Monitoring Distributed, Heterogeneous Data Streams: the Emergence of Safe Zones

Daniel Keren[1], Guy Sagy[2], Amir Abboud[2], David Ben-David[2], Assaf Schuster[2], Izchak Sharfman[2], and Antonios Deligiannakis[3]

[1]Department of Computer Science, Haifa University
[2]Faculty of Computer Science, Israeli Institute of Technology
[3]Department of Electronic and Computer Engineering, Technical Univers ity of Crete

Abstract. In many emerging applications, the data to be monitored is of very high volume, dynamic, and distributed, making it infeasible to collect the distinct data streams to a central node and process them there. Often, the monitoring problem consists of determining whether the value of a global function, which depends on the union of all streams, crossed a certain threshold. A great deal of effort is directed at reducing communication overhead by transforming the monitoring of the global function to the testing of *local* constraints, checked independently at the nodes. Recently, *geometric monitoring* (GM) proved to be very useful for constructing such local constraints for general (non-linear, non-monotonic) functions. Alas, in all current variants of geometric monitoring, the constraints at all nodes share an identical structure and are, thus, unsuitable for handling heterogeneous streams, which obey different distributions at the distinct nodes. To remedy this, we propose a general approach for geometric monitoring of heterogeneous streams (HGM), which defines constraints tailored to fit the distinct data distributions at the nodes. While optimally selecting the constraints is an NP-hard problem, we provide a practical solution, which seeks to reduce running time by hierarchically clustering nodes with similar data distributions and then solving more, but simpler, optimization problems. Experiments are provided to support the validity of the proposed approach.

1 Introduction

For a few years now, processing and monitoring of distributed streams has been emerging as a major effort in data management, with dedicated systems being developed for the task [4]. This paper deals with *threshold queries* over distributed streams, which are defined as "retrieve all items x for which $f(x) \leq T$", where $f()$ is a scoring function and T some threshold. Such queries are the building block for many algorithms, such as top-k queries, anomaly detection, and system monitoring. They are also applied in important data processing and data mining tools, including feature selection, decision tree construction, association rule mining, and computing correlations.

Geometric monitoring (GM) [22, 5, 11, 15] has been recently proposed for handling such threshold queries over distributed data. While a more detailed presentation is deferred until Section 2, we note that GM can be applied to the important case of scoring functions $f()$ evaluated at the average of dynamic data vectors $v_1(t), \ldots, v_n(t)$, maintained at n distributed nodes. Here, $v_i(t)$ is an m-dimensional data vector, often denoted as *local vector*, at the i-th node N_i at time t (often t will be suppressed). In a nutshell, each node monitors a convex subset, often referred to as the node's *safe-zone*, of the *domain* of these data vectors, as opposed to their *range*. What is guaranteed in GM is that the global function $f()$ will not cross its specified threshold as long as all data vectors lie within their corresponding safe-zones. Thus, each node remains silent as long as its data vector lies within its safe zone. Otherwise, in case of a safe-zone breach, communication needs to take place in order to check if the function has truly crossed the given threshold.

The geometric technique can support any scoring function $f()$, evaluated at the average of the data vectors.

A crucial component for reducing the communication required by the geometric method is the design of the safe-zone in each node. Nodes remain silent as long as their local vectors remain within their safe-zone. Thus, good safe-zones increase the probability that nodes will remain silent, while also guaranteeing correctness: a global threshold violation cannot occur unless at least one node's local vector lies outside the corresponding node's safe-zone.

However, prior work on geometric monitoring has failed to take into account the nature of heterogeneous data streams, in which the data distribution of the local vectors at different nodes may vary significantly. This has led to a uniform treatment of all nodes, independently of their characteristics, and the assignment of identical safe-zones (i.e., of the same shape and size) to all nodes.

As we demonstrate in this paper, designing safe-zones that take into account the data distribution of nodes can lead to efficiently monitoring threshold queries at a fraction (requiring an order of magnitude fewer messages) of what prior techniques achieve. However, designing different safe-zones for the nodes is by no means an easy task. In fact, the problem is NP-hard (proof omitted due to lack of space). We thus propose a more practical solution that hierarchically clusters nodes, based on the similarity of their data distributions, and then seeks to solve many small (and easier) optimization problems.

