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

(34 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

1.2 Motivation

1.3 Contributions

1.4 Thesis Outline

Chapter 2

Theoretical Background

The present chapter contains the necessary background knowledge used throughout the length of this thesis. Section 2.1 describes the *"Geometric Approach to Monitoring Threshold Functions over Distributed Data Streams"* in detail, as formulated by I.Sharfman, A.Shuster, D.Keren [**?**]. Section 2.2 presents *multi-objective optimization* and dives into the algorithms used in our implementation. Section 2.4 discusses *graph maximum weight matching* used for node pairing 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 Continuous Distributed Monitoring Model, a.k.a. Data Stream System (definition)

idea: having a real-time overview over the system differing from tradition DBMS: push paradigm vs pull paradigm, continuous queries

application examples: ISP network traffic, distributed sensors etc

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

goal: minimize communication while retaining the highest accuracy possible

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 details of geometric monitoring model

2.1.1 System Architecture

fully distributed node topology

.no coordinator-center node

.communication between nodes

*image

coordinator based node topology

.coordinator-center node

.nodes communicate only with coordinator

*image

2.1.2 Computational Model

stream and node notation

weights

statistics vectors

global statistics vector

monitored function

threshold

estimate vector

drift vector

general operation of distributed algorithm

drift vector definition

general operation of coordinator based algorithm

*balancing process

slack vector

drift vector definition

2.1.3 Geometric Interpretation

node local constraints make sure global violation is accurately monitored

how?

convexity property of drift vectors

theorem of bounded convex hull by local constraints (balls)

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 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 [1]. 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 [3].

- 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.

Algorithm 1: Iterative network operation

```
Data: $monitoringNodes$: a list of Monitoring nodes, $coordinator$: the Coordinator node
begin
    initialization;
    repeat
        foreach $node \in monitoringNodes$ do
            $node.DataVectorUpdate()$;
            $node.ComputeDriftVector()$;
        end
        foreach $node \in monitoringNodes$ 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, 4], 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 [1], 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"* [1], 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).

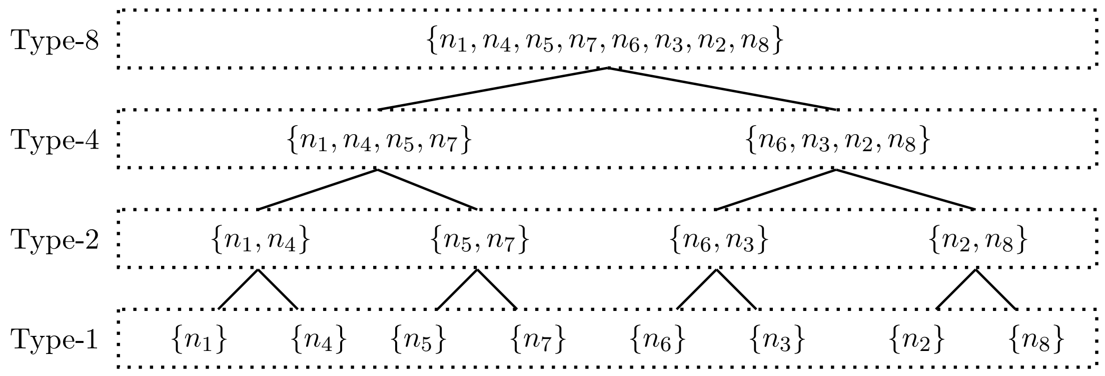


Figure 5.1: Hierarchical pairing scheme example for node set $\{n_1, n_2, n_3, n_4, n_5, n_6, n_7, n_8\}$.

Algorithm 2: Recursively create Monitoring node pairs and hierarchy

