G = (V, E)$, where $V = \{n_1, n_2, ..., n_k\}$ vertex set consists of the initial Monitoring nodes (*"Type-1 nodes"*) and $E = \{(n_i, n_j)\ \forall i, j \in [1, ..., k], i \neq j\}$ edge set contains an edge for every pair of vertices.

2. Weights are assigned to all edges E. The weight of each edge is defined as the cumulative distance of the value of the monitoring function on the mean of each pair of data vectors from the value of the monitoring function on the *global* mean of all Monitoring nodes' data vectors, plus the cumulative distance of each pair of data vectors:

$$w_{i,j} = \sum_{t=t_0}^{t_{end}} [(f(\vec{v}_{global}(t)) - f(\frac{\vec{v_i}(t) + \vec{v_j}(t)}{2})) + (|\vec{v_i}(t) - \vec{v_j}(t)|)] \tag{5.1}$$

   , where $\vec{v_i}(t)$ the data update of node n_i at time t, $\vec{v}_{global}(t)$ the global mean of all Monitoring nodes at time t and $f(\cdot)$ the monitoring function.

3. Maximum weighted matching is performed on the resulting graph, so that nodes are partitioned into disjointed sets M_i, $|M_i| = 2\ \forall i \in [1, ..., \frac{k}{2}]$.

4. Each set $M_i, i \in [1, ..., \frac{k}{2}]$ is considered a single node, so that a new complete graph $G' = (V', E')$ is created, where $V' = \{M_1, ..., M_{\frac{k}{2}}\}$ (*"Type-2 nodes"*) the new vertex set and $E' = \{(M_i, M_j)\ \forall i, j \in [1, ..., \frac{k}{2}]\}$ the new edge set. Weights are assigned to the new edges and the process repeats until the resulting graph contains only a single vertex (*"Type-k node"*), which incorporates all the initial Monitoring nodes.

5. Vertices not matched with any other vertex during the matching process are ignored in future iterations. During the balancing process such unmatched vertices are handled by the traditional random selection balancing algorithm found in [2](also, Section 5.1).

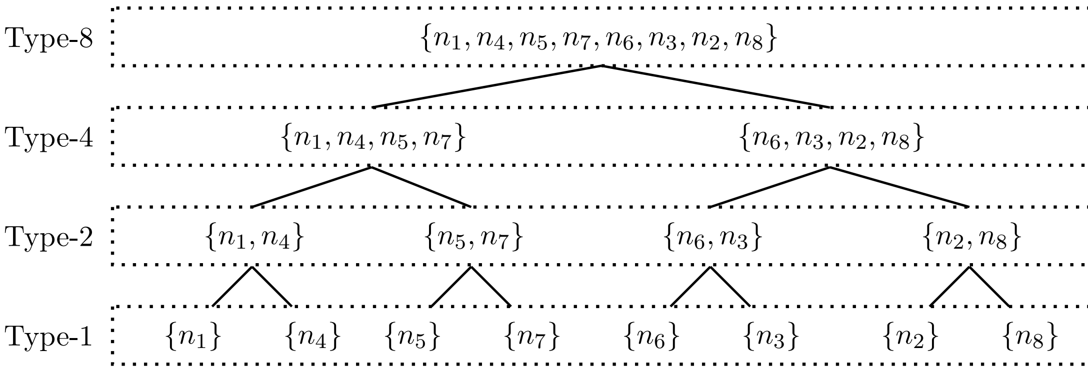


Figure 5.1: Hierarchical pairing scheme example for node set $\{n_1, n_2, n_3, n_4, n_5, n_6, n_7, n_8\}$.

Algorithm 2: 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

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

  `DistancePairer`(*nodeHierarchy*(*Type-i*), $i * 2$);

end

The incentive behind the distance based node pairing scheme comes from the need to track the global data vector as closely as possible, with only a subset of the total node population's data vectors at each balancing attempt. By considering the distance of the mean of a pair of data vectors from the global data vector (distance d_1 in Figure 5.3) the *"quality"* and *"accuracy"* of the tracking ability of each pair is evaluated. Additionally, by taking into account the in-between distance of data vectors of each node pair (distance d_2 in Figure 5.3), pairs from the limits of the data vector velocity spectrum that manage to *"cancel each other out"* more effectively are encouraged.