Hereafter we denote our proposed method for geometric monitoring of heterogeneous streams as **HGM**, in contrast to prior work on geometric monitoring that is denoted **GM**.

2 Related Work

Space limitations allow us to only survey previous work on geometric monitoring (GM). We now describe some basic ideas and concepts of the GM technique, which was introduced and applied to monitor distributed data streams in [22, 23].

As described in Section 1, each node N_i maintains a local vector v_i, while the monitoring function $f()$ is evaluated at the average v of the v_i vectors. Before the monitoring process, each node N_i is assigned a subset of the data space, denoted as S_i – its *safe-zone* – such that, as long as the local vectors are inside their respective safe-zones, it is guaranteed that the global function's value did not cross the threshold; thus the node remains silent as long as its local vector v_i is inside S_i. If $v_i \notin S_i$ (local violation), a violation recovery ("balancing") algorithm [22] can be applied.

For details and scope of GM see [15] and the survey in [6]. Recently, GM was successfully applied to detecting outliers in sensor networks [5], extended to prediction-based monitoring [11], and applied to other monitoring problems [16, 18, 10].

Basic definitions relating to GM. A basic construct is the *admissible region*, defined by $A \triangleq \{v|f(v) \leq T\}$. Since the value we wish to monitor is $f\left(\frac{v_1+...+v_n}{n}\right)$, any viable assignment of safe-zones must satisfy $\bigwedge_{i=1}^{n}(v_i \in S_i) \rightarrow v = (v_1 + ... + v_n)/n \in A$. This guarantees that as long as all nodes are silent, the average of the v_i vectors remains in A and, therefore, the function has not crossed the threshold. The question is, of course, how to determine the safe-zone S_i of each node N_i; in a sense to be made precise in Section 2.1, it is desirable for the safe-zones to be as large as possible.

In [15] it was proved that all existing variants of GM share the following property: each of them defines some convex subset C of A (different methods induce different C's), such that each safe-zone S_i is a translation of C – that is, there exist vectors u_i $(1 \leq i \leq n)$ such that $S_i = \{u_i + c|c \in C\}$ and $\sum_{i=1}^{n} u_i = 0$. This observation unifies the distinct variants of GM, and also allows to easily see why $\bigwedge_{i=1}^{n}(v_i \in S_i)$ implies that $v \in C \subseteq A$ – it follows immediately from the fact that convex subsets are closed under taking averages and from the fact that the u_i vectors sum to zero.

Here we assume that C is given; it can be provided by any of the abovementioned methods. We propose to extend previous work in a more general direction. Our goal here is to handle a basic problem which haunts all the existing GM variants: *the shapes of the safe-zones at different nodes are identical.* Thus, if the data is heterogeneous across the distinct streams (an example is depicted in Figure 1), meaning that the data at different nodes obeys different distributions, existing GM algorithms will perform poorly, causing many local violations that do not correspond to global threshold crossing ("false alarms").

2.1 Safe-Zone Design

In this paper, we present a more general approach that allows to assign *differently shaped* safe-zones to different nodes. Our approach requires tackling a difficult

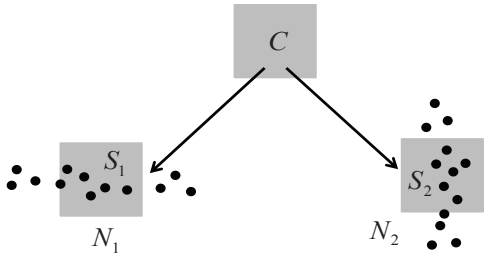

Fig. 1. Why GM may fail for heterogeneous streams. Here C is equal to a square, and the data distribution at the two nodes is schematically represented by samples. In GM, the safe-zones at both nodes are restricted to be a translation of C, and thus cannot cover the data; HGM will allow much better safe-zones (see Section 2.1, and Figure 2).

optimization problem, for which practical solutions need to be devised. We now seek to formulate an optimization problem, whose solution defines the safe-zones at all nodes. The safe-zones should satisfy the following properties:

Correctness: If S_i denotes the safe-zone at node N_i, we must have: $\bigwedge_{i=1}^{n} (v_i \in S_i) \rightarrow (v_1 + ... + v_n)/n \in C$. This ensures that every threshold crossing by $f(v)$ will result in a safe-zone breach in at least one node.

