DEPARTMENT OF INFORMATICS

UNIVERSITY OF PIRAEUS

Monitoring and Mining Distributed Data Streams

PhD thesis

NIKOLAOS GIATRAKOS

BSc in Informatics, U. Piraeus (2006)

MSc in Information Systems, AUEB (2008)

Piraeus, November 2012

DEDICATION

To my father's memory

...completing the above line was the most difficult part of the thesis, I admit.

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

Συμβουλευτική επιτροπή:

Επιβλέπων:
Ιωάννης Θεοδωρίδης,
Αν. Καθηγητής Παν/μίου Πειραιώς

Μέλη:
Μίνως Γαροφαλάκης,
Καθηγητής Πολυτεχνείου Κρήτης

Τιμολέων Σελλής,
Καθηγητής Ε. Μ. Πολυτεχνείου

Πανεπιστήμιο Πειραιώς
Τμήμα Πληροφορικής

Διατριβή

για την απόκτηση
Διδακτορικού Διπλώματος
του Τμήματος Πληροφορικής

ΝΙΚΟΛΑΟΥ ΓΙΑΤΡΑΚΟΥ

«Monitoring and Mining Distributed Data Streams»

(Παρακολούθηση και Εξόρυξη Γνώσης από Κατανεμημένα Ρεύματα Δεδομένων)

Εξεταστική επιτροπή:

Ιωάννης Θεοδωρίδης,
Αν. Καθηγητής Παν/μίου Πειραιώς

Μίνως Γαροφαλάκης,
Καθηγητής Πολυτεχνείου Κρήτης

Τιμολέων Σελλής,
Καθηγητής Ε. Μ. Πολυτεχνείου

Χρήστος Δουληγέρης,
Καθηγητής Παν/μίου Πειραιώς

Θεμιστοκλής Παναγιωτόπουλος,
Καθηγητής Παν/μίου Πειραιώς

Δημήτριος Βέργαδος,

Επίκ. Καθηγητής Παν/μίου Πειραιώς ____________

Αντώνιος Δεληγιαννάκης,

Επίκ. Καθηγητής Πολυτεχνείου Κρήτης ____________

Abstract

Many modern streaming applications, such as online analysis of financial,network, sensor and other forms of data are inherently distributed in nature. Due to the distributed nature of data production in the aforementioned scenarios, the major challenge confronted by algorithms dealing with their manipulation is to reduce communication [21]. This happens because the central collection of data is not feasible in large-scale applications. Furthermore, in the case of sensor deployments, central data accumulation results in depleting the power supply of individual sensors reducing the network lifetime [119, 44, 43, 45].

An important query type that is of the essence in such applications involves a continuous check on the position of a given (arbitrarily complex) function f with respect to a posed threshold T. This monitoring demand may be explicitly placed at the core of applications mission, e.g. in network traffic monitoring scenarios [21, 103] or implicitly stand as an operational component. For instance, while detecting outliers in sensor network settings, motes have to determine the similarity of their samples to those obtained by their neighbors based on a given distance function as well as a chosen similarity threshold.

One approach to achieve the desired communication reduction is to decompose the monitoring problem into local constraints that can be disseminated to the geographically dispersed sites. According to that approach, each site in the network will then have to consult these constraints upon the local dataset is altered. Collecting the data centrally is only required when the local constraint of at least one site is violated [103]. However, the decomposition of the central monitoring problem into a set of local constraints is not always effective. In fact, it may complicate the monitoring processes as well as uncontrollably sacrifice accuracy when functioning over generic network infrastructures such as hierarchical sensor networks where message losses, death or reorganization of nodes affects the network formation [9].

A second approach of performing the monitoring is to allow continuous communication between the necessary network parties but attempt to reduce the bandwidth consumption by applying reduction techniques on the data under transmission. In that, we allow efficient derivation of answers to our tracking procedure by controllably compro-

mising its accuracy. In particular, we aim at inventing techniques capable of providing approximate answers to whether $f > T$ or $f < T$ with predefined accuracy guarantees and simultaneously beware of the fact that the more we reduce the communication cost the looser our accuracy guarantees become.

Regarding the first approach, we focus on monitoring (non-linear) complex functions over distributed data streams. More precisely, in our work [42], we generalize the geometric monitoring approach initially presented in [103] by proposing the adoption of local predictors [22] to be used during the distributed tracking. We present a thorough study regarding prediction models' adoption within the geometric monitoring setting. After identifying the peculiarities exhibited by predictors upon their implementation in the aforementioned environment, we develop a solid theoretic framework composed of sufficient conditions rendering predictors capable of refraining the communication burden. We propose algorithms incorporating those conditions and expand on relaxed versions of them along with extensive theoretical analysis on their expected benefits.

As already noted, the geometric monitoring framework may suffer from inaccuracy in hierarchical sensor network architectures. For instance, its use while trying to detect outliers based on minimum support queries cannot guarantee correctness or provide any predictable approximation [9]. To handle such situations, we reside to the second of the previously discussed approaches and propose an outlier detection framework, namely TACO [44, 45], that trades bandwidth for accuracy in a straightforward manner and supports various similarity metrics (monitored functions of interest).

Eventually, we further elaborate on extensions of the rationales utilized in the previously mentioned approaches. We concentrate on trajectory data streams and perform distributed Representative Trajectory monitoring over a number of monitored objects utilizing the concept of predictors [42]. Additionally, we exploit the properties of the monitored similarity measures used in [44, 45], in the context of detecting movement pattern alterations over streaming movement data [116].

Περίληψη

Πολλές σύγχρονες εφαρμογές ρευμάτων δεδομένων, όπως ανάλυση οικονομικών, δικτυακών, αισθητήρων και άλλων τύπων δεδομένων είναι κατανεμημένης φύσεως. Εξαιτίας της κατανεμημένης φύσης παραγωγής των δεδομένων στα προαναφερθέντα σενάρια, η μεγαλύτερη πρόκληση που αντιμετωπίζουν οι αλγόριθμοι που καλούνται να τα διαχειριστούν είναι η μείωση του κόστους επικοινωνίας [21]. Αυτό συμβαίνει λόγω του ότι η κεντρική συλλογή των δεδομένων δεν είναι εφικτή σε κατανεμημένες εφαρμογές μεγάλης κλίμακας, αφού οδηγεί σε αυξημένη κατανάλωση του εύρους ζώνης των συνδέσμων επικοινωνίας οι οποίοι αργά ή γρήγορα καθίσταντι μη λειτουργικοί. Επιπλέον, σε περιπτώσεις πεδίων εφαρμογής της τεχνολογίας των δικτύων αισθητήρων, η συσσώρευση των δεδομένων κεντρικά έχει ως αποτέλεσμα την εξάντληση της εναπομένουσας ισχύος κάθε συσκευής, μειώνοντας τη διάρκεια ζωής του δικτύου [119, 44, 43, 35].

Ένας σημαντικός τύπος επερωτήσεων που έχει ιδιαίτερο νόημα σε τέτοιες εφαρμογές αφορά το συνεχή έλεγχο της τοποθέτησης της τιμής μιας δοθείσας (οσοδήποτε πολύπλοκης) συνάρτησης *f* σε σχέση με κάποιο τεθέν κατώφλι *T*. Αυτή η απαίτηση παρακολούθησης ενδέχεται να τίθεται ρητά στον πυρήνα της αποστολής κάποιας εφαρμογής κατανεμημένων ρευμάτων δεδομέναν, π.χ. σε σενάρια παρακολούθησης δικτυακής κίνησης [21, 103] ή να αποτελεί λειτουργικό της συστατικό. Επί παραδείγματι, κατά τον προσδιορισμό ακραίων τιμών σε περιβάλλοντα ασύρματων δικτύων αισθητήρων, οι κόμβοι αισθητήρες πρέπει να αποφασίσουν την ομοιότητα των μετρήσεών τουςμε αυτές που έχουν δειγματιστεί από τους γείτονές τους βάσει κάποιας δοθείσας συνάρτησης και ενός επιλεχθέντος κατωφλιού ομοιότητας.

Μια προσέγγιση για να επιτύχη κανείς την επιθυμητή μείωση στην επικοινωνία, είναι η αποσύνθεση του προβλήματος της παρακολούθησης των ρευμάτων

δεδομένων, σε τοπικούς περιορισμούς που μπορούν να δοθούν στις, γεωγραφικά κατανεμημένες, πηγές δεδομένων. Σύμφωνα με αυτή την προσέγγιση, κάθε πηγή δεδομένων θα πρέπει έπειτα να συμβουλεύεται αυτούς τους περιορισμούς σε κάθε αλλαγή του ρεύματος δεδομένων που καταφθάνει τοπικά. Η κεντρική συλλογή των δεδομένων χρειάζεται μόνο όταν παραβιάζεται ο περιορισμός που έχει τεθεί τοπικά σε κάποια πηγή [103]. Παρόλα αυτά, η αποσύνθεση του προβλήματος της παρακολούθησης των ρευμάτων δεδομένων σε ένα σύνολο τοπικών περιορισμών δεν είναι πάντα αποτελεσματική. Στην πραγματικότητα, μπορεί να περιπλέξει τη διαδικασία παρακολούθησης καθώς και να θυσιάζει ανεξέλεγκτα την ακρίβεια της όταν λειτουργεί σε λιγότερο απλές δικτυακές υποδομές όπως ιεραρχικά δίκτυα αισθητήρων όπου απώλειες μηνυμάτων, θάνατος και αναδιοργάνωση των κόμβων του δικτύου μπορεί να λάβουν χώρα [9].

Μια δεύτερη προσέγγιση για την επιτέλεση της παρακολούθησης είναι να επιτραπεί η συνεχείς επικοινωνίας μεταξύ των απαραίτητων δικτυακών μερών αλλά να γίνει προσπάθεια μείωσης της κατανάλωσης του αντίστοιχου εύρους ζώνης με εφαρμογή τεχνικών μείωσης των δεδομένων που πρόκειται να μεταδοθούν. Έτσι, επιτρέπουμε την αποδοτική παροχή απαντήσεων στις διαδικασία παρακολούθησης παράλληλα θυσιάζοντας μέρος της ακρίβειας τους με ελεγχόμενο όμως τρόπο. Συγκεκριμένα, στοχεύουμε στην ανάπτυξη τεχνικών ικανών να παρέχουν προσεγγιστικές απαντήσεις σε ότι αφορά το αν $f > T$ ή $f < T$ με προκαθορισμένες εγγυήσεις ακρίβειας και ταυτόχρονα είμαστε ενήμεροι για το γεγονός ότι όσο περισσότερο μειώνουμε το απαιτούμενο φόρτο επικοινωνίας τόσο χαλαρώνουμε τις εγγυήσεις ακρίβειας των εξαγόμενων απαντήσεων.

Σε ότι αφορά την πρώτη από τις παραπάνω προσεγγίσεις, επικεντρωνόμαστε στην παρακολούθηση (μη γραμμικών) πολύπλοκων συναρτήσεων επί κατανεμημένων ρευμάτων δεδομένων. Πιο συγκεκριμένα, στην εργασία μας [42], γενικεύουμε την προσέγγιση της γεωμετρικής παρακολούθησης που αρχικά παρουσιάστηκε στο [103], προτείνοντας την υιοθέτηση τοπικών μοντέλων πρόβλεψης [22] κατάλληλων να χρησιμοποιηθούν κατα την κατανεμημένη παρακολούθηση. Παρουσιάζουμε διεξοδική μελέτη σχετικά με την υιοθέτηση τέτοιων μοντέλων πρόβλεψης στο πλαίσιο της γεωμετρικής μεθόδου. Αφού προσδιορίσουμε τις ιδιαιτερότητες που παρουσιάζουν τα μοντέλα πρόβλεψης όταν υλοποιηθούν στο προαναφερθέν περιβάλλον, αναπτύσσουμε σχετικό θεωρητικό πλαίσιο αποτελούμενο απο επαρκής συνθήκες που καθιστούν τα μοντέλα πρόβλεψης ικανά να συγκρατήσουν το φόρτο επικοινωνίας. Προτείνουμε αλγορίθμους που ενσωματώνουν αυτές τις συνθήκες και επεκτεινόμαστε σε χαλαρές εκδόσεις των συνθηκών με ταυτόχρονη θεωρητική ανάλυση των αναμενόμενων οφελών τους.

Όπως ήδη σημειώθηκε, το πλαίσιο γεωμετρικής παρακολούθησης μπορεί να οδηγήσει σε έλλειψη ακρίβειας εντός ιεραρχικών αρχιτεκτονικών δικτύων αισθητήρων. Για παράδειγμα, η χρήση του για τον προσδιορισμό ακραίων τιμών βάσει ερωτημαάτων ελάχιστης υποστήριξης δεν μπορεί να εγγυηθεί ορθότητα ή να παρέχει προσέγγιση με προβλέψιμο περιθώριο σφάλματος [9]. Προκειμένου να χειριστούμε τέτοιες περιπτώσεις, καταλήγουμ στη δεύτερη από τις προαναφερθείσες προσεγγίσεις και προτείνουμε ένα πλαίσιο προσδιορισμού ακραίων τιμών, με όνομα TACO [44, 45], το οποίο είναι ικανό να συναλλάσει ευθέως την κατανάλωση εύρους ζώνης με την ακρίβεια στον προσδιορισμό των ακραίων τιμών και μπορεί να ενσωματώσει πληθώρα μέτρων ομοιότητας (παρακολούθηση συναρτήσεων που μας ενδιαφέρουν).

Εν κατακλείδι, επιπρόσθετα στα προηγούμενα αναφερόμαστε σε επεκτάσεις των προηγούμενων λογικών. Επικεντρωνόμενοι σε ρεύματα δεδομένων τροχιών κινούμενων αντικειμένων, πραγματοποιούμε κατανεμημένη παρακολούθηση Αντιπροσωπευτικών Τροχιών επί ενός παριθμού παρακολουθούμενων, κινούμενων αντικειμένων χρησιμοποιώντας έννοιες των μοντέλων πρόβλεψης [42].

Επιπλέον, εκμεταλλευόμαστε τις ιδιότητες των μέτρων ομοιότητας που χρησιμοποιήθηκαν στα [44, 45], για τον εντοπισμό αλλαγών στο μοτίβο κινούμενων αντικειμένων, μέσω των αντίστοιχων ρευμάτων δεδομένων του τρόπου κίνησής τους [116].

Acknowledgments

First, I would like to thank a number of academic persons I had the opportunity to meet during my studies. I choose to do so by chronological order of acquaintance.

Initially, I would like to thank my advisor, Professor Yannis Theodoridis, for the continuous inspiration throughout my studies. During my early years as an undergraduate student, I found his exceptional teaching abilities critical in unfolding the charming world of database related concepts before me. Later on, during my Ph.D., he was always glad to share his priceless experience and knowledge which constitute integral elements of the current thesis.

Special thanks to Assoc. Professor Yannis Kotidis. As my master thesis advisor he exhibited endless patience in communicating the principles a hatching researcher should get to know. I consider the opportunity to work with him as a determinant factor in developing the way of research thinking I possess today. Furthermore, I had the privilege of collaborating with him in most part of this thesis.

I would also like to express my deepest gratitude to Asst. Professor Antonios Deligiannakis. This work in its entirety could not have been fulfilled without his guidance, patience and willingness in criticizing, correcting or even overlooking my imperfections. His help in broadening my horizons, his trust and honest friendship over the years have been invaluable.

Professor Minos Garofalakis is another exceptional researcher and amazing individual that I have had the opportunity to work with and who has provided me with exceptional advice and guidance during the previous two years of my studies. His contribution in the evolution of my research as well as my future career was critical. I would like to thank him for the trust and encouragement he still supports me with.

Separate thanks to the members of the committee, Professor Timos Sellis, Professor Minos Garofalakis, Asst. Professor Antonios Deligiannakis, Professors Christos Douligeris and Themis Panagiotopoulos and Asst. Professor Dimitrios Vergardos for agreeing to serve on my thesis committee and for devoting their precious time to review the current work and provide comments and suggestions.

Over the years I have had the pleasure to cooperate with talented individuals. Zhixian Yan is an amazing person and good friend who I had the opportunity to work with on semantic trajectory extraction issues. Furthermore, I was pleased to collaborate with Izchak Sharfman and Professor Assaf Schuster on our prediction - based geometric monitoring framework development. I also want to thank my colleagues and partners starting with the Information Systems Laboratory (InfoLab) at the University of Piraeus: Elias Frentzos, Evanglelos Katsikaros, Despina Kopanaki, Evangelos Kotsifakos, Gerasimos Marketos, Irene Ntoutsi, Nikos Pelekis and Marios Vodas for the endless working hours and funny moments we shared. Additionally, while visiting the Software Technology and Network Applications Laboratory (SoftNet) at the Technical University of Crete I was delighted to meet Aikaterini Ioannou, Odysseas Papapetrou and Asst. Professor Vasilis Samoladas to whom I give thanks for the fruitful discussions we had together.

During my studies I participated in a number of national as well as European projects. The LIFT (ICT-FP7-LIFT-255951) Project was the funding source of my thesis during the previous two years. Moreover, the FP7/ICT/FET Project MODAP partially supported me during the last three years of my studies. The DIACHORON Project (PEP Attikis) was a good start at my early Ph.D. studies. Finally, by contributing to the FP6/IST Project GEOPKDD, I earned valuable experience and knowledge at the first year of my Ph.D. I give thanks to those in charge of the above research programs.

On a personal level, my close friends were constantly there for me, being willing to tolerate my deeply silent or tense moments (often with turbulent alternations). Nikos, Panagiotis, Aristidis, Tasos and Stavy, I feel fortunate for having you by my side guys.

Finally, I am graceful to my mother for her love, different aspects of support and understanding. No words could express my exact feelings for her.

I frankly apologize to those I have inadvertently left out.

Thank you all!

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Most modern applications continuously receive a huge amount of streaming data from geographically dispersed sources (or sites) such as ATM machines, network routers, sensors, moving objects with GPS enabled devices and so on. On the other hand, the users of these applications are rather concerned with the analysis of those data in an online fashion so as to derive answers in real-time and appropriately trigger decision making procedures. Respective scenarios may include both query monitoring and mining procedures. For instance, a number of sensors placed in a machine room measures the conditions (temperature, humidity, solar radiance) under which machines operate and their mission is to continuously monitor whether the quantities measuring these conditions do not deviate a lot from their expected values. In a similar scenario, a malfunctioning sensor node, whose obtained samples need to be excluded from the query answer, can be detected by having each node in the network checking the similarity it exhibits with the measurements of its neighbors. In the latter case, should a node finds adequate support from its neighbors its acquired values can then be considered valid.

Nonetheless, accomplishing monitoring and mining tasks in such a setting is not trivial. First, operating in a streaming environment yields rapidly changing data distributions in each of the distributed sites so that local data streams vary a lot as time passes. Second, the monitoring and mining processes may involve computations of arbitrarily complex quantities or functions of interest that are defined upon the union of the local data streams. Third, application requirements pose the continuous production of valid answers as an essential part of their function. To further complicate the above situation, a fourth requirement regards bandwidth consumption constraints. In other words, central data collection is not feasible nor desired in distributed streaming settings as it skyrockets the load in the communication links of the underlying network infrastructure.

An important query type that is of the essence in such applications involves a continuous check on the position of a given (arbitrarily complex) function with respect to a posed threshold. This monitoring demand may be explicitly placed at the core of applications mission, e.g. in network traffic monitoring scenarios [21, 103] or implicitly stand as an operational component. As already mentioned, while detecting outliers in sensor network settings, motes have to determine the similarity of their samples to those obtained by their neighbors based on a given distance function as well as a chosen similarity threshold [44, 45, 43, 30].

In the current thesis we tackle with the above issues and present monitoring and mining techniques for distributed streaming settings. One approach to achieve the desired communication reduction is to decompose the monitoring problem into local constraints that can be disseminated to the geographically dispersed sites. According to that approach, each site in the network will then have to consult these constraints upon the local dataset is altered. Collecting the data centrally is only required when the local constraint of at least one site is violated [103].

Regarding the first approach, we focus on monitoring (non-linear) complex functions over distributed data streams. More precisely, in our work [42], we generalize the geometric monitoring approach initially presented in [103] by proposing the adoption of local predictors [22] to be used during the distributed tracking. We present a thorough study regarding prediction models' adoption within the geometric monitoring setting. After identifying the peculiarities exhibited by predictors upon their implementation in the aforementioned environment, we develop a solid theoretic framework composed of sufficient conditions rendering predictors capable of refraining the communication burden. We propose algorithms incorporating those conditions and expand on relaxed versions of them along with extensive theoretical analysis on their expected benefits. More precisely, our contributions are:

- We introduce the adoption of prediction models in the setting of tracking complex, non-linear functions utilizing the geometric approach [103, 105]. We exhibit the way prediction models can be locally adopted by sites and we show the characteristics they attribute to the geometric approach. We then illustrate that the initial geometric monitoring framework of [103, 105] is a special case of our, more general, prediction-based geometric monitoring framework.
- We point out the failure of conventional notions of good predictors to be applied in this setting and manage to establish a solid theoretic framework consisting of sufficient conditions that do render prediction models capable of guaranteeing reduced bandwidth consumption.
- We expose a number of novel tracking mechanisms relaxing the previously (hard to verify in a distributed manner) identified sufficient conditions. Using the simplest possible primitives regarding prediction models' behavior, we thoroughly study the potentials of our new tracking techniques to achieve communication preservation.
- We present an extensive experimental analysis using a variety of real data sets,

parameters and functions of interest. Our evaluation shows that our approaches can provide significant communication load reduction with savings ranging from 2 times and in some cases reaching 3 orders of magnitude compared to the transmission cost of the original bounding algorithm.

- We provide extensions of the prediction - based geometric monitoring framework for the special case of monitoring representative trajectories over spatiotemporal data streams produced by a number of tracked moving objects.

However, the decomposition of the central monitoring problem into a set of local constraints is not always effective. In fact, it may complicate the monitoring processes as well as uncontrollably sacrifice accuracy when functioning over generic network infrastructures such as hierarchical sensor networks where message losses, death or reorganization of nodes affects the network formation. In particular, [9] comments that when the geometric approach, utilized in the monitoring part of our study, comes to perform outlier detection in generic sensor architectures, it may sacrifice the accuracy of the techniques beyond control.

A second approach of performing this mining task, which involves monitoring similarity functions of pairs of sensor nodes, is to allow continuous communication between the necessary network parties but attempt to reduce the bandwidth consumption by applying reduction techniques on the data under transmission. In that, we allow efficient derivation of answers to our tracking procedure by controllably compromising its accuracy. In particular, we aim at inventing techniques capable of providing approximate answers to whether a similarity measure among sensor readings exceeds a given threshold with predefined accuracy guarantees and simultaneously beware of the fact that the more we reduce the communication cost the looser our accuracy guarantees become. Hence, we employ this second approach and propose an outlier detection framework, namely TACO [44, 45]. Our contributions to the task of mining outliers in sensor network architectures are as follows:

- We present TACO, an outlier detection framework that trades bandwidth for accuracy in a straightforward manner. TACO supports various popular similarity measures used in different application areas. Examples of such measures include, but are not limited to, the cosine similarity, the correlation coefficient and the Jaccard coefficient.
- We present an extensive theoretical study on the trade offs occurring between bandwidth and accuracy during TACO's operation.
- We subsequently devise a boosting process that provably improves TACO's accuracy under no additional communication costs.
- We devise novel load balancing and comparison pruning mechanisms, which alleviate certain (leading) nodes from excessive processing and communication load. These mechanisms result in a more uniform, power consumption and prolonged

network unhindered operation, since the more evenly spread power consumption results in an infrequent need for network reorganization.

- We present a detailed experimental analysis of our techniques for a variety of data sets and parameter settings. Our results demonstrate that our methods can reliably compute outliers, while at the same time significantly reducing the amount of transmitted data, with average recall and precision values exceeding 80% and often reaching 100%. It is important to emphasize that the above results often correspond to bandwidth consumptions that are lower than what is required by a simple continuous aggregate query, using a method like TAG [78]. We also demonstrate that TACO may result in prolonged network lifetime, up to a factor of 3 in our experiments. We further provide comparative results with the recently proposed technique of [30] that uses an equivalent outlier definition and supports common similarity measures. Overall, TACO appears to be more accurate up to 10% in terms of the F-Measure metric while resulting in lower bandwidth consumption.

Motivated by the properties of monitored functions such as the correlation coefficient or the cosine similarity we used for outlier detection, we subsequently study the application of generalized versions of these measures for mining semantic trajectories from spatiotemporal data streams. The aforementioned measures possess the property of detecting similar trends in the values of the encapsulated quantities and can thus serve as the means for detecting movement pattern alterations. These alterations in turn correspond to homogeneous portions of motion that can be semantically annotated.

We begin introducing a complete, centralized framework, namely SeTraStream [116], for semantic aware trajectory extraction and subsequently comment on extensions of SeTraStream to distributed settings. The distributed version of our framework utilizes the second of the discussed approaches (incorporates data reduction techniques), so as to alleviate bandwidth consumption but we come up with smart ways tailored for our exact needs, which manage to reduce communication without affecting accuracy. Towards the objective of real-time semantic trajectory extraction, our core contributions are:

- As a prior step for extracting semantic trajectories, we redesign trajectory data preprocessing in the real-time context, including *online cleaning* and *online compression*. Our cleaning includes an one-loop procedure for removing outliers and alleviating errors based on a *Kernel smoothing* method. SeTraStream's compression scheme uses a combination of the *Synchronized Euclidean Distance* (sed) and the novel definition of a *Synchronized Correlation Coefficient* (scc).
- We design techniques for finding division points which infer trajectory episodes during online semantic trajectory extraction. SeTraStream's outcomes are later easy to handle and a *semantic tagging* classifier can then be applied for tag assign-

ment on identified homogeneous portions of movement, e.g. “driving”, “jogging”, “dwelling for shopping” and so on.

- We implement SeTraStream’s multi-layer procedure for semantic trajectory extraction and evaluate it, considering different real life trajectory datasets. The results demonstrate the ability of SeTraStream to accurately provide computed semantic-aware trajectories in real-time, readily available for applications’ querying purposes.
- We extend the previously introduced centralized framework to distributed settings and provide techniques for communication load reduction by data compression which, however, do not compromise accuracy at all. Our distributed techniques provide applications the ability to predetermine the worst case bandwidth consumption and thus enable a priori installation of proper network infrastructure.

This thesis is organized as follows. Initially, in the next chapter, we comment on works related to the issues we elaborate throughout our study. The rest of the thesis is divided into two parts. In the first part, we concentrate on distributed monitoring approaches. We present our prediction - based geometric monitoring framework [42] (Chapter 3). Furthermore, in Chapter 4 we provide a case study regarding the adoption of the previously proposed monitoring mechanism to the special case of distributed representative trajectory monitoring. The second part, focuses on distributed mining techniques. We present our TACO framework [44, 45] for detecting outliers in sensor network settings in Chapter 5, while in Chapter 6 we examine semantic trajectory extraction issues based on generalizations of similarity measures - monitored functions utilized in Chapter 5. Eventually, Chapter 7 includes concluding remarks and future work considerations.

Chapter 2

Related Work

2.1 Distributed Monitoring of Data Streams

Recently, substantial efforts have been devoted on tracking and querying distributed data streams [21]. The geometric monitoring framework which is leveraged by our approaches in Chapter 3 was introduced in [103, 105] and was later enhanced in [106]. The optimizations proposed in [106] are orthogonal to our approaches, but note that the techniques of [106] either require data to conform with a multivariate normal distribution or entail a number of solutions to a series of optimization problems that may increase the computational load. The latter renders their adoption unaffordable in resource constraint environments such as [104]. On the contrary, our approaches are based on simple predictors' adoption that remain adaptable to changing data distributions and are easy to maintain even when resource constraints exist. In other work related to the geometric monitoring approach, [104] discusses an application of the framework of [103, 105] to clustered sensor network settings. The more recent work of [100] adopts the geometric approach and proposes a tentative bound algorithm to monitor threshold queries in distributed databases (rather than distributed data streams) for functions with bounded deviation.

Prediction models in the context of distributed data streams have already been fostered in previous work to monitor one-dimensional quantiles [23] and randomized sketch summaries [22]. Their adoption has been proven beneficial in terms of reducing the communication burden. Contrary to previous approaches our focus is on the benefits they can provide in the context of the geometric monitoring framework for tracking non-linear threshold functions.

In related work regarding distributed trigger monitoring, [67] provides a framework for monitoring thresholded counts over distributed data streams, while [55] designs techniques that decompose the problem of detecting when the sum of a distributed set of variables exceeds a given threshold. Based on [55] anomaly detection techniques are studied in [51] and [52]. The recent work of [24] provides upper and lower communication bounds for approximate monitoring of thresholded F_p moments, with $p = 0, 1, 2$.

Other works focus on tracking specific types of functions over distributed data streams. The work of [84] considers simple aggregation queries over multiple sources, while [3] focuses on monitoring top-k values. Furthermore, [26] monitors set-expression cardinalities in a distributed system using a scheme for charging local changes against single site's error tolerance. [121] considers the problem of tracking heavy hitters and quantiles in a distributed manner establishing optimal algorithms to accomplish the task. Eventually, [25] studies the problem of clustering distributed data streams, while [125] generalizes the previous approach to hierarchical environments.

2.2 Representative Trajectories and Location Predictors

In Chapter 4 we present a specialization of the prediction - based monitoring developed in Chapter 3 for the case of distributed representative trajectory tracking over spatiotemporal data streams. The concept of a representative trajectory providing a concise summary of the movement of a number of monitored objects that is adopted in our approach, resembles the one that was initially introduced in TRACLUS [73]. According to the TRACLUS clustering framework, a representative trajectory of a cluster of line segments (partial trajectories) is computed as an average direction vector similar to a rotated centroid. The representative trajectory notion is reconsidered in [92] where uncertain trajectories with general, instead of linear, types of movement are clustered as a whole.

In the context of privacy-aware querying, [91] utilizes representative trajectories to enable the production of fake trajectories that resemble the average movement pattern of the objects participating in the query answer. In that, the returned query answers do not reveal the actual trajectory of a moving object and simultaneously ensure that the introduction of fake trajectories does not uncontrollably distort the query answer.

An approach for expressing the ”representativeness”, via a voting process that is applied for each segment of a given trajectory is presented in [87]. A simplistic trajectory ranking method selects the trajectories of highest voting and ignores trajectories in low density regions. The previous scheme is improved in [88] by handling trajectory segmentation and sub-trajectory sampling aspects. Representative trajectories have also been used for sampling purposes in [93].

In [99] online, distributed hot motion path extraction is studied. The hot motion paths are defined as frequently traveled trails of moving objects. The framework, similar to our techniques, assumes a distributed system of moving objects communicating with a central server. The location measurements of each object are modeled with some uncertainty tolerance ϵ and a one-pass greedy algorithm, termed RayTrace, which is supposed to run on each object independently is introduced. However, the focus is on pinpointing frequently preferred paths instead of tracking the average movement behavior of the monitored objects.

Finally, Chapter 4 utilizes the predictors presented in Chapter 3 (and [23, 22]) for estimating the future location of moving objects. Specific techniques for predicting the upcoming locations of moving objects have been proposed in [120, 59, 83]. However, these approaches are developed to perform over centralized databases and cannot efficiently function in (distributed) streaming settings where we are interested in online, continuous tracking of the objects as well as in adapting our predictions to frequently changing movement distributions.

2.3 Outlier Detection in Sensor Networks

The emergence of sensor networks as a viable and economically practical solution for monitoring and intelligent applications has prompted the research community to devote substantial effort to define and design the necessary primitives for data acquisition based on sensor networks [78, 119]. Different network organizations have been considered, such as using hierarchical routes (i.e., the aggregation tree [109, 123]), cluster formations [13, 50, 96, 122], or even completely ad-hoc formations [4, 65, 70]. The framework we present in Chapter 5 assumes a clustered network organization. Such networks have been shown to be efficient in terms of energy dissipation, thus resulting in prolonged network lifetime [96, 122].

Sensor networks can be rather unreliable, as the commodity hardware used in the development of the motes is prone to environmental interference and failures. As a result, substantial effort has been devoted to the development of efficient outlier detection techniques that manage to pinpoint motes exhibiting extraordinary behavior [126].

The recent work of [9] adopts the basic (without incorporating predictors) geometric monitoring framework we are going to discuss in Chapter 3 during outlier detection in a sensor network. However, in a generic sensor setting where message losses as well as addition and removal of nodes may happen, fostering the geometric approach may compromise the accuracy of the technique beyond control [9]. As a result in Chapter 5 we are going to present our TACO framework [44, 45] that fosters a different rationale achieving efficiency by incorporating data reduction techniques accompanied by respective accuracy (depending on the reduction ratio) guarantees. Until then, we review a number of works that tackle with the outlier detection process in sensor network architectures.

The authors of [56, 57] introduce a declarative data cleaning mechanism over data streams produced by the sensors. Similarly, the work of [35] introduces a data cleaning module designed to capture noise in sensor streaming data based on the prior data distribution and a given error model $N(0, \delta^2)$. In [81] kalman filters are adopted during data cleaning or outlier detection procedures. Nonetheless, without prior knowledge of the data distribution the parameters and covariance values used in these filters are difficult to set. The data cleaning technique presented in [129] makes use of a weighted moving average which takes into account both recent local samples and corresponding values by neighboring motes to estimate actual measurements. A wavelet-based value correction process is discussed in [128] while outliers are determined utilizing the Dynamic Time Warping (*DTW*) distance of neighboring motes' values. A different approach is presented in [15], where Pairwise Markov Networks are used as a tool to derive a subset of motes sufficient to infer the values obtained by the whole network. However, this technique requires an energy draining learning phase. In other related work, [112] proposes a fuzzy approach to infer the correlation among readings from different sensors, assigns a confidence value to each of them, and then performs a fused weighted average scheme. A histogram-based method to detect outliers with reduced communication cost is presented in [107].

In [68], the authors discuss a framework for cleaning input data errors using integrity constraints, while in [7, 124] unsupervised outlier detection techniques are used to report the top-k values that exhibit the highest deviation in a network's global sample. Amongst these techniques, of particular interest is the technique of [7], as it is flexible with respect to the outlier definition. However, in contrast to our techniques, it provides no means of directly controlling the bandwidth consumption, thus often requiring comparable bandwidth to centralized approaches for outlier detection [7].

In [58], a probabilistic technique for cleaning RFID data streams is presented. The framework of [30] is used to identify and remove outliers during the computation of aggregate and group-by queries posed to an aggregation tree [19, 78]. Its definition of what constitutes an outlier, based on the notion of minimum support and the use of recent history, is adopted in Chapter 5 by our framework. It further demonstrates that common similarity metrics such as the correlation coefficient and the Jaccard coefficient can capture the types of dirty data encountered by sensor network applications. Similarly to TACO (Chapter 5), the PAO framework [43] operates on top of clustered network organizations and attempts to restrain communication costs during outlier identification by detecting trends on mote measurements and applying linear-regression based compression. In [111] the authors introduce a novel definition of an outlier, as an observation that is sufficiently far from most other observations in the data set. A similar definition is adopted in [85] where a distributed outlier detection approach for dynamic data sets is presented. However, in cases where the motes observe physical quantities (such as noise levels, temperature) the absolute values of the readings acquired depend, for example, on the distance of the mote from the cause of the monitored event (i.e., a passing car or a fire respectively). Thus, correlations among readings in space and time are more important than the absolute values, used in [85, 111].