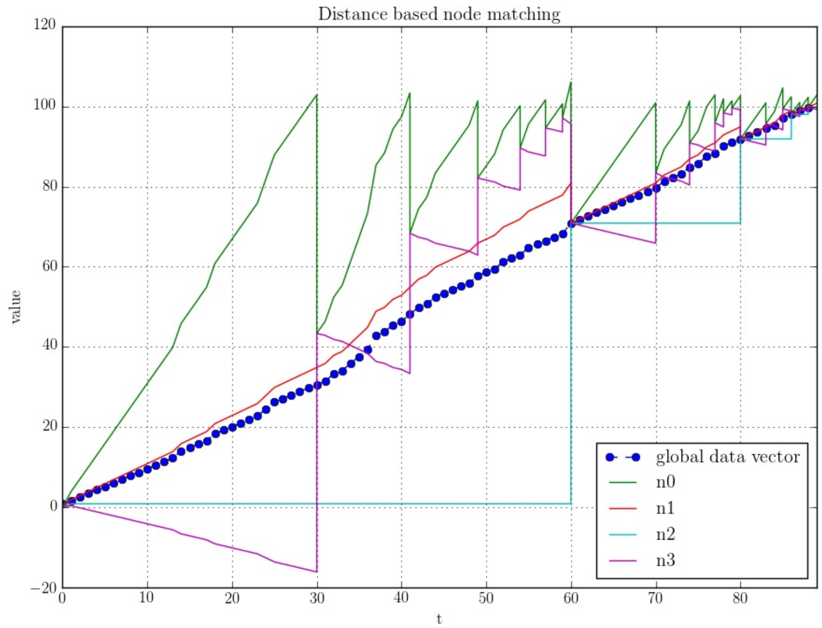


Figure 5.2: The drift vectors during Geometric Monitoring operation until a Global Violation. Distance based node matching is used on 4 nodes ($\{n_0, n_1, n_2, n_3\}$), with 1-dimensional data vectors, threshold $T = 100$ and $f(x) = x$ as the monitoring function. The *Type-2* node pairs are $\{n_0, n_3\}$ and $\{n_1, n_2\}$.