Expansiveness: Every safe-zone breach (local violation) triggers communication, so the safe-zones should be as "large" as possible. We measure the "size" of a safe-zone S_i by its probability volume, defined as $\int_{S_i} p_i(v)dv$ where p_i is the pdf of the data at node N_i. Probabilistic models have proved useful in predicting missing and future stream values in various monitoring and processing tasks [7, 24, 13], including previous geometric methods [15], and their incorporation in our algorithms proved useful in monitoring real data (Section 4). To handle these two requirements, we formulate a constrained optimization problem as follows:

Given:(1) probability distribution functions $p_1, \ldots, p_n$ at n nodes
(2) A convex subset C of the admissible region A

$$\text{Maximize} \int_{S_1} p_1 dv_1 \cdot ... \cdot \int_{S_n} p_n dv_n \quad \text{(expansiveness)}$$

$$\text{Subject to } \frac{S_1 \oplus \cdots \oplus S_n}{n} \subseteq C \quad \text{(correctness)}$$

where $\frac{S_1 \oplus \cdots \oplus S_n}{n} = \left\{ \frac{v_1 + \cdots + v_n}{n} \,|v_1 \in S_1, \ldots, v_n \in S_n \right\}$, or the *Minkowski sum* [20] of $S_1, \ldots, S_n$, in which every element is divided by n (the *Minkowski average*). Introducing the Minkowski average is necessary in order to guarantee correctness, since v_i must be able to range over the entire safe-zone S_i. Note that instead of using the constraint $\frac{S_1 \oplus ... \oplus S_n}{n} \subseteq A$, we use $\frac{S_1 \oplus ... \oplus S_n}{n} \subseteq C$. This preserves correctness, since $C \subseteq A$. The reason we chose to use C is that typically it's

much easier to check the constraint for the Minkowski average containment in a convex set; this is discussed in Section 3.4.

To derive the target function $\int_{S_1} p_1 dv_1 \cdot \ldots \cdot \int_{S_n} p_n dv_n$, which estimates the probability that the local vectors of all nodes will remain in their safe-zones, we assumed that the data is not correlated between nodes (hence we multiply the individual probabilities), as it was the case in the experiments in Section 4 (see also [24] and the discussion therein). If the data is correlated, the algorithm is essentially the same, with the expression for the probability that data at some node breaches its safe-zone modified accordingly.

Note that correctness and expansiveness have to reach a "compromise": figuratively speaking, the correctness constraint restricts the size of the safe-zones, while the probability volume increases as the safe-zones become larger. This trade-off is central in the solution of the optimization problem.

The advantage of the resulting safe-zones is demonstrated by a schematic example (Figure 2), in which C and the stream pdfs are identical to those in Figure 1. In HGM, however, the individual safe-zones can be shaped very differently from C, allowing a much better coverage of the pdfs, while adhering to the correctness constraint. Intuitively speaking, nodes can trade "geometric slack" between them; here S_1 trades "vertical slack" for "horizontal slack".

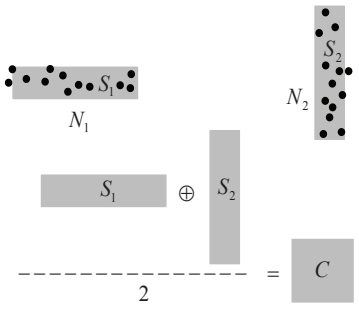

Fig. 2. Schematic example of HGM safe-zone assignment for two nodes, which also demonstrates the advantage over previous work. The convex set C is a square, and the pdf at the left (right) node is uniform over a rectangle elongated along the horizontal (vertical) direction. HGM can handle this case by assigning the two rectangles S_1, S_2 as safe-zones, which satisfies the correctness requirement (since their Minkowski average is equal to C). GM (Figure 1) will perform poorly in this case.

3 Constructing the Safe-Zones

We now briefly describe the overall operation of the distributed nodes. The computation of the safe-zones is initially performed by a coordinator node, using

a process described in this section. This process is performed infrequently, since there is no need to change the safe-zones of a node unless a global threshold violation occurs. As described in Section 2.1, the input to the algorithm is: (1) The probability distribution functions $p_1, \ldots, p_n$ at the n nodes. These pdfs can be of any kind (e.g., Gaussian [23], random walk [21], uniform, etc). (2) A convex subset C of the admissible region A.