The algorithms in [7, 30, 56, 124, 43] provide no easily tunable parameters in order to limit the bandwidth consumed while detecting and processing outliers. On the contrary, the techniques of Chapter 5 have a direct way of controlling the number of bits used for encoding the values observed by the motes. While [30] takes a best effort approach for detecting possible outliers and [56] requires transferring all data to the base station in order to accurately report them, controlling the size of the encoding allows our framework to control the accuracy of the outlier detection process.

The work in [14, 113] addresses the problem of identifying faulty sensors using a localized voting protocol. However, localized voting schemes are prone to errors when motes that observe interesting events generating outlier readings are not in direct communication [30]. Furthermore, the framework of [113] requires a *correlation network* to be maintained, while the TACO framework can be implemented on top of commonly used clustered network organizations.

In Section 5.7 we extend TACO by a message suppression strategy that fledges bandwidth consumption preservation. Message suppression schemes in sensor networks for continuous aggregate queries have been studied in [28, 29, 102]. Our work differs in that we do not suppress raw measurements but extracted, compact bitmap representations instead.

The Locality Sensitive Hashing (LSH) scheme used in TACO was initially introduced in the rounding scheme of [47] to provide solutions to the MAX-CUT problem. Since then, LSH has been adopted in similarity estimation [12, 33], clustering [97] or indexing techniques for set value attributes [46]. Additionally, LSH has also been fostered in approximate nearest neighbor (NN) queries [54] while [2] introduces a novel hash-based indexing scheme for approximate NN retrieval that, unlike [54], can be applied to non-metric spaces as well. Eventually, the recent work of [38] extends the Random Hyperplain Projection LSH scheme [12] by automatically detecting correlations, thus computing embeddings tailored to the provided data sets.

2.4 Semantic Trajectory Extraction

In Chapter 6 our objective is to generalize the rationale of the similarity measures used in Chapter 5, but this time so as to detect movement pattern alterations of moving objects and design online methods for real-time semantic trajectory extraction with extensions to distributed settings.

We start by trajectory construction which is the procedure of reconstructing trajectories from the original sequence of spatiotemporal records of moving objects. Tasks involved in this procedure mainly include *data cleaning* and *data compression*. *Data cleaning* is dealing with trajectory errors which are quite common in GPS alike trajectory recordings. There are two types of errors: the outliers which are far away from the true values and need to be removed; the noisy data that should be corrected and smoothed. Several works [80, 101, 117] design specific filtering methods to remove outliers and smoothing methods to deal with small random errors. Regarding network-constrained moving objects, a number of map matching algorithms have been designed to refine the raw GPS records [6, 64].

Trajectory data are generated continuously, in a high frequency and sooner or later grow beyond systems' computational and memory capacity. Therefore, *data compression* is a fundamental task for supporting scalable applications. The spatiotemporal compression methods for trajectory data can be classified into four types: i.e. *top-down*, *bottom-up*, *sliding window*, and *opening window*. The top-down algorithm recursively splits the trajectory sequence and selects the best position in each sub-sequence. A representative top-down method is the Douglas-Peucker (DP) algorithm [34], with many extended implementation techniques. The bottom-up algorithm starts from the finest possible representation, and merges the successive data points until some halting conditions are met. Sliding window methods compress data in a fixed window size; whilst open window methods use a dynamic and flexible window size for data segmentation. To name but a few methods: Meratnia et al. propose *Top-Down Time Ratio* (TD-TR) and *OPen Window Time Ratio* (OPW-TR) for the compression of spatiotemporal trajectories [82]. In addition, the work of [95] provides two sampling based compression methods: *threshold-guided sampling* and *STTrace* to deal with limited memory capacity.

Recently, semantic-based trajectory model extraction has emerged as a hot topic involving the addition of a semantic level on top of the mere spatiotemporal trajectories. [110] models a semantic trajectory as a sequence of stops and moves. From a semantic point of view, a raw trajectory as a sequence of GPS points can be abstracted to a sequence of meaningful episodes (e.g. *begin*, *move*, *stop*, *end*) along with their spatiotemporal extend. [117][118] design a computing platform to progressively extract spatiosemantic trajectories from the raw GPS tracking feeds. In that approach, different levels of trajectories are constructed, from the construction of *spatiotemporal trajectories* and *structured trajectories* to the final *semantic trajectory* extraction, in four computational layers, i.e. *data preprocessing*, *trajectory identification*, *trajectory structure* and *semantic enrichment*.

Trajectory episodes like stops and moves can be computed with given geographic artifacts [1] or only depend on spatiotemporal criteria like density, velocity, direction etc. [86, 98, 117]. [1] develops a mechanism for the automatic extraction of stops that is based on the intersection of trajectories and geometries of geographical features considered relevant to the application. In that approach the semantic information is limited to geographic data that intersect the trajectories for a certain time interval. Thus, it is restricted to applications in which geographic information can help to identify places visited by the moving objects which play an essential role. Recently, more advanced methods use spatiotemporal criteria to perform trajectory segmentation which is equivalent to identifying episodes like stops/moves: [117] designs a velocity-based method providing a dynamic velocity threshold on stop computation, where the minimal stop duration is used to avoid false positives (e.g. congestions); several clustering-based stop identification methods have been developed, e.g. using the velocity [86] and direction features [98] of movement. Finally, [8] provides a theoretical framework for trajectory segmentation and claim that the segmentation problem can be solved in $O(n\ell ogn)$ time.

Online segmentation concepts can be traced back to the time series and signal processing fields [66], but not initially for trajectories. Although, some of the above works are capable of adapting to an online context [6][64], none of them focuses on revealing the profound semantics present in the computed trajectories in real-time.

Part I

Monitoring Distributed Data Streams

Chapter 3

Prediction - Based Geometric Monitoring over Distributed Data Streams

3.1 Introduction

A wide variety of modern applications relies on the *continuous* processing of vast amounts of arriving data in order to support decision making procedures in real time. Examples include network administration, stock market analysis, environmental, surveillance and other application scenarios. These settings are, more often than not, inherently *distributed* in nature. For instance, consider the case of a network operation center where data is produced by hundreds or thousands of routers [20, 23, 22] or the case of environmental as well as control applications where wireless sensor network adoption has become of great importance [79].

Due to the distributed nature of data production in the aforementioned scenarios, the major challenge confronted by algorithms dealing with their manipulation is to reduce communication [20, 23, 22, 103, 105, 106, 26]. This happens because the central collection of data is not feasible in large-scale applications. Furthermore, in the case of sensor network deployments, central data accumulation results in depleting the power supply of individual sensors reducing the network lifetime [79].

An important query type that is of the essence in the aforementioned fields regards the monitoring of a trigger condition defined upon the range of values a function of interest receives [103, 105, 106, 51, 55, 67, 52]. For instance, in order to perform spam detection on a number of dispersed mail servers, algorithms base their decisions on whether the value of the information gain function globally exceeds a given threshold [105]. Moreover, in the example of the network operation center, Denial-of-Service attacks are detected by attempting to pinpoint strangely high (based on a given threshold) number of distinct source addresses routing packets across various destinations within the network [26].

Recently, the work in [103, 105] has introduced a generic paradigm for monitoring complex (non-linear) functions defined over the average of local vectors maintained at distributed sites. Their proposed geometric approach essentially monitors the area of the input domain where the average vector may lie, rather than monitoring the function's value itself. The monitoring is performed in a distributed manner, by assigning each node a monitoring zone, expressed as a hypersphere, which is nothing more than a subset of the input domain where the average vector may lie. Communication is shown to be necessary only if at least one site considers it likely that the condition of the monitored function may have changed since the last communication between the sites.

In this chapter [42], we examine the potentials of a simple (yet powerful), easy to locally maintain approach in order to further reduce transmissions towards the central source. In particular, we foster prediction models so as to describe the evolution of local streams. The adoption of prediction models has already been proven beneficial in terms of bandwidth preservation [20, 23, 22] in distributed settings. Initially, we extend the geometric monitoring framework of [103, 105] and illustrate how it can incorporate predictors, in order to forecast the evolution of local data vectors of sites. We exhibit the way the geometric monitoring framework is modified to encompass constructed predictors and identify the peculiarities occurred upon predictors' adoption. In contrast to the findings of prior works [20, 23, 22], we prove that the mere utilization of local predictions is hardly adequate to guarantee communication preservation even when predictors are quite capable of describing local stream distributions. We then proceed by establishing a theoretically solid monitoring framework that incorporates conditions managing to guarantee fewer contacts with the central source. Eventually, we develop a number of mechanisms, along with extensive speculative analysis, that relax the previously introduced framework, base their function on simpler criteria, and in practice yield significant transmission reduction.

The chapter proceeds as follows. In the next section we formally present the geometric monitoring framework and we exhibit exemplary prediction models useful for the applications we consider. Section 3.3 introduces the idea of prediction - based geometric monitoring and shows how conventional notions of good prediction models fail to adapt in the current setting. Section 3.4 presents theoretic frameworks that manage to dictate the sufficient conditions for prediction models adoption with provable communication reduction while in Section 3.5 we propose practical alternatives built upon as simple as possible assumptions on the ability of prediction models to describe incoming data distributions. Our experimental analysis is incorporated in Section 3.6.

3.2 Preliminaries

In this section, we first provide helpful background work related to function monitoring using the geometric approach. We then describe local stream predictors, which have been utilized in past work. The notation used in this chapter appears in Table 3.1.

3.2.1 The Geometric Monitoring Framework

As in previous works [23, 105, 20, 22, 106], we assume a distributed, two-tiered setting, where data arrives continuously at n geographically dispersed sites. At the top tier, a central coordinator exists that is capable of communicating with every site, while pairwise site communication is only allowed via the coordinating source.

Each site S_i, $i \in [1..n]$ participating at the bottom tier receives updates on its local stream and maintains a d-dimensional local measurements vector $v_i(t)$. The *global measurements vector* $v(t)$ at any given timestamp t, is calculated as the weighted average of $v_i(t)$ vectors, $v(t) = \frac{\sum_{i=1}^{n} w_i v_i(t)}{\sum_{i=1}^{n} w_i}$, where $w_i \geq 0$ refers to the weight associated with a site. Usually, w_i corresponds to the number of data points received by S_i [103]. Our aim is to continuously monitor whether the value of a function $f(v(t))$, defined upon $v(t)$, lies above/below a given threshold T. We use the term *threshold surface* to denote the area of the input domain where $f(v(t)) = T$.

During the monitoring task using the geometric approach [103, 105], the coordinator may request that all sites transmit their local measurements vectors and subsequently calculates $v(t)$, performs the required check on $f(v(t))$, and transmits the $v(t)$ vector to all sites. The previous process is referred to as a *synchronization* step. Let $v_i(t_s)$ denote the local measurements vector that S_i communicated during the last synchronization process at time t_s. The global measurements vector computed during a synchronization step is denoted as the *estimate vector* e, where $e = \sum_{i=1}^{n} w_i v_i(t_s) / \sum_{i=1}^{n} w_i$.

After a synchronization, sites keep up receiving updates of their local streams and accordingly maintain their $v_i(t)$ vectors. At any given timestamp, each site S_i individually computes $v_i(t) - v_i(t_s)$ and the local *drift vector* $u_i(t) = e + (v_i(t) - v_i(t_s))$. Since

$$v(t) = \frac{\sum_{i=1}^{n} w_i v_i(t)}{\sum_{i=1}^{n} w_i} = e + \frac{\sum_{i=1}^{n} w_i (v_i(t) - v_i(t_s))}{\sum_{i=1}^{n} w_i} = \frac{\sum_{i=1}^{n} w_i u_i(t)}{\sum_{i=1}^{n} w_i}$$

$v(t)$ constitutes a convex combination of the drift vectors. Consequently, $v(t)$ will always lie in the convex hull formed by the $u_i(t)$ vectors: $v(t) \in Conv\ (u_1(t), \ldots, u_n(t))$, as depicted in Figure 3.1.

Symbol	Description
n	The number of sites
S_i	The i-th site
t_s	Timestamp of the last synchronization
$v(t)$	Global measurements vector at time t ($\sum_{i=1}^{n} w_i v_i(t) / \sum_{i=1}^{n} w_i$)
$e(t)$	Estimate vector at time t (equal to $v(t_s)$)
$e^p(t)$	The predicted estimate vector ($\sum_{i=1}^{n} w_i v_i^p(t) / \sum_{i=1}^{n} w_i$)
$v_i(t)$	Local measurements vector at S_i at time t
w_i	Number of data points at S_i
$u_i(t)$	Drift vector (equals to $e(t) + v_i(t) - v_i(t_s)$)
$v_i^p(t)$	Local predictor of S_i at time t
$u_i^p(t)$	Prediction deviation vector ($e^p(t) + v_i(t) - v_i^p(t)$)
$B_c^{\|r\|}$	Local constraint (ball) centered at c with radius $\|r\|$

Table 3.1: Notation of Chapter 3

Note that each site can compute the last known value of the monitored function as $f(e)$ and can, thus, determine whether this value lies above/below the threshold T. Since $v(t) \in Conv\ (u_1(t), \ldots, u_n(t))$, if the value of the monitored function in the entire convex hull lies in the same direction (above/below the threshold T) as $f(e)$, then it is guaranteed that $f(v(t))$ will lie in that side. In this case, the function will certainly not have crossed the threshold surface. The key question is: "how can the sites check the value of the monitored function in the entire convex hull, since each site is unaware of the current drift vectors of the other sites"? This test can be performed in a distributed manner as described in Theorem 3.2.1, while an example (in 2-dimensions) is included in Figure 3.1.

Theorem 3.2.1. *[103, 105] Let $x, y_1, \ldots, y_n \in R^d$ be a set of d dimensional vectors. Let $Conv\ (x, y_1, \ldots, y_n)$ be the convex hull of $x, y_1, \ldots, y_n$. Let $B_{\frac{x+y_i}{2}}^{\|\frac{x-y_i}{2}\|}$ be a ball centered at $\frac{x+y_i}{2}$ with a radius of $\|\frac{x-y_i}{2}\|$ that is, $B_{\frac{x+y_i}{2}}^{\|\frac{x-y_i}{2}\|} = \{z \in R^d : \|\|z - \frac{x+y_i}{2}\| \leq \|\frac{x-y_i}{2}\|\}$. Then, $Conv\ (x, y_1, \ldots, y_n) \subset \bigcup_{i=1}^{n} B_{\frac{x+y_i}{2}}^{\|\frac{x-y_i}{2}\|}$.*

With respect to our previous discussion x corresponds to e while y_i vectors refer to the drift vectors $u_i(t)$. Hence, sites need to compute their local constraints in the form of $B_{\frac{e+u_i(t)}{2}}^{\|\frac{e-u_i(t)}{2}\|}$ and independently check whether a point within these balls may cause a threshold crossing. If this indeed is the case, a synchronization step takes place. Note that since $Conv\ (e, u_1, \ldots, u_n)$ is a subset of the union of local ball constraints, the framework may produce a synchronization in cases where the convex hull has not actually crossed the threshold surface (false positives).

In summary, each site in the geometric monitoring framework manages to track a subset of the input domain. The overall approach achieves communication savings since the coordinator needs to collect the local measurement vectors of the sites only

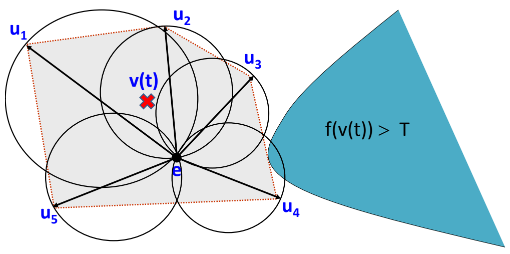


Figure 3.1: Demonstration of the geometric framework rationale. $Conv\ (u_1, \ldots, u_n)$ is depicted in gray, while the actual position of e and the current $v(t)$ are shown as well. Black spheres refer to the local constraints constructed by sites to assess possible threshold crossing. $v(t)$ is guaranteed to lie within the union of these locally constructed spheres. Since one of the spheres crosses the threshold surface, $f(v(t))$ and $f(e)$ may not lie at the same side relative to the threshold T. Hence, a synchronization needs to be performed.

when a site locally detects (in its monitored area of the input domain) that a threshold crossing may have occurred.

3.2.2 Local Stream Predictors

We now outline the properties of some prediction model options that have already been proven useful in the context of distributed data streams [22, 23]. Note, beforehand, that the concept of their adoption is to keep such models as simple as possible, and yet powerful enough to describe local stream distributions. It can easily be conceived that more complex model descriptors can be utilized, which however incur extra communication burden when sites need to contact the coordinating source [22, 23]. In our setting, this translates to an increased data transmission overhead during each synchronization step. In our discussion, hereafter, we utilize the term *predictor* to denote a prediction estimator for future values of a local measurements vector. Using a similar notation to the one of Section 3.2.1, we employ $v_i^p(t)$ to denote the prediction for the local measurements vector of site S_i at timestamp t.

The Static Predictor. The simplest guess a site may take regarding the evolution of its local measurements vector is that its coordinates will remain unchanged with respect to the values they possessed in the last synchronization: $v_i^p(t) = v_i(t_s)$. It is also evident that this predictor is trivial to maintain in both the sites and the coordinator. Moreover,

Predictor	**Info.**	**Pred. Local Vector** (v_i^p)
Static	$\emptyset$	$v_i(t_s)$
Linear Growth	$\emptyset$	$\frac{t}{t_s} v_i(t_s)$
Velocity/Acceleration	vel_i	$v_i(t_s) + (t - t_s)vel_i + (t - t_s)^2 accel_i$

Table 3.2: Local Stream Predictors' Summary

it requires no additional information to be communicated towards the coordinator upon a synchronization step. Using the static predictor, in the absence of a synchronization step, the coordinator estimates that $v(t) = e$.

The static predictor may be a good choice only in settings where the evolution of the values in each local measurements vector is unpredictable, or local measurements vectors change rarely.

The Linear Growth Predictor. The next simple, but less restrictive, assumption that can be made is that local vectors will scale proportionally with time. In particular, $v_i^p(t) = \frac{t}{t_s} v_i(t_s)$ which is the only calculation individual sites and the coordinating source need to perform in order to derive an estimation of $v_i(t)$ at any given time. Note that, using this predictor, the best guess that a coordinator can make for the value of $v(t)$ is equal to $\frac{t}{t_s} v(t_s) = \frac{t}{t_s} e$. As with the Static Predictor, the Linear Growth Predictor requires no additional information to be transmitted upon a synchronization.

We can deduce that the Linear Growth Predictor is built on the assumption that $v_i(t)$ vectors evolve, but that their evolution involves no direction alterations. Consequently, it can be adopted to approximate local streams in which $v_i(t)$ vectors' coordinates are expected to increase uniformly over time.

The Velocity/Acceleration Predictor. The Velocity/Acceleration (*VA*) Predictor is a much more expressive predictor. VA employs additional vectors that attempt to capture both the scaling and directional change that $v_i(t)$ may undertake. More precisely, in the *VA* predictor the future value of the local measurements vector is estimated as $v_i^p(t) = v_i(t_s) + (t - t_s) vel_i + (t - t_s)^2 accel_i$. Since the velocity vel_i and the acceleration $accel_i$ of the local stream are capable of expressing both possible types of $v_i(t)$ alterations, it provides an enriched way to approximate its behavioral pattern.

In a way similar to [22], when a synchronization is about to take place, S_i is required to compute the velocity vector vel_i utilizing a window of the W most recent updates it received. Given that window, the velocity vector can be calculated by computing the overall disposition as the difference between $v_i(t)$ and the local vector instance corresponding to the first position of the window.[1] Scaling this outcome by the time difference between the window extremes provides vel_i. In addition, the $accel_i$ value can be computed as the difference between the current velocity and corresponding velocity calculated in the previous synchronization. Scaling the previous result by $1/(t - t_s)$ computes a proper $accel_i$ vector. Additional approaches based on use of vel_i and $accel_i$ values can be found in [22].

It is easy to see that the flexibility provided by the *VA* Predictor comes at the cost of the transmission of vel_i (along with $v_i(t)$) during each synchronization. We note that $accel_i$ does not need to be communicated to the coordinator, since the coordinator is already aware of the previously computed velocity vectors of each site.

[1]Note that each update may not arrive at each timestamp. Thus, the timestamp of the first update in the window may in general be different than $t - W + 1$.

Table 3.2 summarizes the described predictor characteristics. It is important to emphasize that the prediction-based monitoring framework described in the next sections can utilize any predictor and is, thus, not restricted to the predictors presented in this section.

3.3 Prediction-Based Monitoring

In this section, we first motivate the need to incorporate predictors in the geometric monitoring framework and then demonstrate how this can be achieved. We then illustrate that the initial geometric monitoring framework of [103, 105] is a special case of our, more general, prediction-based geometric monitoring framework. Subsequently, we define the notion of a good predictor and demonstrate that good predictors lead to monitoring a smaller subset of the domain space, thus potentially leading to fewer synchronizations and, hence, fewer transmitted messages.

3.3.1 Motivation for Predictors

Figure 3.1 demonstrates a motivating example of why it may be beneficial to incorporate predictors in the geometric monitoring framework. In the illustrated example, the sphere of u_4 has crossed the threshold surface. By definition, the direction of each drift vector essentially depicts how the values of the corresponding local measurements vector have changed since the last synchronization. Using the geometric monitoring approach, a synchronization will take place because S_4 will detect a threshold crossing. A plausible question is: "Could we avoid such a synchronization step, if the changes in the values of the five local measurements vectors could have been predicted fairly accurately"? For example, if we could have predicted the change (drift) in the local measurements vectors of each site fairly accurately, then we would have determined that $v(t)$ has probably not moved closer to the threshold surface and, thus, avoid the synchronization step. The above example motivates the need for prediction-based geometric monitoring.

3.3.2 How to Incorporate Predictors

As explained in Section 3.2.2, the coordinator can receive, during a synchronization step, information regarding the predicted local measurements vector $v_i^p(t)$ of each site. Thus, the coordinator will be able to compute an estimation of $v(t)$ provided by the local predictors as: $e^p(t) = \frac{\sum\limits_{i-1}^{n} w_i v_i^p(t)}{\sum\limits_{i-1}^{n} w_i}$, which we will term as the *predicted estimate vector*. Based on $e^p(t)$, we now show that the coordinator can continuously check for potential threshold crossings. However, in this case a synchronization is required only when $e^p(t)$ and $v(t)$ are likely to be placed in different sides of the threshold surface.

In the context of the geometric monitoring framework, we first observe that:

$$v(t) = \frac{\sum_{i=1}^{n} w_i v_i(t)}{\sum_{i=1}^{n} w_i} = e^p(t) + \frac{\sum_{i=1}^{n} w_i (v_i(t) - v_i^p(t))}{\sum_{i=1}^{n} w_i} = \frac{\sum_{i=1}^{n} w_i u_i^p(t)}{\sum_{i=1}^{n} w_i}$$

where $u_i^p(t) = e^p(t) + (v_i(t) - v_i^p(t))$ denotes the vector expressing the *prediction deviation*. Thus, similar to our analysis in Section 3.2.1, $v(t) \in Conv\ (u_1^p(t), \ldots, u_n^p(t)) \subset \bigcup_{i=1}^{n} B^{\|\frac{e^p(t)-u_i^p(t)}{2}\|}_{\frac{e^p(t)+u_i^p(t)}{2}}$. Since $v(t)$ lies in the convex hull $Conv(u_1^p(t), \ldots, u_n^p(t))$, each site S_i can monitor the ball that has as endpoints of its diameter the estimated predicted vector $e^p(t)$ and its prediction deviation $u_i^p(t)$.

Note that the geometric monitoring approach of [103, 105] corresponds to utilizing a static predictor (this leads to $v_i^p(t) = v(t_s)$, $e^p(t) = e$ and $u_i^p(t) = u_i(t)$) and is, thus, a special case of our, more general, prediction-based monitoring framework.

3.3.3 Defining a Good Predictor

Upon utilizing a predictor, as long as local forecasts ($v_i^p(t)$) remain sound, we expect that they will approximate the true local vectors $v_i(t)$ to a satisfactory degree at any given timestamp. This means that each $v_i^p(t)$ will be in constant proximity to the $v_i(t)$ vector, when compared to $v_i(t_s)$. Formally:

Property 3.3.1. *A* **Good Predictor** *possesses the property:*

$$\|v_i(t) - v_i^p(t)\| \leq \|v_i(t) - v_i(t_s)\| \ \forall t \geq t_s$$

Property 3.3.1 lies, implicitly or not, in the core of predictors' adoption in distributed stream settings. It expresses the notion of a useful, in terms of bandwidth consumption reduction, predictor present in previous works [20, 22, 23] which have managed to exhibit important improvements by exploiting the above fact. Hence, we start by exploring the benefits of the notion of good predictors expressed by Property 3.3.1 within the geometric monitoring setting.

Predictors satisfying Property 3.3.1 yield stricter local constraints for the bounding algorithm compared to the original monitoring mechanism (Section 3.2.1). This happens because $\|v_i(t) - v_i^p(t)\| \leq \|v_i(t) - v_i(t_s)\| \Leftrightarrow \|u_i^p(t) - e^p(t)\| \leq \|u_i(t) - e\|$ and the radius of the constructed balls will always be smaller. An example of prediction-based monitoring is depicted in Figure 3.2.

Consequently, a good predictor results in the sites monitoring a tighter convex hull, namely $Conv\ (u_1^p(t), \ldots, u_n^p(t))$, than the corresponding convex hull of the original geometric monitoring framework. This yields the construction of tighter local constraints and, as already mentioned, a synchronization is required only when $e^p(t)$ is likely to be placed in a different side of the threshold surface to the one of $v(t)$. Hence, a synchronization is again caused when any ball $B_{\frac{e^p(t)+u_i^p(t)}{2}}^{\|\frac{e^p(t)-u_i^p(t)}{2}\|}$ crosses the threshold surface.

Despite the fact that this mechanism may in practice be useful, it cannot guarantee fewer synchronizations because $Conv\ (u_1^p(t), \ldots, u_n^p(t))$, although tighter, might still be placed closer than $Conv\ (u_1(t), \ldots, u_n(t))$ to the threshold surface. This in turn will cause some $B_{\frac{e^p(t)+u_i^p(t)}{2}}^{\|\frac{e^p(t)-u_i^p(t)}{2}\|}$ to cross the threshold before any $B_{\frac{e+u_i(t)}{2}}^{\|\frac{e-u_i(t)}{2}\|}$ does (Figure 3.2). This observation shows that the conventional concept of good predictors fails to adapt in the current setting since it does not guarantee by itself fewer synchronizations.

3.4 Strong Monitoring Models

The concluding observations of Section 3.3 raise a concern regarding the sufficient conditions that should be fulfilled for the predictors to always yield fewer synchronizations than the original framework. Apparently, this happens when the surface of the monitoring framework devised by the predictors is contained inside the monitored surface of the original framework. In other words, we need to define the prerequisites for constructing local constraints that are always included in $\bigcup_{i=1}^{n} B_{\frac{e+u_i(t)}{2}}^{\|\frac{e-u_i(t)}{2}\|}$ utilized by the original framework. Let $Sur(P)$ be the surface monitored by any alternative mechanism that adopts predictors while operating. A monitoring model is defined as *strong* if the following property holds:

Property 3.4.1. *A* **Strong** *predictor-based* **Monitoring Model** *possesses the property:* $Sur(P) \subseteq \bigcup_{i=1}^{n} B_{\frac{e+u_i(t)}{2}}^{\|\frac{e-u_i(t)}{2}\|}$

3.4.1 Containment of Convex Hulls

According to Theorem 3.2.1, after computing the local drift vectors and prediction deviations, we are free to choose any common, *reference vector* in order to perform the monitoring task. Thus, it is not mandatory for the sites to use e and $e^p(t)$ as a common reference point in order to construct their monitoring zones. In fact, the sites could use any common point as an endpoint of the diameter of their monitoring zones.

An important observation that we prove in this section is that a predictor-based monitoring model satisfies Property 3.4.1 when (1) every prediction deviation vector is contained in the convex hull of the estimate vector and the drift vectors defined by the original bounding algorithm, and (2) an appropriate reference vector is selected.

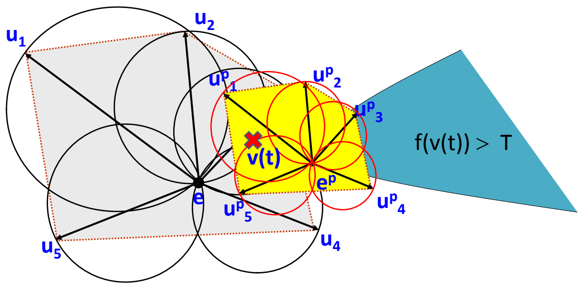


Figure 3.2: The red balls demonstrate the local constraints of sites when using a sample good predictor. A good predictor results in the tighter convex hull $Conv\ (u^p_1(t), \ldots, u^p_n(t))$ (depicted in yellow). Here, fewer synchronizations are not guaranteed, since the balls of the predicted convex hull cross the threshold before any ball of the original convex hull does so.

Before proving our observation, we first show that for any triplet of vectors $z, y, x \in R^d$, the condition $z \in B^{\|\frac{y-x}{2}\|}_{\frac{y+x}{2}}$ is equivalent to $\langle x - z, y - z\rangle \leq 0$ where the notation $\langle ., .\rangle$ refers to the inner product of two vectors. Whenever it is appropriate, we omit the temporal reference symbol (t) in the vectors to simplify the exposition.

Lemma 3.4.1. *$z \in B^{\|\frac{y-x}{2}\|}_{\frac{y+x}{2}}$ if and only if $\langle x - z, y - z\rangle \leq 0$.*

Proof. Recall that if $z \in B^{\|\frac{y-x}{2}\|}_{\frac{y+x}{2}}$, then $\|z - \frac{x+y}{2}\| \leq \|\frac{x-y}{2}\|$. This is equivalent to $\frac{1}{4}\langle 2z - (x + y), 2z - (x + y)\rangle - \frac{1}{4}\langle x - y, x - y\rangle \leq 0$. Recall that the inner product is distributive, i.e. $\langle a + b, c\rangle = \langle a , c\rangle + \langle b , c\rangle$, and symmetric, i.e. $\langle a , b\rangle = \langle b , a\rangle$ Therefore: $\frac{1}{4}\langle 2z - (x + y), 2z - (x + y)\rangle - \frac{1}{4}\langle x - y, x - y\rangle =$
$\frac{1}{4}\langle (z - x) + (z - y), (z - x) + (z - y)\rangle -$
$\frac{1}{4}\langle (z - x) - (z - y), (z - x) - (z - y)\rangle =$
$\langle z - x, z - y\rangle = \langle x - z, y - z\rangle$. □

We now proceed to prove in Lemma 3.4.2 that a predictor-based monitoring model that maintains each prediction deviation vector contained in the convex hull of the drift vectors defined by the original bounding algorithm is a strong predictor-based monitoring model if it also selects the same reference vector (e.g., e instead of e^p) as the original framework. A direct result is that the area monitored by the sites is a subset of the corresponding area of the original framework. This, in turn, leads to fewer synchronizations, since every time a site detects a potential threshold crossing in the predictor-based monitoring model, at least one site would also have detected the same threshold crossing (for the same vector of the input domain) in the original framework.

Lemma 3.4.2. *Let $u^p_i \in Conv(u_1, ..., u_n)\ \forall i \in \{1..n\}$. Then*

$$B^{\|\frac{e-u^p_i}{2}\|}_{\frac{e+u^p_i}{2}} \subseteq \bigcup_{i=1}^{n} B^{\|\frac{e-u_i}{2}\|}_{\frac{e+u_i}{2}}.$$

Proof. For each $u_i^p \in Conv(u_1, ..., u_n)$ there exist λ_1,λ_2, …, λ_n such that $\lambda_i \geqslant 0$ $(i \in \{1..n\})$, $\sum_{i=1}^{n} \lambda_i = 1$ and $u_i^p = \sum_{i=1}^{n} \lambda_i u_i$. Let $h \in B_{\frac{e+u_i^p}{2}}^{\|\frac{e-u_i^p}{2}\|}$. We show that for at least one of the u_i vectors, $h \in B_{\frac{e+u_i}{2}}^{\|\frac{e-u_i}{2}\|}$. According to Lemma 3.4.1:$\langle h-e, h-u_i^p\rangle \leq 0$. Therefore:

$$\langle h-e, h-u_i^p\rangle = \left\langle h-e, \sum \lambda_i h - \sum \lambda_i u_i \right\rangle \quad = \\ \left\langle h-e, \sum \lambda_i (h-u_i)\right\rangle = \sum \lambda_i \langle h-e, h-u_i\rangle \quad \leq \quad 0$$

Since $\lambda_i \geqslant 0$, it follows that for at least one u_i with $\lambda_i > 0$, $\langle h-e, h-u_i\rangle \leq 0$, which implies (Lemma 3.4.1) that $h \in B_{\frac{e+u_i}{2}}^{\|\frac{e-u_i}{2}\|}$. □

A trivial example of a strong predictor-based monitoring model is the static predictor which, as mentioned in Section 3.3, is equivalent to the original framework of [103, 105]. Unfortunately, the containment constraints are not easily abided by any other chosen predictor and, even if they are, it appears hard to dictate a way that allows sites to identify that fact in a distributed manner. We will revisit the convex hull containment issues in Section 3.5.1.

3.4.2 Convex Hull Intersection Monitoring

An important observation that we make is that, as $v(t) \in Conv(u_1, \ldots, u_n)$ and $v(t) \in Conv(u_1^p, \ldots, u_n^p)$, these two convex hulls cannot be disjoint (Fig. 3.2). One could, thus, seek ways to exploit this fact, which limits the possible locations of $v(t)$, in order to reduce the size of the monitoring zones of each site which, in turn, will potentially lead to fewer detected threshold crossings. We thus seek to come up with new local constraints in the context of predictor-based monitoring models that cover the intersection of the two convex hulls and which also fulfill Property 3.4.1. To proceed towards that goal we first formally formulate an enhanced version of Property 3.3.1.

Property 3.4.2. *A* **Universally Good Predictor** *possesses the pro-perty:* $\|v_i(t) - v_i^p(t)\| \leq \|v_j(t) - v_j(t_s)\|$ *for any pair of sites* S_i, S_j

In other words for universally good predictors:

$$\underset{k=1..n}{min} \|e - u_k\| \geq \underset{k=1..n}{max} \|e^p - u_k^p\| \tag{3.1}$$

Property 3.4.2 yields $\|u_i^p - e^p\| \leq \|u_j - e\|$ for any pair of sites S_i, S_j. The latter result is produced by simply adding as well as subtracting e^p, e to the left and right side of its inequality, respectively. The following lemma utilizes this fact to devise appropriate local constraints to be fostered at each site S_i.