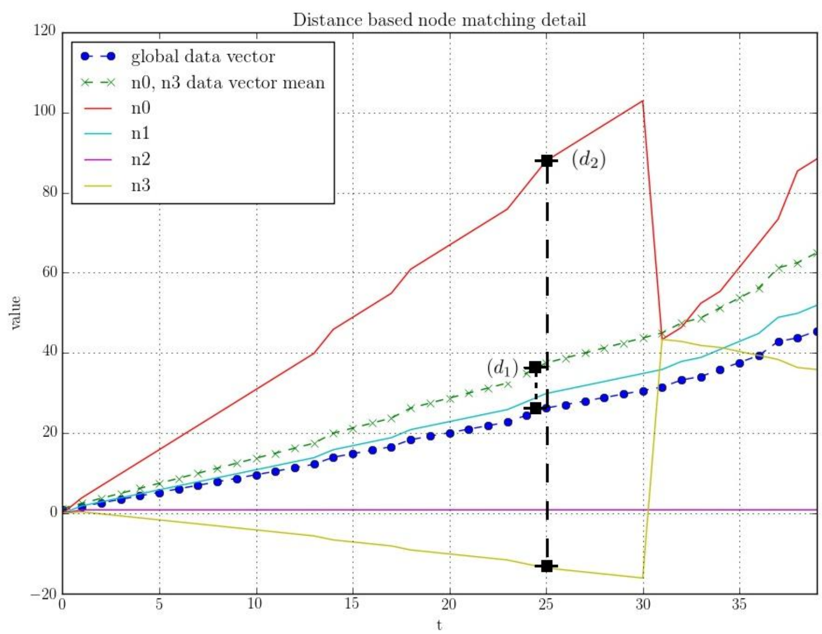


Figure 5.3: Detailed depiction of the Geometric Monitoring operation of Figure 5.2. Distance based node matching operating on 4 nodes ($\{n_0, n_1, n_2, n_3\}$), with 1-dimensional data vectors, threshold $T = 100$ and $f(x) = x$ as the monitoring function. Distance d_1 denotes the distance of the data vector mean of the paired nodes n_0 and n_3 from the global mean (*global data vector*) at $t = 25$, whereas distance d_2 denotes the in-between distance of data vectors $\vec{v_0}(t)$ and $\vec{v_3}(t)$ of the node pair at time $t = 25$ (before a Local Violation occurs, where $\vec{e} = 0$ and $\vec{u_i}(t) = \vec{v_i}(t)\ \forall\ i \in [0, 1, 2, 3], t < 30$). Both distances are taking part in the edge weighting process, according to Equation 5.1.

5.3 Heuristic Balancing

The balancing method incorporated into the *coordinator based* algorithm of the Geometric Monitoring method [2] (Section 2.1) attempts to minimize the communication overhead of local violations by computing the, so called, *balancing vector*. The *balancing vector* is defined as the weighted mean of the drift vectors of the nodes contained in the balancing set, and, in case of a successful balance, it is guaranteed that $B(\vec{e}, \vec{b})$ is monochromatic. Consequently, by setting the drift vectors of the nodes in the balancing set to be equal to the balance vector, all local constraints are fulfilled and the convexity property of the drift vectors is satisfied.

While this method partially succeeds in reducing the communication burden of false alarms either by requesting only a subset of the total node set each time a Local Violation occurs or by setting the drift vectors to a safe point (represented by the balance vector), major drawbacks can be noted regarding vector positioning and bounding ball construction. Updated vector assignment as a result of the "optimization" procedure does not take into account the idiosyncrasies of the monitoring function and the admissible region it produces. Additionally, all nodes taking part in the balancing process are handled identically, without taking advantage of the differences in the behavior of each node.

Previous work proposed selecting an optimal reference vector, instead of the estimate vector for bounding ball construction, along with shape customization of the local constraints at the nodes according to the node's needs [6]. Local constraint customization served as the basis for the now popular *Safe-Zone* framework [4, 7], which diverges from the traditional bounding sphere setting, while maintaining the same fundamental idea of distance computation of a point from a set of support vectors [8], preserving the essence of the admissible region and retaining the balancing process of the coordinator based scenario.

This thesis proposes a novel heuristic approach for optimal positioning of drift vectors, which takes into account both the temporal behavior of each node's data stream, as well as the peculiarities of the monitoring function over said data streams. Aim of the heuristic optimization is the maximization of the estimated time until the following Local Violation occurs, which, expressed as an optimization formula, receives the following form:

$$\max\min \frac{2 * (T - x_i)}{vel_i(t_{lv})\sqrt{2 * (T - x_i) * accel_i(t_{lv}) + {vel_i}^2(t_{lv})}}, \forall n_i \in P' \tag{5.2}$$

where:

T : monitoring threshold

x_i : the maximum value of the monitoring function $f(\cdot)$ over the bounding ball $B(\vec{e}(t_{lv}), \vec{u_i}(t_{lv}))$, where t_{lv} is the time a Local Violation occurred and i the index of node n_i

$vel_i(t_{lv})$: the estimated velocity of the maximum value of the monitoring function $f(\cdot)$ when applied to the bounding ball created by the data stream update of node n_i and the estimate vector $\vec{e}$ at time t_{lv}