Given this input, the coordinator applies the algorithm described in Sections 3.1 to 3.5 to compute $S_1 ... S_n$ which solve the optimization problem defined in Section 2.1. Then, node N_k is assigned S_k.

3.1 Solving the Optimization Problem

In order to efficiently solve our optimization problem, we need to answer several questions:

- What kinds of shapes to consider for candidate safe-zones?
- The target function is defined as the product of integrals of the respective pdfs on the candidate safe-zones. Given candidate safe-zones, how do we efficiently compute the target function?
- Given candidate safe-zones, how do we efficiently test if their Minkowski average lies in C?
- As we will point out, the number of variables to optimize over is very large, with this number increasing with the number of nodes. It is well-known that the computational cost of general optimization routines increases at a super-linear rate with the number of variables. To remedy this issue, we propose in Section 3.5 a hierarchical clustering approach, which uses a divide-and-conquer algorithm to reduce the problem to that of recursively computing safe-zones for small numbers of nodes.

3.2 Shape of Safe-Zones to Consider

The first step in solving an optimization problem is determining the parameters to optimize over. Here, the space of parameters is huge – *all* subsets of the Euclidean space are candidates for safe-zones. For one-dimensional (scalar) data, intervals provide a reasonable choice for safe-zones, but for higher dimensions no clear candidate exists.

To achieve a practical solution, we choose the safe-zones from a parametric family of shapes, denoted by S. This family of shapes should satisfy the following requirements:

- It should be broad enough so that its members can reasonably approximate every subset which is a viable candidate for a safe-zone.
- The members of S should have a relatively simple shape. In practice, this means that they are polytopes with a restricted number of vertices, or can be defined by a small number of implicit equations (e.g., polynomials [14]).
- It should not be too difficult to compute the integral of the various pdfs over members of S (Section 3.3).

- It should not be too difficult to compute, or bound, the Minkowski average of members of S (Section 3.4).

The last two conditions allow efficient optimization. If computing the integrals of the pdf or the Minkowski average are time consuming, the optimization process may be lengthy. We thus considered and applied in our algorithms various polytopes (such as triangles, boxes, or more general polytope) as safe-zones; this yielded good results in [15].

The choices of S applied here have provided good results in terms of safe-zone simplicity and effectiveness. However, the challenge of choosing the best shape for arbitrary functions and data distributions is quite formidable, and we plan to continue studying it in the future.

3.3 Computing the Target Function

The target function is defined as the product of integrals of the respective pdfs on the candidate safe-zones. Typically, data is provided as discrete samples. The integral can be computed by first approximating the discrete samples by a continuous pdf, and then integrating it over the safe-zone. We used this approach, fitting a GMM (Gaussian Mixture Model) to the discrete data and integrating it over the safe-zones, which were defined as polytopes. To accelerate the computation of the integral, we used Green's Theorem to reduce a double integral to a one-dimensional integral over the polygon's boundary, for the two-dimensional data sets in the experiments. For higher dimensions, the integral can also be reduced to integrals of lower dimensions, or computed using Monte-Carlo methods.

3.4 Checking the Constraints

A simple method to test the Minkowski sum constraint relies on the following result [9]:

Lemma 1. *If P and Q are convex polytopes with vertices $\{P_i\}$, $\{Q_j\}$, then $P \oplus Q$ is equal to the convex hull of the set $\{P_i + Q_j\}$.*

Now, assume we wish to test whether the Minkowski average of P and Q is contained in C. Since C is convex, it contains the convex hull of every of its subsets; hence it suffices to test whether the points $(P_i + Q_j)/2$ are in C, for all i, j. If not all points are inside C, then the constraint violation can be measured by the maximal distance of a point $(P_i + Q_j)/2$ from C's boundary. The method easily generalizes to more polytopes: for three polytopes it is required to test the average of all triplets of vertices, etc.

3.5 Hierarchical Clustering

While the algorithms presented in Sections 3.3-3.4 reduce the running time for computing the safe-zones, our optimization problem still poses a formidable difficulty. For example, fitting octagonal safe-zones [15] to 100 nodes with two-dimensional data requires to optimize over 1,600 variables (800 vertices in total,

each having two coordinates), which is quite high. To alleviate this problem, we first organize the nodes in a hierarchical structure, which allows us to then solve the problem recursively (top-down) by reducing it to sub-problems, each containing a much smaller number of nodes.