Lemma 3.4.3. *If Property 3.4.2 holds, each site S_i needs to examine whether $B_{\frac{e+u_i}{2}}^{\|\frac{e-u_i}{2}\|} \cap B_{e^p}^{\|e-u_i\|}$ crosses the threshold, since:*

$$Conv(u_1, \ldots, u_n) \cap Conv(u_1^p, \ldots, u_n^p) \subset \bigcup_{k=1}^{n} B_{\frac{e+u_k}{2}}^{\|\frac{e-u_k}{2}\|} \cap B_{e^p}^{\|e-u_k\|}$$

Proof. We demonstrate that any vector $h \in R^d$ which lies in $Conv\ (u_1, \ldots, u_n) \cap Conv\ (u_1^p, \ldots, u_n^p)$ is also included in at least one intersection $B_{\frac{e+u_k}{2}}^{\|\frac{e-u_k}{2}\|} \cap B_{e^p}^{\|e-u_k\|}$ of a site S_k ($k \in \{1..n\}$). What is certain is that, due to Property 3.4.2, $h \in Conv\ (u_1^p, \ldots, u_n^p) \Rightarrow h \in \bigcup_{k=1}^{n} B_{\frac{e^p+u_k^p}{2}}^{\|\frac{e^p-u_k^p}{2}\|} \Rightarrow h \in \bigcup_{k=1}^{n} B_{e^p}^{\|e^p-u_k^p\|} \Rightarrow h \in B_{e^p}^{\|e-u_k\|}$. Since h is definitely contained as well in at least one of the balls $B_{\frac{e+u_k}{2}}^{\|\frac{e-u_k}{2}\|}$ (Theorem 3.2.1) constructed by the sites, which implies that h will be examined by at least one site. The proof follows immediately. □

According to Lemma 3.4.3, universally good predictors guarantee Property 3.4.1, thus leading to a decreased synchronization frequency. Nonetheless, in practice we cannot safely assume that predictors are always universally good. Hence, sites need to constantly check Inequality 3.1. This check can only be done by gathering all $\|e - u_i\|$, $\|e^p - u_i^p\|$ values to the coordinator so as to compute the respective minimum and maximum. But if we allow that, the number of messages will be equivalent to that of continuous, central data collection. Thus, in Section 3.5.4, we seek to devise alternative implementations for intersection monitoring that employ more relaxed conditions and call for a synchronization only when a violation of newly devised local constraints occurs.

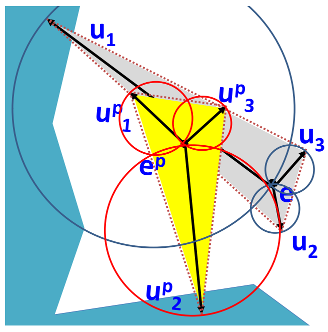
3.5 Simplified Alternatives

3.5.1 Relaxing the Containment Condition

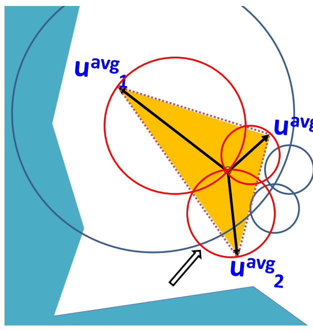
The containment of convex hulls (Section 3.4.1), as a sufficient prerequisite to achieve accordance with Property 3.4.1 is seemingly hard to achieve, let alone come up with ways to continuously check it in a distributed manner. To confront the above drawbacks we investigate an alternative approach which relaxes that condition. Rather than implementing distributed checks for the containment condition, we direct our interest to the more practical alternative of making it likely. Intuitively, we are looking for a way to monitor $v(t)$ such that:

Requirement 1: the local constraints in the shape of constructed balls are tighter than those of the original framework (Section 3.2.1)

Requirement 2: the choice of the reference point should be as close as possible to e (due to the establishment of Lemma 3.4.2)



(a) The original and the prediction-based convex hulls cross the threshold. The blue and red balls depict the areas monitored by each site, correspondingly, for the original and the prediction-based frameworks. S_2 violates Property 3.3.1 producing a larger prediction deviation (u_2^p) than the corresponding drift vector's (u_2) length.



(b) Convex hull and local constraints of the Average Model. Threshold crossing is prevented with stricter local constraints (except for S_2) and increased $\bigcup_{i=1}^{n} B^{\|\frac{e^p+e}{4}-\frac{u_i^p+u_i}{4}\|}_{\frac{e^p+e+u_i^p+u_i}{4}}$ area contained in the constraints of the original bounding algorithm.

Figure 3.3: The effect of the Average Model Adoption

since this pair of requirements renders the containment of new constraints in $\bigcup_{i=1}^{n} B^{\|\frac{e-u_i}{2}\|}_{\frac{e+u_i}{2}}$ more likely. Furthermore, we wish to invent an algorithm that avoids any communication among the sites, unless a threshold crossing is observed.

Since $v(t) = \frac{\sum_{i=1}^{n} w_i u_i^p}{\sum_{i=1}^{n} w_i}$ and $v(t) = \frac{\sum_{i=1}^{n} w_i u_i}{\sum_{i=1}^{n} w_i}$, for any $\mu \in R$ we can express the true global vector as $v(t) = \frac{\sum_{i=1}^{n} w_i(\mu u_i^p + (1-\mu)u_i)}{\sum_{i=1}^{n} w_i}$. So, in order to monitor the current status of the true global vector we may reside to a new convex hull, namely $Conv\ (\mu u_1^p + (1-\mu)u_1, \ldots, \mu u_n^p + (1-\mu)u_n)$. We then find ourselves concerned with identifying a value for μ that may fulfill Requirements 1 and 2.

Lemma 3.5.1. *For any $\frac{1}{2} \le \mu \le 1$, when Property 3.3.1 holds, tighter local constraints compared to the framework of Section 3.2.1 are guaranteed, i.e.: $\|(\mu u_i^p + (1-\mu)u_i) - (\mu e^p + (1-\mu)e)\| \le \|u_i - e\|$*

Proof.

$$\|u_i^p - e^p\| \le \|u_i - e\| \overset{\mu \ge 0}{\Leftrightarrow} \|\mu u_i^p - \mu e^p\| \le \|\mu u_i - \mu e\| \Leftrightarrow$$

$$\|\mu u_i^p - \mu e^p + (1-\mu)(u_i - e) + (\mu - 1)(u_i - e)\| \le \|\mu u_i - \mu e\|$$

By the triangle inequality:

$$\|\mu u_i^p - \mu e^p + (1-\mu)(u_i - e)\| - \|(\mu - 1)(u_i - e)\| \le \|\mu u_i - \mu e\| \Leftrightarrow$$

$$\|\mu u_i^p - \mu e^p + (1-\mu)(u_i - e)\| \leq (2\mu - 1)\|u_i - e\|$$

Obviously, $(2\mu - 1) \geq 0 \Leftrightarrow \mu \geq \frac{1}{2}$. The balls that are built by the original framework possess a radius of $\frac{\|u_i - e\|}{2}$ and for $\mu \leq 1$:

$$\|\mu u_i^p - \mu e^p + (1-\mu)(u_i - e)\| \leq (2\mu - 1)\|u_i - e\| \leq \|u_i - e\|$$

The latter inequality completes the proof. $\square$

The lemma above provides a rough upper, as well as lower, bound to the value of μ such that $\|(\mu u_i^p + (1-\mu)u_i) - (\mu e^p + (1-\mu)e)\| \leq \|u_i - e\|$ in every site. This means that sites construct tighter constraints than the ones they possessed using the original framework.

3.5.2 The Average Model

Lemma 3.5.1 shows that setting $\mu = \frac{1}{2}$ meets Requirement 1 and simultaneously provides beforehand some minimum knowledge with respect to the closest we can move $\mu e^p + (1-\mu)e$ towards e for Requirement 2 to be satisfied as well. Based on these, we are able to devise a first simpler alternative to the containment of convex hulls notion, which we term as the *"average model"*. The average model monitors $Conv\ (\frac{u_1^p + u_1}{2}, \ldots, \frac{u_n^p + u_n}{2}) \subset \bigcup_{i=1}^{n} B_{\frac{e^p + e + u_i^p + u_i}{4}}^{\|\frac{e^p+e}{4} - \frac{u_i^p + u_i}{4}\|}$ by a priori picking a value of $\mu = \frac{1}{2}$.

Figure 3.3 depicts an example of the Average Model adoption, where both the original and the prediction-based convex hulls cross the threshold surface in different areas (we used three sites to simplify the exposition). In Figure 3.3(a), notice that for S_2 Property 3.3.1 is violated. Despite this fact, as shown in Figure 3.3(b), the Average Model can still ward off threshold crossing, nearly achieving containment of its spheres in those of the original bounding algorithm.

3.5.3 The Safer Model

We now discuss an alternative model that relaxes Requirement 2. Following a rationale similar to [106], we observe that at any given time, the sites can individually choose the reference point $\mu e^p + (1-\mu)e$, $\frac{1}{2} \leq \mu \leq 1$ which is farther from the threshold surface and, simultaneously, ensures smaller local constraints. Note that by being far from the threshold surface, a reference point makes the local constraints of any predictor based monitoring model less likely to cause a crossing [106]. This second alternative is termed the *"safer model"*.

At the first step of the algorithm, every site starts with $\mu_1 = \frac{1}{2}$ and calculates $\mu_1 e^p + (1-\mu_1)e$. In addition, let e_1^* denote the vector lying on the threshold surface and being the closest to $\mu_1 e^p + (1-\mu_1)e$. Every site is capable of individually computing

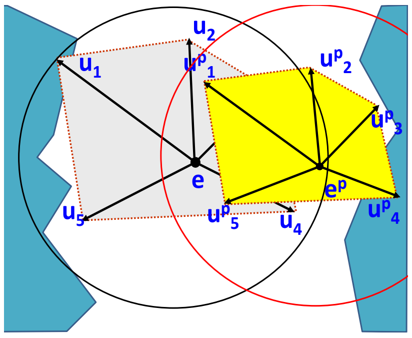


Figure 3.4: Loosened Intersection Monitoring. $\underset{i=1..n}{max}\, B_e^{\|e-u_i\|}$, $\underset{i=1..n}{max}\, B_{e^p}^{\|e-u_i\|}$ are produced by S_1 which is the one that checks $\underset{i=1..n}{max}\, B_e^{\|e-u_i\|} \cap \underset{i=1..n}{max}\, B_{e^p}^{\|e-u_i\|}$. No threshold crossing occurs despite that individual convex hulls violate the threshold surface.

$\|\mu_1 e^p + (1-\mu_1)e - e_1^*\|$ and, thus, determine the distance the first examined reference point yields. To reduce the computational requirements of the technique, we define a number of allowed steps θ, such that in every subsequent step $1 \leq j \leq \theta$ the sites employ a value of $\mu_j = \mu_{j-1} + \frac{1}{2\theta}$ until $\mu_\theta = 1$. Eventually, the μ_j value that induces the largest distance is chosen. Notice that using this framework, the sites can reach a consensus regarding μ without any additional communication. This happens due to the fact that the choice of the final μ is based on common criteria related to the threshold surface and the e, e^p vectors that are known to all sites.

3.5.4 Loosened Intersection Monitoring

So far in this section we have proposed simplistic alternatives that relax the convex hull containment condition that was discussed in Section 3.4.1. The presented (*average* and *safer*) predictor based monitoring models do manage to avoid any direct communication between the sites unless a threshold crossing is detected. Although they do not necessarily abide by Property 3.4.1, these models encompass Requirements 1 and 2 (for the average model) and are, thus, in practice likely to substantiate a condition that is hard to check in a distributed manner.

We next aim at inventing a loosened version for the intersection monitoring model of Section 3.4.2. As before, we wish to come up with a mechanism that avoids any communication between sites unless a threshold crossing happens and simultaneously makes Property 3.4.1 very likely. Property 3.3.1 is again set as a simple prerequisite, but note that all our algorithms in this section remain correct even if it does not hold, since local constraints still totally cover the monitored area of the input domain. In Section 3.4.2 we saw that $v(t) \in Conv(u_1, \cdots, u_n) \cap Conv(u_1^p, \cdots, u_n^p)$ while in this section we demonstrated that $v(t)$ also lies in any $Conv\ (\mu u_1^p + (1-\mu)u_1, \cdots, \mu u_n^p + (1-\mu)u_i)$ which for $\frac{1}{2} \leq \mu \leq 1$ possesses the desired characteristics formulated in Requirements 1 and 2. The following lemma provides a primitive result on how the intersection monitoring can be achieved using the aforementioned logic. For ease of exposition, we use $Conv^3_\cap$ to denote the triple intersection of these three (original, predicted and weighted) convex hulls, while $Sur(Conv^3_\cap) \equiv \underset{i=1..n}{max}\, B_e^{\|e-u_i\|}$ $\cap \underset{i=1..n}{max}\, B_{e^p}^{\|e^p-u_i^p\|} \cap \underset{i=1..n}{max}\, B_{\mu e^p+(1-\mu)e}^{\|\mu e^p+(1-\mu)e-\mu u_i^p-(1-\mu)u_i\|}$, where $\underset{i=1..n}{max}\, B_c^{\|r\|}$ denotes the corresponding (in each maximization term) ball of maximum radius.

Lemma 3.5.2. *For any $\mu \in R$, the area inscribed in $Conv^3_\cap$ is covered by the region induced by $Sur(Conv^3_\cap)$.*

Proof. Initially notice that:

$$Conv(u_1, \dots, u_n) \subset \bigcup_{i=1}^{n} B_{\frac{e+u_i}{2}}^{\|\frac{e-u_i}{2}\|} \subset \underset{i=1..n}{max}\, B_e^{\|e-u_i\|} \tag{3.2}$$

$$Conv(u_1^p, \dots, u_n^p) \subset \bigcup_{i=1}^{n} B_{\frac{e^p+u_i^p}{2}}^{\|\frac{e^p-u_i^p}{2}\|} \subset \underset{i=1..n}{max}\, B_{e^p}^{\|e^p-u_i^p\|} \tag{3.3}$$

$$Conv(\mu u_1^p + (1-\mu)u_1, \dots, \mu u_n^p + (1-\mu)u_n) \subset$$
$$\bigcup_{i=1}^{n} B_{\frac{\mu e^p+(1-\mu)e+\mu u_i^p+(1-\mu)u_i}{2}}^{\|\frac{\mu e^p+(1-\mu)e-\mu u_i^p-(1-\mu)u_i}{2}\|} \subset \underset{i=1..n}{max}\, B_{\mu e^p+(1-\mu)e}^{\|\mu e^p+(1-\mu)e-\mu u_i^p-(1-\mu)u_i\|} \tag{3.4}$$

So each time the maximum balls cover the corresponding convex hulls. We want to prove that the intersection of the latter balls also covers the intersection of the convex hulls. The proof will be derived by contradiction.

Suppose that a vector $h \in Conv^3_\cap$ exists. Now assume that the vector h does not lie in at least one of $\underset{i=1..n}{max}\, B_e^{\|e-u_i\|}$, $\underset{i=1..n}{max}\, B_{e^p}^{\|e^p-u_i^p\|}$, or $\underset{i=1..n}{max}\, B_{\mu e^p+(1-\mu)e}^{\|(\mu e^p+(1-\mu)e)-(\mu u_i^p+(1-\mu)u_i)\|}$. However, this would violate at least one of the Propositions 3.2,3.3,3.4, which is a contradiction. This concludes the proof. □

Note, however, that the intersection cover identified in Lemma 3.5.2 cannot be tracked in a distributed manner, since the site that determines each maximum ball may be different. Should Property 3.3.1 and, thus, (for $\frac{1}{2} \leq \mu \leq 1$) Lemma 3.5.1 hold, what sites actually need to perform so that they can track (in a distributed manner) $Conv^3_\cap$ is to use $B_e^{\|e-u_i\|}$, $B_{e^p}^{\|e-u_i\|}$ and $B_{\mu e^p+(1-\mu)e}^{\|e-u_i\|}$. To understand this, observe that if both $\|e^p - u_i^p\|$, $\|(\mu e^p +(1-\mu)e) - \mu u_i^p - 1 - \mu)u_i\| \leq \|e - u_i\|$ (due to Property 3.3.1 and Lemma 3.5.1, respectively), it is evident that: $Sur(Conv^3_\cap) \subset \underset{i=1..n}{max}\, B_e^{\|e-u_i\|} \cap$ $\underset{i=1..n}{max}\, B_{e^p}^{\|e-u_i\|} \cap \underset{i=1..n}{max}\, B_{\mu e^p+(1-\mu)e}^{\|e-u_i\|}$.

Hence, the site that possesses the maximum $\|e - u_i\|$ will check whether the intersection of its locally constructed balls crosses the threshold. Provided that the intersection of the local balls does not cross the threshold at any site (and, thus, at the site with the maximum $\|e - u_i\|$ as well), synchronization can safely be avoided. At this point, we would be interested in identifying proper values for μ that refine the range ($\frac{1}{2} \leq \mu \leq 1$) established in Lemma 3.5.1. Nonetheless, the following corollary shows that if we have to employ $\|e - u_i\|$ as the radius of the balls, $\underset{i=1..n}{max} B^{\|e-u_i\|}_{\mu e^p + (1-\mu)e}$ does not reduce the volume of the intersection.

Corollary 3.5.1. *When Property 3.3.1 holds and for any* $0 \leq \mu \leq 1$, $\underset{i=1..n}{max} B^{\|e-u_i\|}_{\mu e^p + (1-\mu)e}$ *does not reduce the volume of the region induced by* $\underset{i=1..n}{max} B^{\|e-u_i\|}_{e} \cap \underset{i=1..n}{max} B^{\|e-u_i\|}_{e^p}$.

Proof. Assume that there exists a vector $h \in \underset{i=1..n}{max} B^{\|e-u_i\|}_{e} \cap \underset{i=1..n}{max} B^{\|e-u_i\|}_{e^p}$. Then:

$$\|h - e\| \leq \underset{i=1..n}{max} \|e - u_i\| \overset{1 \geq \mu}{\Rightarrow} (1-\mu)\|h - e\| \leq (1-\mu) \underset{i=1..n}{max} \|e - u_i\|$$

and $\|h - e^p\| \leq \underset{i=1..n}{max} \|e - u_i\| \overset{\mu \geq 0}{\Rightarrow} \mu\|h - e^p\| \leq \mu \underset{i=1..n}{max} \|e - u_i\|$.

Summing the above and merely applying the triangle inequality we obtain:

$$\|h - (\mu e^p + (1-\mu)e)\| \leq \mu\|h - e^p\| + (1-\mu)\|h - e\| \leq \underset{i=1..n}{max} \|e - u_i\|$$

The latter result exhibits that if a vector lies in the intersection of $\underset{i=1..n}{max} B^{\|e-u_i\|}_{e}$ $\cap \underset{i=1..n}{max} B^{\|e-u_i\|}_{e^p}$, it will definitely lie in the $\underset{i=1..n}{max} B^{\|e-u_i\|}_{\mu e^p + (1-\mu)e}$ as well. Consequently that ball does not contribute to refining the monitored area. □

Corollary 3.5.1 is true for any $0 \leq \mu \leq 1$, but note that in order to ensure $\|e^p - u^p_i\|$ $\leq \|e - u_i\|$ and $\|(\mu e^p + (1-\mu)e) - \mu u^p_i - (1-\mu)u_i\| \leq \|e - u_i\|$, we had already assumed that $\frac{1}{2} \leq \mu \leq 1$. Hence it suffices to check whether the pair $B^{\|e-u_i\|}_{e} \cap B^{\|e-u_i\|}_{e^p}$ crosses the threshold in at least one site [2]. Figure 3.4 provides an exemplary application of the intersection monitoring procedure described so far, where $\underset{i=1..n}{max} B^{\|e-u_i\|}_{e}$, $\underset{i=1..n}{max} B^{\|e-u_i\|}_{e^p}$ are produced by S_1.

On the other hand, following an intuition similar to the one utilized in the average model, an alternative is to track $Conv\ (u^p_1, \cdots, u^p_n) \cap Conv\ (\frac{u^p_1 + u_1}{2}, \cdots, \frac{u^p_n + u_n}{2})$ instead. In other words, this time each site S_i needs to individually construct two balls using e^p and $\frac{e^p + e}{2}$ as centers and $M = max\{\|e^p - u^p_i\|, \|\frac{e^p + e}{2} - \frac{u^p_i + u_i}{2}\|\}$ as the common radius (note that M refers to the maximum of the pair of local radii). Subsequently, a synchronization is caused when at least one S_i detects that the locally constructed intersection crosses the threshold.

We conclude our study by showing the condition which makes the latter intersection tracking preferable as it results in smaller local constraints compared to $\underset{i=1..n}{max} B^{\|e-u_i\|}_{e}$ $\cap \underset{i=1..n}{max} B^{\|e-u_i\|}_{e^p}$.

[2] Even if the site that determines the maximum radius finds that Property 3.3.1 does not hold, Corollary 3.5.1 is still valid upon replacing $\|e - u_i\|$ with $\|e^p - u^p_i\|$.

Proposition 3.5.1. *When Property 3.3.1 holds and* $\underset{i=1..n}{max}\, B_e^{\|e-u_i\|} \supseteq \underset{i=1..n}{max}\, B^M_{\frac{e^p+e}{2}}$, *then:*

$$\underset{i=1..n}{max}\, B^M_{e^p} \cap \underset{i=1..n}{max}\, B^M_{\frac{e^p+e}{2}} \subseteq \underset{i=1..n}{max}\, B_e^{\|e-u_i\|} \cap \underset{i=1..n}{max}\, B_{e^p}^{\|e-u_i\|}$$

Proof. For any vector $h \in \underset{i=1..n}{max}\, B^M_{e^p} \cap \underset{i=1..n}{max}\, B^M_{\frac{e^p+e}{2}}$ we have:

$$\|h - e^p\| \leq M \overset{\|e^p-u_i^p\| \leq \|e-u_i\|}{\leq} max\|e - u_i\|$$

which entails that $h \in \underset{i=1..n}{max}\, B_{e^p}^{\|e-u_i\|}$. If in addition $\underset{i=1..n}{max}\, B_e^{\|e-u_i\|} \supseteq \underset{i=1..n}{max}\, B^M_{\frac{e^p+e}{2}}$ then $h \in \underset{i=1..n}{max}\, B_e^{\|e-u_i\|}$ as well. Hence $h \in \underset{i=1..n}{max}\, B_e^{\|e-u_i\|} \cap \underset{i=1..n}{max}\, B_{e^p}^{\|e-u_i\|}$ □

3.5.5 Choosing Amongst Alternatives

So far, we investigated a number of simpler alternatives that loosen the strong monitoring frameworks of Section 3.4, i.e., the convex hull containment as well as the intersection monitoring framework. We based our analysis on Property 3.3.1 as an intuitive assumption also employed in past studies [23, 20, 22] and evolved it to practical tracking mechanisms together with appropriate speculative analysis. Nonetheless, upon relaxing the monitoring conditions we also relaxed their conformity to Property 3.4.1, i.e., the prerequisite for strong predictor-based monitoring models. Since the coordinator is supposed to a priori dictate the predictor-based tracking alternative that should be uniformly utilized by sites at least until the next synchronization, we need to provide a decision making mechanism that enables it choose among the available options and adjust its decisions on their anticipated performance with respect to communication savings.

The available tracking options that do not belong (excluding the trivial choice of the original framework) to the strong predictor-based monitoring models' class include:

- Monitoring of $Conv(u_1, \ldots u_n)$ as in Section 3.2.1
- Monitoring of $Conv(u_1^p, \ldots, u_n^p)$ as in Section 3.3
- Adoption of the average model
- Adoption of the safer model
- Tracking of $\underset{i=1..n}{max}\, B_e^{\|e-u_i\|} \cap \underset{i=1..n}{max}\, B_{e^p}^{\|e-u_i\|}$
- Tracking of $\underset{i=1..n}{max}\, B^M_{e^p} \cap \underset{i=1..n}{max}\, B^M_{\frac{e^p+e}{2}}$

In order to provide an appropriate decision making mechanism, we require that sites keep up monitoring all the six options mentioned above. This monitoring will take place **only** for models that would not result in **any** local transmission since the last synchronization (i.e., we stop monitoring an alternative model for which we detect that a transmission would have been caused). Notice that one model has been chosen as the main model after the last synchronization. Thus, for each of the 6 alternatives, the sites maintain 6 bits, where the i-th bit is set iff the corresponding monitoring mode would have resulted in at least one transmission since the last synchronization. A synchronization can still be caused only by the main model. Upon a synchronization, however, together with $v_i(t)$ and the velocity vector in the case of the velocity acceleration model choice, sites attach 5 bits (they do not need to send a bit for the current model being used) on their messages. Note that the following facts hold in our adaptive algorithm:

- No site had a violation using the current model in a previous time instance (since the previous synchronization).
- An alternative model that has its corresponding bit to 1 in **any** of the sites would not have been better than the model currently being used, since it would have resulted in a transmission in a prior (or the current) time instance.
- Based on the above observation, we decide to switch to an alternative model only if the corresponding bits for this model were equal to 0 in **all** the sites.

In case of multiple alternatives with unset bits, a random choice among such alternatives is performed.

3.6 Evaluation Results

In order to evaluate our algorithms we developed a simulation environment in Java. We utilized two real data sets to derive data stream tuples arriving at every site in the network. "Corpus", consists of 804,414 records present in the Reuters Corpus (RCV1-v2) [76] collection. Each record corresponds to a news story to which a list of terms (features) and appropriate categorization have been attributed. As in [106, 105] we focused on the following features: *Bosnia, Ipo, Febru* while monitoring their co-existence with the *CCAT* (the CORPORATE / INDUSTRIAL) category. Aiming at identifying the relevance of these features to the CCAT category at any given time, we monitored two different functions involving the Chi-Square(χ^2) and Mutual Information(*MI*) score. We utilized the Corpus data set in order to test our techniques using the Cash Register streaming paradigm i.e. taking into consideration the whole history of the tuples arriving at the various sites. In any given timestamp, after the receipt of a new tuple each site forms a vector which consists of four dimensions for the χ^2 and three dimensions for the *MI* case. These vectors have one of their positions set, while the rest remain zero. In particular, for both the functions the first position of the vector is set if the term and the category co-occur, the second if the term occurs without the CCAT category, the third in case CCAT is present without the term, while the fourth (only for χ^2 score) if neither of them appeared in the incoming tuple.

Due to the nature of the incoming (binary) vectors and the utilized Cash Register paradigm the previously described environment may be considered moderate to change and be thought of as easily predictable by our techniques. In order to test our algorithms in more dynamic conditions we utilized one more data set. The ”Weather” data set includes Solar Irradiance, Wind Speed and Wind Peak measurements from the station in the University of Washington and for the year 2002 [27], where each file incorporates 523,439 records of measurements. We used the Weather data sets so as to monitor the Variance (Var) and the Signal to Noise Ratio (StN) functions. We utilized Var since it has already been used within the geometric monitoring framework [104]. In addition, the StN function equals the ratio between the mean and the standard deviation ($\frac{\mu}{\sigma}$) in a given window of measurements and can, thus, be applicable to globally quantify the noise present in the measurements.

In each experiment we first measure the number of messages transmitted in the network across different thresholds for a network configuration consisting of 10 sites. We then use the middle case threshold and plot the number of transmitted messages when increasing the scale of the distributed environment (the number of sites). We denote the performance of the original bounding algorithm (Section 3.2.1) by ”Model 0”, while ”Model 1” refers to the mere application and monitoring of the prediction-based bounding algorithm (Section 3.3). Eventually, ”CAA” shows the performance of the Choosing Amongst Alternatives framework that was introduced in Section 3.5.5. Moreover, for each of the lines in the graphs, we enclosed the chosen predictor using LG to denote the Linear Growth predictor and $VA - W$ so as to declare a Velocity / Acceleration predictor with a window of W measurements [3].

3.6.1 Corpus Data Set - Cash Register Paradigm

We begin our study by examining the performance of our techniques in the Corpus data set on par with the Cash Register paradigm adoption. Figure 3.5 depicts the performance of Model 0 and of the CAA approach when using the LG and the VA predictors. Since almost 6000 documents are received within a period of a month [106], we choose a $W = 200$ window for the VA predictor which is expected to be roughly the amount of news stories received daily. Each column of the figure corresponds to the case of the terms “Bosnia”, “Ipo” and “Febru”, respectively.

Sensitivity to Threshold - Chi Square. As shown in the first row of Figure 3.5, where the χ^2 function for the term ”Bosnia” is monitored, Model 0 appears to always be about 2 and 1.85 times worse in terms of the number of transmitted messages when

[3]We focus on comparing the performance of our prediction-based geometric monitoring techniques against [103, 105] (Model 0), since we expect our prediction-based methods to give similar benefits when operating over the ellipsoidal bounding regions of [106] to those seen in our current study using spherical constraints as in [103, 105].

compared to the CAA(LG) and CAA(VA-200) approaches, respectively, for different threshold values (Fig. 3.5(a)) using 10 sites.

Sensitivity to Threshold - Mutual Information (MI). Moving to the second and third rows of Figure 3.5(a) we investigate the cases of the ”Ipo” as well as ”Febru” terms, monitoring the *MI* function across different threshold values for a 10 site configuration (note that *MI* is calculated as a logarithm, therefore the negative threshold values in that axis). In these graphs the peak that occurs at 0.4 and 0 for the ”Ipo” and ”Febru” involves an accumulation of synchronizations around the average value the *MI* function possesses along the run. We again observe that CAA(LG) performs 1.75-2.1 times better than the Model 0 case for both monitored terms. Despite the fact that CAA(VA-200) is proved slightly worse compared to CAA(LG), it is still able to better amend the peak that occurs in ”Febru” monitoring for a 0 threshold.

Sensitivity to Number of Sites. Eventually, switching to Figure 3.5(b), for the “Bosnia” term (first row) the relative benefits remain almost the same across all network scales. For the ”Ipo” term monitoring (middle row) we observe that Model 0 is steadily more than 1.8 times worse than the CAA(LG) choice across different scales. CAA(VA-200) performs worse than the CAA(LG) case for network configurations up to 80 sites. Nonetheless, the introduction of additional sites (along with their respective sub-streams) in 90, 100 site cases, causes the *MI* function to always lie below the posed zero threshold since the ”Ipo” term becomes more rare. This fact is perfectly read by the CAA(VA-200) approach, the transmitted messages of which approach zero. A similar behavior appears early in the third row of Figure 3.5(b) for the Febru case where the introduction of more than 10 sites causes *MI* to be negative, which is again accurately pinpointed by the CAA(VA-200) monitoring model reaching savings of 3 orders of magnitude size.

In the previously presented graphs we omitted the lines for the mere application of Model 1 to keep the diagram readable, since Model 1 shows almost identical (actually CAA can occasionally save a few tens of extra messages) behavior with its corresponding CAA applications. CAA possesses the ability to recognize the utility of Model 1 (the mere application of predictors as described in Section 3.3) in this setting and encompass it throughout its operation. On the other hand, this fact exhibits the ability of Model 1 to provide an efficient solution in environments where v_i values evolve relatively slowly. Nonetheless, as we will shortly present, this is not always true in scenarios where more dynamic updates occur.

3.6.2 Weather Data - Sliding Window Paradigm

We proceed to the sliding window operation, using the Weather data set and monitoring the Var and StN functions. Note that in all the graphs presented in the current subsection the Linear Growth predictor is not applicable since it assumes that local vectors

(v_i) uniformly evolve by a time dependent factor (Section 3.2.2), which is obviously unrealistic for the physical measurements in the Weather data and the sliding window application scenario. We thus compare the performance of Model 0, Model 1(VA-W) and CAA(VA-W) cases in our study. For the CAA(VA-W) monitoring model we again choose the window based on natural time units' division. We uniformly utilize a prediction window $W = 10$ which corresponds to the latest minutes of received observations, except for the Wind Peak data where $W = 50$ was chosen to adjust predictions to the expected frequency of the peaks in the wind blasts. For each data set-function pair, the default value of the sliding window size, over which the corresponding function is computed, is 200 measurements. However, we also perform a sensitivity analysis on this parameter as well.

Variance Monitoring. Figure 3.6(a) plots the performance of the techniques in the case of Var monitoring in the Solar Irradiance Data. In the first row of the figure we observe that the cost of Model 0 ranges between 11 and 600 times larger than the cost yielded by CAA(VA-10) monitoring model, while CAA(VA-10) ensures up to 500 times lower cost even when compared with Model 1 across different thresholds. A case of particular interest shows up for a threshold of 30000. There, Model 1 shows a peak in the number of transmitted messages which are higher even when compared to Model 0. This happens due to the existence of specific sites whose drift vectors approach the threshold surface as noted in Figure 3.2. Obviously, increasing the threshold to 40000 alters the threshold surface and thus hinders the same sites to cause threshold violations. Nonetheless, CAA(VA-10) maintains low transmission cost due to the loosened intersection monitoring capabilities (Section 3.5.4) that it embodies. We will revisit this issue in the next subsection where we look into the operational details of the CAA monitoring model. In the meantime we note that the same applies for the second row of Figure 3.6(a) where increasing the scale of the network results in CAA(VA-10) savings that reach a number of 30 times compared to Model 0 and they become even larger when compared to Model 1. Eventually, the third row of the same figure, shows the resilience of our techniques when altering the employed size of observations encapsulated in the sliding window for 10 sites. CAA(VA-10) shows similar behavior when enlarging the window. Model 1 yields more synchronizations for a window of 200 observations since enlarging the window causes the variance values within it to increase and thus some sites approach the posed threshold of 50000. The lack of the alternative mechanisms that are incorporated in CAA leads sites merely utilizing Model 1 to threshold crossings. Finally, Model 0 exhibits high sensitivity to the number of values that local vectors (v_i) are built upon.

Figure 3.6(b) presents corresponding results for Var monitoring in the Wind Speed data set. Model 1 is slightly worse (almost 1%) in terms of transmitted messages compared to CAA(VA-10) when varying the threshold (first row in the figure) and across different network scales (second row), yet both result in savings ranging between 3

Threshold	Model 0	Model 1	Average	Safer	Intersection 1	Intersection 2
10000	0	15	0	3	0	0
30000	105	0	0	0	4	25
50000	7	0	0	0	5	3
70000	0	0	1	0	0	1
90000	0	0	1	0	0	1

Table 3.3: Case study: Solar-Var Vs Threshold Monitoring

and 13 times compared to the message cost of Model 0. Furthermore, in the third row of Figure 3.6(b) we observe that both CAA(VA-10) and Model 1 remain resilient to altering the sliding window size ensuring significant benefits when compared to Model 0. Notice that for a window of 100 observations, Model 1 performs better than CAA(VA-10). Recall that CAA resolves ties in the choice of the monitoring mechanism (Section 3.5.5) by picking a random model among those which did not cause a threshold crossing. Thus, when a particular model is always the appropriate choice, the adaptive CAA algorithm may sometimes end up transmitting slightly more messages. The results are similar for the Wind Peak data set.

Signal to Noise Monitoring. In our next experiment we utilized the same motif for analyzing the performance of our techniques in monitoring the StN function. We begin our discussion with the Solar Irradiance data set in Figure 3.7(a). CAA(VA-10) performs up to 3 times better than Model 0 when varying the threshold for 10 sites (first row in the figure) and up to 5 times across network configurations of 10-100 sites (second row). Model 1 is again the worst choice as it yields 2-5 times higher cost compared to Model 0 across different thresholds and appears over 2 times worse than Model 0 for different scales. In the third row of Figure 3.7(a), it is evident that CAA(VA-10) again remains mostly unaffected to different window sizes, while Model 1 exhibits a wavy behavior depending on the accuracy of the employed VA-10 predictor.