$accel_i(t_{lv})$: the estimated acceleration of the maximum value of the monitoring function $f(\cdot)$ when applied to the bounding ball created by the data stream update of node n_i and the estimate vector $\vec{e}$ at time t_{lv}

t_{lv} : time of Local Violation occurrence

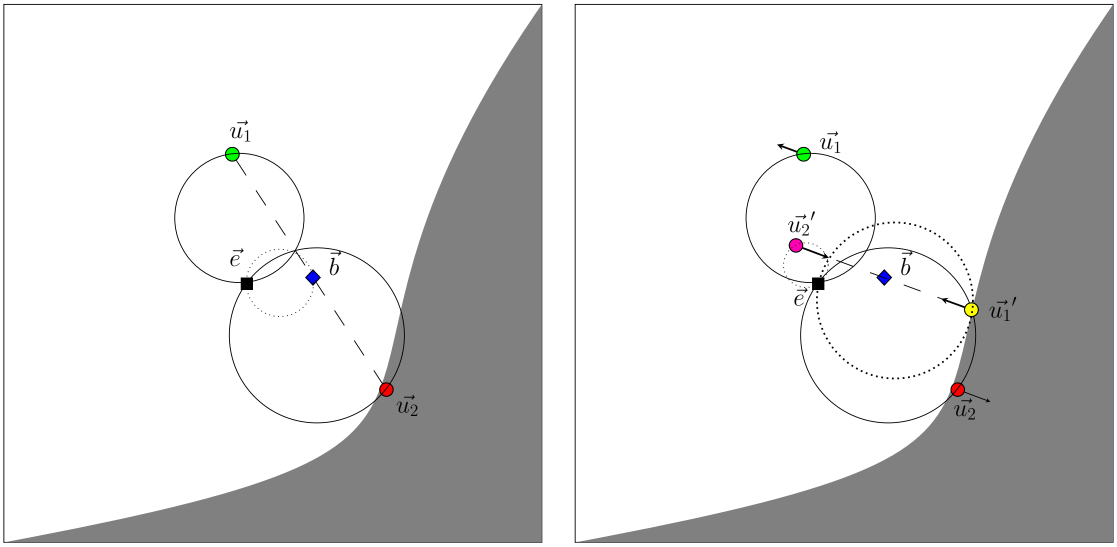
P' : the balancing set

The Equation 5.2 originates from the the combination of elementary kinematic equations, as such:

Assume a moving object i at point x_i, with acceleration a_i and current velocity v_i. Let v_f be the object's final velocity when it reaches a threshold point T at time t, from which it deviates by $d = T - x_i$. Let current time be $t = 0$.

Distance (or *Displacement*) is described by:

$$d = \frac{v_i + v_f}{2} * t \tag{5.3}$$



(a) The classic balancing method. As long as $B(\vec{e}, \vec{b})$ is monochromatic (i.e. within the Admissible region), balance is successful and the updated drift vectors are set to $\vec{u_1}' = \vec{u_2}' = \vec{b}$.

(b) The heuristic balancing method. Arrows depict the velocities of each drift vector. After a successful balance is achieved ($B(\vec{e}, \vec{b})$ is monochromatic), the optimal points in which the updated drift vectors $(\vec{u_1}', \vec{u_2}')$ should be positioned are computed by maximizing the estimated time until the next Local Violation, based on the current drift vector positions and the estimated velocities. Balance vector $\vec{b}$ remains unchanged.

Figure 5.4: Balancing methods

Final velocity is defined as:

$$v_f^2 = v_i^2 + 2 * a * d \tag{5.4}$$

By solving Equation 5.4 for v_f, plugging it into Equation 5.3 and solving the resulting Equation for t, we extract the desired result (Equation 5.2).

The newly defined heuristic optimization formula (5.2) aims to maximize the time until the next Local Violation concerning any of the nodes belonging in the balancing set. By taking into account the maximum value of the monitoring function $f(\cdot)$ inside the bounding ball created by each data stream update and the estimate vector, and by computing acceleration and velocity measures of this value over time, an approximate mapping of the data stream space to the one dimensional space of the arbitrary monitoring function is achieved. This permits the computation of the optimal positions the balanced drift vectors should take in order to maximize the time they reach the monitoring threshold, as depicted in Figure 5.4b.