We first perform a bottom-up hierarchical clustering of the nodes. To achieve this, a distance measure between nodes needs to be defined. Since a node is represented by its data vectors, a distance measure should be defined between subsets of the Euclidean space. We apply the method in [8], which defines the distance between sets by the L^2 distance between their moment vectors (vectors whose coordinates are low-order moments of the set). The moments have to be computed only once, in the initialization stage. The leaves of the cluster tree are individual nodes, and the inner vertices can be thought of as "super nodes", each containing the union (Minkowski average) of the data of nodes in the respective sub-tree. Since the moments of a union of sets are simply the sum of the individual sets' moments, the computation of the moment for the inner nodes is very fast.

After the hierarchical clustering is completed, the safe-zones are assigned top-down: first, the children of the root are assigned safe-zones under the constraint that their
Minkowski average is contained in C. In the next level, the grandchildren of the root are assigned safe-zones under the constraint that their Minkowski average is contained in their parent nodes' safe-zones, etc. The leaves are either individual nodes, or clusters which are uniform enough and can all be assigned safe-zones with identical shapes.

4 Experiments

HGM was implemented and compared with the GM method, as described in [15], which is the most recent variant of previous work on geometric monitoring that we know of. We are not aware of other algorithms which can be applied to monitor the functions treated here (the ratio queries in [12] deal with accumulative ratios and not instantaneous ones as in our experiments).

4.1 Data, Setup and Monitored Functions

Data and Monitored Functions Our data consists of air pollutant measurements taken from "AirBase – The European Air Quality Database" [3], measured in micrograms per cubic meter. Nodes correspond to sensors at different geographical locations. The data at different nodes greatly varies in size and shape and is irregular as a function of time. The monitored functions were chosen due to their practical importance, and also as they are non-linear and non-monotonic and, thus, cannot be handled by most existing methods. In Section 4.2 results are presented for monitoring the ratio of NO to NO_2, which is known to be an important indicator in air quality analysis [17]. An example of monitoring a quadratic function in three variables is also presented (Section 4.3);

quadratic functions are important in numerous applications (e.g., the variance is a quadratic function in the variables, and a normal distribution is the exponent of a quadratic function, hence thresholding it is equivalent to thresholding the quadratic).

Choosing the Family of Safe-Zones To solve the optimization problem, it is necessary to define a parametric family of shapes S from which the safe-zones will be chosen. Section 3.2 discusses the properties this family should satisfy. In [15], the suitability of some families of polytopes is studied for the simpler, but related, problem of finding a safe-zone common to all nodes. The motivations for choosing S here were:

- Ratio queries (Section 4.2) – the triangular safe-zones (Figure 3) have the same structure, but not size or location, as C, and are very simple to define and apply.

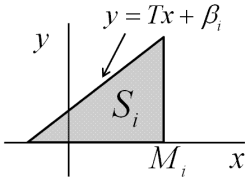

Fig. 3. Triangular safe-zones used for ratio monitoring.

- Quadratic function (Section 4.3) – here we allowed general polytopes, and tested the results for increasing numbers of vertices. The model selected was with 12 vertices, in which the target function to optimize was "saturated" (i.e. adding more vertices increased the value by less than 0.1%).

Optimization Parameters and Tools The triangular safe-zones (Section 4.2) have two degrees of freedom each (M_i and β_i, see Figure 3), hence for n nodes we have $2n$ parameters to optimize over. The safe-zones in Section 4.3 require 36 parameters each. In all cases we used the Matlab routine fmincon to solve the optimization problem [1]. To compute the integral of the pdf on the safe-zones, data was approximated by a Gaussian Mixture Model (GMM), using a Matlab routine [2].

4.2 Ratio Queries

This set of experiments concerned monitoring the ratio between two pollutants, NO and NO_2, measured in distinct sensors. Each of the n nodes holds a vector (x_i, y_i) (the two concentrations), and the monitored function is $\frac{\sum y_i}{\sum x_i}$ (in [12] ratio is monitored but over aggregates over time, while here we monitor the

instantaneous ratio for the current readings). An alert must be sent whenever this function is above a threshold T (taken as 4 in the experiments), and/or when the NO_2 concentration is above 250. The admissible region A is a triangle, therefore convex, so $C = A$. The safe-zones tested were triangles of the form depicted in Figure 3, a choice motivated by the shape of C. The half-planes method (Section 3.4) was used to test the constraints. An example with four nodes, which demonstrates the advantage of allowing different safe-zones at the distinct nodes, is depicted in Figure 4.