```
1  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 $Type\text{-}k$ pairs
2      if $length(nodes) = 1$ then                          // recursion stopping condition
3          return $nodeHierarchy$;
4      end
5      $g = CreateCompleteGraph(nodes)$;     // complete graph with $nodes$ as vertices
6      foreach $(n_i, n_j) \in g.Edges()$ do                     // assign weights to edges
7          $w_{i,j} = \sum_{t=t_0}^{t_{end}} [(f(\vec{v}_{global}(t)) - f(\frac{\vec{v_i}(t)+\vec{v_j}(t)}{2})) + (|\vec{v_i}(t) - \vec{v_j}(t)|)]$;
8          $g.edge(n_i, n_j).weight = w_{i,j}$;
9      end
10     $nodeHierarchy$(Type-i) $= g.maximalWeightMatching()$;  // node pairs of Type-i
11     DistancePairer($nodeHierarchy(Type\text{-}i), i * 2$);
12 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.

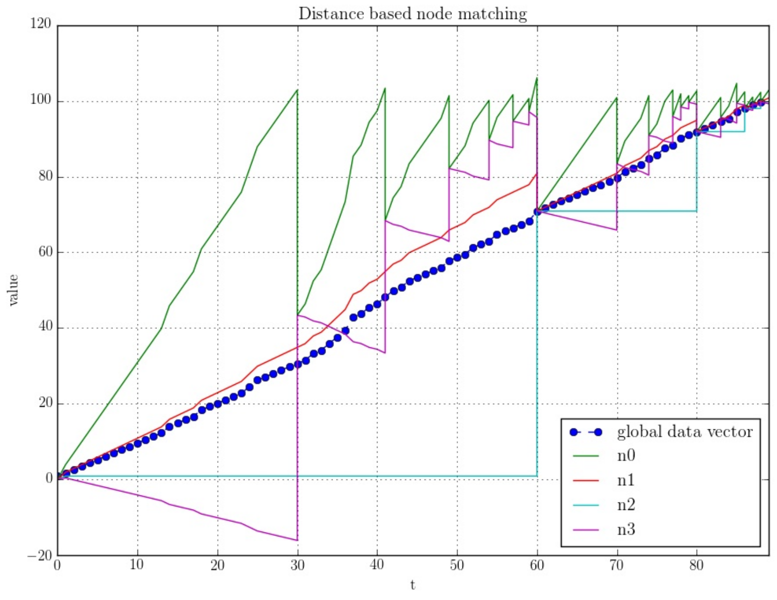


Figure 5.2: The drift vectors during Geometric Monitoring operation until a Global Violation. Distance based node matching is used on 4 nodes ($\{n_0, n_1, n_2, n_3\}$), with 1-dimensional data vectors, threshold $T = 100$ and $f(x) = x$ as the monitoring function. The *Type-2* node pairs are $\{n_0,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