5.3.1 Implementation of the Heuristic Balancing

In order transform the heuristic optimization formula (5.2) into an applicable setting, *multi-objective optimization* (Section 2.2) is used. The optimization function is now defined as such:

$$
\begin{aligned}
\min\; & -z \\
\text{s.t.}\quad & z \leq g(h(\vec{e}, \vec{u_0}), vel_0, accel_0, T) \\
& z \leq g(h(\vec{e}, \vec{u_1}), vel_1, accel_1, T) \\
& \vdots \\
& z \leq g(h(\vec{e}, \vec{u_n}), vel_n, accel_n, T) \\
& \vec{b} = \frac{1}{\sum_{i=0}^{n} w_i} \sum_{i=0}^{n} (w_i * \vec{u_i}) \qquad\qquad , \forall n_i \in P'
\end{aligned}
\tag{5.5}
$$

where:

$g : \mathbb{R}^4 \to \mathbb{R}$, the heuristic optimization function as defined in Equation 5.2

$h : \mathbb{R}^d \to \mathbb{R}$, the function computing the maximum value of the monitoring function $f(\cdot)$ in $B(\vec{e}, \vec{u_i})$, which is an optimization problem by itself

d : the data vector dimensionality

T : the monitoring threshold

$\vec{u_i}$: the drift vector of node n_i

w_i : the weight of node n_i

vel_i : the velocity of the maximum value of the monitoring function when applied to the ball defined by node's n_i drift vector $\vec{u_i}$ and the estimate vector $\vec{e}$

$accel_i$: the acceleration of the maximum value of the monitoring function when applied to the ball defined by node's n_i drift vector $\vec{u_i}$ and the estimate vector $\vec{e}$

$\vec{b}$: the balancing vector

P' : the balancing set

Solution to the above optimization problem (5.5) is given by the *Sequential Least Squares Programming (SLSQP)* solver, which is described in Subsection 2.2.1. The problem is decomposed and formulated using an additional helping parameter z in order to avoid non-differentiable functions (such as min and max) and to aid computation by the solver.

In the heuristic optimization problem defined previously (5.5) the nested optimization of detecting the maximum value of an arbitrary monitoring function inside the bounding ball $B(\vec{e}, \vec{u_i})$ is existent. This optimization problem is formed as follows:

$$\max \quad f \tag{5.6}$$

$$\text{s.t.} \quad \sum_{i=1}^{d}(x_i - c_i)^2 = r^2 \tag{5.7}$$

where:

f : the monitoring function $f(\cdot)$

x_i : element i of d-dimensional vector $\vec{x}$

c_i : element i of d-dimensional vector $\vec{c}$, which represents the center of the sphere

r : the radius of the sphere

d : the space dimensionality

Eq. 5.7 : a $(d + 1)$ dimensional sphere in $\mathbb{R}^d$

The optimization problem of detecting the maximum value of a function inside a sphere (5.6) is solved using *Constrained Function Minimization (CONMIN)*, which implements the method of feasible directions, as described in Subsection 2.2.2.

The resulting heuristic balancing algorithmic implementation is summarized in the following Algorithm:

Algorithm 3: 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

5.3.2 Smoothing, Velocity and Acceleration Estimation via Savitzky-Golay

The heuristic balancing method proposed previously (Section 5.3) requires an efficient estimation of the velocity and the acceleration of the output of the monitoring function over the maximum value of the bounding ball. Additionally, a smoothing operation over the data stream series would be beneficial, in order to grasp the trend (increasing or decreasing) of the data stream without letting noisy updates and extreme fluctuations misguide the optimization operation.