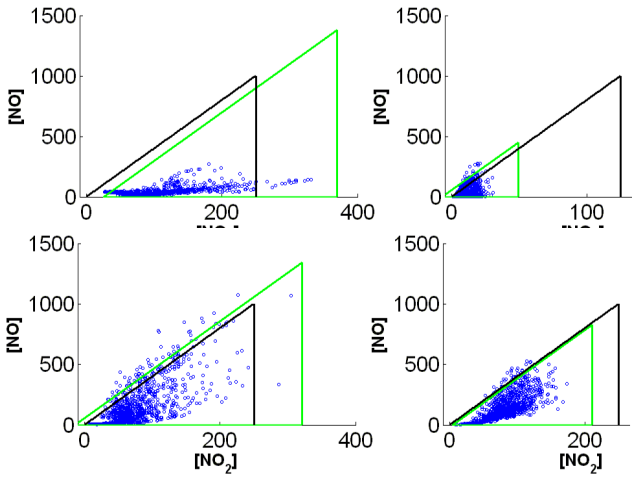

Fig. 4. Example of safe-zones with four nodes. The convex set C is the triangle outlined in black, safe-zones are outlined in green. Nodes with more compact distributions are assigned smaller safe-zones, and nodes with high values of the monitored function (NO/NO_2 ratio) are assigned safe-zones which are translated to the left in order to cover more data. This is especially evident in the top right node, in which the safe-zone is shifted to the left so it can cover almost all the data points. In order to satisfy the Minkowski sum constraint, the safe-zone of top left node is shifted to the right, which in that node hardly sacrifices any data points; also, the larger safe-zones are balanced by the smaller ones. Note that HGM allows safe-zones which are *larger* than the admissible region A, as opposed to previous work, in which the safe-zones are subsets of A.

Improvement Over Previous GM Work. We compared HGM with GM in terms of the number of produced local violations. In Figure 5, the number of safe-zone violations is compared for various numbers of nodes. HGM results in significantly fewer local violations, even for a small number of nodes. As the number of nodes increases, the benefits of HGM over GM increase. For a modest network size of 10 nodes, HGM requires less than an order of magnitude fewer messages than GM.

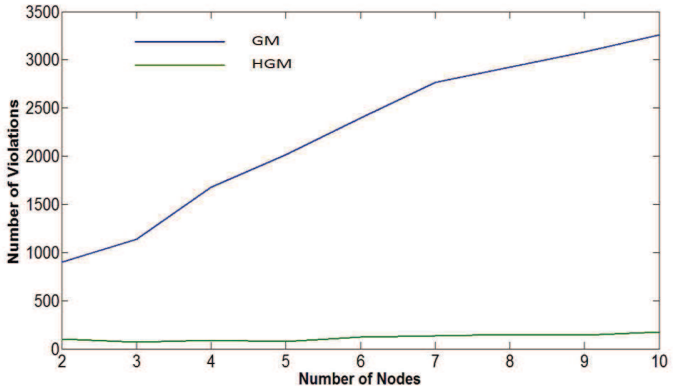

Fig. 5. Comparison of our HGM (green) to GM [15] (blue) in terms of number of violations, up to 10 nodes.

4.3 Monitoring a Quadratic Function

Another example consists of monitoring a quadratic function with more general polyhedral safe-zones in three variables (Figure 6). The data consists of measurements of three pollutants (NO, NO_2, SO_2), and the safe-zones are polyhedra with 12 vertices. The admissible region A is the ellipsoid depicted in pink; since it is convex, $C = A$. As the extent of the data is far larger than A, the safe-zones surround the regions in which the data is denser. Here we did not compare to previous methods.

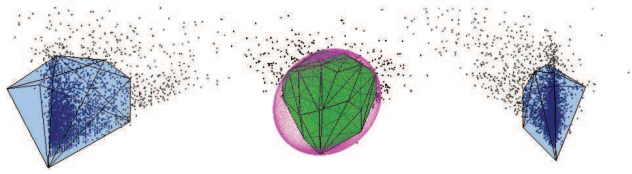

Fig. 6. Monitoring a quadratic function. The set C is the pink ellipsoid, the safe-zones are polyhedra with 12 vertices each (in pale blue), and their Minkowski average is in green.