We then analyze the performance of the Wind Peak Data in StN monitoring (Figure 3.7(b)) (the Wind Speed data had similar behavior). Model 1 and CAA possess similar performance across different thresholds with savings ranging between 4 and 85 times compared to the cost of Model 0. The same holds in large part when varying the network scale (middle row in Fig 3.7(b)) where savings reach a factor of 5. An exception occurs for 40 and 50 site configuration cases. This is another occasion where site predictors lie close to the threshold surface for the given threshold of 0.5 and CAA manages to achieve increased savings due to the intersection monitoring capacity. As more sites are added in the subsequent steps (60-100 site configurations) the predicted estimate's (e^p) position is affected and thus the sites that were previously causing synchronizations (despite their restricted local constraints - balls) were ousted from the threshold surface, stabilizing the cost of Model 1. Eventually, as with the previously examined functions-data set pairs, the CAA monitoring model is not sensitive to changes in the window size (third row of Fig 3.7(b)) while Model 0 and Model 1

# Sites	Model 0	Model 1	Average	Safer	Intersection 1	Intersection 2
10	35	16	25	35	1	3
40	8	6	12	19	0	1
50	13	10	7	17	0	1
80	12	14	9	14	0	1
90	9	9	12	21	0	1

Table 3.4: Case study: Wind Peak-StN Vs # Sites Monitoring

exhibit opposite trends upon enlarging it. Overall, Model 0's cost is 5 to 35 times the transmission cost of CAA, while savings against Model 1 range between 4 to 9 times across different window sizes.

3.6.3 CAA Operational Insights

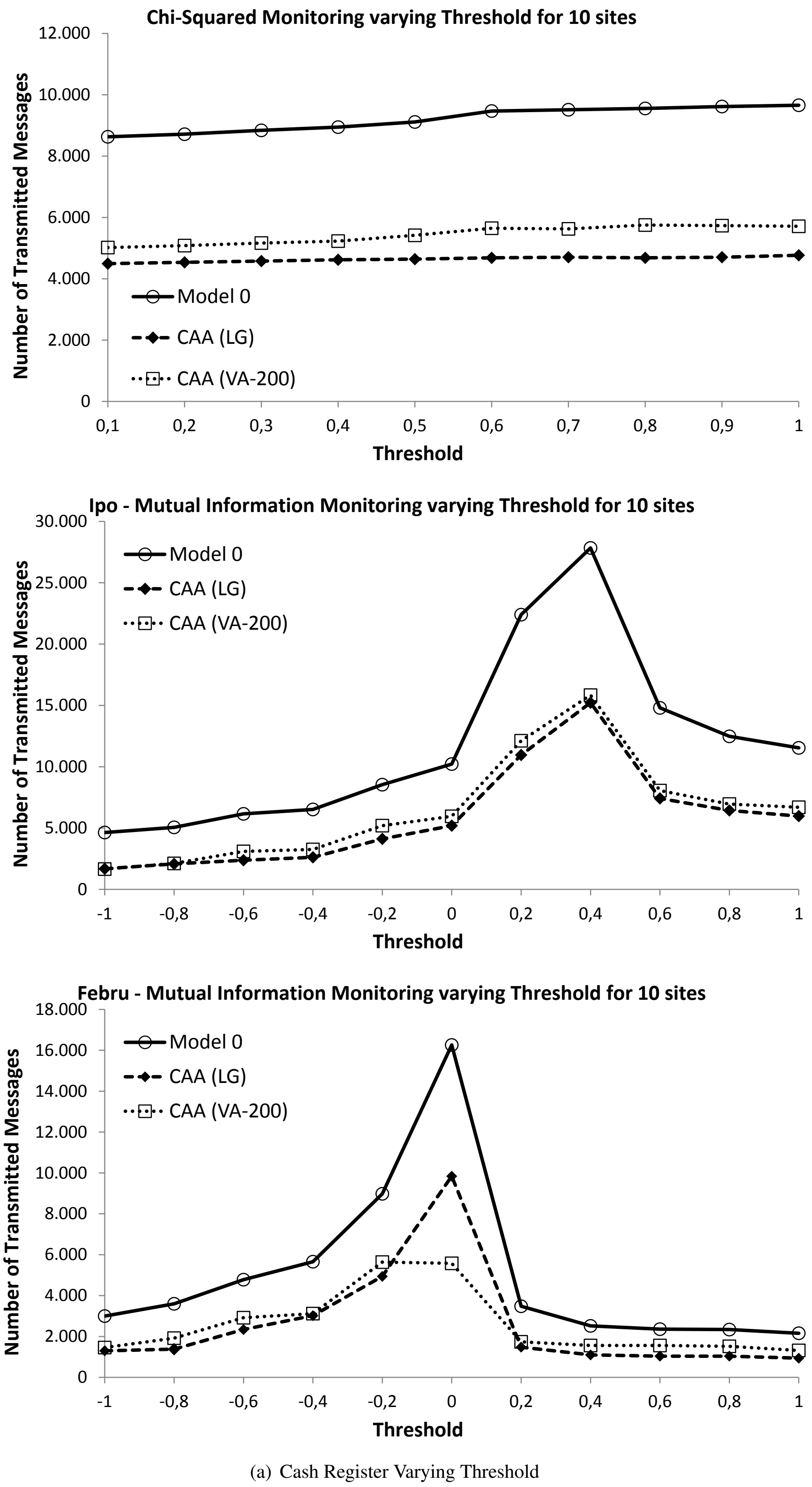
We are now providing additional details regarding the choices that CAA makes throughout its operation to investigate the stem of its benefits. Since it is hard to present analytic statistics of alternative models' usage for every single case of the previously discussed graphs, we focus on two situations where Model 1 exhibits possibly unexpected peaks in the number of messages and examine the tools that CAA utilizes to avoid similarly high message exchange.

The first of the aforementioned cases regards the Solar Irradiance under Var monitoring against different thresholds and for 10 sites(first row of Fig 3.6(a)). Table 3.3 shows the CAA choices for different thresholds. Intersection1 refers to monitoring the intersection between the original and the predicted convex hull, while Intersection2 refers to monitoring the intersection between the average convex hull and the predicted one. We point out that for threshold ¿10000 (where it exhibits low costs) Model 1 is never employed by CAA. For the threshold 30000 case, Model 0 appears as the most frequent choice but it is only used during the first synchronizations until predictors are stabilized around the threshold surface (if Model 0 was continuously picked, CAA would have had similar cost to Model 0). Afterwards, the loosened intersection framework is chosen which safely leads the monitoring procedure to the decrement of the transmission cost as shown in Fig 3.6(a).

The second case we distinguished during our discussion was the peak that occurs when monitoring the Wind Peak data under the StN function for network configurations of different scale (middle row of Fig. 3.7(b)). As Table 3.4 shows, for 10 sites the savings CAA provides are mostly attributed to the average and safer model usage, while for 40, 50 and more sites, after a few synchronizations, the single time that Intersection2 is employed by CAA hinders communication for a considerable amount of time.

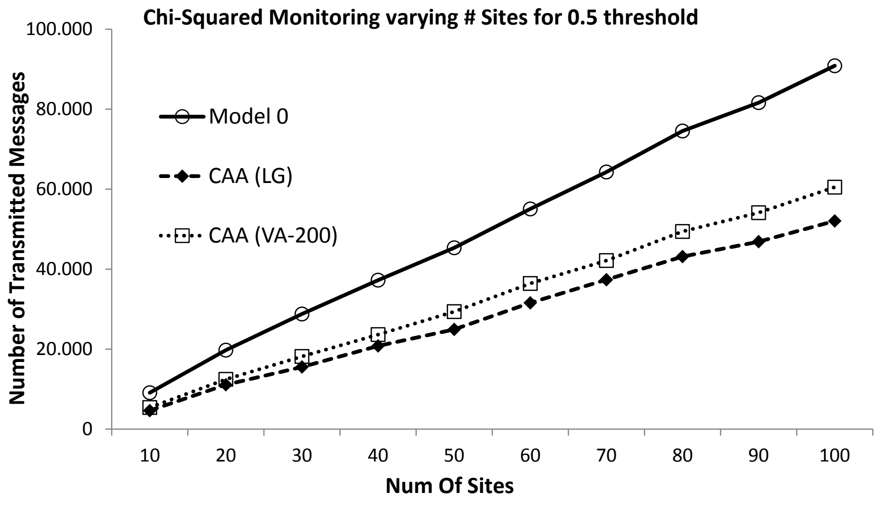
3.7 Synopsis

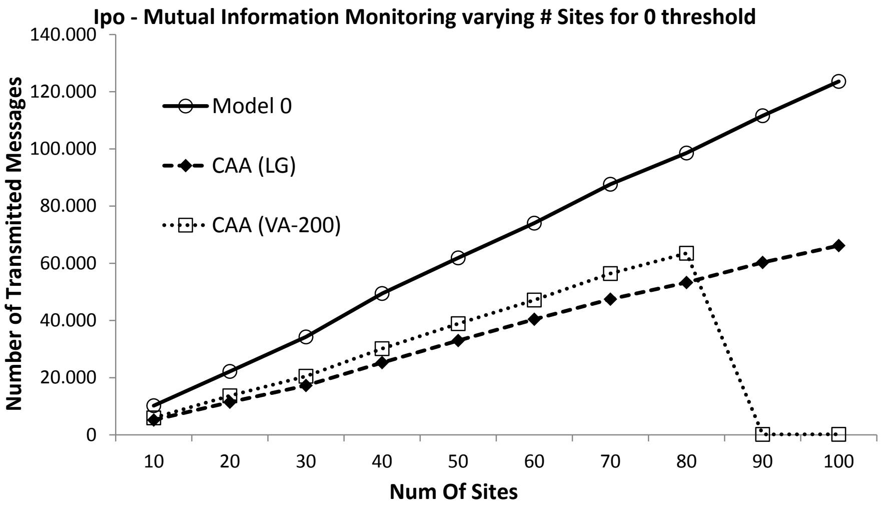
In this chapter, we presented a thorough study regarding prediction models' adoption within the geometric monitoring setting. After identifying the peculiarities exhibited by predictors upon their implementation in the aforementioned environment, we developed a solid theoretic framework composed of sufficient conditions rendering predictors capable of refraining the communication burden. We proposed algorithms based on relaxed versions of these conditions along with extensive theoretical analysis on their expected benefits. In the next chapter, we present the adoption of our prediction - based geometric monitoring techniques to the special application scenario of distributively tracking representative trajectories over a number of monitored moving objects.

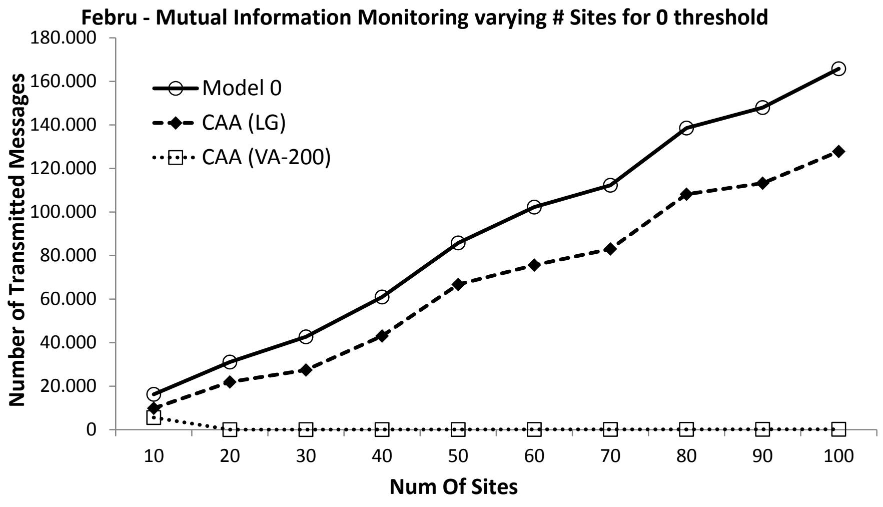


(a) Cash Register Varying Threshold

Figure 3.5: Corpus Data Set: Performance of our Techniques in the Cash Register Streaming Paradigm

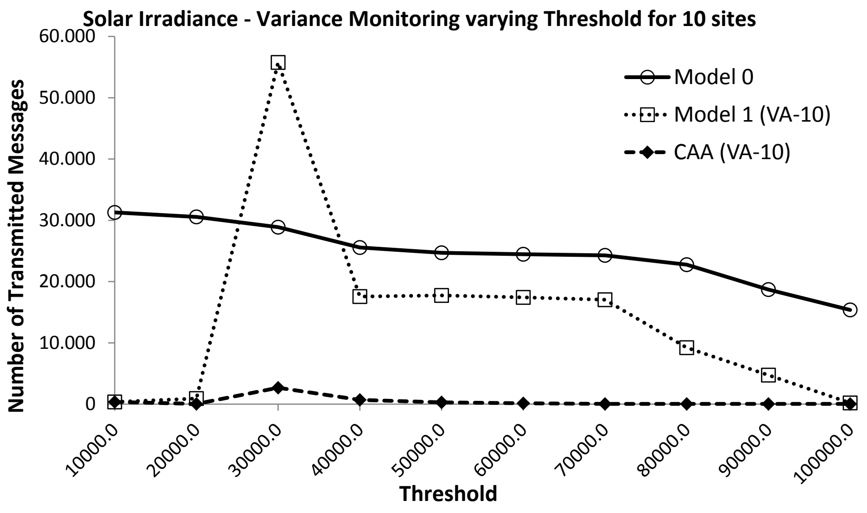


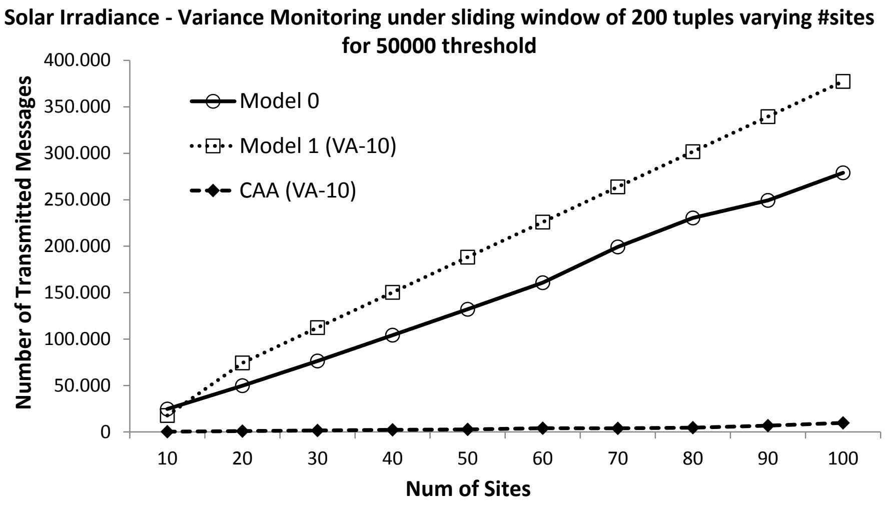


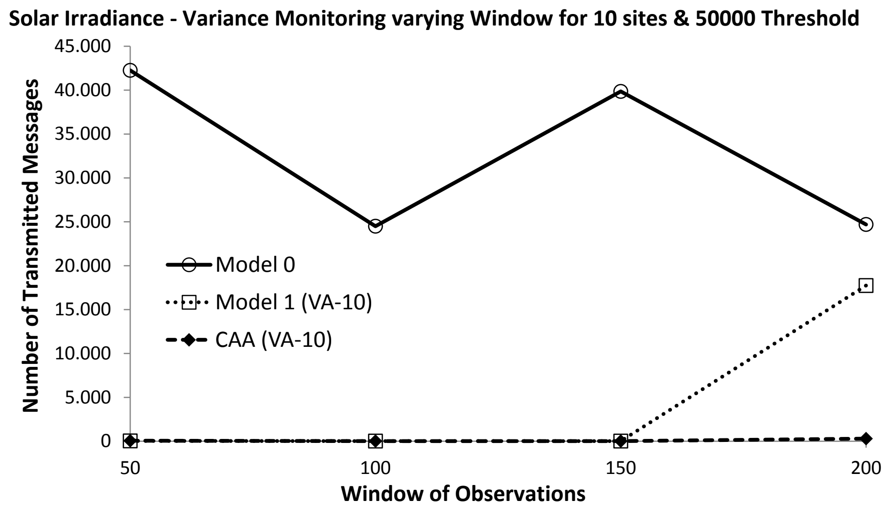


(b) Cash Register Varying Sites

Figure 3.5: Corpus Data Set: Performance of our Techniques in the Cash Register Streaming Paradigm (cont.)

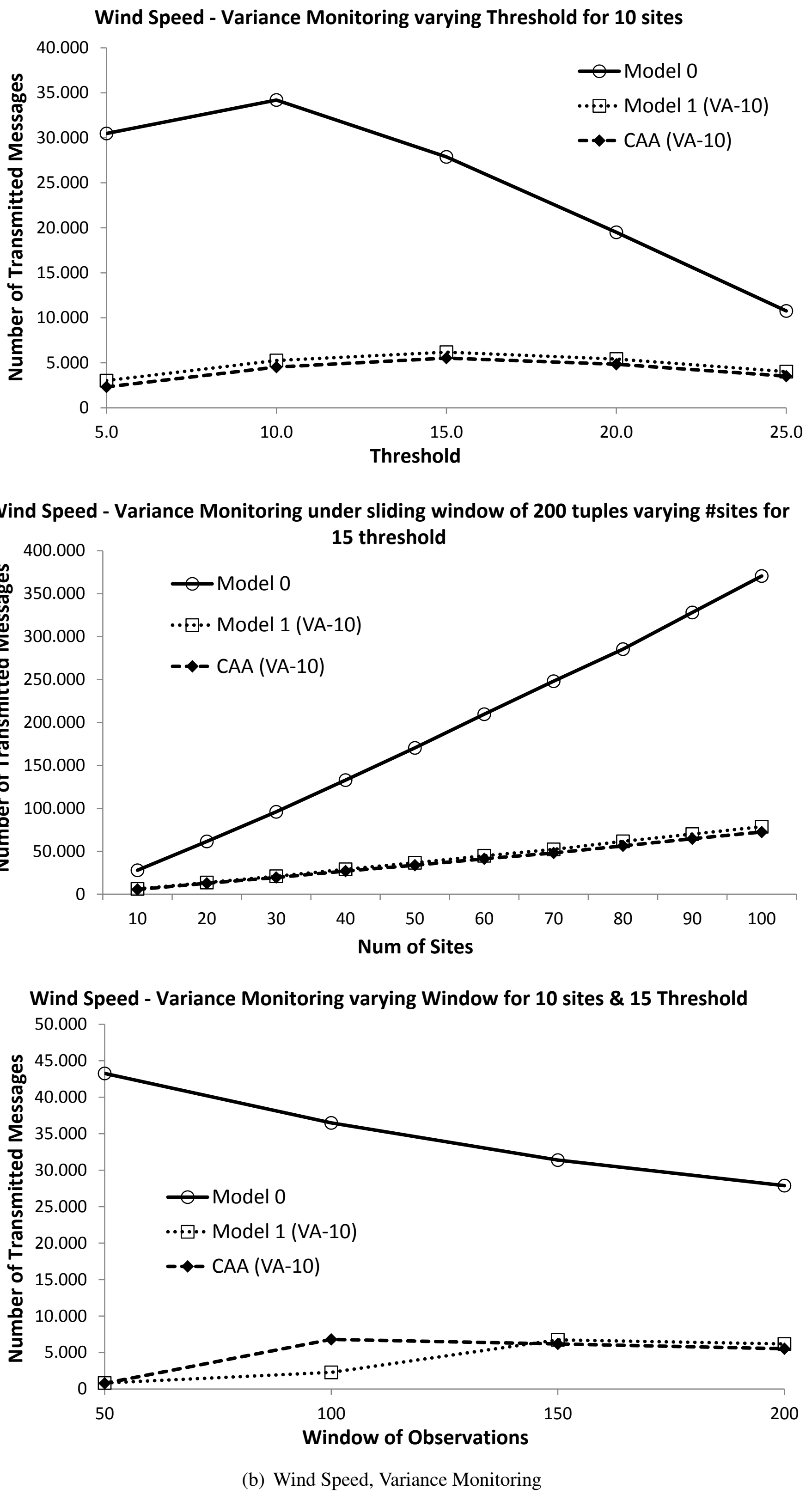






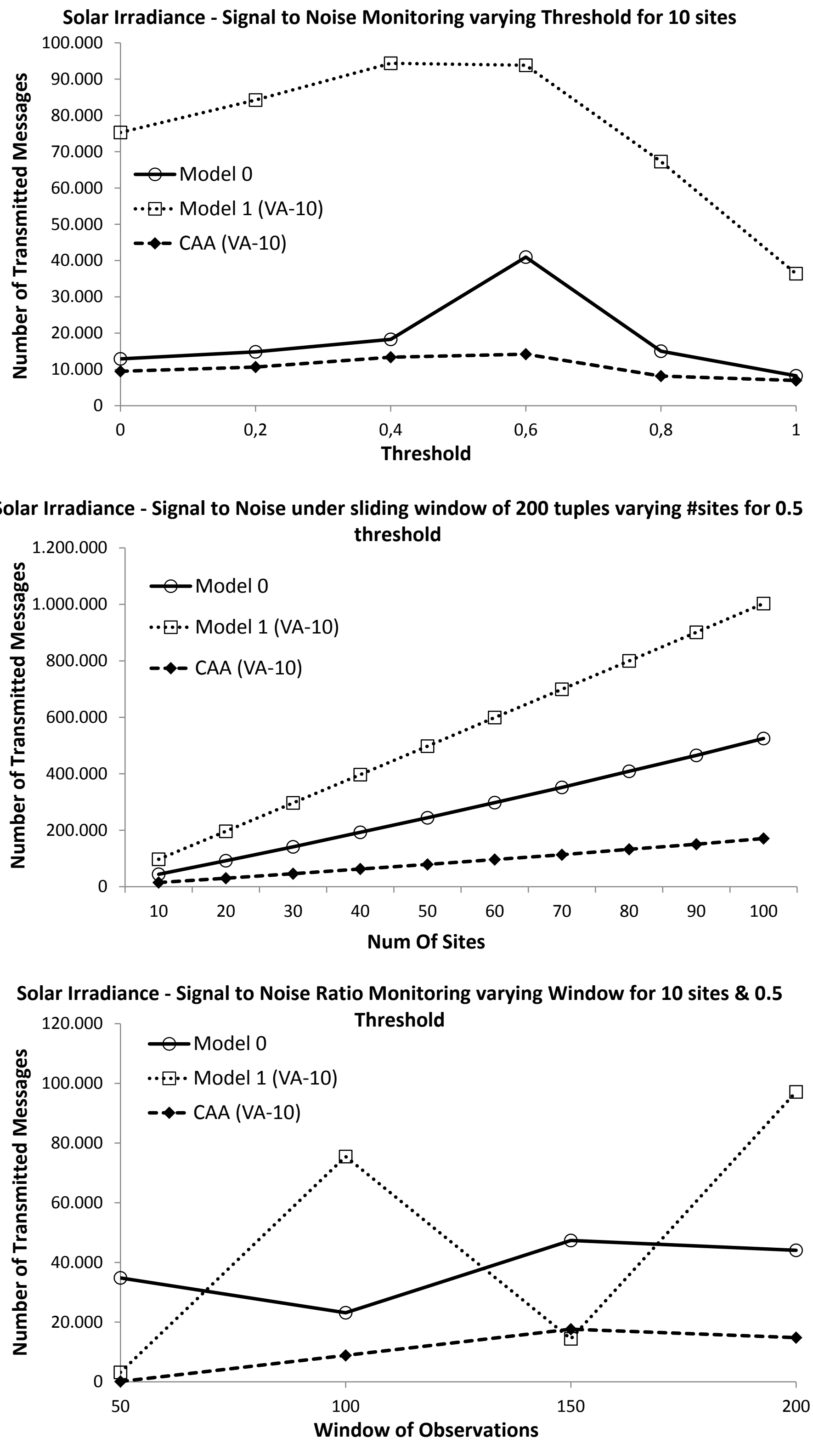
(a) Solar Irradiance, Variance Monitoring

Figure 3.6: Weather Data Set: Performance of our Techniques in the Sliding Window Streaming Paradigm for Variance Monitoring



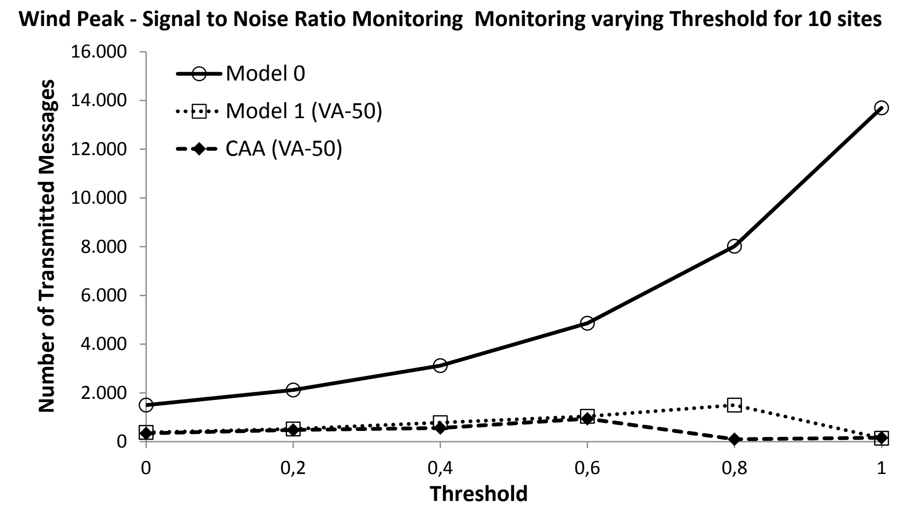
(b) Wind Speed, Variance Monitoring

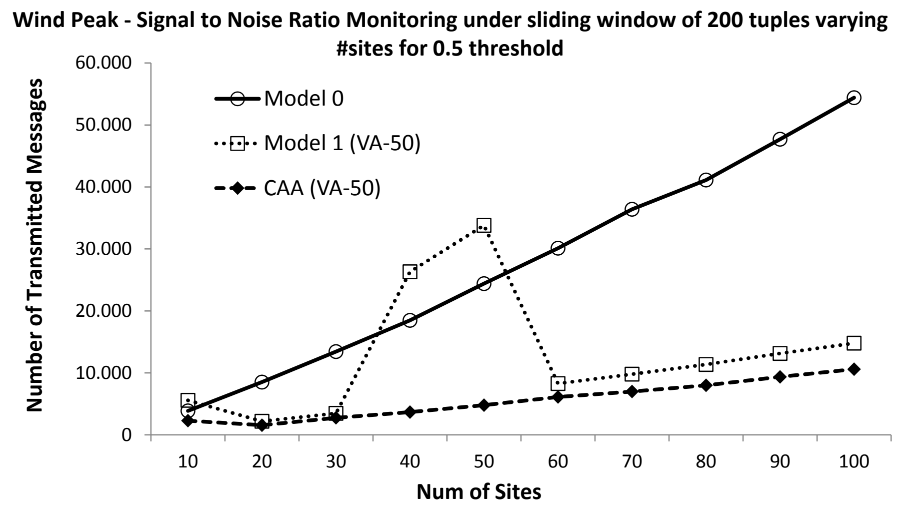
Figure 3.6: Weather Data Set: Performance of our Techniques in the Sliding Window Streaming Paradigm for Variance Monitoring (cont.)

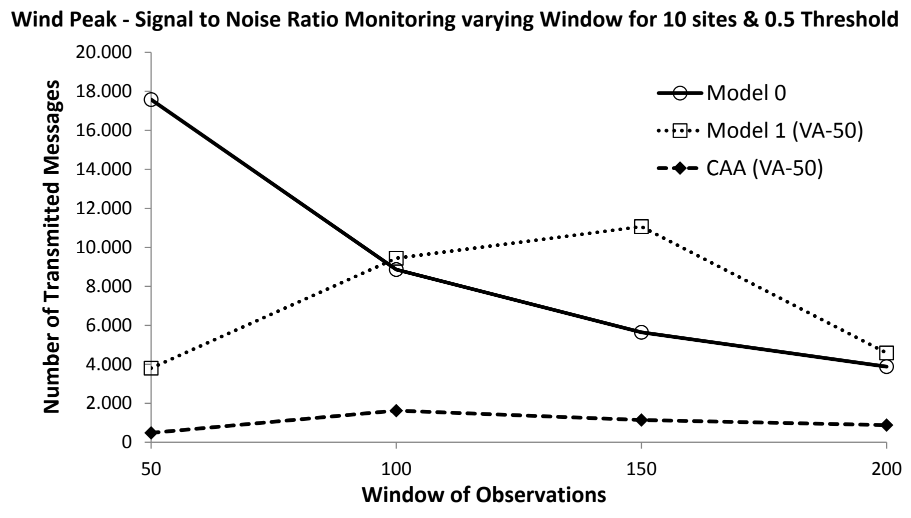


(a) Solar Irradiance, StN Monitoring

Figure 3.7: Weather Data Set: Performance of our Techniques in the Sliding Window Streaming Paradigm for StN Monitoring







(b) Wind Peak, StN Monitoring

Figure 3.7: Weather Data Set: Performance of our Techniques in the Sliding Window Streaming Paradigm for StN Monitoring (cont.)

Chapter 4

A Case Study on Prediction - Based Distributed Monitoring of Representative Trajectories

4.1 Introduction

The extraction of a Representative Trajectory (ReTra) capable of providing a concise summary of the movement of a group of monitored objects, constitutes the main tool of important trajectory querying and mining tasks. More precisely, in trajectory cluster analysis [92, 73] the calculated representative trajectories are utilized as the major descriptors of the *average* movement of the trajectories of a specified cluster [73]. In addition, during privacy aware query answering [91], ReTra is used as the starting point so as to enable the production of fake trajectories that resemble the average movement pattern of the objects participating in the query answer. Hence, those trajectories essentially ensure that the returned query answer does not reveal the actual trajectory of a moving object. Furthermore, they construct an outcome which abides by the expected movement characteristics avoiding uncontrollable distortion of the query answer.

In streaming settings, location updates of moving objects arrive *online* at a coordinating source which is expected to provide analytic results of the monitored movement pattern in *real-time*. In such a scenario, the continuous maintenance of a representative trajectory is capable of providing analysts a compact picture of the average behavior of the motion at any given time interval.

The definition of a ReTra utilized in this chapter is inspired by the respective conceptualization presented in [73]. In particular, [73] defines the representative trajectory as an average direction vector (similar to a rotated centroid). Hence, in our study we are going to compute every time marked 2D-point $(x_{ReTra}(t), y_{ReTra}(t))$ of the

ReTra, as the centroid of the respective 2D-points of the tracked moving objects i.e., $(x_i(t), y_i(t))$ coordinates for the $i-$th object O_i. Notice that the moving objects are usually set to send their location updates on predefined time intervals, thus averaging the spatial information of the trajectories is sufficient (and valid) to attribute the desired ”average” behavior to the final representative.

Based on the above description, the issue of computing a ReTra having collected all location updates at a central server is trivial. It simply involves the computation (average) of a series of two dimensional centroids whose timestamp lies in the monitored time interval. Nonetheless, in a distributed setting where thousands or millions of moving objects constantly transmit their location towards the central source, a major problem arises. Sooner or later the communication links are overloaded and may become non operational. Thus, the execution of the continuous monitoring is stalled and ReTra's delivery in real-time is precluded. Recall that as we pointed out in Chapter 3, bandwidth consumption issues occur in any distributed streaming environment where the major challenge confronted by algorithms dealing with data manipulation is to reduce communication [20, 23, 22, 103, 106, 26, 42]. Nonetheless, in our case where streaming tuples refer to location updates of thousands or millions of moving objects, the amount of distribution is much larger (e.g., compared to the case of networked company servers studied in conventional distributed scenarios) the problem becomes much more intense.

In the current chapter, we aim at confronting the aforementioned challenges. We manage to provide efficient techniques that enable continuous, distributed ReTra monitoring within user-defined accuracy guarantees, reducing the bandwidth consumption charged to the underlying network infrastructure. In our effort to do so, we utilize the notion of predictors for the future locations of the monitored objects developing prediction - based distributed ReTra tracking techniques.

The rest of the chapter is organized as follows. In the next section we describe the architectural infrastructure of our distributed setting, we present basic concepts regarding representative trajectory computation and exhibit how predictors can be installed in this scenario. In Section 4.3 we present two approaches for distributed ReTra monitoring and perform a theoretic comparison with respect to their characteristics.

4.2 Basics

4.2.1 Network Model

We assume a three-tiered network infrastructure as depicted in Figure 4.1. At the bottom tier a number of moving objects are equipped with GPS enabled devices, possessing typical communication and processing capabilities. Monitored objects periodically report their timestamped location updates in the form of $\langle x, y, t \rangle$ triplets to a group of

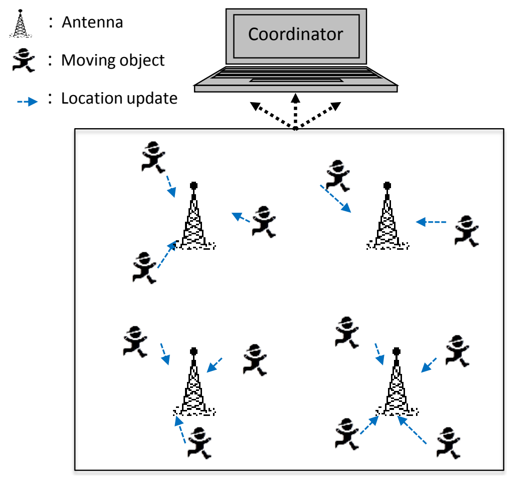


Figure 4.1: Exemplary Network Architecture

geographically dispersed antennas which stand in the middle tier and passively relay the updates to a central source. The top tier includes a central coordinating source which is capable of communicating with the moving objects via the second tier participants. Pairwise site communication is only allowed through the central source.

4.2.2 Representative Trajectory Concepts

In our introductory section we briefly referred to the notion of the representative trajectory that will be utilized throughout our study. In the current subsection, we start by providing a formal definition of a ReTra.

Definition 4.2.1. *A Representative Trajectory (ReTra) over N monitored moving objects within a time window of T size is defined as the time-ordered sequence of pairs:*

$$(x_{ReTra}(t), y_{ReTra}(t)) = (\frac{\sum_{i=1}^{N} x_i(t)}{N}, \frac{\sum_{i=1}^{N} y_i(t)}{N})$$

for any $t \in [t_{now} - T, t_{now}]$, where t_{now} refers to the time of the most recent location update.

The above definition is inspired by the one initially introduced in [73], which establishes a series of rotated centroids produced out of clustered line segments. Since in a streaming scenario we are usually interested for the most recent part of a trajectory, our definition utilizes a time window of T size to express that need. For real time analysis purposes, it is important to observe that apart from a visualization of the series of 2D centroids that compose the ReTra, useful movement characteristics such as average velocity, acceleration, azimuth of directional change etc can be made readily available to analysts.