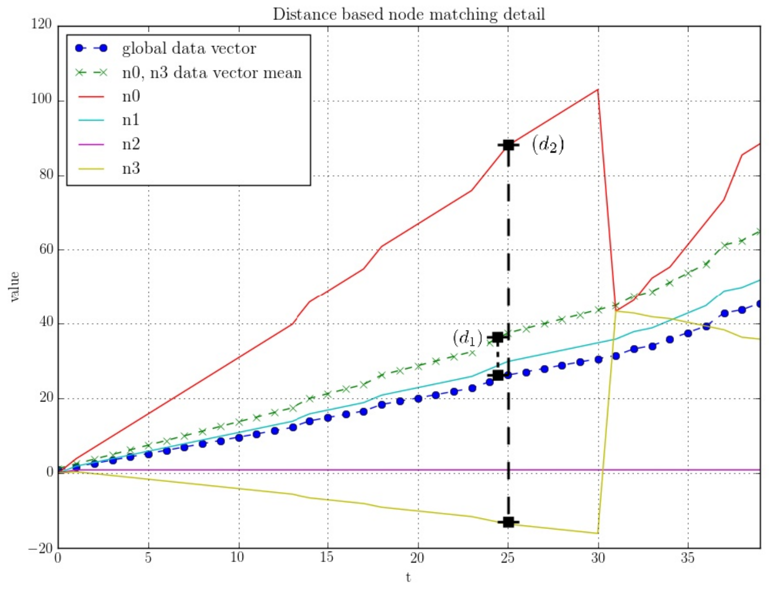


Figure 5.3: Detailed depiction of the Geometric Monitoring operation of Figure 5.2. Distance based node matching operating on 4 nodes ($\{n_0, n_1, n_2, n_3\}$), with 1-dimensional data vectors, threshold $T = 100$ and $f(x) = x$ as the monitoring function. Distance d_1 denotes the distance of the data 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 computation, along with shape customization of the local constraints at the nodes according to the node's needs [4]. Local constraint customization served as the basis for the now popular *Safe-Zone* framework [1, 5], 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 [**?**], preserving the essence of the admissible region and retaining the balancing process of the coordinator based scenario.

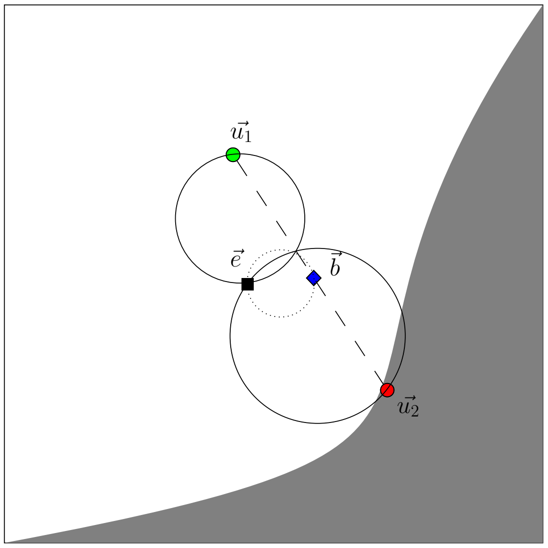
balancing process in original gm(1 paragraph)

why the original method looses points

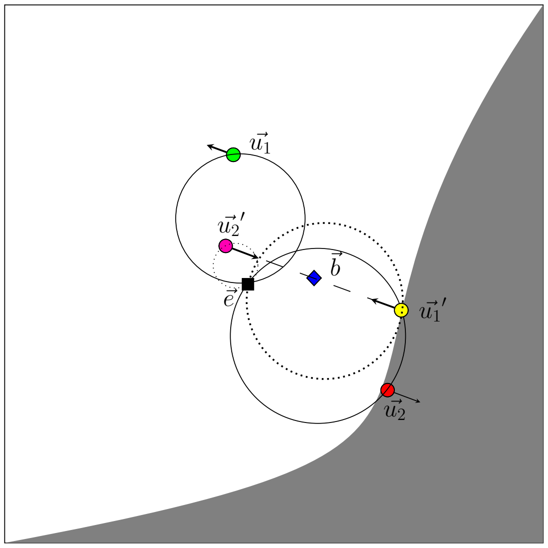
mention estimate vector moving and custom shapes as attempts to optimize balance vector computation

our approach: optimaly position new drift vectors to maxinize the time until a local violation. use simple velocity and acceleration estimations of drift vectors to enhance results.

the heuristic formula and explain



(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

images of optimal positions in 2-d (3-d?)

sketch of convex hull and balls and otimal positions vs traditional positioning (2-d)

analyze the method (bilevel multiobjective optimization), write the formulas of optimization (stackoverflow style)

mention solvers used

algorithm

5.3.1 Smoothing, Velocity and Acceleration Estimation via Savitzky-Golay

why S-G: smoothing and derivation at the same time

precomputed coefficients (real time)

easy implementation

customizable(window and order)

S-G implementation and algorithm

Example images of same data smoothed, velocity and acceleration (3 figures or all in one?)

5.4 Implementation Challenges

training data

can be overcome, worst case scenario our algorithm operates just like the original GM

complexity of optimization (i.e. optimal point location)

complexity of optimization (i.e. node matching)

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] 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

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

[4] 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.

[5] 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

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