

# 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

(30 pages)

# Public Abstract

BLAH BLAH

# Acknowledgments

my mum

# Contents

	Page
Abstract . . . . .	iii
Public Abstract . . . . .	iv
Acknowledgments . . . . .	v
List of Tables . . . . .	viii
List of Figures . . . . .	ix
 <b>I INTRODUCTION AND PRELIMINARIES</b>	 <b>1</b>
1 Introduction . . . . .	2
1.1 Overview . . . . .	2
1.2 Motivation . . . . .	2
1.3 Contributions . . . . .	2
1.4 Thesis Outline . . . . .	2
2 Theoretical Background . . . . .	3
2.1 Geometric Monitoring of Distributed Streams . . . . .	3
2.2 Multiobjective Optimization . . . . .	5
2.3 Savitzky-Golay Filtering . . . . .	5
2.4 Maximum Weight Matching in Graphs . . . . .	6
3 Related Work . . . . .	7
 <b>II PROBLEM DEFINITION AND IMPLEMENTATION</b>	 <b>8</b>
4 Problem Statement . . . . .	9
5 Implementation . . . . .	10
5.1 Geometric Monitoring Implementation . . . . .	10
5.2 Distance Based Node Matching . . . . .	12
5.3 Heuristic Balancing . . . . .	13
5.4 Implementation Challenges . . . . .	14
 <b>III RESULTS AND CONCLUSIONS</b>	 <b>15</b>
6 Experimental Results . . . . .	16
6.1 Experimental Setting . . . . .	16
6.2 Distance Based Node Matching . . . . .	16

6.3	Heuristic Balancing . . . . .	16
6.4	Overall Results . . . . .	16
7	Conclusions and Future Work . . . . .	18
7.1	Conclusions . . . . .	18
7.2	Future Work . . . . .	18
	References . . . . .	19
	Appendix . . . . .	20
	Chapter A Geometric Monitoring Python Implementation . . . . .	21
A.1	Python . . . . .	21
A.2	Numpy and Scipy . . . . .	21
A.3	Openopt . . . . .	21
A.4	NetworkX . . . . .	21
A.5	Putting It All Together . . . . .	21

# List of Tables

Table	Page
-------	------



# List of Figures

Figure

Page

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

---

```
1 begin
2   initialization;
3   repeat
4     forall Monitoring_Nodes do
5       node.data_update();
6       node.compute_drift_vector();
7     end
8     forall Monitoring_Nodes do
9       node.check_for_violation();
10      if Local_Violation then
11        node.report();
12        coordinator.balance();
13      end
14    end
15  until Global_Violation;
16 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  $i$ ,  $w_i = 1$ , so that the bounding ball  $B(\vec{e}(t), \vec{u}_i(t))$  is not monochromatic, and the randomly requested node  $j$ ,  $w_j = 1$ , so that the newly formed bounding ball is  $B(\vec{e}(t), \frac{\vec{u}_i(t) + \vec{u}_j(t)}{2})$ . 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 ??) reduces the communication overhead of false alarms, with the vast majority of violation resolutions requiring only the specified 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 with a different, distance based, criterion for pairing nodes into disjoint pairs and creating the hierarchical structure.

our method:

describe in detail, mentioning that its the same process as in deligiannakis

\*image of disjoint pairs

mention our criterion for matching

\*image of matching results on classic balancing method

emphasize global statistics vector following, emphasize matching of nodes on the extremes (i.e. higher velocity one with lower velocity one)

algorithm

## 5.3 Heuristic Balancing

balancing process geometric interpretation

\*image

balancing process algorithm

### 5.3.1 Smoothing, Velocity and Acceleration Estimation via Savitzky-Golay

S-G implementation

\*image

## 5.4 Implementation Challenges

training data

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] I. Sharfman, A. Schuster, and D. Keren, “Shape sensitive geometric monitoring,” in *Proceedings of the Twenty-seventh ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, ser. PODS ’08. New York, NY, USA: ACM, 2008, pp. 301–310. [Online]. Available: <http://doi.acm.org/10.1145/1376916.1376958>

# 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