The challenge that arises upon trying to continuously maintain an accurate ReTra in the setting of Section 4.2.1, relates to the communication cost of having monitored moving objects transmitting each and every location update that participates in $(x_{ReTra}(t),\ y_{ReTra}(t))$. It is not difficult to see that if our objective is to constantly keep an exact ReTra image at the coordinator, the problem is insuperable. Nonetheless, in online querying procedures it is more often than not satisfactory to obtain an approximate (within user specified accuracy bounds) answer to a posed query [84, 21, 22, 20, 23, 25]. This simultaneously provides the primitives for reducing the bandwidth consumption, since transmissions can be suppressed unless the answer - 2D points belonging to ReTra in our scenario - is likely to have exceeded those accuracy guarantees.

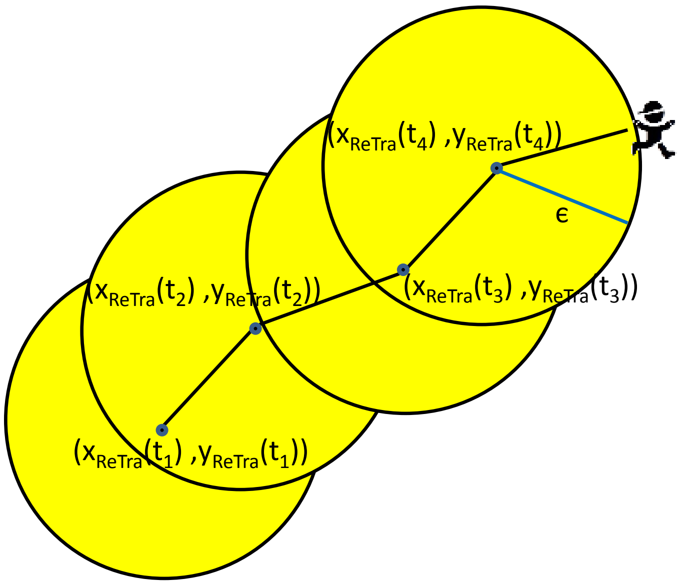

$(x_{ReTra}(t_4), y_{ReTra}(t_4))$
ϵ
$(x_{ReTra}(t_2), y_{ReTra}(t_2))$
$(x_{ReTra}(t_3), y_{ReTra}(t_3))$
$(x_{ReTra}(t_1), y_{ReTra}(t_1))$

Figure 4.2: An $\epsilon-$approximate ReTra representation.

Definition 4.2.2. *An estimation of a ReTra constitutes an $\epsilon-$approximation of the true representative trajectory, if its estimated $(\widehat{x_{ReTra}}(t), \widehat{y_{ReTra}}(t))$ points adhere to the following condition:*

$$\|(\widehat{x_{ReTra}}(t), \widehat{y_{ReTra}}(t)) - (x_{ReTra}(t), y_{ReTra}(t))\| \leq \epsilon$$

for a constant $\epsilon > 0$, at any given timestamp $t \in [t_{now} - T, t_{now}]$.

An example of an $\epsilon-$approximate representative trajectory represenation is depicted in Figure 4.2. Therefore, in what follows we concentrate on monitoring an

$\epsilon-$approximation of the true ReTra at the coordinator. Our methods base their efficiency in avoiding site to coordinator contact unless the quality of the ReTra sustained at the central source may have been reduced beyond the given threshold ϵ.

4.2.3 ReTra's Query Processing

At the beginning of the monitoring process, a query of the following type [89] is registered at the coordinator:

```
SELECT ReTra(m.mpoint, $\epsilon$)
FROM mpoints AS m
{WHERE Overlaps3D(m.mpoint,$\langle AreaOfInterest, T\rangle$)=$true$}
{USING [prediction_model] WITH [Parameters]}
SAMPLE PERIOD $\tau$;
```

The above query is continuously processed until it is explicitly terminated by the users. The coordinating source is responsible to disseminate the specifications of the representative trajectory tracking to the monitored objects. The virtual table $mpoints$ stores the information of object trajectories, while the $Overlaps3D$ operator essentially performs a spatiotemporal range query that restricts those trajectories a) temporally - keeping the parts that lie in the desired time window T, b) spatially - filtering parts placed out of the area we wish to focus on.

Note that the WHERE compartment of the query is placed in brackets which are used to show that the whole compartment or its counterparts are optional. In case T is omitted or set to $T = 0$, the query answer that will be provided every τ time units will include information only about t_{now}. On the other hand, leaving the $AreaOfInterest$ field empty will cause all reachable (from the coordinator) moving objects to participate in ReTra's calculation.

The USING compartment is also optional and refers to the chosen prediction model adoption along with its parameters. We are going to discuss the details regarding this parameter in the next section. For now, we mention that the desired prediction model will be used in order to derive $(\widehat{x_{ReTra}(t)}, \widehat{y_{ReTra}(t)})$ while performing the $\epsilon-$approximate ReTra tracking. Omitting this compartment is equivalent to the static predictor adoption which was presented in Section 3.2.2 and Table 3.2.

Eventually, the ReTra calculation in the SELECT compartment is derived according to Definition 4.2.1 and 4.2.2, where omitting the tolerance parameter ϵ accounts for continuous central data collection as discussed in the previous sections.

4.2.4 Incorporating Location Predictors

In the current section we refer to the utility of predictors that may be utilized so as to derive the corresponding $\epsilon-$ approximation during distributed ReTra tracking. The talk regards the estimations $(\widehat{x_{ReTra}}(t),\, \widehat{y_{ReTra}}(t))$ moving objects need to continuously possess in order to ensure the $\epsilon-$ proximity according to Definition 4.2.2.

Initially, notice that the static e.g. in case of objects unpredictably moving in the plane, as well as the velocity acceleration predictors that we presented in Section 3.2.2 of Chapter 3 (also see Table 3.2) can be utilized so as to estimate the future location of a monitored object. In our current scenario the monitored vectors are two dimensional, composed of $(x_i(t), y_i(t))$ coordinates of a monitored object O_i.

The global estimation for a representative trajectory point is computed by

$$(x^p_{ReTra}(t), y^p_{ReTra}(t)) = (\frac{\sum_{i=1}^{N} x^p_i(t)}{N}, \frac{\sum_{i=1}^{N} y^p_i(t)}{N})$$

Hence, in what follows we essentially consider $(\widehat{x_{ReTra}}(t),\, \widehat{y_{ReTra}}(t)) = (x^p_{ReTra}(t),$ $y^p_{ReTra}(t))$. And correspondingly for local estimations: $\hat{x_i}(t) = x^p_i(t)$, $\hat{y_i}(t) = y^p_i(t)$. The latter constitutes the $\epsilon-$approximation of any representative trajectory point within the chosen window of T size. This enables the coordinator to continuously provide corresponding answers to the query of Section 4.2.3 at least until the next synchronization upon which new predictions about the global vectors will be derived.

In order to efficiently process the tracking process and ward off unnecessary contacts with the coordinating source, again our ultimate goal is to come up with constraints that monitored objects may locally check before deciding to transmit their location updates. As long as those constraints are not violated at all objects, we can guarantee that the current ReTra constitutes an $\epsilon-$ approximation of the true one. Of course, in case the imposed constraints are violated in at least one object, it certainly needs to contact the coordinating source.

4.3 Distributed ReTra Monitoring

4.3.1 ReTra Monitoring by Decomposition to Local Constraints

We now proceed to the core of the distributed ReTra monitoring algorithm. In order to achieve the $\epsilon-$approximation target, we are going to decompose the problem of distributively ensuring $\|(x^p_{ReTra}(t),\, y^p_{ReTra}(t)) - (x_{ReTra}(t),\, y_{ReTra}(t))\| \leq \epsilon$ of Definition 4.2.2, to local constraints that can be consulted by moving objects so as to decide whether a synchronization process needs to take place. The upcoming lemma elaborates on the later issue introducing an appropriate sufficient condition.

Lemma 4.3.1. *When the local constraint*

$$\|(x^p_i(t), y^p_i(t)) - (x_i(t), y_i(t))\| \leq \epsilon$$

holds for any monitored object O_i at any $t \in [t_{now} - T, t_{now}]$, then

$$\|(x^p_{ReTra}(t), y^p_{ReTra}(t)) - (x_{ReTra}(t), y_{ReTra}(t))\| \leq \epsilon$$

i.e., the estimation of the ReTra that is kept at the coordinating source, is always an $\epsilon-$approximation of the true representative trajectory.

Proof. Starting by the local constraint of a single monitored object O_i we obtain:

$$\|(x^p_i(t), y^p_i(t)) - (x_i(t), y_i(t))\| \leq \epsilon$$

Summing for N objects, we have

$$\sum_{i=1}^{N} \|(x^p_i(t), y^p_i(t)) - (x_i(t), y_i(t))\| \leq N \cdot \epsilon \overset{\substack{triangle\\inequality}}{\Rightarrow}$$

$$\|(\sum_{i=1}^{N} x^p_i(t), \sum_{i=1}^{N} y^p_i(t)) - (\sum_{i=1}^{N} x_i(t), \sum_{i=1}^{N} y_i(t))\| \leq N \cdot \epsilon \Rightarrow$$
$$\|(\frac{\sum_{i=1}^{N} x^p_i(t)}{N}, \frac{\sum_{i=1}^{N} y^p_i(t)}{N}) - (\frac{\sum_{i=1}^{N} x_i(t)}{N}, \frac{\sum_{i=1}^{N} y_i(t)}{N})\| \leq \epsilon \Rightarrow$$
$$\|(x^p_{ReTra}(t), y^p_{ReTra}(t)) - (x_{ReTra}(t), y_{ReTra}(t))\| \leq \epsilon$$

which concludes the proof. □

As long as Lemma 4.3.1 holds, no transmission is required. The monitored objects keep up acquiring updates of their location and locally maintain the required information for predictors' computation during the next synchronization as described in Section 3.2.2. In case $\|(x^p_i(t), y^p_i(t)) - (x_i(t), y_i(t))\| \leq \epsilon$ is not satisfied in at least one site central data collection is taking place, after which the tracking process is continued.

4.3.2 ReTra Monitoring Using the Prediction-Based Geometric Approach

In order to apply the prediction-based geometric monitoring framework (Chapter 3) in the representative trajectory monitoring case, we first need to identify the basic elements of the tracking process. In what follows we re-employ the notation used in Chapter 3 and was summarized in Table 3.1. We start by studying the function of interest according to Definition 4.2.2.

$$\begin{aligned} f(v(t)) &= \|(x^p_{ReTra}(t), y^p_{ReTra}(t)) - (x_{ReTra}(t), y_{ReTra}(t))\| \\ &= \sqrt{(x^p_{ReTra} - x_{ReTra})^2 + (y^p_{ReTra} - y_{ReTra})^2} \end{aligned}$$

and the specified threshold is ϵ. Furthermore, the true global vector at any given timenstamp is composed of $v(t) = (\frac{\sum_{i=1}^{N} x_i(t)}{N}, \frac{\sum_{i=1}^{N} y_i(t)}{N})$. Note that $(x^p_{ReTra}(t), y^p_{ReTra}(t))$ is made known to all the moving objects after each synchronization process and thus it can be treated as a constant in the monitored function. We observe that the global vector $v(t)$ is indeed the average (and thus a convex combination) of the vectors $v_i(t) = (x_i(t), y_i(t))$ possessed by individual moving objects and thus the geometric approach is applicable. As regards the rest of the elements of the monitoring process, the predicted estimate $e^p(t) = (\frac{\sum_{i=1}^{N} x_i^p(t)}{N}, \frac{\sum_{i=1}^{N} y_i^p(t)}{N})$, while $e = (\frac{\sum_{i=1}^{N} x_i(t_s)}{N}, \frac{\sum_{i=1}^{N} y_i(t_s)}{N})$. Eventually, local predictors are expressed by $v_i(t) = (x_i^p(t), y_i^p(t))$ as well.

We finally briefly outlying the steps of the monitoring process since the details have already been provided while presenting our prediction - based geometric monitoring framework in Chapter 3. Again assume that the coordinator has collected all the location updates of monitored objects at a previous timestep t_s. Then, the central source extracts e and e^p which are broadcasted to all moving objects. Having received the estimate as well as the predicted estimate vectors, every object O_i individually computes the drift vector $u_i(t) = (\frac{\sum_{i=1}^{N} x_i(t_s)}{N}, \frac{\sum_{i=1}^{N} y_i(t_s)}{N}) + (x_i(t), y_i(t)) - (x_i(t_s), y_i(t_s))$ and the prediction deviation vector $u_i^p(t) = (\frac{\sum_{i=1}^{N} x_i^p(t)}{N}, \frac{\sum_{i=1}^{N} y_i^p(t)}{N}) + (x_i(t), y_i(t))$ $- (x_i^p(t), y_i^p(t))$. The tracking of the resulted convex hull is performed according to the techniques devised in Chapter 3 employing the CAA strategy we described in Section 3.5.5.

4.3.3 Comparison of the Approaches

We are now concerned with identifying the characteristics of the two devised approaches and decide which of the two is more preferable for a ReTra tracking application. For ease of exposition, we start by assuming a static predictor choice i.e., $x_i^p(t) = y_i(t_s)$ and $y_i^p(t) = y_i(t_s)$.

Notice that both the approach of Section 4.3.1 and that of Section 4.3.2 are geometric in nature. As regards the first, the decomposition to local constraints resulted in having moving objects essentially consult (Lemma 4.3.1) disks centered at $(x_i(t_s), y_i(t_s))$ with a radius of ϵ size (Figure 4.3). On the other hand, the approach of Section 4.3.2 devises that moving objects check if the disk centered at $\frac{e+u_i(t)}{2}$ crosses the threshold surface. Nonetheless, it worths focusing on the latter approach (that of prediction - based geometric monitoring) and study the shape of the threshold surface in more detail.

In this particular case, the region where a threshold crossing can occur is beyond ϵ distance of the $e = (\frac{\sum_{i=1}^{N} x_i(t_s)}{N}, \frac{\sum_{i=1}^{N} y_i(t_s)}{N})$ vector. As a consequence, that region where the monitored convex hull $Conv\ (u_1(t), \ldots, u_n(t))$ is allowed to lie is again a disk (depicted in Figure 4.4) centered at e with radius ϵ, B_e^{ϵ}, which is convex. But if the region where $Conv\ (u_1(t), \ldots, u_N(t))$ may lie without causing a threshold crossing

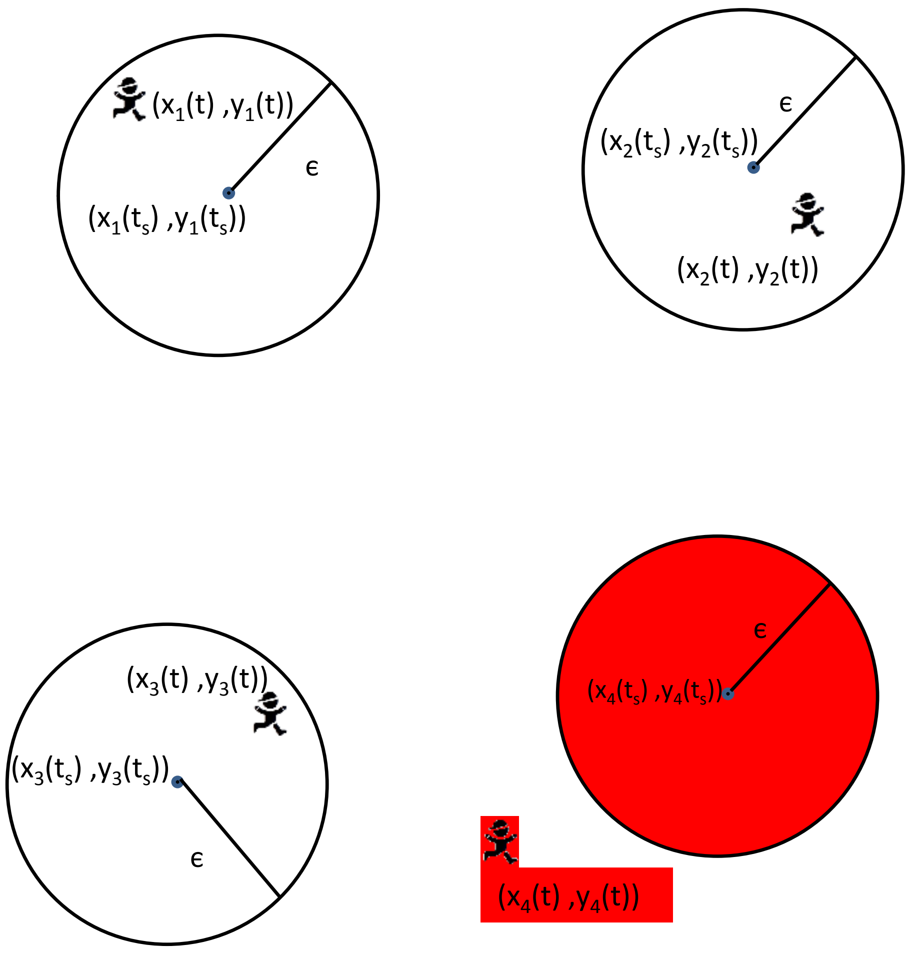


Figure 4.3: Rationale of the ReTra Monitoring by Decomposition to Local Constraints. Moving Object O_4 moves outside its local constraint(disk) and thus causes central data collection.

is convex, we do not actually need $\overset{N}{\underset{i=1}{\cup}} B_{\frac{e+u_i(t)}{2}}^{\frac{e-u_i(t)}{2}}$ to monitor it. By convexity of B_e^ϵ, the convex hull of interest cannot cause threshold crossing without having at least one of its vertices $(u_1(t), \ldots, u_N(t))$ doing so. Hence, it suffices for every moving object to check if $e + (x_i(t), y_i(t)) - (x_i(t_s), y_i(t_s))$ is included in B_e^ϵ. Overall, we again check a circular constraint of length ϵ which is violated only when $\|(x_i(t), y_i(t)) - (x_i(t_s), y_i(t_s))\| > \epsilon$.

This analysis is valid even in the more general case of choosing any other (but the same), instead of the static, predictor in both approaches. Moreover, our previous discussion renders the presented approaches equivalent in terms of the size of the consulted local constraints. Note that as explained in Section 3.3, the size of the

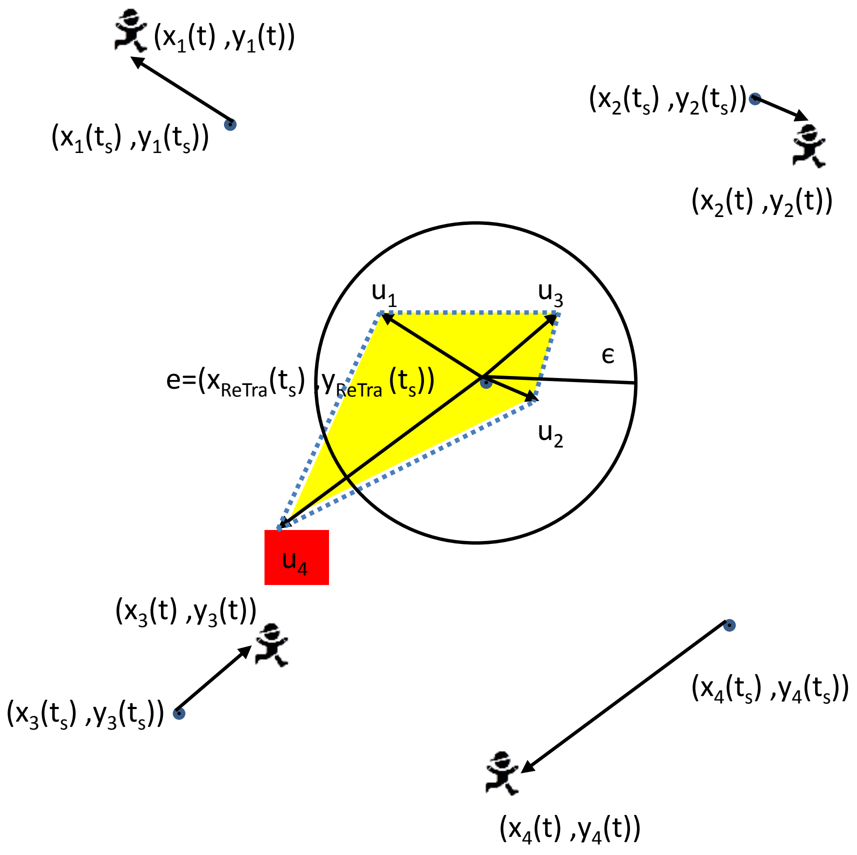


Figure 4.4: Rationale of the ReTra Monitoring Utilizing the Prediction - Based Geometric Approach. Vertex u_4 corresponding to the moving object O_4 moves outside B_e^ϵ causing a synchronization process.

constructed constraints is an important factor of the communication cost of a given monitoring scheme. However, we also noted that the choice of the reference point is the other major criterion in placing the tracked convex hull farther or nearer to the threshold surface. As a result, the CAA approach presented in Section 3.5.5 provides additional flexibility to the distributed tracking process as it is capable of tuning both the aforementioned couple of parameters depending on the recent performance of the corresponding alternative tracking mechanisms.

4.4 Synopsis

In the current chapter we presented two alternative approaches for efficiently performing ϵ-approximate, distributed ReTra monitoring. Since the monitored function of this special case is relatively easy to handle, in Section 4.3.1 we managed to decompose the tracking of the non linear function $\|(x^p_{ReTra}(t), y^p_{ReTra}(t)) - (x_{ReTra}(t), y_{ReTra}(t))\|$ to a set of simple local constraints, while in Section 4.3.2 we utilized the framework of [42]. Furthermore, we developed a theoretic analysis exhibiting the equivalence of the two approaches as well as the flexibility the CAA strategy provides upon utilizing the prediction - based geometric monitoring framework. We thus presented a specialization of the concepts of Chapter 3 to the distributed, representative trajectory monitoring over streaming spatiotemporal data streams.

Part II

Mining Distributed Data Streams

Chapter 5

Tunable Approximate Computation of Outliers In Wireless Sensor Networks

5.1 Introduction

Pervasive applications are increasingly supported by networked sensory devices that interact with people and themselves in order to provide the desired services and functionality. Because of the unattended nature of many applications and the inexpensive hardware used in the construction of the sensors, sensor nodes often generate imprecise individual readings due to interference or failures [56]. Sensors are also often exposed to severe conditions that adversely affect their sensing elements, thus yielding readings of low quality. For example, the humidity sensor on the popular MICA mote is very sensitive to rain drops [30].

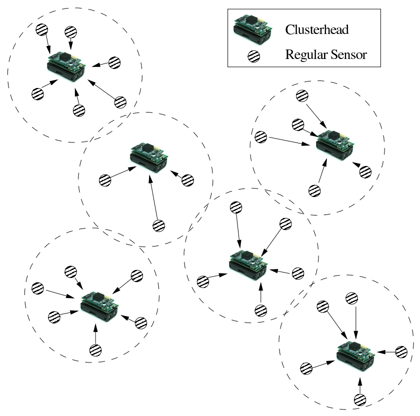
The development of a flexible layer that will be able to detect and flag outlier readings, so that proper actions can be taken, constitutes a challenging task. Conventional outlier detection algorithms [5, 39] are not suited for our distributed, resource-constrained environment of study. First, due to the limited memory capabilities of sensor nodes, in most sensor network applications, data is continuously collected by motes and maintained in memory for a limited amount of time. Moreover, due to the frequent change of the data distribution, results need to be generated continuously and computed based on recently collected measurements. Furthermore, a central collection of sensor data is not feasible nor desired, since it results in high energy drain, due to the large amounts of transmitted data. Hence, what is required are continuous, distributed and in-network approaches that reduce the communication cost and manage to prolong the network lifetime.

One can provide several definitions of what constitutes an outlier, depending on the application. For example in [111], an outlier is defined as an observation that is sufficiently far from most other observations in the data set. However, such a definition is inappropriate for physical measurements (like noise or temperature) whose absolute values depend on the distance of the sensor from the source of the event that triggers the measurements. Moreover, in many applications, one cannot reliably infer whether a reading should be classified as an outlier without considering the recent history of values obtained by the nodes. Thus, in our framework we propose a more general method that detects outlier readings taking into account the recent measurements of a node, as well as spatial correlations with measurements of other nodes.

Similar to recent proposals for processing declarative queries in wireless sensor networks, our techniques employ an *in-network processing* paradigm that fuses individual sensor readings as they are transmitted towards a *base station*. This fusion, dramatically reduces the communication cost, often by orders of magnitude, resulting in prolonged network lifetime. While such an in-network paradigm is also used in proposed methods that address the issue of data cleaning of sensor readings by identifying and, possibly, removing outliers [14, 30, 56, 113], none of these existing techniques provides a straightforward mechanism for controlling the burden of the nodes that are assigned to the task of outlier detection.

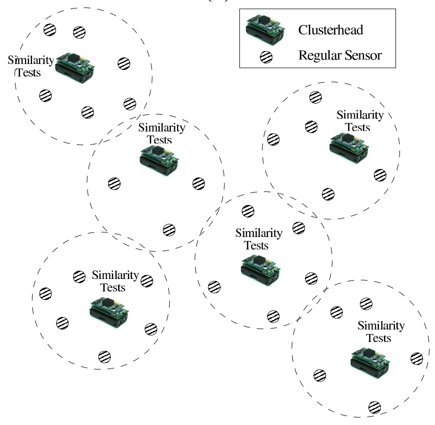
An important observation that we make in this chapter [44, 45] is that existing in-network processing techniques cannot reduce the volume of data transmitted in the network to a satisfactory level and lack the ability of tuning the resulting overhead according to the application needs and the accuracy levels required for outlier detection. Please note that it is desirable to reduce the amount of transmitted data in order to also significantly reduce the energy drain of sensor nodes. This occurs not only because radio operation is by far the biggest culprit in energy drain [78], but also because fewer data transmissions also result in fewer collisions and, thus, fewer re-transmissions by the sensor nodes.

In this chapter we present a novel outlier detection scheme termed TACO (TACO stands for Tunable Approximate Computation of Outliers). TACO [44] adopts two levels of hashing mechanisms. The first is based on locality sensitive hashing (LSH) [12], which is a powerful method for dimensionality reduction [12, 53, 54]. We first utilize LSH in order to encode the latest W measurements collected by each sensor node as a bitmap of $d \ll W$ bits. This encoding is performed locally at each node. The encoding that we utilize trades accuracy (i.e., probability of correctly determining whether a node is an outlier or not) for bandwidth, by simply varying the desired level of dimensionality reduction and provides tunable accuracy guarantees based on the d parameter mentioned above. Assuming a clustered network organization [13, 50, 96, 122], motes communicate their bitmaps to their clusterhead, which can estimate the similarity amongst the latest values of any pair of sensors within its cluster by comparing their



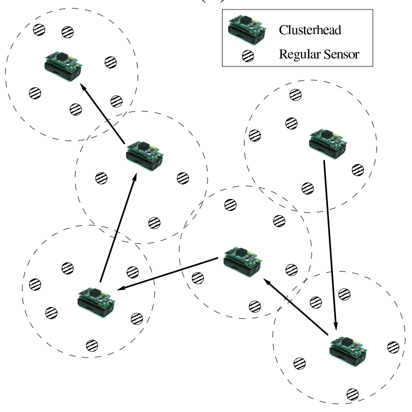
Each sensor node uses LSH to encode its latest W measurements using just d bits. The node transmits the compressed d–bit representation to the clusterhead.

(a)



Each clusterhead performs similarity tests on the received meaurements. Nodes that do not gain enough support are considered as potential outliers.

(b)



An approximate TSP problem is solved. Lists of potential outliers are exchanged (along with their compressed representations and current support). The final list is transmitted to the base station.

(c)

Figure 5.1: Main Stages of the TACO Framework

bitmaps, and for a variety of similarity metrics, which essentially constitute the monitored functions of interest, that are useful for the applications we consider. Based on the performed similarity tests, and a desired minimum support specified by the posed query, each clusterhead generates a list of *potential* outlier nodes within its cluster. At a second (inter-cluster) phase of the algorithm, this list is then communicated among the clusterheads, in order to allow potential outliers to gain support from measurements of nodes that lie within other clusters. This process is sketched in Figure 5.1.

The second level of hashing (omitted in Figure 5.1) adopted in TACO's framework comes during the intra-cluster communication phase. It is based on the hamming weight of sensor bitmaps and provides a pruning technique (regarding the number of performed bitmap comparisons) and a load balancing mechanism alleviating clusterheads from communication and processing overload. We choose to discuss this load balancing and comparison pruning mechanism separately, for ease of exposition, as well as to better exhibit its benefits.

This chapter proceeds as follows. Initially, Section 5.2 introduces our basic framework. In Sections 5.3 and 5.4 we analyze TACO's operation in detail. Our load balancing and comparison pruning mechanisms are described in Section 5.5, while Section 5.6 demonstrates how a variety of similarity measures can be utilized by TACO. In Section 5.7 we elaborate on interesting extensions to TACO that are capable of further reducing the communication cost. Section 5.8 presents our experimental evaluation.

5.2 Basic Framework

5.2.1 Outlier Definition

As in [30], we do not aim to compute outliers based on a mote's latest reading but, instead, take into consideration its most recent measurements. In particular let u_i denote the latest W readings obtained by node S_i. Then, given a similarity metric $sim{:}R^W \rightarrow [0, 1]$ and a similarity threshold Φ we consider the readings by motes S_i and S_j similar if

$$sim(u_i, u_j) > \Phi. \tag{5.1}$$

In TACO, we classify a mote as an outlier if its latest W measurements are not found to be similar with the corresponding measurements of at least *minSup* other motes in the network. The parameter *minSup*, thus, dictates the minimum support (either in the form of an absolute, uniform value or as a percentage of motes, i.e per cluster) that the readings of the mote need to obtain by other motes in the network, using Equation 5.1. Consequently, as in Chapter 3, the chosen similarity measure $sim(u_i, u_j)$ constitutes the monitored function of interest given the Φ threshold. The difference is the additional introduction of the minimum support parameter. The de-

Similarity Metric	Calculation of Similarity
Cosine Similarity	$cos(\theta(u_i, u_j)) = \frac{u_i \cdot u_j}{\|\|u_i\|\| \cdot \|\|u_j\|\|} \Rightarrow$ $\theta(u_i, u_j) = arccos \frac{u_i \cdot u_j}{\|\|u_i\|\| \cdot \|\|u_j\|\|}$
Correlation Coefficient	$corr(u_i, u_j) = \frac{cov(u_i, u_j)}{\sigma_{u_i} \sigma_{u_j}} =$ $\frac{E(u_i u_j) - E(u_i)E(u_j)}{\sqrt{E(u_i^2) - E^2(u_i)} \sqrt{E(u_j^2) - E^2(u_j)}} =$ $\frac{\sum_{\ell=1}^{W} (u_{i\ell} - E(u_i)) \cdot (u_{j\ell} - E(u_j))}{\sqrt{\sum_{\ell=1}^{W} (u_{i\ell} - E(u_i))^2} \cdot \sqrt{\sum_{\ell=1}^{W} (u_{j\ell} - E(u_j))^2}}$
Jaccard Coefficient	$J(u_i, u_j) = \frac{\|u_i \cap u_j\|}{\|u_i \cup u_j\|}$
Extended Jaccard Coefficient	$T(u_i, u_j) = \frac{u_i \cdot u_j}{\|\|u_i\|\|^2 + \|\|u_j\|\|^2 - u_i \cdot u_j}$
Euclidean Distance	$dist(u_i, u_j) = \sqrt{\sum_{\ell=1}^{W} (u_{i\ell} - u_{j\ell})^2}$

Table 5.1: Computation of some supported similarity metrics between vectors u_i, u_j containing the latest W measurements of nodes S_i and S_j.

mand for *minSup* makes the geometric approach fail in controlling accuracy when message losses or addition as well as death of sensor nodes takes place [9]. On the contrary, given the above definition, the framework we are about to present manages to control accuracy and provide a direct way to link it with the desired bandwidth conservation.

By allowing the user/application to control the value of *minSup*, our techniques are resilient to environments where spurious readings originate from multiple nodes at the same epoch, due to a multitude of different, and hence unpredictable, reasons. Our framework can also easily incorporate additional *witness criteria* based on non-dynamic grouping characteristics (such as the node identifier or its location), in order to limit, for each sensor, the set of nodes that are tested for similarity with it. For example, one may not want sensor nodes located in different floors to be able to witness each other's measurements.

5.2.2 Supported Similarity Metrics - Monitored Functions

The definition of an outlier, as presented in Section 5.2.1, is quite general to accommodate a number of intuitive similarity tests between the latest W readings of a pair of sensor nodes S_i and S_j. Examples of such monitored functions of interest, include the *cosine similarity*, the *correlation coefficient* and the *Jaccard coefficient* [12, 30]. The *Euclidean distance* and the *extended Jaccard coefficient* of standardized value vectors are supported as well. Table 5.1 demonstrates the formulas for computing the aforementioned metrics over the two vectors u_i, u_j containing the latest W readings of sensors S_i and S_j, respectively[1].

It is important to emphasize that our framework is not limited to using just one

[1]$E(.)$, σ and $cov(.)$ in the table stand for mean, standard deviation and covariance, respectively.

of the metrics presented in Table 5.1. On the contrary, as it will be explained in Section 5.3.1, any similarity metric satisfying a set of common criteria may be incorporated in our framework.

5.2.3 Network Organization

We adopt an underlying network structure where motes are organized into clusters (shown as dotted circles in Figure 5.1). Queries are propagated by the base station to the clusterheads, which, in turn, disseminate these queries to sensors within their cluster.

Various algorithms [13, 50, 96, 122] have been proposed to clarify the details of cluster formation, as well as the clusterhead election and substitution (rotation) during the lifetime of the network. All these approaches have been shown to be efficient in terms of energy dissipation, thus resulting in prolonged network lifetime. The aforementioned algorithms differ in the way clusters and corresponding clusterheads are determined, though they all share common characteristics since they primarily base their decisions on the residual energy of the sensor nodes and their communication links.

An important aspect of our framework is that the choice of the clustering algorithm is orthogonal to our approach. Thus, any of the aforementioned algorithms can be incorporated in our framework. An additional advantage of our techniques is that they require no prior state at clusterhead nodes, thus simplifying the processes of clusterhead rotation and re-election.

5.2.4 Operation of the Algorithm

We now outline the various steps involved in our TACO framework. These steps are depicted in Figure 5.1.

Step 1: Data Encoding and Reduction. At a first step, the sensor nodes encode their latest W measurements using a bitmap of d bits. In order to understand the operation of our framework, the actual details of this encoding are not important (they are presented in Section 5.3). What is important is that:

- As we will demonstrate, the similarity function between the measurements of any pair of sensor nodes can be evaluated using their encoded values, rather than using their uncompressed readings.
- The used encoding trades accuracy (i.e., probability of correctly determining whether a node is an outlier or not) for bandwidth, by simply varying the desired level of dimensionality reduction (i.e., parameter d mentioned above). Larger values of d result in increased probability that similarity tests performed on the

encoded representation will reach the same decision as an alternative technique that would have used the uncompressed measurements instead.

After encoding its measurements, each sensor node transmits its encoded measurements to its clusterhead.