The *Savitzky-Golay smoothing filter* [9] (Section 2.3) is ideal in the heuristic Geometric Monitoring setting, for it smooths and derivates the signal without much additional computational burden, allowing it to be applied directly at the Monitoring Nodes' data streams. By assuming equidistant data points the precomputation of convolution coefficients becomes trivial when specifying the window size, the window center, the order of the polynomial and the desired derivative. Following that, the precomputed coefficients are applied to the desired signal by a simple convo-

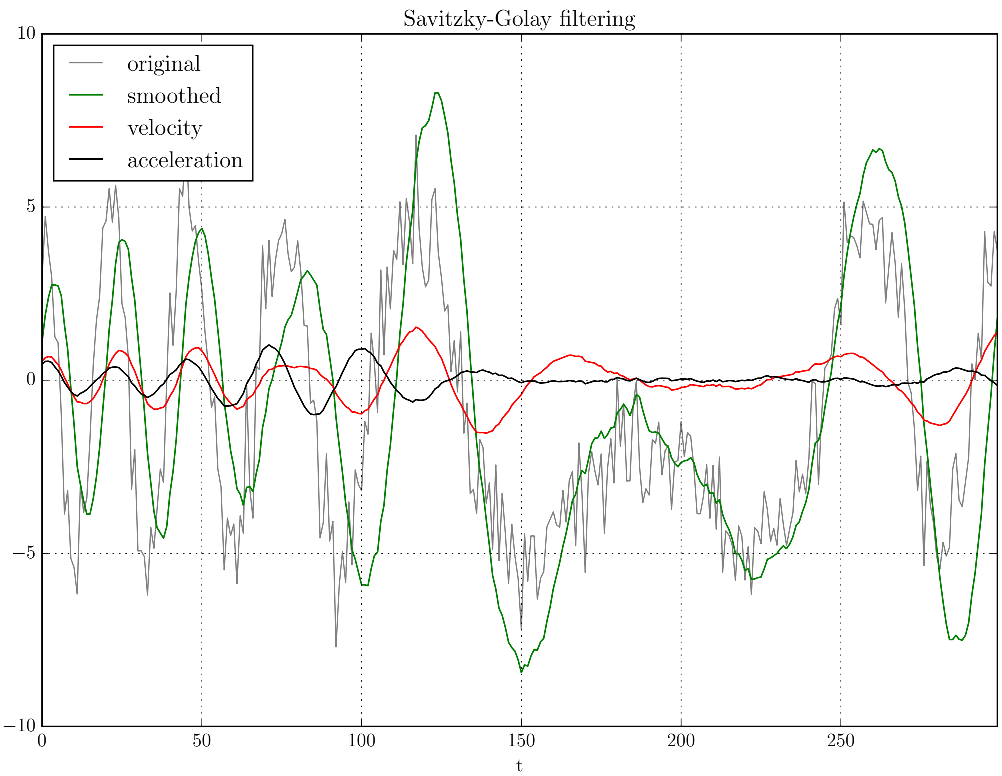


Figure 5.5: Savitzky-Golay filtering of a signal with added Gaussian noise. The smoothing window is 50 points in length, centered at the far end, as in the real-time smoothing applied to the Geometric Monitoring setting. The polynomial order is 2 for the smoothed signal, 3 for the velocity estimation and 5 for the acceleration estimation of the original signal.

lution, which is both fast and, if required, on-line. The application of the filter on a noisy signal, along with the velocity and the acceleration computation of this signal, is shown in Figure 5.5.

5.4 Implementation Challenges

The proposed methods and algorithms incur some implementation challenges, which, on the greater part, can be managed.

Regarding the *distance based node matching* presented in Section 5.2:

- In order to extract the optimal node pairs training data must be available. This situation can be handled in two ways. One way is to initiate execution of the Geometric Monitoring task using the original method of randomly requesting nodes during balancing, until the necessary amount of data to train the model has been cumulated. Then the model can be trained and the operation can be switched to the distance based node matching scheme. A more appropriate solution could be to incrementally update the node pairs using the data provided by message passing in the standard Geometric Monitoring execution, or by occasionally polling the monitoring nodes during low network activity until a satisfiable amount of data has been gathered.

Regarding the *heuristic balancing* method presented in Section 5.3:

- The bi-level multi-objective optimization incorporated into the method can become computationally expensive when dealing with a large balancing set or with highly dimensional data streams. Attention must be paid to the selected solvers responsible for the optimization task, for some solvers can be more effective than others in different settings and different monitoring function applications. Additionally, some solvers provide customization parameters, such as *tolerance* and *iteration count*, among others, that greatly influence the execution time of the optimization routine, as well as the precision of the results.