References

1. http://tinyurl.com/kxssfgl.
2. DCPR (Data Clustering and Pattern Recognition) Toolbox, http://tinyurl.com/nxospq2.
3. The european air quality database. In *http://tinyurl.com/ct9bh7x*.

4. Daniel J. Abadi, Yanif Ahmad, Magdalena Balazinska, Ugur Çetintemel, Mitch Cherniack, Jeong-Hyon Hwang, Wolfgang Lindner, Anurag Maskey, Alex Rasin, Esther Ryvkina, Nesime Tatbul, Ying Xing, and Stanley B. Zdonik. The design of the borealis stream processing engine. In *CIDR*, 2005.
5. Sabbas Burdakis and Antonios Deligiannakis. Detecting outliers in sensor networks using the geometric approach. In *ICDE*, 2012.
6. Graham Cormode. Algorithms for continuous distributing monitoring: A survey. In *AlMoDEP*, 2011.
7. Amol Deshpande, Carlos Guestrin, Samuel Madden, Joseph M. Hellerstein, and Wei Hong. Model-driven data acquisition in sensor networks. In *VLDB*, 2004.
8. M. Elad, A. Tal, and S. Ar. Content based retrieval of vrml objects: an iterative and interactive approach. In *Proceedings of the sixth Eurographics workshop on Multimedia 2001*, 2002.
9. Efi Fogel and Dan Halperin. Exact and efficient construction of minkowski sums of convex polyhedra with applications. *Computer-Aided Design*, 39(11), 2007.
10. Minos N. Garofalakis, Daniel Keren, and Vasilis Samoladas. Sketch-based geometric monitoring of distributed stream queries. *PVLDB*, 2013.
11. Nikos Giatrakos, Antonios Deligiannakis, Minos N. Garofalakis, Izchak Sharfman, and Assaf Schuster. Prediction-based geometric monitoring over distributed data streams. In *SIGMOD*, 2012.
12. Rajeev Gupta, Krithi Ramamritham, and Mukesh K. Mohania. Ratio threshold queries over distributed data sources. In *ICDE*, 2010.
13. Bhargav Kanagal and Amol Deshpande. Online filtering, smoothing and probabilistic modeling of streaming data. In *ICDE*, 2008.
14. Daniel Keren, David B. Cooper, and Jayashree Subrahmonia. Describing complicated objects by implicit polynomials. *IEEE Trans. Pattern Anal. Mach. Intell.*, 16(1), 1994.
15. Daniel Keren, Izchak Sharfman, Assaf Schuster, and Avishay Livne. Shape sensitive geometric monitoring. *IEEE Trans. Knowl. Data Eng.*, 24(8), 2012.
16. J. Kogan. Feature selection over distributed data streams through optimization. In *SDM*, 2012.
17. M.R. Kurpius and A.H. Goldstein. Gas-phase chemistry dominates o3 loss to a forest, implying a source of aerosols and hydroxyl radicals to the atmosphere. *Geophysical Research Letters*, 30(7), 2007.
18. Odysseas Papapetrou, Minos N. Garofalakis, and Antonios Deligiannakis. Sketch-based querying of distributed sliding-window data streams. *PVLDB*, 5(10), 2012.
19. G. Sagy, D. Keren, I. Sharfman, and A. Schuster. Distributed threshold querying of general functions by a difference of monotonic representation. *PVLDB*, 4(2), 2010.
20. J.P. Serra. Image analysis and mathematical morphology. In *Academic Press, London*, 1982.
21. Shetal Shah and Krithi Ramamritham. Handling non-linear polynomial queries over dynamic data. In *ICDE*, 2008.
22. Izchak Sharfman, Assaf Schuster, and Daniel Keren. A geometric approach to monitoring threshold functions over distributed data streams. *ACM Trans. Database Syst.*, 32(4), 2007.
23. Izchak Sharfman, Assaf Schuster, and Daniel Keren. Shape sensitive geometric monitoring. In *PODS*, 2008.
24. Mingwang Tang, Feifei Li, Jeff M. Phillips, and Jeffrey Jestes. Efficient threshold monitoring for distributed probabilistic data. In *ICDE*, 2012.