Step 2: Outlier Detection at the Cluster Level. Each clusterhead receives the encoded measurements of the sensors within its cluster. It then performs similarity tests amongst all pairs of sensor nodes that may witness each other (please note that the posed query may have imposed restrictions on this issue), in order to determine nodes that cannot reach the desired support level and are, thus, considered to be outliers at a cluster level.

Step 3: Intercluster Communication. After processing the encoded measurements within its cluster, each clusterhead has determined a set of potential outliers, along with the support that it has computed for each of them. Some of these potential outliers may be able to receive support from sensor nodes belonging to other clusters. Thus, a communication phase is initiated where the potential outliers of each clusterhead are communicated (along with their current support) to other clusterheads in which their support may increase. Please note that depending on the restrictions of the posed queries, only a subset of the clusterheads may need to be reached. The communication problem is essentially modeled as a TSP problem, where the origin is the clusterhead itself, and the destination is the base station.

The extensible definition of an outlier in our framework enables the easy application of semantic constraints on the definition of outliers. For example, we may want to specify that only movement sensors trained on the *same location* are allowed to witness each other, or similarly that only readings from vibration sensors attached to identical engines in a machine room are comparable. Such static restrictions can be easily incorporated in our framework (i.e., by having clusterheads maintain the corresponding information, such as location and type, for each sensor id) and their evaluation is orthogonal to the techniques that we present in this chapter.

5.3 Data Encoding and Reduction

In this section we provide the definition and properties of the Locality Sensitive Hashing schemes. We further investigate the details of a particular, the Random Hyperplain Projection, LSH scheme which serves as the basis for incorporating in TACO most of the similarity measures presented in Table 5.1 (the details of the latter issue are included in the current section as well as Section 5.6).

Symbol	Description
S_i	the $i - th$ sensor node
u_i	the value vector of node S_i
W	tumble size (length of u_i)
$\theta(u_i, u_j)$	the angle between vectors u_i, u_j
X_i	the bitmap encoding produced after applying LSH to u_i
d	bitmap length
$D_h(X_i, X_j)$	the hamming distance between bitmaps X_i, X_j
Φ	similarity threshold used
Φ_θ	threshold based on angle similarity
Φ_{D_h}	threshold based on hamming distance similarity
minSup	the minimum support parameter

Table 5.2: Notation of Chapter 5

5.3.1 Definition and Properties of LSH

A *Locality Sensitive Hashing scheme* is defined in [12] as a distribution on a family F of hash functions that operate on a set of objects, such that for two objects u_i, u_j:

$$P_{h \epsilon F}[h(u_i) = h(u_j)] = sim(u_i, u_j)$$

where $sim(u_i, u_j) \epsilon [0, 1]$ is some similarity measure. In [12] the following necessary properties for existence of an LSH family function for given similarity measures are proved:

Lemma 5.3.1. *For any similarity function $sim(u_i, u_j)$ that admits an LSH function family, the distance $1 - sim(u_i, u_j)$ satisfies the triangle inequality.*

Lemma 5.3.2. *Given an LSH function family F corresponding to a similarity function $sim(u_i, u_j)$, we can obtain an LSH function family F' that maps objects to $\{0, 1\}$ and corresponds to the similarity function $\frac{1+sim(u_i,u_j)}{2}$.*

Lemma 5.3.3. *For any similarity function $sim(u_i, u_j)$ that admits an LSH function family, the distance $1 - sim(u_i, u_j)$ is isometrically embeddable in the hamming cube.*

As a result, the above lemmas simultaneously summarize the conditions any candidate similarity measure should satisfy so as to be incorporated in our outlier detection framework.

5.3.2 Data Reduction at the Sensor Level

In our setting, TACO applies LSH to the value vectors of physical quantities sampled by motes. It can be easily deduced that LSH schemes have the property of dimensionality reduction while preserving similarity between these vectors. Dimensionality reduction can be achieved by introducing a hash function family such that (Lemmas 5.3.2,5.3.3) for any vector $u_i \epsilon R^W$ consisting of W sampled quantities, $h(u_i) : R^W \rightarrow [0,1]^d$ with $d \ll W$.

In what follows we first describe an LSH scheme for estimating the cosine similarity between motes (please refer to Table 5.1 for the definition of the cosine similarity metric).

Theorem 5.3.1. *[Random Hyperplane Projection (RHP) [12, 47]]*
Assume we are given a collection of vectors defined on the W dimensional space. We choose a family of hash functions as follows: We produce a spherically symmetric random vector r of unit length from this W dimensional space. We define a hash function h_r as:

$$h_r(u_i) = \begin{cases} 1 & ,\textit{if}\, r \cdot u_i \geq 0 \\ 0 & ,\textit{if}\, r \cdot u_i < 0 \end{cases}$$

For any two vectors $u_i, u_j \epsilon R^W$:

$$P = P[h_r(u_i) = h_r(u_j)] = 1 - \frac{\theta(u_i, u_j)}{\pi} \square \tag{5.2}$$

Equation 5.2 can be rewritten as:

$$\theta(u_i, u_j) = \pi \cdot (1 - P) \tag{5.3}$$

Note that Equation 5.3 expresses angle similarity as the product of the potential range of the angle between the two vectors (π), with the probability of equality in the result of the hash function application (P). Thus, after repeating a stochastic procedure using d random vectors r, the final embodiment in the hamming cube results in [114]:

$$D_h(X_i, X_j) = d \cdot (1 - P) \tag{5.4}$$

where $X_i, X_j \epsilon [0,1]^d$ are the bitmaps (of length d) produced and $D_h(X_i, X_j) = \sum_{\ell=1}^{d} |X_{i\ell} - X_{j\ell}|$ is their hamming distance. Hence, we finally derive:

$$\frac{\theta(u_i, u_j)}{\pi} = \frac{D_h(X_i, X_j)}{d} \tag{5.5}$$

Equation 5.5 provides the means to compute the angle (and thus the cosine similarity) between the initial value vectors based on the hamming distance of their corresponding bitmaps. The exact details of bitmap comparison with respect to outlier detection will be provided in the next section.

Given two vectors u_i, u_j on the R^W space, and a desired similarity metric, one can deterministically compute the similarity of the two vectors. However, the transition from the continuous (R^W) space to the hamming cube ($[0,1]^d$) results in imprecision, since the estimation of the angle similarity depends on the selection of the spherically symmetric random vectors. We now determine the number of bits required in each encoding so that the resulting scheme achieves an $(\epsilon, \delta) - approximation$ of the actual similarity, where ϵ denotes a tolerance on the distortion of $\frac{\theta(u_i,u_j)}{\pi}$ and δ is the probability with which ϵ may be exceeded.

Theorem 5.3.2. *To estimate* $\frac{\theta(u_i,u_j)}{\pi}$ *with precision* ϵ *and probability at least* $1-\delta$*, sensor nodes need to produce bitmaps of* $O(\ell og(2/\delta)/(2\epsilon^2))$ *length.*

Proof. Assume a pair of bitmaps X_i, X_j, produced by nodes S_i, S_j, each consisting of d bits. $(X_{i_1}, X_{j_1}), \ldots, (X_{i_d}, X_{j_d})$ denotes corresponding positions of the two bitmaps.

As already mentioned (Theorem 5.3.1), using LSH on bitmap production at the sensor level ensures that bits at the same position of X_i, X_j are designed to be equal with a probability proportional to the angle similarity of the initial vectors. Nonetheless, bits at different positions are produced using independent, random vectors r. As a result, checks at each position k during the hamming distance calculation can be considered as independent random variables Y_k which yield 1, if the corresponding bits differ, and 0 otherwise.

This yields $Y_1, \ldots, Y_d$ random variables, with $Y_i = 0$ or $Y_i = 1$. Obviously, $\sum_{i=1}^{d} \frac{Y_i}{d} = \frac{D_h(X_i,X_j)}{d}$, the expectation of which is $\frac{\theta(u_i,u_j)}{\pi}$ (Equation 5.5). Hoeffding's inequality [37] states that for any $\epsilon > 0$:

$$Pr[|\frac{D_h(X_i,X_j)}{d} - \frac{\theta(u_i,u_j)}{\pi}| \geq \epsilon] \leq 2e^{-2d\epsilon^2} \tag{5.6}$$

Let the right side of the inequality be δ $(0 < \delta < 1)$. Then, one can see that the estimation $\frac{D_h(X_i,X_j)}{d}$ obtained using bitmaps X_i, X_j is beyond $\pm\epsilon$ of the initial value $\frac{\theta(u_i,u_j)}{\pi}$ with probability $\leq \delta$. Hence, $\epsilon = \sqrt{\ell n(2/\delta)/(2d)}$ and $d = \ell n(2/\delta)/(2\epsilon^2)$ which completes the proof. □

So far, we discussed the general operational aspects of our framework (Section 5.2). Moreover, we formally presented the preliminaries of LSH schemes and looked into the primitives for RHP application on mote values(Section 5.3). In the upcoming section we bind these together and primly analyze the details of the outlier identification process throughout its various stages.

5.4 Detecting Outliers with TACO

We now present the operation of our TACO framework in detail. As discussed in Section 5.1, outlying values often cannot be reliably deduced without considering the correlation of a sensor's recent measurements with those of other sensor nodes. Hereafter,

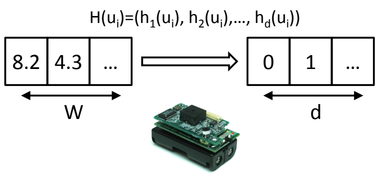


Figure 5.2: LSH application to mote's value vector

we propose a generic technique that takes into account the aforementioned parameters providing an energy efficient way to detect outlying values.

5.4.1 Running Example and Query Format

In what follows, we will use as a running example the case where

- the angle similarity between two vectors u_i and u_j is chosen; and
- the angle similarity threshold Φ_θ is set equal to Φ.

Thus, in our running example $sim(u_i, u_j) = \theta(u_i, u_j)$, and $\Phi_\theta = \Phi$. In Section 5.6 we demonstrate the way that other similarity measures, included in Table 5.1, can be utilized in TACO.

We now describe the syntax of the queries supported by TACO. Query sections enclosed in "[...]" denote optional sections that may be omitted. The meaning of each query section will be explained shortly.

Assume that the sensor network's base station poses a query of the form:

SELECT $c.S_i$
FROM Clusterheads c
WHERE $c.Support_{S_i} < minSup$
USING
TUMBLE_SIZE=W
ENCODING_BITS=d SEEDED_BY seed
TESTS($sim()$, Φ)
[CHECKS_ON ¡specifications on $sim()$ tests¿]
[BOOSTING_GROUPS=μ]

This query is executed continuously in the network until it is explicitly terminated by the base station. The *seed* parameter in the USING compartment is encapsulated in the query so as to enable every mote in the network to produce the same set of random vectors r, which will be utilized during the LSH encoding. In addition, in the CHECKS_ON line of the query, users are able to declare a set of non dynamic

Algorithm 1 TACO at Individual Motes

```
Require: Event of Query Reception
 1: compute $h_r$ Functions($W$, $d$, seed)
 2: {¡¡Sampling¿¿}
 3: repeat
 4:     call getData()
 5:     if dataReady() then
 6:         $u_i \leftarrow newSample$
 7:     end if
 8: until Tumble of W size is formed
 9: compute$X_i(u_i, h_r$ functions) {$X_i$ denotes the LSH bitmap of the node}
10: IntraclusterMessage.send($S_i$,$X_i$)
11: {Go to ¡¡Sampling¿¿}
```

specifications regarding restrictions on motes that can witness each other as discussed in Section 5.2.4. As an additional example, users may specify that motes within a certain radius can witness each other for similarity, irrespectively of whether they are assigned to the same cluster, as they are expected to sense similar conditions. Eventually, the BOOSTING_GROUPS specification regards the application of our proposed boosting process which will be presented in Section 5.4.6. As already mentioned, the CHECKS_ON and the BOOSTING_GROUPS lines (surrounded by "[...]") are optional. In case the CHECKS_ON specification is omitted, users allow comparisons of any pair of motes in the network, while the absence of BOOSTING_GROUPS declaration is equivalent to a $\mu = 1$ choice.

The rest of the query parameters have already been discussed in Section 5.2. The previously presented query is broadcasted to the sensor network having clusterheads disseminating the corresponding information within their cluster and towards other peers.

5.4.2 TACO at Individual Motes

Query reception at sensor nodes triggers a sampling procedure. Recalling Section 5.3.2, W recent measurements form a mote's tumble [11]. Sending a W-dimensional value vector as is, exacerbates the communication cost, which is an important factor that impacts the network lifetime. TACO thus applies LSH in order to reduce the amount of transmitted data. In particular, after having collected W values, each mote applies d $h_r()$ functions on it so as to derive a bitmap of length d (Figure 5.2), where the ratio of d to the size of the W measurements determines the achieved reduction. The derived bitmap is then transmitted to the corresponding clusterhead. The whole process is sketched in Algorithm 1. We additionally note that in case of severe memory constraints, motes do not need to pre-compute and locally store the random vectors (Line 1 of Algorithm 1), but instead sensor nodes are capable of generating those vectors on the fly using the common seed [38].

Algorithm 2 TACO's Intra-cluster Processing at Clusterhead C_i

```
Require: Bitmap Reception from Motes in Cluster
 1: for all pairs of motes ($S_i$,$S_j$) do
 2:     if $D_h(X_i, X_j) \leq \Phi_{D_h}$ then
 3:         $Support_{S_i} \leftarrow Support_{S_i} + 1$
 4:         $Support_{S_j} \leftarrow Support_{S_j} + 1$
 5:     end if
 6: end for
 7: for all $S_i$s in Cluster do
 8:     if $Support_{S_i} < minSup$ then
 9:         $PotOut_{C_i} \leftarrow < S_i, X_i, Support_{S_i} >$
10:     end if
11: end for
```

5.4.3 Intra-Cluster Processing

In the next phase, each clusterhead is expected to report outlying values. To do so, it would need to compare pairs of received vectors in order to determine their similarity based on Equation 5.1 and the Φ_θ similarity threshold. On the contrary, the information that reaches the clusterhead is in the form of compact bitmap representations. Note that Equation 5.5 provides a way to express the angle similarity in terms of the hamming distance and also the similarity threshold $\Phi_{D_h} = d \cdot \frac{\Phi_\theta}{\pi}$. Thus, clusterheads can obtain an approximation of the initial similarity by examining the hamming distance between pairs of bitmaps.

Hence, after the reception of intracluster messages by motes in its cluster, every clusterhead proceeds according to Algorithm 2. If the hamming distance between two tested bitmaps is lower than or equal to Φ_{D_h}, then the two initial vectors will be considered similar, and each sensor in the pair will be able to witness the measurements of the other sensor, thus being able to increase its support by 1 (Lines 1-6). At the end of the procedure, each clusterhead determines a set of potential outliers ($PotOut$) within its cluster, and extracts a list of triplets in the form $\langle S_i, X_i, support \rangle$ containing for each outlier S_i its bitmap X_i and the current support that X_i has achieved so far (Lines 7-11).

5.4.4 Inter-Cluster Processing

Local outlier lists extracted by clusterheads take into account both the recent history of values and the neighborhood similarity (i.e., motes with similar measurements in the same cluster). However, this list of outliers is not final, as the posed query may have specified that a mote may also be witnessed by motes assigned to different clusterheads. Thus, an *inter-cluster communication* phase must take place, in which each clusterhead communicates information (i.e., its produced triplets) regarding its local, potential outlier motes that do not satisfy the *minSup* parameter. In addition, if the required support is different for each sensor (i.e., $minSup$ is expressed as a percentage of nodes in the cluster), then the desired $minSup$ parameter for each potential outlier also needs to be

Algorithm 3 TACO's Inter-cluster Processing at Clusterhead C_i

```
1: while more motes in $PotOut_{C_i}$ do
2:     InterclusterMessage.send(¡$S_i, X_i, Support_{S_i}$¿ $\in PotOut_{C_i}$)
       {towards the next TSP hop}
3: end while
 {Intercluster Message Reception Handling}
1: if InterclusterMessage.receive(¡$S_i, X_i, Support_{S_i}$¿) then
2:     for all $S_j$s in the current cluster do
3:         if $D_h(X_i, X_j) \leq \Phi_{D_h}$ then
4:             $Support_{S_i} \leftarrow Support_{S_i} + 1$
5:         end if
6:     end for
7:     if $Support_{S_i} < minSup$ then
8:         InterclusterMessage.send(¡$S_i, X_i, Support_{S_i}$¿)
           {towards the next TSP hop}
9:     end if
10: end if
```

transmitted. Please note that the number of potential outliers is expected to only be a small portion of the total motes participating in a cluster.

During the intercluster communication phase (sketched in Algorithm 3) each clusterhead thus transmits its potential outliers to those clusterheads where its locally determined outliers may increase their support (based on the restrictions of the posed query - first three lines of Algorithm 3). This is achieved by using a circular network path computed by solving a TSP problem that has as origin the clusterhead, as endpoint the base station, and as intermediate nodes those sensors that may help increase the support of this clusterhead's potential outliers. The TSP can be computed either by the basestation after clusterhead election, or in an approximate way by imposing GPSR [63] to aid clusterheads make locally optimal routing decisions. However, note that such a greedy algorithm for TSP may result in the worst route for certain point - clusterhead distributions [48].

Any item in the $PotOut_{C_i}$ set of potential outliers of clusterhead C_i received by a clusterhead C_j is compared to local sensor bitmaps and the support parameter of nodes within $PotOut_{C_i}$ is increased appropriately upon a similarity occurrence (Lines 1-6 in Intercluster Message Reception Handling of Algorithm 3). In this phase, upon a successful similarity test, we do not increase the support of motes within the current cluster (i.e., the cluster of C_j), since at the same time the potential outliers produced by C_j have been correspondingly forwarded to neighboring clusters in search of additional support. Any potential outlier that reaches the desired *minSup* support is excluded from the list of potential outliers that will be forwarded to the next clusterhead (Lines 7-9 of Intercluster Message Reception Handling).

Eventually, motes that do not manage to accumulate adequate witnesses are identified as outliers. The corresponding information that reaches the base station comes in the form of the S_i of each node that shows abnormal behavior. Nevertheless, the base station may additionally require inspection of the sampled values obtained by these

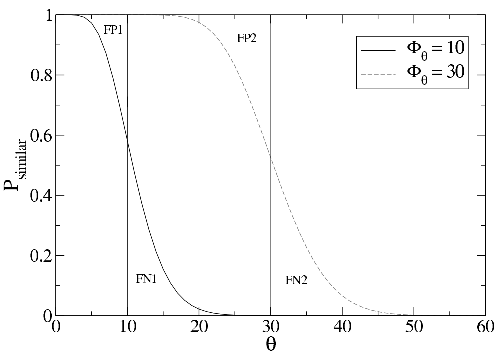


Figure 5.3: Probability $P_{similar}$ of judging two bitmaps as similar, depending on the angle (θ) of the initial vectors and for two different thresholds Φ_θ (W=16, reduction ratio=1/4).

motes. An efficient way to derive estimations of those values is by having clusterheads forwarding the bitmap X_i, on par with the identifier of the nodes, to the query source. Subsequently, sampled values' estimations can be extracted by applying a reverse LSH process [38]. Of course, a need for exact mote sample acquisition introduces an additional step of directly querying the pinpointed outlier sensors.

5.4.5 Analysis

The similarity tests that take place during TACO's intra- and intercluster processing are approximate in nature, as their decisions rely on hamming distance tests on pairs of mote bitmaps, instead of the angle similarity of initial mote vectors. In this section, we elaborate on the quality guarantees provided by TACO with respect to the aforementioned similarity estimations for the given Φ_θ threshold. We initially discuss some intuition on these quality guarantees and present graphical analysis results, when different parameters are varied. We then provide an analysis based on the use of Chernoff bounds.

We first demonstrate how one could estimate the expected error rate of the performed checks. Recall that for any pair of vectors u_i, u_j, the probability P that the corresponding bits in their bitmap encoding are equal is given by Equation 5.2. Thus, the probability of satisfying the similarity test, via LSH manipulation can be expressed by the cumulative function of a binomial distribution:

$$P_{similar} = \sum_{i=0}^{\Phi_{D_h}} \binom{d}{i} P^{d-i} \cdot (1-P)^i \qquad (5.7)$$

As an example, Figure 5.3 plots the value of $P_{similar}$ as a function of $\theta(u_i, u_j)$ (recall

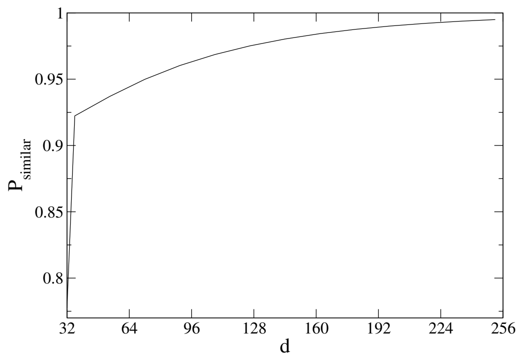


Figure 5.4: Probability $P_{similar}$ of judging two bitmaps (of vectors that pass the similarity test) as similar, depending on the number of bits d used in the LSH encoding (W=16, θ=5, Φ_θ=10)

that P is a function of the angle between the vectors) for two different values of Φ_θ. The area FP_1 above the line on the left denotes the probability of classifying any two vectors as dissimilar, even though their $theta$ angle is less than the threshold (false positive). Similarly, the area FN_1 denotes the probability of classifying the encodings as similar, when the corresponding initial vectors are not (false negative). The areas denoted as FP_2 and FN_2 correspond to the same probabilities for an increased value of Φ_θ. We can observe that the method is more accurate (i.e., leads to smaller areas for false positive and negative detections) for more strict definitions of an outlier implied by smaller Φ_θ thresholds. In Figure 5.4 we depict the probability that TACO correctly identifies two similar ($\theta = 5, \Phi_\theta = 10$) vectors as similar, varying the length d of the bitmap. As expected, using more LSH hashing functions (i.e., choosing a higher value of d), increases the probability of resulting in a successful test.

We can, therefore, estimate the expected false positive and false negative rate in similarity tests as a fraction of those regions over the whole graph area i.e. $\frac{FP}{\pi}, \frac{FN}{\pi}$, with:

$$\begin{cases} FP = \Phi_\theta - \int_0^{\Phi_\theta} P_{similar}(\theta) d\theta \\ FN = \int_{\Phi_\theta}^{\pi} P_{similar}(\theta) d\theta \end{cases}$$

Additionally, in case some rough, prior knowledge of the probability density function df of $\theta(u_i, u_j)$ exists, we are able to derive more accurate FP, FN estimations. In our previous discussion, we assumed that every $\theta(u_i, u_j)$ value may appear with equal probability. $df(\theta)$ information, however, provides more precise information about the angle values frequency. As a result, we can incorporate this knowledge in our estimation by substituting $P_{similar}(\theta)$ with $df(\theta)P_{similar}(\theta)$ in the presented integrals.

We proceed by demonstrating that the probability of incorrect similarity estimation of two vectors u_i, u_j decreases exponentially with the difference $|\theta(u_i, u_j) - \Phi_\theta|$.

Theorem 5.4.1. *For any $\theta(u_i,u_j) > 0$ and $\epsilon = |\frac{\theta(u_i,u_j)-\Phi_\theta}{\theta(u_i,u_j)}|$, clusterheads perform a correct similarity test for u_i, u_j by means of $D(X_i,X_j)$ with probability at least $1-\delta$, where $\delta = e^{-\frac{(\theta(u_i,u_j)-\Phi_\theta)^2}{\theta(u_i,u_j)}\frac{d}{2\pi}}$.*

Proof. First, please note that for $\theta(u_i,u_j) = 0$, our scheme will always return the same bitmaps for u_i and u_j, thus always correctly classifying them as similar.

Let Y_i be a random variable that takes the value of 0 if the bits at the $i-th$ position of the bitmaps X_i, X_j (received by a clusterhead) agree and 1 otherwise. We can then introduce a new random variable $Y = \sum_{i=1}^{d} Y_i$. Substituting $\sum_{i=1}^{d} Y_i$ with $D_h(X_i,X_j)$, we obtain: $Y = D_h(X_i,X_j)$. Since, by Equation 5.5, $E[Y_i] = \frac{\theta(u_i,u_j)}{\pi}$ $\Rightarrow E[Y] = d\frac{\theta(u_i,u_j)}{\pi}$.

We prove the theorem for the case when the initial value vectors are dissimilar ($\theta(u_i,u_j) > \Phi_\theta$). The proof for the reverse case is symmetric. The proof will be based on the use of the Chernoff bound [18]. Let $\epsilon = |\frac{\theta(u_i,u_j)-\Phi_\theta}{\theta(u_i,u_j)}|$. Thus, for the case of $\theta(u_i,u_j) > \Phi_\theta$, $\Phi_\theta = \theta(u_i,u_j)(1-\epsilon)$.

$$Pr[E[Y] \le d\frac{\Phi_\theta}{\pi}] = Pr[D_h(X_i,X_j) \le d\frac{\theta(u_i,u_j)(1-\epsilon)}{\pi}] =$$

$$Pr[d\frac{\theta(u_i,u_j)}{\pi} - D_h(X_i,X_j) \ge \epsilon d\frac{\theta(u_i,u_j)}{\pi}] \le e^{-\epsilon^2 d\frac{\theta(u_i,u_j)}{2\pi}} \Rightarrow$$

$$Pr[\frac{D_h(X_i,X_j)}{d}\pi \le \Phi_\theta] \le e^{-\frac{(\theta(u_i,u_j)-\Phi_\theta)^2}{\theta(u_i,u_j)}\frac{d}{2\pi}} \tag{5.8}$$

The bound of Inequality 5.8 is a strictly increasing function of $\theta(u_i,u_j)$ in the interval $[0,\Phi_\theta]$ and a strictly decreasing function in the interval $[\Phi_\theta,\pi]$. Thus, the probability of incorrect estimation using TACO decreases (exponentially) with $|\theta(u_i,u_j) - \Phi_\theta|$. This concludes our proof. □

5.4.6 Boosting TACO Encodings

We note that the process described in Sections 5.4.3, 5.4.4 can accurately compute the support of a mote in the network (assuming a reliable communication protocol that resolves conflicts and lost messages). Thus, if the whole process was executed using the initial measurements (and not the LSH vectors) the resulting list of outliers would be exactly the same with the one that would be computed by the base station, after receiving all measurements and performing the calculations locally. The application of LSH however results in imprecision during pair-wise similarity tests. We previously presented how this imprecision can be bounded in a controlable manner. We also noted (Figure 5.4) that increasing the size of the bitmaps produced by motes, improves TACO's accuracy. Nevertheless, larger bitmaps imply higher energy consumption. To avoid extra communication burden, we propose an alternative technique to achieve improved accuracy.

Assume that a clusterhead has received a pair of bitmaps X_i, X_j, each consisting of d bits. We split the initial bitmaps X_i, X_j in μ groups (X_{i_1}, X_{j_1}), (X_{i_2}, X_{j_2}), $\ldots$, (X_{i_μ}, X_{j_μ}), such that X_i is the concatenation of X_{i_1},$\ldots$,X_{i_μ}, and similarly for X_j. Each of X_{i_κ} and X_{j_κ} is a bitmap of n bits such that $d = \mu \cdot n$. For each group g_κ we obtain an estimation θ_κ of angle similarity using Equation 5.5 and, subsequently, an answer to the similarity test based on the pair of bitmaps in the group. We then provide as an answer to the similarity test, the answer provided by the majority of the μ similarity tests.[2]

Two questions that naturally arise are: (i) Does the aforementioned partitioning of the hash space help improve the accuracy of successful classification?; and (ii) Which value of μ should one use? Let us consider the probability of correctly classifying two similar vectors in TACO (the case of dissimilar vectors is symmetric). In our original (unpartitioned) framework, the probability of correctly determining the two vectors as similar is $P_{similar}(d)$, given by Equation 5.7. Thus, the failure probability of incorrectly classifying two similar vectors as dissimilar is $P_{wrong}(d) = 1 - P_{similar}(d)$.

By separating the initial bitmaps to μ groups, each containing $\frac{d}{\mu}$ bits, one can view the above classification as using μ independent Bernoulli trials, which each return 1 (similarity) with a success probability of $P_{similar}(\frac{d}{\mu})$, and 0 (dissimilarity), otherwise. Let Y denote the random variable that computes the sum of these μ trials. In order for our classification to incorrectly classify the two vectors as dissimilar, more than half of the μ similarity tests must fail. The average number of successes in these μ tests is $\overline{Y} = \mu \times P_{similar}(\frac{d}{\mu})$. A direct application of the Chernoff bounds gives that more than half of the Bernoulli trials can fail with a probability $P_{wrong}(d, \mu)$ at most: $P_{wrong}(d, \mu) \leq e^{-2\mu(P_{similar}(\frac{d}{\mu}) - \frac{1}{2})^2}$. Given that the number of bits d and, thus, the number of potential values for μ is small, we may compare $P_{wrong}(d, \mu)$ with $P_{wrong}(d)$ for a small set of μ values and determine whether it is more beneficial to use this boosting approach or not. We also need to make two important observations regarding the possible values of μ: (i) The number of possible μ values is further restricted by the fact that our above analysis holds for μ values that provide a (per group) success probability > 0.5 and (ii) Increasing the value of μ may provide worse results, as the number of used bits per group decreases. We explore this issue further in our experiments.

5.4.7 Discussion

The robustness of TACO's outlier definition, as well as the tuning capabilities that it provides, render the framework straightforwardly applicable to a wide variety of

[2]Ties are resolved by taking the median estimate of θ_ks.

application classes. The setting of TACO's parameters is in direct relation to the application context. In particular, the first of these parameters regards the window size W which can be tuned depending on the application's desire to base its decisions on short- or long-term observations. The second parameter refers to the length d of the LSH bitmaps, which affects the number of transmitted bits and for which we have extensively analyzed (Sections 5.3,5.4) its impact on the accuracy of our techniques. The desired similarity threshold (Φ) and the level of support ($minSup$) are essentially those values that determine sensitive or more relaxed outlier definitions referring to neuralgic or ordinary deployments of TACO, respectively.

A popular deployment field where wireless sensor networks suit themselves in, relates to habitat monitoring applications [94]. As an example, consider a monitoring application that aims at investigating bird breeding conditions with motes placed in nests. Since nest temperatures may affect monitored behaviors, such mote samples are considered of particular utility. The apparition of outlying temperature measurements of nodes near nests may attract researchers' interest to further look into caused reactions. As scientists usually base their investigation on long term observations [94], large window sizes may be utilized. In TACO's setting, W values between 24-32 measurements can be applicable, with $\Phi_\theta = 30$ degrees and $minSup$ values of 4 motes withing a radius of $\sim$10 meters near bird nests. To prolong the scientific observation interval and thus the lifetime of the whole sensor network, d values can be squeezed to yield data reduction ratios of $[1/8, 1/16]$.

As another example, consider motes as machine particles in industrial applications where controllers need to pinpoint machines that exhibit high vibrations indicating malfunctioning conditions. Their aim is to timely diagnose those machines and intervene to prevent the production process to be ceased due to permanent casualty. Consequently, sampling rates are high while accuracy is of importance to avoid false negative or positive alarms. The first requirement may lead to selecting smaller window sizes of $W = 16$ measurements, while the second requirement premises d values ensuring moderate (e.g. 1/2 or 1/4) reduction ratios and sensitive similarity definitions i.e., $\Phi_\theta = 10$ degrees.

The above are representative examples of TACO's adoption and parameter tuning. Of course, the framework itself is not limited to the discussed scenarios but can serve as an outlier detection tool in any deployment field.

5.5 Load Balancing and Comparison Pruning

In our initial framework, clusterhead nodes are required to perform data collection and reporting, as well as bitmap comparisons. As a result, clusterheads are overloaded with extra communication and processing costs, which entails larger energy drain, when compared to other nodes. In order to avoid draining the energy of clusterhead nodes, the network structure will need to be frequently reorganized (by electing new clusterheads). While protocols such as HEED [122] limit the number of messages required during the clusterhead election process, this election process still requires bandwidth. In this section, we tackle with this issue and describe efficient mechanisms provided by TACO for limiting clusterheads' load.

5.5.1 Leveraging Additional Motes for Outlier Detection

In order to limit the overhead of clusterhead nodes, we extend our framework by incorporating the notion of *bucket nodes*. *Bucket nodes* (or simply buckets) are motes within a cluster the presence of which aims at distributing communication and processing tasks and their associated costs. Besides selecting the clusterhead nodes, within each cluster the election process continues to elect additional B bucket nodes. This election process is easier to carry out by using the same algorithm (i.e., HEED) that we used for the clusterhead election.

After electing the bucket nodes within each cluster, our framework determines a mechanism that distributes the outlier detection duties amongst them. Our goal is to group similar bitmaps in the same bucket so that the comparisons that will take place within each bucket produce just a few local outliers. To achieve this, we introduce a second level of hashing.

Proposition 5.5.1. *Let $W_h(X_i) = \sum_{\ell=1}^{d} X_{i\ell}$ be the hamming weight of bitmap X_i with $0 \leq W_h(X_i) \leq d$. For any pair of bitmaps X_i and X_j, it holds that $D_h(X_i, X_j) \geq |W_h(X_i) - W_h(X_j)|$.*

Proof. For any pair of bitmaps X_i, X_j:

$$|W_h(X_i) - W_h(X_j)| = |\sum_{\ell=1}^{d} X_{i\ell} - \sum_{\ell=1}^{d} X_{j\ell}| = |\sum_{\ell=1}^{d}(X_{i\ell} - X_{j\ell})| \overset{\substack{triangle\\inequality}}{\leq} \sum_{\ell=1}^{d} |X_{i\ell} - X_{j\ell}| = D_h(X_i, X_j)$$ □

Corollary 5.5.1. *If $|W_h(X_i) - W_h(X_j)| > \Phi_{D_h}$, then the bitmaps X_i and X_j cannot witness each other.*

Our second level of hashing takes into consideration Colorrary 5.5.1 to hash highly dissimilar bitmaps to different buckets. More precisely:

- Consider a partitioning of the hash key space to the elected buckets, such that each hash key is assigned to the $\lfloor \frac{W_h(X_i)}{\frac{d}{B}} \rfloor$-th bucket. Motes with similar bitmaps will have nearby Hamming weights, thus hashing to the same bucket with high probability.

- Please recall that encodings that can support each other in our framework have a Hamming distance lower or equal to Φ_{D_h}. In order to guarantee that a node's encoding can be used to witness any possible encoding within its cluster, it needs to be sent to all buckets that cover the hash key range $\lfloor\frac{\max\{W_h(X_i)-\Phi_{D_h},0\}}{\frac{d}{B}}\rfloor$ to $\lfloor\frac{\min\{W_h(X_i)+\Phi_{D_h},d\}}{\frac{d}{B}}\rfloor$. Thus, the value of B determines the number of buckets to which an encoding must be sent. Larger values of B reduce the range of each bucket, but result in more encodings being transmitted to multiple buckets. In our framework, we select the value B (whenever at least B nodes exist in the cluster) by setting $\frac{d}{B} > \Phi_{D_h} \implies B < \frac{d}{\Phi_{D_h}}$. As we will shortly show, this guarantees that each encoding will need to be transmitted to at most one additional bucket, thus avoiding hashing the measurements of each node to multiple buckets.