- The *Savitzky-Golay smoothing filter*, responsible for smoothing and differentiating the signals representing the maximum value of the monitoring function over the bounding spheres, is directly affected by the selected window length and the polynomial order. That being the case, care must be taken to select appropriate values that effectively track the general trends without compromising detail important to the optimization routine.

Part III

RESULTS AND CONCLUSIONS

Chapter 6

Experimental Results

experimental result showcase

6.1 Experimental Setting

dataset used

reference appendix for tools, mention in short

6.2 Distance Based Node Matching

comparison with random matching

comparison with distribution node matching deligiannakis

!use same balancing, both classic and heuristic! (i.e. 1st all with classic, then all with heuristic)

explain

6.3 Heuristic Balancing

comparison with classic balancing

!random matching!

explain

how S-G affects results

6.4 Overall Results

summarise results

compare classic random and classic distribution optpair with heuristic distance optpair observe how S-G affects results again

explain

Chapter 7

Conclusions and Future Work

conclusions

7.1 Conclusions

problem statement in short

what has been done in short

our contributions

short explanation of contributions

7.2 Future Work

References

[1] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom, "Models and issues in data stream systems," in *Proceedings of the Twenty-first ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, ser. PODS '02. New York, NY, USA: ACM, 2002, pp. 1–16. [Online]. Available: http://doi.acm.org/10.1145/543613.543615

[2] I. Sharfman, A. Schuster, and D. Keren, "A geometric approach to monitoring threshold functions over distributed data streams," in *Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '06. New York, NY, USA: ACM, 2006, pp. 301–312. [Online]. Available: http://doi.acm.org/10.1145/1142473.1142508

[3] M. N. Garofalakis, D. Keren, and V. Samoladas, "Sketch-based geometric monitoring of distributed stream queries." *PVLDB*, vol. 6, no. 10, pp. 937–948, 2013. [Online]. Available: http://dblp.uni-trier.de/db/journals/pvldb/pvldb6.html#GarofalakisKS13

[4] D. Keren, G. Sagy, A. Abboud, D. Ben-David, A. Schuster, I. Sharfman, and A. Deligiannakis, "Geometric monitoring of heterogeneous streams." *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 8, pp. 1890–1903, 2014. [Online]. Available: http://dblp.uni-trier.de/db/journals/tkde/tkde26.html#KerenSABSSD14

[5] B. Babalis, "A simulator for monitoring data streams," Master's thesis, Technical University of Crete, Chania, Greece, 2013.

[6] D. Keren, I. Sharfman, A. Schuster, and A. Livne, "Shape sensitive geometric monitoring," *IEEE Transactions on Knowledge and Data Engineering*, vol. 24, no. 8, pp. 1520–1535, Aug 2012.

[7] D. Keren, G. Sagy, A. Abboud, D. Ben-David, I. Sharfman, and A. Schuster, "Safe-zones for monitoring distributed streams," in *Proceedings of the First International Workshop on Big*

Dynamic Distributed Data, Riva del Garda, Italy, August 30, 2013, 2013, pp. 7–12. [Online]. Available: http://ceur-ws.org/Vol-1018/paper11.pdf

[8] V. Samoladas, "Unification fo safe zones and the geometric method with application to generalized median monitoring," October 2013.

[9] A. Savitzky and M. J. E. Golay, "Smoothing and differentiation of data by simplified least squares procedures." *Analytical Chemistry*, vol. 36, no. 8, pp. 1627–1639, 1964. [Online]. Available: http://dx.doi.org/10.1021/ac60214a047

Appendix

Chapter A

Geometric Monitoring Python Implementation

A.1 Python

what is python

why python

A.2 Numpy and Scipy

what are they

why use them and how

A.3 Openopt

what is it

details about framework

A.4 NetworkX

what is it

details about framework

A.5 Putting It All Together

code description

UML

how to run