- The transmission of an encoding to multiple bucket nodes ensures that it may be tested for similarity with any value that may potentially witness it. Therefore, the support that a node's measurements have reached is distributed in multiple buckets needing to be combined.

- Moreover, we must also make sure that the similarity test between two encodings is not performed more than once. Thus, we impose the following rules: (a) For encodings mapping to the same bucket node, the similarity test between them is performed only in that bucket node; and (b) For encodings mapping to different bucket nodes, their similarity test is performed only in the bucket node with the lowest id amongst these two. Given these two requirements, we can thus limit the number of bucket nodes to which we transmit an encoding to the range $\lfloor\frac{\max\{W_h(X_i)-\Phi_{D_h},0\}}{\frac{d}{B}}\rfloor$ to $\lfloor\frac{W_h(X_i)}{\frac{d}{B}}\rfloor$. The above range for $B < \frac{d}{\Phi_{D_h}}$ is guaranteed to contain at most 2 buckets.

Thus, each bucket reports the set of outliers that it has detected, along with their support, to the clusterhead. The clusterhead performs the following tests:

- Any encoding reported to the clusterhead by at least one, but not all bucket nodes to which it was transmitted, is guaranteed not to be an outlier, since it must have reached the required support at those bucket nodes that did not report the encoding.

- For the remaining encodings, the received support is added, and only those encodings that did not receive the required overall support are considered to be outliers.

5.5.2 Load Balancing Among Buckets

Despite the fact that the introduction of bucket nodes does alleviate clusterheads from comparison and message reception load, it does not guarantee by itself that the portion

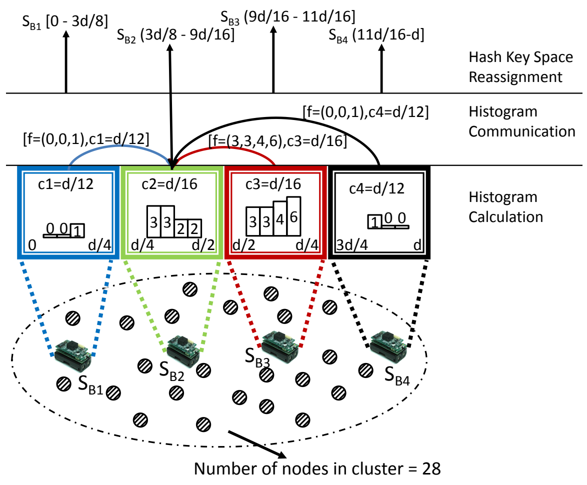


Figure 5.5: Exemplary (bottom-up) demonstration of the 3 phases of load balancing

of load taken away from the clusterheads will be equally distributed between buckets. In particular, we expect that motes sampling ordinary values of measured attributes will produce similar bitmaps, thus directing these bitmaps to a limited subset of buckets, instead of equally utilizing the whole arrangement. In such a situation, an equi-width partitioning of the hash key space to the bucket nodes is obviously not a good strategy. On the other hand, if we wish to determine a more suitable hash key space allocation, we require information about the data distribution of the monitored attributes and, more precisely, about the distribution of the hamming weight of the bitmaps that original value vectors yield. Based on the above observations, we can devise a load balancing mechanism that can be used after the initial, equal-width partitioning in order to repartition the hash key space between bucket nodes. Our load balancing mechanism fosters simple equi-width histograms and consists of three phases: a) *histogram calculation* per bucket, b) *histogram communication* between buckets, and c) *hash key space reassignment*.

During the *histogram calculation* phase, each bucket locally constructs equi-width histograms by counting $W_h(X_i)$ frequencies belonging to bitmaps that were hashed to them. The range of histogram's domain value is restricted to the hash key space portion assigned to each bucket. Obviously, this phase takes place side by side with the normal operation of the motes. We note that this phase adds minimum computation overhead since it only involves increasing by one the corresponding histogram bucket counter for each received bitmap.

In the *histogram communication* phase, each bucket communicates to its clusterhead (a) its estimated frequency counts attached, and (b) the width parameter c that it used in its histogram calculation. From the previous partitioning of the hash key space, the clusterhead knows the hash key space of each bucket node. Thus, the transmission of the width c is enough to determine (a) the number of received bars/values, and (b) the range of each bar of the received histogram. As a result, the clusterhead can easily reconstruct the histograms that it received.

The final step involves the adjustment of the hash key space allocation that will eventually provide the desired load balance based on the transmitted histograms. Relying on the received histograms, the clusterhead determines a new space partitioning and broadcasts it to all nodes in its cluster. The aforementioned phases can be periodically (but not frequently) repeated to adjust the bounds allocated to each bucket, adapting the arrangement to changing data distributions. Figure 5.5 depicts an example of the load balancing procedure. To simplify the figure, in this example we assume that the second bucket node is also the clusterhead.

The mechanisms described in this section better balance the load among buckets and also refrain from performing unnecessary similarity checks between dissimilar pairs of bitmaps, which would otherwise have arrived at the clusterhead. This stems from the fact that hashing bitmaps based on their hamming weight ensures that dissimilar bitmaps are hashed to different buckets. We experimentally validate the ability of this second level hashing technique to prune the number of comparisons in Section 5.8.5.

5.6 TACO under Other Supported Similarity Measures

We have already noted the ability of our framework to encompass a wide variety of popular similarity measures. So far, in our running example, we showed TACO's function using the angle (and, thus, the cosine similarity) between sensor value vectors. In this subsection we provide a detailed discussion on how the other measures presented in Table 5.1 can be incorporated in TACO.

Correlation Coefficient. Let $E(u_i)$ notate the mean value and σ_{u_i} the standard deviation of vector u_i. Moreover for mote value vectors u_i, u_j we denote $u_i^* = u_i - E(u_i)$, $u_j^* = u_j - E(u_j)$.

Proposition 5.6.1. *The correlation coefficient (*corr*) can be used as a similarity measure in TACO by using the same family of hashing functions as with the cosine similarity in the Random Hyperplane Projection LSH scheme.*

Proof. To prove the proposition it suffices to show that some kind of equivalence between the two measures (i.e., cosine similarity and correlation coefficient) exists. In particular,

$$corr(u_i, u_j) = corr(u_i^*, u_j^*) = cos(\theta(u_i^*, u_j^*)) \tag{5.9}$$

holds. We now provide a simple proof for the previous equation, starting from the first part, and based on the observation that $E(u_i^*) = E(u_j^*) = 0$, while also $\sigma_{u_i^*} = \sigma_{u_i}$

and $\sigma_{u_j^*} = \sigma_{u_j}$:

$$\begin{aligned}
corr(u_i^*, u_j^*) &= \frac{E(u_i^* u_j^*) - E(u_i^*)E(u_j^*)}{\sigma_{u_i^*}\sigma_{u_j^*}} \\
&= \frac{E((u_i - E(u_i))(u_j - E(u_j))) - E(u_i^*)E(u_j^*)}{\sigma_{u_i^*}\sigma_{u_j^*}} \\
&= \frac{E(u_i \cdot u_j - u_j \cdot E(u_i) - u_i \cdot E(u_j) + E(u_i) \cdot E(u_j))}{\sigma_{u_i}\sigma_{u_j}} \\
&= \frac{E(u_i \cdot u_j) - E(u_j \cdot E(u_i)) - E(u_i \cdot E(u_j)) + E(u_i) \cdot E(u_j)}{\sigma_{u_i}\sigma_{u_j}} \\
&= \frac{E(u_i \cdot u_j) - E(u_i) \cdot E(u_j)}{\sigma_{u_i}\sigma_{u_j}}
\end{aligned}$$

which equals $corr(u_i, u_j)$. Furthermore:

$$\begin{aligned}
cos(\theta(u_i^*, u_j^*)) = \frac{u_i^* \cdot u_j^*}{||u_i^*|| \cdot ||u_j^*||} = \frac{\frac{1}{W}\sum_{\ell=1}^{W} u_{i\ell}^* \cdot u_{j\ell}^*}{\frac{1}{W}\sqrt{\sum_{\ell=1}^{W} u_{i\ell}^{*2}} \cdot \sqrt{\sum_{\ell=1}^{W} u_{j\ell}^{*2}}} = \frac{E(u_i^* u_j^*)}{\sigma_{u_i^*}\sigma_{u_j^*}} \\
= corr(u_i^*, u_j^*)
\end{aligned}$$

□

In other words, an outlier detection query may specify a similarity threshold Φ_{corr} based on the correlation coefficient for u_i, u_j. Since $corr(u_i, u_j) = corr(u_i^*, u_j^*)$, Φ_{corr} also holds for u_i^*, u_j^*. Additionally, because $corr(u_i^*, u_j^*) = cos(\theta(u_i^*, u_j^*))$, Φ_{corr} can be transformed into a threshold for the hamming distance between bitmaps using Equation 5.5. Hence, after the collection of u_is, motes produce and apply LSH on u_i^*s so as to obtain appropriate bitmaps preserving the angle and, subsequently, the $corr$-similarity of the initial value vectors. The rest of the outlier detection process presented in the previous sections remains unaffected.

Euclidean Distance of Standardized Vectors. A popular similarity measure often used in distance based outlier identification is the Euclidean distance. Nonetheless, this measure relies on absolute values to determine the similarity of u_i, u_j while we have previously reasoned that in our setting the emphasis should be set on the correlations of the motes measurements in space and time rather than on the absolute sampled values. Nevertheless, we are able to adjust the Euclidean distance so as to serve our purposes by considering standardized vectors.

Let $u_i' = \frac{u_i - E(u_i)}{\sigma_{u_i}}, u_j' = \frac{u_j - E(u_j)}{\sigma_{u_j}}$ and $dist(u_i', u_j')$ the Euclidean distance between u_i', u_j' which is calculated by $dist(u_i', u_j') = \sqrt{\sum_{\ell=1}^{W}(u_{i\ell}' - u_{j\ell}')^2}$.

Proposition 5.6.2. *The Euclidean distance* dist() *of standardized mote vectors can capture correlations among sensor readings and is incorporated in TACO by using the* *same family of hashing functions as with the cosine similarity in the Random Hyperplane Projection LSH scheme.*

Proof. Initially, we ought to show that the Euclidean distance of standardized vectors can capture existing correlations between motes' values. In fact, $corr(u_i, u_j) = 1 - \frac{dist^2(u'_i, u'_j)}{2W}$ holds. Observe that upon standardizing mote value vectors, their mean is zero and their standard deviation is 1. As a result, $corr(u'_i, u'_j)$ is reduced to $\frac{1}{W}\sum_{\ell=1}^{W} u'_{i\ell}u'_{j\ell}$ and simple calculations yield that $corr(u_i, u_j) = corr(u'_i, u'_j)$. Moreover, $dist^2(u'_i, u'_j) = \sum_{\ell=1}^{W}(u'_{i\ell} - u'_{j\ell})^2 = \sum_{\ell=1}^{W} u'^2_{i\ell} + \sum_{\ell=1}^{W} u'^2_{j\ell} - 2\sum_{\ell=1}^{W} u'_{i\ell}u'_{j\ell} = 2W - 2W corr(u'_i, u'_j)$. Combining the previous pair of equivalences leads to the discussed $corr(u_i, u_j) = 1 - \frac{dist^2(u'_i, u'_j)}{2W}$ equality.

Since the correlation coefficient captures the correlation among vectors and we exposed a formula connecting it with $dist(u'_i, u'_j)$, the Euclidean distance of standardized vectors can also capture potential interrelations. It remains to exhibit that $dist(u'_i, u'_j)$ is encompassed by the Random Hyperplane Projection scheme which is derived in a straightforward manner according to Equation 5.9:

$$corr(u_i, u_j) = cos(u^*_i, u^*_j) = 1 - \frac{dist^2(u'_i, u'_j)}{2W}$$

. □

Summarizing, it suffices for the user query to place a threshold for $dist(u'_i, u'_j)$ which is transformed into an equivalent Φ_{corr}. That point forward, TACO operates in exactly the same way as with the $corr$ similarity measure choice.

Extended Jaccard Coefficient of Standardized Vectors. Similarly, the Extended Jaccard (or Tanimoto) Coefficient of u'_i, u'_j expressed by the ratio

$$T(u'_i, u'_j) = \frac{u'_i \cdot u'_j}{||u'_i||^2 + ||u'_j||^2 - u'_i \cdot u'_j}$$

is commutatively supported in TACO by means of their Euclidean distance. In particular, given a specific $\Phi_{Tanimoto} \in [0, 1]$ the similarity test $T(u'_i, u'_j) \geq \Phi_{Tanimoto}$ can be performed checking the condition:

$$dist(\frac{\Phi_{Tanimoto} + 1}{2\Phi_{Tanimoto}} u'_i, u'_j) \leq \frac{\sqrt{(\Phi_{Tanimoto} + 1)^2 - 4\Phi^2_{Tanimoto}}}{2\Phi_{Tanimoto}} \sqrt{W}$$

instead [72].

Jaccard Coefficient. Jaccard coefficient is another measure that can be used in applications that require motes sample discrete quantities such as types of objects in the network realm, spatial features etc. In [46] the authors introduce a mechanism for transforming sets of values into dimensionally reduced bitmaps with preserved Jaccard similarity. To achieve that, they use minwise independent permutations and simplex codes. Here, we provide a summary of the main results of [46] for clarity and discuss the way the MinHash scheme [16, 17] utilized there can be embodied in the outlier detection procedure. Moreover, we extend the results of [46] by providing a mechanism to determine appropriate bitmap sizes upon utilizing that scheme in TACO's context.

Let $\pi()$ denote a random permutation on $u_i \epsilon N^W$ (assuming elements of value sets are labeled by Natural numbers) and $min\{\pi(u_i)\} = min\{\pi(u_{i\ell}) | u_{i\ell} \epsilon u_i\}$. According to the min hashing scheme [16, 17, 46]:

$$Pr[min\{\pi(u_i)\} = min\{\pi(u_j)\}] = J(u_i, u_j)$$

After using k random permutations the resulted signatures are expected to agree in $k \cdot J(u_i, u_j)$ values. Taking one step further, signatures can be embedded in the hamming space utilizing error correcting codes. Error correcting codes (ECCs) have the property of transforming the b-lengthed binary representation of each signature element to bitmaps of fixed $(b + s)/2$ distance, for some $s > 0$. The final bitmap is produced by concatenating the ECC outcomes. Eventually, the following equivalence can be proved [46]:

$$\frac{D_h(X_i, X_j)}{d} = \frac{1 - J(u_i, u_j)}{2} \tag{5.10}$$

with bitmap size $d = (b + s) \cdot k$ and $0 \leq D_h(X_i, X_j) \leq d/2$.

In [17] the authors dictate an appropriate value for k to control the number of false positives/false negatives during similarity tests using the signatures with given $\Phi_{Jaccard}$ threshold. Nonetheless, the transition to the hamming space results in additional imprecision. Signature elements are chosen as numbers of fixed precision which determines the length b of their binary representation. On the other hand, normally, when ECCs are used to correct bit errors in communication channels, the value of s is chosen on the basis of the number of errors that an ECC is able to amend. Nevertheless, in the current utilization no such criterion may be applied as our goal is different and regards accurate similarity preservation. Additionally, appropriate k values dictated by [17] premise that the similarity between value vectors is lower bounded by some constant number. Obviously, such an assumption cannot be guaranteed in our setting. In other words, the bounds provided in [17] are not sufficient for the current context since they do not take into consideration the length of the binary representation of the signature elements and the chosen ECC to determine the value of k and subsequently control the imprecision of the Jaccard coefficient based similarity test using Equation 5.10.

Notice that bits at corresponding positions of bitmaps X_i, X_j are not independent as they are produced based on the chosen error correcting code specifications and signature element binary formats. For this reason the boosting process of Section 5.4.6 cannot be used for this kind of bitmaps as we cannot freely partition them into groups. However, the k groups, each of $b+s$ bit length, inside a bitmap are independent since they originate from different signature elements. We exploit this fact to prove the following theorem.

Theorem 5.6.1. *Given the choice of signature element universe and the specifications of the chosen ECC (that is, given $b+s$), to estimate $\frac{1-J(u_i,u_j)}{2}$ with precision $\frac{\epsilon}{b+s}$ and probability at least $1-\delta$ the number of signature elements k should be set to $O(log(2/\delta)(b+s)^2/(8\epsilon^2))$.*

Proof. Assume a pair of bitmaps X_i, X_j produced by applying min hashing and a chosen ECC to sets u_i, u_j correspondingly. Let $Y_1, Y_2, \dots, Y_k$ be independent random variables with $Y_i = 0$ or $Y_i = (b+s)/2$. The average of the sum of these variables is $\sum_{i=1}^{k} \frac{Y_i}{k} = \frac{D_h(X_i,X_j)}{k}$ and the expectation of the previous average, derived by Equation 5.10, is $(b+s)\frac{1-J(u_i,u_j)}{2}$. Utilizing Hoeffding's inequality [37] for some $\epsilon > 0$:

$$Pr[|\frac{D_h(X_i,X_j)}{k} - (b+s)\frac{1-J(u_i,u_j)}{2}| \geq \epsilon] \leq 2e^{-\frac{8k\epsilon^2}{(b+s)^2}}$$

Substituting $(b+s)\cdot k$ with d:

$$Pr[|\frac{D_h(X_i,X_j)}{d} - \frac{1-J(u_i,u_j)}{2}| \geq \frac{\epsilon}{b+s}] \leq 2e^{-\frac{8k\epsilon^2}{(b+s)^2}}$$

Setting the right side of the inequality equal to δ and performing simple calculations, completes the proof. □

The above theorem provides the means to determine the value of k and simultaneously incorporates the effect of the ECC choice (the value of s) in the desired precision of the estimation. After determining the overall d value, the following theorem elaborates on the accuracy of the similarity test performed at the clusterheads' level.

Theorem 5.6.2. *For any $J(u_i,u_j) < 1$ and $\epsilon = \frac{|J(u_i,u_j)-\Phi_{Jaccard}|}{1-J(u_i,u_j)}$, clusterheads perform a correct similarity test for u_i, u_j by means of $D(X_i,X_j)$ with probability at least $1-\delta$, where $\delta = e^{-\frac{d(J(u_i,u_j)-\Phi_{Jaccard})^2}{2(1-J(u_i,u_j))}}$.*

A proof can be obtained as in Theorem 5.4.1 and is omitted. Note again that $J(u_i,u_j) = 1$ leads to identical bitmaps and the probability of incorrect similarity test decreases exponentially only this time with the $|J(u_i,u_j) - \Phi_{Jaccard}|$ difference. Upon motes transform initial sets of values to bitmaps, the outlier detection process using the Jaccard coefficient is quite analogous to the case of cosine similarity with Equation 5.10 (instead of Equation 5.5) as the main tool.

We further note [12] that there exist popular similarity metrics that do not accept an LSH scheme. For instance, Lemma 5.3.1 implies that the $Dice(u_i,u_j) = \frac{2|u_i \cap u_j|}{|u_i|+|u_j|}$

and $Overlap(u_i, u_j) = \frac{|u_i \cap u_j|}{min(|u_i|,|u_j|)}$ coefficients do not admit an LSH scheme since they do not satisfy the triangle inequality.

5.7 Extensions

An obvious way to further decrease communication costs during the intra- and inter-cluster processing of TACO is by suppressing mote messages when bitmaps are not altered in a number of successive tumbles. In particular, let X_i^{last} denote the last bitmap that mote S_i transmitted to the clusterhead (or bucket node) and X_i^{new} the bitmap produced in the current tumble. When $D_h(X_i^{last}, X_i^{new}) = 0$, S_i does not need to communicate any information to the clusterhead. This reduces the burden of intracluster communication. In the intercluster processing, as far as X_i is not modified, should it happen to be included in $PotOut_{C_i}$, X_i^{last} needs to be transmitted only once. Please notice that the result of similarity tests where S_i participates will not be affected at all.

Relaxing the previous condition, we may allow motes suppress their messages in case $D_h(X_i^{last}, X_i^{new}) \leq f$, where $0 \leq f \leq d$. As a consequence, additional savings in bandwidth consumption are yielded, however, with the make-weight of distorting the result of the tests which S_i takes place in. Despite this fact, we can still guarantee that a portion of these tests cannot be affected. Hereafter, we outline the cases for which no distortion in S_i's similarity test outcomes is introduced.

Assume that motes S_i, S_j are to be compared during the intra- or intercluster processing and S_i suppressed its message in the current tumble while S_j did not. The result of the similarity test will rely on $D_h(X_i^{last}, X_j^{new})$. Due to the fact that the hamming distance possesses the property of satisfying the triangle inequality and bearing that $D_h(X_i^{last}, X_i^{new}) \leq f$:

$$|D_h(X_i^{last}, X_j^{new}) - f| \leq D_h(X_i^{new}, X_j^{new}) \leq D_h(X_i^{last}, X_j^{new}) + f \qquad (5.11)$$

Consequently:

$(D_h(X_i^{last}, X_j^{new}) + f \leq \Phi_{D_h}) \vee ((D_h(X_i^{last}, X_j^{new}) \geq f) \wedge (D_h(X_i^{last}, X_j^{new}) - f \geq \Phi_{D_h})) \models$undistorted test.

Provided that both sensor nodes S_i, S_j suppress their messages, $D_h(X_i^{last}, X_j^{last})$ is taken into consideration so as to decide their similarity. In this case:

$$|D_h(X_i^{last}, X_j^{last}) - f| \leq D_h(X_i^{last}, X_j^{new}) \leq D_h(X_i^{last}, X_j^{last}) + f \qquad (5.12)$$

Combining inequalities (5.11),(5.12), for the case of pairs of motes that mutually suppress their messages, we overall obtain:

$(D_h(X_i^{last}, X_j^{last})+2f \leq \Phi_{D_h}) \vee ((D_h(X_i^{last}, X_j^{last}) \geq 2f) \wedge (D_h(X_i^{last}, X_j^{last})-2f \geq \Phi_{D_h})) \models$ undistorted test.

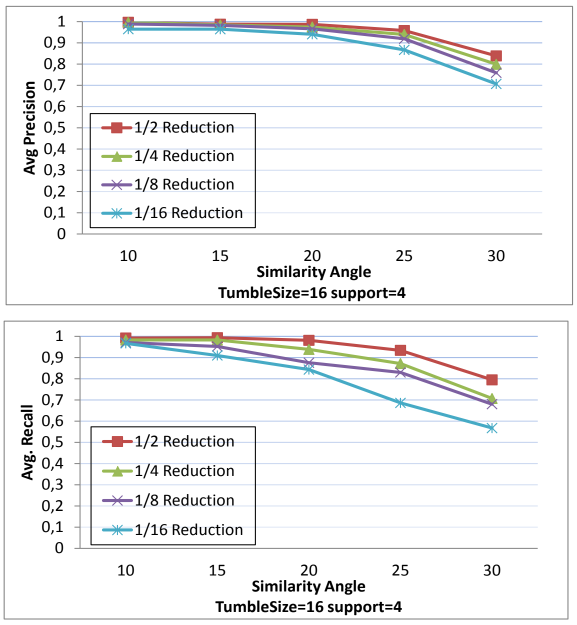
In any other case, depending on the actual changes of the corresponding hamming distance, the adoption of the message suppression strategy may cause alteration (compared to the utilization of $D_h(X_i^{new}, X_j^{new})$) in the result of a test or not. Obviously, increasing the value of f provides increased communication savings but also weakens the guarantees on the distortion of similarity test outcomes. On the other hand, smaller fs produce tighter upper and lower bounds in the presented inequalities (5.11),(5.12) while yielding more moderate bandwidth consumption preservation.

5.8 Experiments

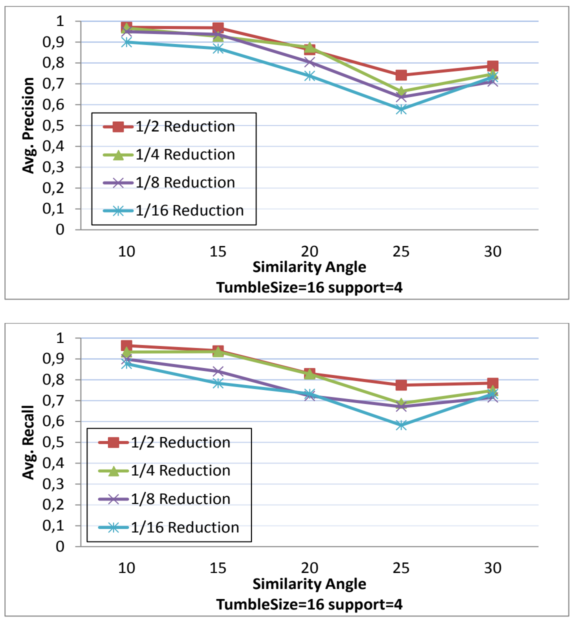
5.8.1 Experimental Setup

In order to evaluate the performance of our techniques we implemented our framework on top of the TOSSIM network simulator [75]. Since TOSSIM imposes restrictions on the network size and is rather slow in simulating experiments lasting for thousands of epochs, we further developed an additional lightweight simulator in Java and used it for our sensitivity analysis, where we vary the values of several parameters and assess the accuracy of our outlier detection scheme. The TOSSIM simulator was used in smaller-scale experiments, in order to evaluate the energy and bandwidth consumption of our techniques and of alternative methods for computing outliers. Through these experiments we examine the performance of all methods, while taking into account message loss and collisions, which in turn result in additional retransmissions and affect the energy consumption and the network lifetime.

In our experiments we utilized two real world data sets. The first, termed Intel Lab Data, includes temperature and humidity measurements collected by 48 motes for a period of 633 and 487 epochs, respectively, in the Intel Research, Berkeley lab [32]. The second, termed Weather Data, includes air temperature, relative humidity and solar irradiance measurements from the station in the University of Washington and for the year 2002 [27]. We used these measurements to generate readings for 100 motes for a period of 2000 epochs. In both data sets we increased the complexity of the temperature and humidity data by specifying for each mote a 6% probability that it will fail dirty at some point. We simulated failures using a known deficiency [30] of the MICA2 temperature sensor: each mote that fails-dirty increases its measurement (in our experiment this increase occurs at an average rate of about 1 degree per epoch), until it reaches a MAX_VAL parameter. This parameter was set to 100 degrees for the



(a) Intel.Temperature Precision, Recall vs Similarity Angle



(b) Intel.Humidity Precision,Recall vs Similarity Angle

Figure 5.6: Average Precision, Recall in Intel Data Set

Intel lab data set and 200 degrees for the Weather data (due to the fact that the Weather data set contains higher average values). To prevent the measurements from lying on a straight line, we also impose a noise up to 15% at the values of a node that fails dirty.

Additionally, each node with probability 0.4% at each epoch obtains a spurious measurement which is modeled as a random reading between 0 and MAX_VAL degrees. Finally, for solar irradiance measurements, we randomly injected values obtained at various time periods to the sequence of readings, in order to generate outliers.

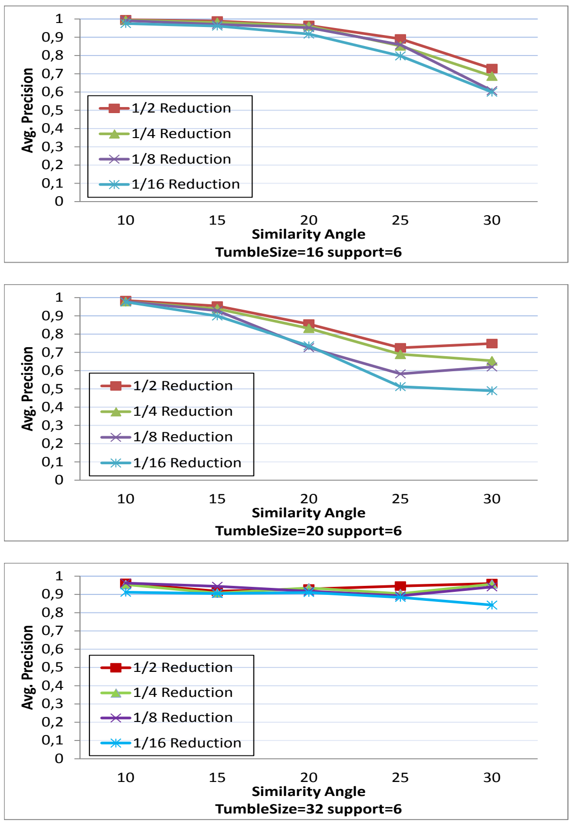
We need to emphasize that increasing the complexity of the real data sets actually represents a worst-case scenario for our techniques. It is easy to understand that the amount of transmitted data during the intracluster communication phase is independent of the data sets' complexity, since it only depends on the specified parameter d that controls the dimensionality reduction. On the other hand, the amount of data exchanged during the intercluster phase of our framework depends on the number of generated outlier values. Thus, the added data set complexity only increases the transmitted data (and, thus, the energy consumption) of our framework. Despite this fact, we demonstrate that our techniques can still manage to drastically reduce the amount of transmitted data, in some cases even below what a simple aggregate query (i.e., MIN, MAX or SUM) would require under TAG [78].

In the Intel Lab and Weather data sets we organized the sensor nodes in four and ten clusters, correspondingly. Please note that we selected a larger number of clusters for the Weather data set, due to the larger number of sensor nodes that appear in it. The sensor nodes were organized in clusters using the HEED algorithm.

5.8.2 Sensitivity Analysis

We first present a series of sensitivity analysis experiments using our Java simulator in order to explore a reasonably rich subset of the parameter space. To evaluate the accuracy of TACO in the available data sets we initially focus on the precision and recall metrics. In a nutshell, the precision specifies the percentage of reported outliers that are true outliers, while the recall specifies the percentage of outliers that are reported by our framework. The set of true outliers was computed offline (i.e. assuming all data was locally available), based on the selected similarity metric and threshold, specified in each experiment. The goal of these experiments is to measure the accuracy of the TACO scheme and of the boosting process, and to assess their resilience to different compression ratios.

We used different tumble sizes ranging between 16 and 32 measurements and Φ_θ thresholds between 10 and 30 degrees. Moreover, we experimented with a reduction ratio up to 1/16 for each (W, Φ_θ) combination. In the Intel Lab data sets we found little fluctuations by changing the *minSup* parameter from 3-5 motes, so henceforth we consider a fixed *minSup*=4 (please recall that there are 48 motes in this data set). Due to a similar observation in the Weather data set, *minSup* is set to 6 motes. All the experiments were repeated 10 times. Figures 5.6 and 5.7 depict the accuracy of our methods presenting the average precision and recall for the used data sets, for different

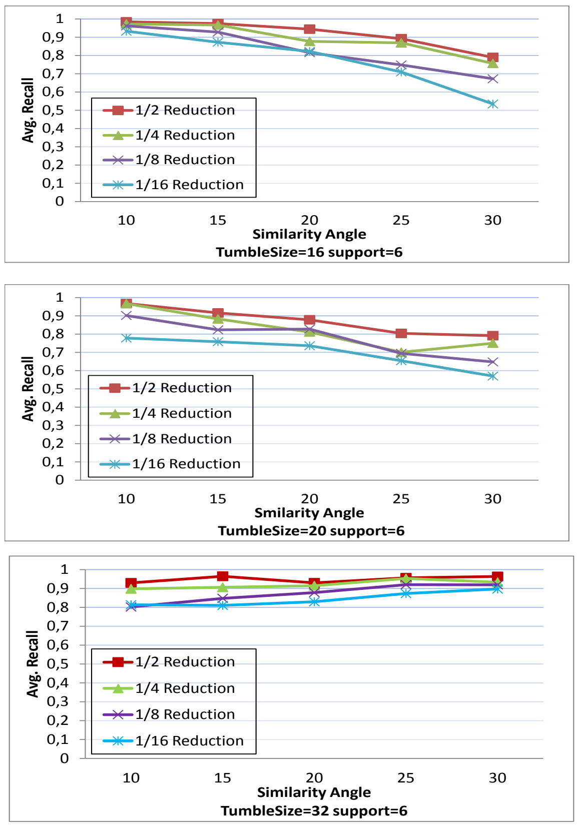


(a) Weather Precision: Temperature, Humidity and Solar Irradiance vs Similarity Angle

Figure 5.7: Average Precision, Recall in Weather Data Set

similarity angles and reduction ratios. To acquire these, we obtained precision and recall values per tumble and calculated the average precision, recall over all tumbles in the run. Finally, we proceeded by estimating averages over 10 repetitions, using a different random set of hash functions in each iteration.

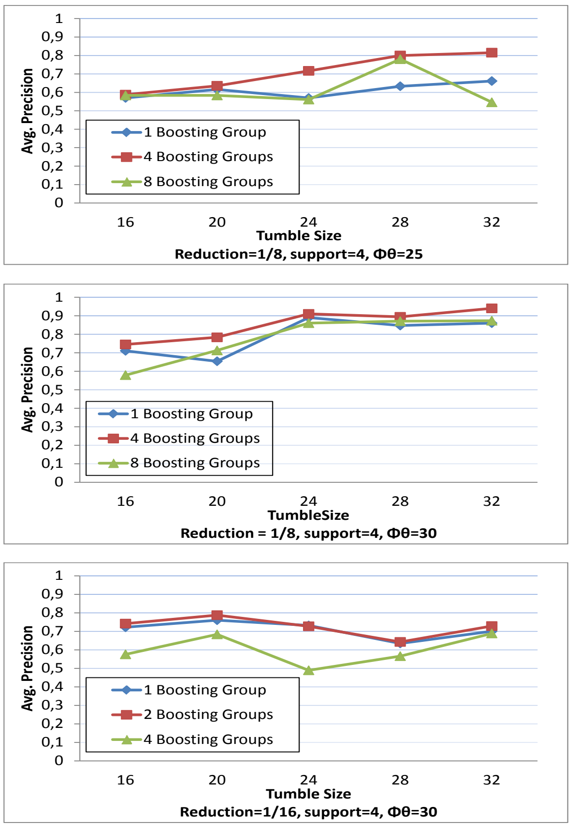
As it can be easily observed, in most of the cases, motes producing outlying values can be successfully pinpointed by our framework with average precision and recall $> 80\%$, even when imposing a 1/8 or 1/16 reduction ratio, for similarity angles up to 20 degrees. The TACO scheme is much more accurate when asked to capture strict, sensitive definitions of outlying values, implied by a low Φ_θ value. This behavior is expected based on our formal analysis (see Figure 5.3 and Equation 5.2). We also note that the model may slightly swerve from its expected behavior depending on the number of near-to-threshold outliers (those falling in the areas FP, FN in Figure 5.3) that exist in the data set. That is, for instance, the case when switching from 25 to 30 degrees in the humidity data sets of Figures 5.6 and 5.7.



(b) Weather Recall: Temperature, Humidity and Solar Irradiance vs Similarity Angle

Figure 5.7: Average Precision, Recall in Weather Data Set (cont)

Obviously, an improvement in the final results may arise by increasing the length d of each bitmap (i.e., consider more moderate reduction ratios, Figure 5.4). Another way to improve performance is to utilize the boosting process discussed in Section 5.4.6. All previous experiments were ran using a single boosting group during the comparison procedure. Figure 5.8 depicts the improvement in the values of precision and recall for the Intel humidity and temperature datasets as more groups are considered and for a variety of tumble sizes (the trends are similar for the other data sets, which are omitted). Points in Figure 5.8 corresponding to the same tumble size W use bitmaps of the same length, so that the reduction ratio is 1/8 (Intel.Humidity) and 1/16 (Intel.Temperature), but differ in the number of groups utilized during the similarity estimation. It can easily be deduced that using 4 boosting groups in the humidity dataset is the optimal solution for all the cited tumble sizes, while the 4 group line tends to ascend by increasing the W parameter. This comes as no surprise since the selection of higher W values results in larger (but still 1/8 reduced) bitmaps, which in turn equip the overall comparison

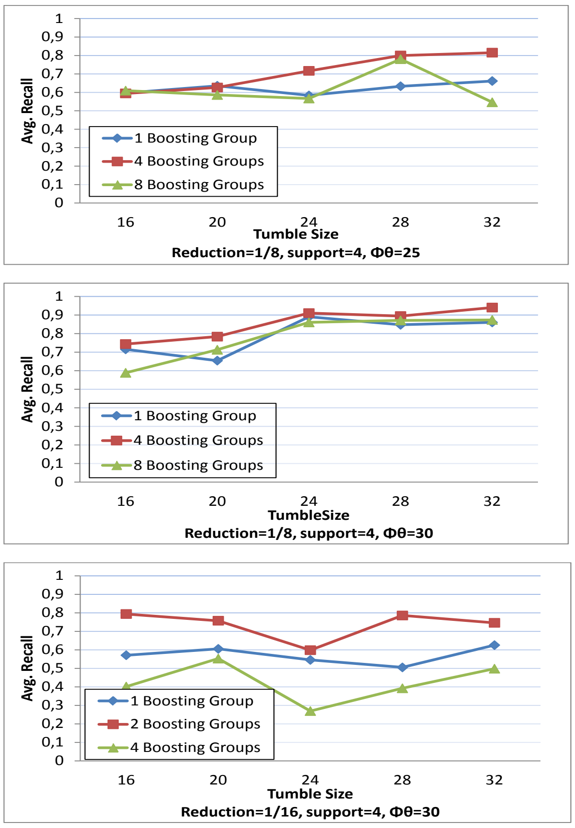


(a) Boosting Precision: Intel.Humidity (Φ_θ=25 & 30 degrees for 1/8 Reduction) and Intel.Temperature (Φ_θ=30 degrees for 1/16 Reduction) vs Tumble Size

Figure 5.8: Boosting Application on Intel datasets

model with more accurate submodels. Moreover, notice that using 8 groups may provide worse results since the number of bits per group becomes smaller, thus resulting in submodels that are prone to produce low quality similarity estimations. For the same reason, in the Intel temperature dataset shown in Figure 5.8, where the imposed reduction ratio is 1/16, employing 2 boosting groups provides better results compared to the 4 boosting group case. In the latter plot, deviations upon switching between tumble sizes may appear to be steeper (i.e. for tumble size 24), since the 1/16 reduction causes larger discontinuities when transiting from the continuous space to the hamming cube. Notice that in the Intel temperature dataset, the application of boosting has a marginal effect on the average precision (¡+4%). On the contrary, average recall values are skyrocketed with the improvement reaching a 30% percentage.

Taking one step further we extracted 95% confidence intervals for each tumble across multiple data sets. We omit the corresponding graphs, however, we note TACO



(b) Boosting Recall: Intel.Humidity (Φ_θ=25 & 30 degrees for 1/8 Reduction) and Intel.Temperature (Φ_θ=30 degrees for 1/16 Reduction) vs Tumble Size

Figure 5.8: Boosting Application on Intel datasets (cont.)

exhibits little deviations (± 0.04) from its average behavior in a tumble in all of the data sets.

5.8.3 Performance Evaluation Using TOSSIM

Due to limitations in the simulation environment of TOSSIM, we restricted our experimental evaluation to the Intel Lab data set. We used the default TOSH_DATA_LENGTH value set to 29 bytes and applied 1/4, 1/8 and 1/16 reduction ratios to the original binary representation of tumbles containing W=16 values each.

We measured the performance of our TACO framework against two alternative approaches. The first approach, termed as *NonTACO*, performed the whole intra- and inter-cluster communication procedure using the initial value vectors of motes "as is". In the TACO and NonTACO approaches, motes producing outlying values were identified in-network, following precalculated TSP paths, and were subsequently sent to

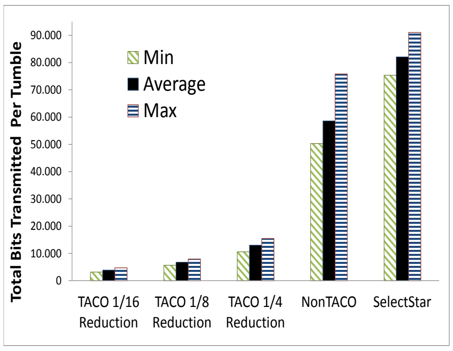


Figure 5.9: Total Bits Transmitted per approach

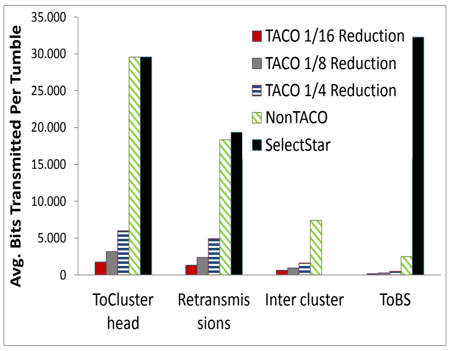


Figure 5.10: Transmitted bits categorization

the base station by the last clusterhead in each path. In the third approach, termed as *SelectStar*, motes transmitted original value vectors to their clusterheads and, omitting the intercluster communication phase, clusterheads forwarded these values as well as their own vector to the base station.

Besides simply presenting results involving these three approaches (TACO, NonTACO and SelectStar), we further seek to analyze their bandwidth consumption during the different phases of our framework. This analysis yields some interesting comparisons. For example, the number of bits transmitted during the intracluster phase of NonTACO provide a *lower bound* for the bandwidth consumption that a simple continuous aggregate query (such as MAX, MIN or SUM query) would require under TAG for all epochs, as this quantity: (a) Simply corresponds to transmitting the data observations of each sensor to one-hop neighbors (i.e., the clusterheads), and (b) Does not contain bandwidth required for the transmission of data from the clusterheads to the base station. Thus, if TACO requires fewer transmitted bits than the intracluster phase of NonTACO, then it also requires less bandwidth than a continuous aggregate query.

Note that in our setup for TACO, during the first tumble, the base station broadcasts a message encapsulating the parameters (W,d, seed etc) of the query. The overhead of transmitting these values is included in the presented graphs.

Figure 5.9 depicts the average, maximum and minimum number of total bits transmitted in the network in a tumble for the TACO (with different reduction ratios), NonTACO and SelectStar approaches. Comparing, for instance, the performance of the middle case of 1/8 Reduction and the NonTACO executions, we observe that, in terms of total transmitted bits the reduction achieved by TACO is on the average 1/9 per tumble, thus exceeding the imposed 1/8 reduction ratio. The same observation holds for the other two reduction ratios. This comes as no surprise, since message collisions entailing retransmissions are more frequent with increased message sizes used in NonTACO,

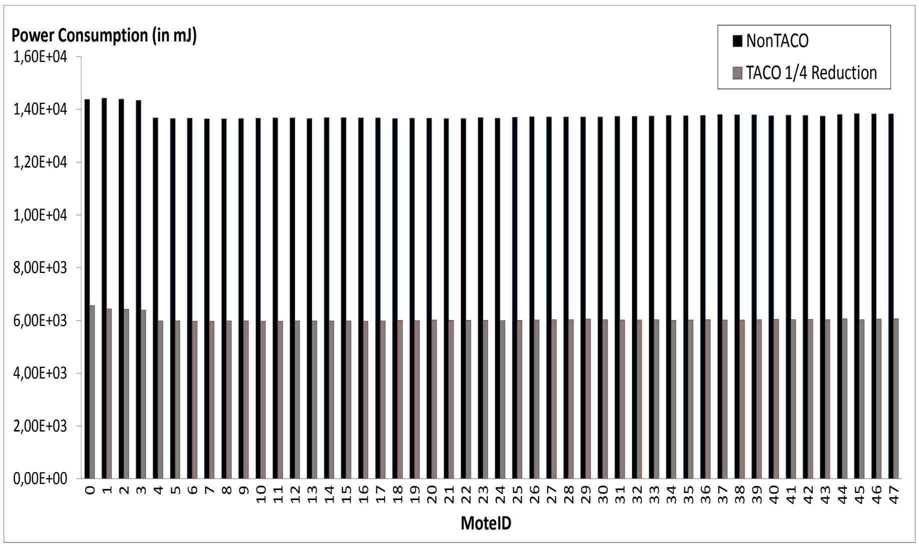


Figure 5.11: Power Consumption vs. MoteID

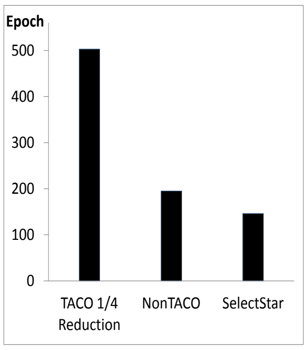


Figure 5.12: Average Lifetime

augmenting the total number of bits transmitted. Furthermore, comparing these results with the SelectStar approach exhibits the efficiency of the proposed inter-cluster communication phase for in-network outlier identification. The achieved reduction ratio of TACO 1/8 Reduction, when compared to the SelectStar approach is, on average 1/12, with a maximum value of 1/15. This validates the expected benefit derived by TACO.

Figure 5.10 presents a categorization of the average number of bits transmitted in a tumble. For each of the approaches, we categorize the transmitted bits as: (1) ToClusterhead: bits transmitted to clusterheads during the intra-cluster communication phase; (2) Intercluster: bits transmitted in the network during the inter-cluster communication phase (applicable only for TACO and NonTACO); (3) ToBasestation: bits transmitted from clusterheads towards the base station; (4) Retransmissions: additional bits resulting from message retransmission due to lossy communication channels or collisions. In Figure 5.10, please notice that the bits classified as Intercluster are always less than those in the ToClusterhead category. Moreover, the total bits of TACO (shown in Figure 5.9), are actually less than what NonTACO requires in its intracluster phase (Figure 5.10), even without including the corresponding bits involving retransmissions during this phase (73% of its total retransmission bits). Based on our earlier discussion, this implies that TACO under collisions and retransmissions is able to identify outlier readings at a fraction than what even a simple aggregate query would require.

Also, we used PowerTOSSIM [108] to acquire power measurements yielded during simulation execution. Figure 5.11 provides an additional quantitative representation of the energy savings provided by our framework, presenting the power consumption in motes for the TACO using a reduction ratio of 1/4, and NonTACO approaches (motes 0-3 in the Figure are clusterheads). To keep the graph readable we omit the Select-Star approach, which had a much larger energy drain. Overall, the TACO application reduces the power consumption up to a factor of 1/2.7 compared to the NonTACO approach. The difference between the selected reduction ratio (1/4) and the corresponding power consumption ratio (1/2.7) stems from the fact that motes need to periodically turn on/off their radio to check whether they are recipients of any transmission attempts. This fact mainly affects the TACO implementation since in the other two approaches, where more bits are delivered in the network, the amount of time that the radio remains turned on is indeed devoted to message reception. We leave the development of a more efficient transmission/reception schedule, tailored for our TACO scheme as future work.

As a final exhibition of the energy savings provided by our framework, in Figure 5.12 we used the previously extracted power measurements to plot the average network lifetime for motes initialized with 5000 mJ residual energy. Network lifetime is defined as the epoch on which the first mote in the network totally drains its energy. Obviously, network lifetime is proportional to the energy savings provided by the TACO approach compared to the other techniques.

5.8.4 TACO vs Hierarchical Outlier Detection Techniques

In the previous sections we experimentally validated the ability of our framework to tune the amount of transmitted data while simultaneously accurately predicting outliers. On the contrary, existing in-network outlier detection techniques, such as the algorithm of [30, 111] cannot tune the amount of transmitted information. Moreover, these algorithms lack the ability to provide guarantees since they both base their decisions on partial knowledge of recent measurements received by intermediate nodes in the hierarchy from their descendant nodes.

In this subsection, we perform a comparison to the recently proposed algorithm

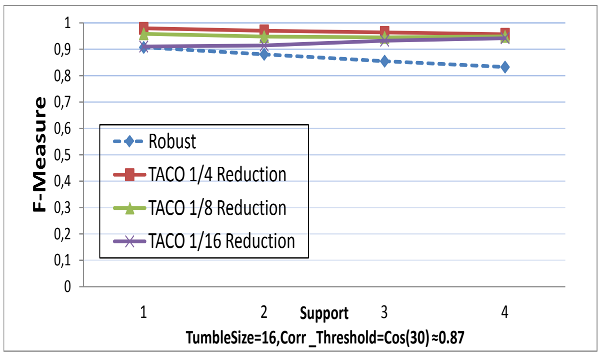


Figure 5.13: Intel.Temperature TACO vs Robust Accuracy varying $minSup$

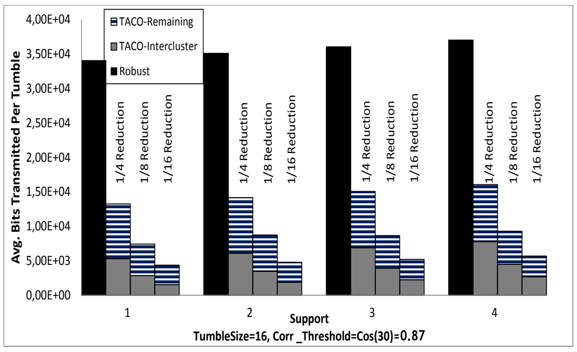


Figure 5.14: Intel.Temperature TACO vs Robust transmitted bits varying $minSup$

of [30], which we will term as *Robust*. We use *Robust* as the most representative example to extract comparative results related to accuracy and bandwidth consumption since it uses an equivalent outlier definition and bases its decisions on common similarity measures. As in the previous subsection, we utilized the Intel Lab data set in our study, keeping the TACO framework configuration unchanged.

In order to achieve a fair comparison, the Robust algorithm was simulated using a tree network organization of three levels (including the base station) with a $CacheSize = 24$ measurements. Note that such a configuration is a good scenario for Robust since most of the motes that can witness each other often share common parent nodes. Thus, the loss of witnesses as data ascend the tree organization is reduced. Please refer to [30] for further details.

In the evaluation, we employed the correlation coefficient-$corr$ (see Table 5.1) as a common similarity measure equivalent to the cosine similarity as mentioned in Section 5.6. We chose to demonstrate results regarding the temperature measurements in the data set. However, we note that the outcome was similar for the humidity data and proportional for different Φ_{corr} thresholds. Figure 5.13 depicts the accuracy of Robust compared to TACO with different reduction ratios varying the $minSup$ parameter. To acquire a holistic performance view of the approaches, we computed the *F-Measure* metric as *F-measure=2/(1/Precision+1/Recall).* Notably, TACO behaves better even for the extreme case of 1/16 reduction, while Robust falls short up to 10%. To complete the picture, Figure 5.14 shows the average bits transmitted by motes in the two different settings. Notice that the stacked bars in the TACO approach form the total number of transmitted bits which comprises the bits devoted to intercluster communication (*TACO-Intercluster*) and those termed as *TACO-remaining* for the remainder. The increment of the $minSup$ parameter in the graph correspondingly causes an increment in the *TACO-Intercluster* bits as more motes do not manage to find adequate support in their cluster and subsequently participate in the intercluster communication phase. TACO ensures less bandwidth consumption with a ratio varying from 1/2.6 for a reduction ratio of 1/4, and up to 1/7.8 for 1/16 reduction.

5.8.5 Bucket Node Exploitation

In order to better perceive the benefits derived from bucket node introduction, Table 5.3 summarizes the basic features ascribed to network clusters for different numbers B of bucket nodes. The table provides measurements regarding the average number of comparisons along with the average number of messages resulting from multi-hashed bitmaps. Moreover, it presents the average number of bitmaps received per bucket for different cluster sizes and Φ_θ thresholds. Focusing on the average number of comparisons per tumble (Comparisons in the Table), this significantly decreases as new bucket nodes are introduced in the cluster. From this point of view, we have achieved our goal since, as mentioned in Section 5.5.1, not only bucket nodes do alleviate the clusterhead from comparison load, but also the hash key space distribution amongst them preserves the redundant comparisons.

Studying the number of multi-hash messages (Multihash Messages in the Table) and the number of bitmaps received per bucket (Bitmaps Per Bucket) a trade-off seems to appear. The first column regards a message transmission cost mainly charged to the regular motes in a cluster, while the second involves load distribution between buckets. As new bucket nodes are adopted in the cluster, the Multihash Messages increases with a simultaneous decrease in Bitmaps Per Bucket. In other words, the introduction of more bucket nodes causes a shift in the energy consumption from clusterhead and bucket nodes to regular cluster motes. Achieving appropriate balance, aids in maintaining uniform energy consumption in the whole cluster, which in turn leads to infrequent network reorganization.

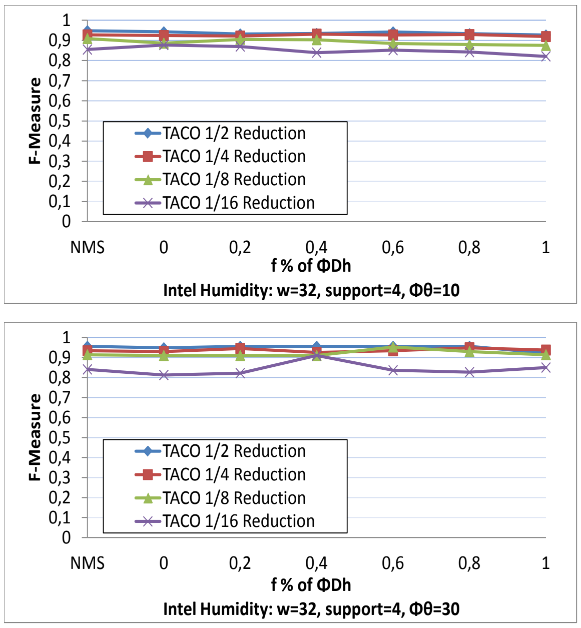
		Φ_θ					
		10			20		
Cluster Size	Buckets	Comparisons	Multihash Messages	Bitmaps Per Bucket	Comparisons	Multihash Messages	Bitmaps Per Bucket
12	1	66.00	0	12	66	0	12
	2	38.08	0.90	6.45	40.92	1.36	6.68
	4	24.55	7.71	3.65	30.95	8.88	4.08
24	1	276.00	0	24	276	0	24
	2	158.06	1.62	12.81	171.80	2.76	13.38
	4	101.10	14.97	7.27	128.63	17.61	8.15
36	1	630	0	36	630	0	36
	2	363.64	2.66	19.33	394.97	4.30	20.15
	4	230.73	22.88	10.88	291.14	26.28	12.19
48	1	1128	0	48	1128	0	48
	2	640.10	3.14	25.57	710.95	5.85	26.93
	4	412.76	30.17	14.49	518.57	34.64	16.21

Table 5.3: The effect of Bucket Nodes Introduction (W=16, d=128)

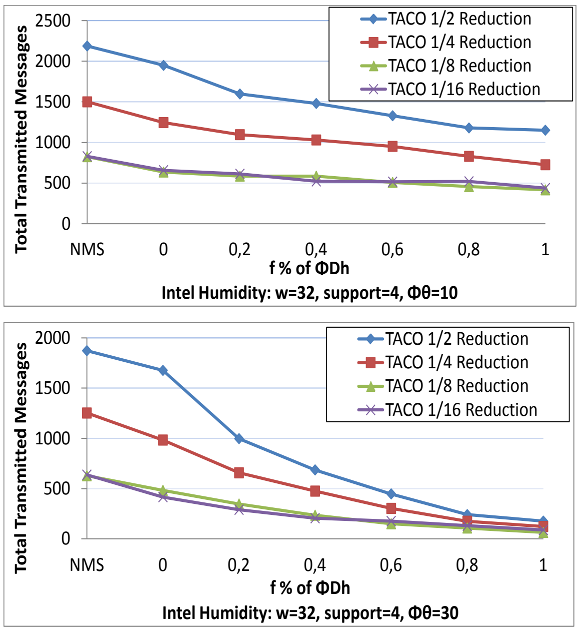
5.8.6 Message Suppression

Eventually, we study the effect of the message suppression strategy, introduced in Section 5.7, with respect to the accuracy it attributes to TACO as well as the reduction it yields on the amount of communicated data within the sensor network setting. Figure 5.15 plots corresponding experimental results for different reduction ratios utilizing the Intel Lab Humidity data as a representative example since the other datasets exhibit analogous behavior. We measured TACO's accuracy in terms of the F-measure metric (Section 5.8.4) and computed the total number of transmitted messages throughout the network operation varying the f value (which is expressed as a percentage of the respective Φ_{D_h} in the Figure) and the posed Φ_θ threshold. Particularly, we choose to present results where we set the tumble size equal to 32 since, given the default TOSH_DATA_LENGTH value of 29 bytes, the aforementioned size entails that in a single tumble motes are required to transmit 3 messages for 1/2 reduction, 2 messages for 1/4 reduction and a single message otherwise during both the intra- and inter-cluster communication. Consequently, we are able to acquire a well formed picture of the bandwidth consumption preservation provided under different circumstances.

Based on Figure 5.15(a), it can be observed that the accuracy of the framework under message suppression mostly remains unaffected for reduction ratios up to 1/8. On the other hand, in Figure 5.15(b) the decrease of message transmissions reaches a factor of 1/2, and 1/11 for Φ_θ =10 and 30 degrees, respectively. The sign *NMS* in the horizontal axis of the graphs expresses the case where No Message Suppression is applied. Moreover, notice that due to the fact that f is declared as a percentage of Φ_{D_h} the greater the angle threshold, the larger the number of suppressed messages. To sum up the above discussion, we mention that message suppression is proven to equip TACO with significantly increased communication savings without precluding



(a) TACO's Accuracy under Message Suppression



(b) TACO's Number of Transmitted Messages under Message Suppression

Figure 5.15: Message Suppression on Intel Humidity datasets

the accurate outlier identification as it does not exhibit high deviations from its *NMS* behavior.

5.9 Synopsis

In this chapter we presented TACO, a framework for detecting outliers in wireless sensor networks. Our techniques exploit locality sensitive hashing as a means to compress individual sensor readings and use a novel second level hashing mechanism to achieve intracluster comparison pruning and load balancing. TACO is largely parameterizable, as it bases its operation on a small set of intuitive application defined parameters: (i) the length of the LSH bitmaps (d), which controls the level of desired reduction; (ii) the number of recent measurements that should be taken into account when performing the similarity test (W), which can be fine-tuned depending on the application's desire to put more or less emphasis to past values; (iii) the desired similarity threshold (Φ); and (iv) the required level of support for non-outliers. Given the fact that TACO is not restricted to a monolithic definition of an outlier but, instead, it supports a number of intuitive similarity tests, the application can specialize and fine-tune the outlier detection process by choosing appropriate values for these parameters. We have also presented novel extensions to the basic TACO scheme that boost the accuracy of computing outliers. Our framework processes outliers in-network, using a novel intercluster communication phase. Our experiments demonstrated that our framework can reliably identify outlier readings using a fraction of the bandwidth and energy that would otherwise be required, resulting in significantly prolonged network lifetime.

In the next chapter we leverage properties of the similarity measures utilized for outlier detection in order to detect movement pattern alterations and extract semantic trajectories over spatiotemporal data streams of moving objects. In particular, based on the observation that measures like the correlation coefficient or the (equivalent) cosine similarity depend on the trends of the encapsulated data rather than their absolute values, we employ the $RV-$ coefficient measure which is a generalization of the correlation coefficient in order to detect homogeneous portions in the motion of monitored objects which are formed by identifying the division points where the motion pattern is altered. We first develop a centralized framework and subsequently echibit how our framework can be modified so as to function in a distributed manner. Unfortunately, LSH based data reduction cannot perform in that case for efficient distributed monitoring. However, we again manage to invent smart ways to reduce the amount of transmitted data controlling the accuracy of the semantic trajectory extraction process.

Chapter 6

Semantic Trajectory Extraction over Streaming Movement Data

6.1 Introduction

With the growth of location-based tracking technology like GPS, RFID and GSM networks, an enormous amount of trajectory data are generated from various real-life applications, including traffic management, urban planning and geo-social networks. A lot of studies have already been established on trajectories, ranging from data management to data analysis. The focus of trajectory data management includes building data models, query languages and implementation aspects, such as efficient indexing, query processing, and optimization techniques [49][90]. The analysis aims at trajectory data mining including issues like classification, clustering, outlier detection, as well as trajectory pattern discovery (e.g. sequential, periodic and convoy patterns) [41][60][74][77].

Recently, semantic trajectory extraction has attracted the research interest [1, 8, 110, 115, 117, 118]. The focus of semantic trajectory extraction is initially on the extraction of *meaningful* trajectories from the raw positioning data like GPS feeds. Moreover, sensory elements placed on vehicles can provide additional lower-scale information about their movement. Semantic trajectories manage to encompass both objects' spatiotemporal movement characteristics (at a certain level of abstraction) as well as useful information regarding objects' movement patterns (e.g dwelling, speeding, tailgating) and social activities (see Fig. 6.1) assigned to different time intervals throughout their lifespan. Current methods of such kind of trajectory extraction are mainly offline [1][8][110][115][117][118], which is not enough for modern, real life applications, because positioning data of moving objects are continuously generated

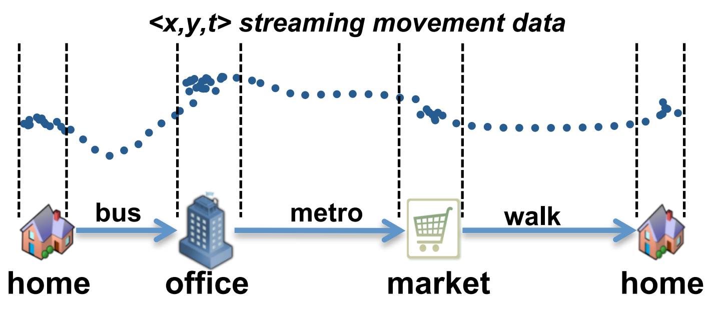


Figure 6.1: From streaming movement data to semantic trajectory

as streams and corresponding querying operations often demand result delivery in an online and continuous fashion.

Motivating Examples. Online semantic trajectory extraction can be useful in many traffic monitoring scenarios where authorities are interested in identifying apart from recent (i.e., within a restricted time window) objects' trajectory representation, the behavior of the drivers by posing queries of the form: *"Report every τ secs the movement and driving behavior of the objects within area A during the last T minutes"*. In that, authorities are able to continuously diagnosing streets where the density of vehicles whose drivers tend to have aggressive (speeding, tailgating, driving at the edges of the lanes etc.) behavior has recently become high, thus enabling suitable placement and periodic rearrangement of traffic wardens and patrol cars. As another example, state-of-the-art navigation services (http://world.waze.com/) provide the potential for combining traditional routing functionality with social networking facilities. Online semantic trajectory extraction allows users to acquire a compact picture of the movement and the social activities of interconnected friends around their moving area.

In this chapter, we introduce *SeTraStream* [116], a real-time platform that can progressively process raw mobility data arriving within a restricted time window and compute semantic-aware trajectories online. Before that, a number of data preparation steps need to be considered so as to render data easy to handle and ready to reveal profound movement patterns. The talk regards data cleaning and compression that precede the online segmentation and semantic trajectory extraction procedures. Data cleaning is dealing with trajectory errors, including systematic errors (outlier removal) and random errors (smooth noise) [80][117]. Compression considers data reduction because trajectory data grow rapidly and lack of compression sooner or later leads to exceeding system capacity [64][82]. Segmentation is used for dividing trajectories into episodes where each episode is in some sense homogeneous (e.g. sharing similar velocity, direction etc.) [8] and thus expresses unchanged movement pattern. Semantic computation can further extract high-level trajectory concepts like stops/moves [110], and even provide additional tagging support like the activity for stops (e.g. home, office, shopping) and the transportation mode (e.g. metro, bus, walking) for moves [1][115][117][127].

Challenges. It is non-trivial to establish a real-time semantic trajectory extraction platform. There exist new technical challenges compared to the existing offline solutions: (1) *Efficient Computation:* Large amounts of movement data are generated continuously, therefore we need to come up with more efficient algorithms which can handle different levels of trajectories in an acceptable time – including all data processing aspects like data cleaning, compression, segmentation, and semantic tagging; (2) *Suitable Trajectory Segmentation Decision Making:* Algorithms in offline trajectory extraction typically tune a lot of thresholds placed on movement features (like *acceleration, direction alteration, stop duration etc.*) to find their most suitable values, sometimes in a per object fashion. However, in the real-time context the movement attribute distribution may tremendously vary over time and continuous parameter tuning is prohibitive for real-time semantic trajectory extraction. Thus, suitable techniques should not rely on many predefined thresholds on certain movement features but instead consider pattern alterations during the trajectory computation process. (3) *Semantic Trajectory Tagging:* After trajectory segmentation, the outcomes should provide the potentials for semantic tags to be explored, e.g. characterization of the activity (shopping, work) or means of movement that is taking place in episodes (e.g. car, metro, bus in Fig. 6.1).

The rest of the chapter proceeds as follows. Section 6.2 describes the preliminaries for semantic trajectory extraction in SeTraStream, while in section 6.3 we present the data preparation procedures regarding incoming data cleaning and compression. In Section 6.4 we present SeTraStream's online segmentation algorithms and in Section 6.5 we experimentally evaluate our techniques. Eventually, section 6.6 includes extensions of the framework to distributed streaming settings.

6.2 Preliminaries

6.2.1 Data and Semantic Trajectory Models

In our setting, a central server continuously collects the status updates of moving objects that move inside an area of interest – monitoring area of moving objects. First, such updates involving an object O_i contain spatiotemporal $\langle x, y, t\rangle$ points forming its "*Raw Location Stream*".

Definition 6.2.1 (Raw Location Stream)**.** *The continuous recording of spatiotemporal points that update the status of a moving object O_i, i.e. $\langle Q_1^{\ell s}, Q_2^{\ell s}, \ldots, Q_n^{\ell s}\rangle$, where $Q_i^{\ell s} = \langle x, y, t\rangle$ is a tuple including moving object's O_i, position $\langle x, y\rangle$ and timestamp t.*

By means of the raw location streams, we can derive information of movement features such as acceleration, speed, direction etc., which make up a "*Location Stream Feature Vector*" ($Q^{\ell f}$). Moreover, depending on the application, updates include additional attributes such as heading, steering wheel activity, lane position, distance to headaway vehicle (e.g to assess tailgating), displacement and so on. These features formulate a "*Complementary Feature Vector*" (Q^{cf}). Consequently, the two types of feature vectors combined together are forming the "*Movement Feature Vector*" ($\mathcal{Q} = \langle Q^{\ell f}, Q^{cf} \rangle$) of d dimension describing d attributes of O_i's movement at a specific timestamp.

Definition 6.2.2 (Movement Feature Vector)**.** *The movement attributes of object O_i at timestamp t can be described by a d-dimensional vector that is the concatenation of the location stream feature vector and the complementary feature vector $\mathcal{Q} = \langle Q^{\ell f}, Q^{cf} \rangle$.*
– Location Stream Feature Vector $(Q^{\ell f})$: The movement features of object O_i that can be derived from the raw location stream tuple $Q^{\ell s}$.
– Complementary Feature Vector (Q^{cf}): The movement features that cannot be derived from the location stream but are explicitly included in O_i's status updates.

To provide better understanding and mobility data abstraction, in [110][117] the concept of *semantic trajectories* is introduced, where the trajectory is thought of as a sequence of meaningful episodes (e.g. stop, move, and other self-contained and self-correlated trajectory portions).

Definition 6.2.3 (Semantic Movement)**.** *A semantic movement or trajectory consists of a sequence of meaningful trajectory units, called "episodes", i.e. $\mathcal{T}_{sem} = \{e_{first}, \ldots, e_{last}\}$.*
– An episode (e) groups a subsequence of the location stream (a number of consecutive $\langle x, y, t \rangle$ points) having similar movement features.
– From a semantic data compression point of view, an episode stores the subsequence's temporal duration as well as its spatial extent $e_i = (time_{from}, time_{to}, geometry_{bound}, tag)$.

The $geometry_{bound}$ is the geometric abstraction of the episode, e.g. the bounding box of a stop area or the shape trace of roads that the moving object has followed. The term *tag* in the last part of the previous definition refers to the semantics of the episode, i.e. characterization of the activity or means of movement that is taking place in an episode (see Fig. 1).

6.2.2 Window Specifications

In our context, the time window size T expresses the most recent portion of semantic trajectories the server needs to be informed about. An additional parameter τ specifies a time interval in which client side devices, installed on moving objects, are required

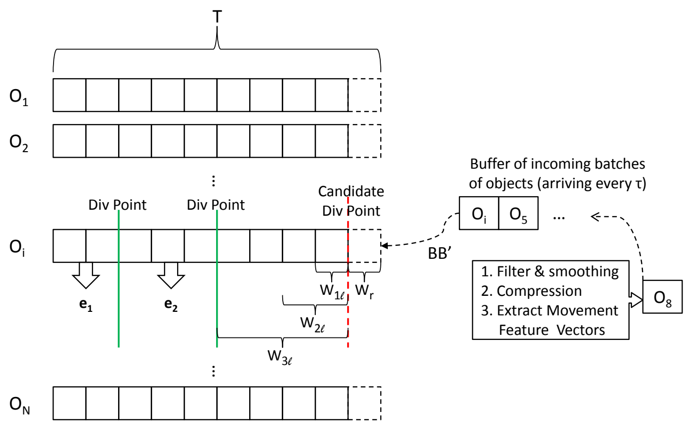


Figure 6.2: The SeTraStream Framework

to collect and report batches of their time ordered status updates [40]. Thus, $\frac{T}{\tau}$ batches are included in the window. Obviously, posed prerequisites are: 1) $\tau \ll T$ and 2) $T \bmod \tau = 0$. As the window slides, for each monitored object O_i, the most aged batch expires and a newly received one is appended to it. The size of τ may vary from a few seconds to minutes depending on the application's sampling frequency. Small τ values enable fine-tuned episode extend determination with the make-weight of increased processing costs, while larger τ values reduce the processing load by increasing the granules that are assigned to episodes.

6.2.3 SeTraStream Overview

Having presented the primitive concepts utilized by our framework, in this subsection we outline SeTraStream's general function. Details will be provided in the upcoming sections. The whole process is depicted in Fig. 6.2. Upon the receipt of a batch containing the status updates including $Q^{\ell s}, Q^{cf}$ vectors at different timestamps in τ, a cleaning and smoothing technique is applied on it (Step 1 on the right part of the figure). Consequently, a novel compression method (Step 2) is applied on the batch considering both $Q^{\ell s}, Q^{cf}$ characteristics while performing the load shedding. Finally, at a third step $Q^{\ell f}, Q^{cf}$ feature vectors are extracted, a corresponding matrix is formed and the batch is buffered until it is processed at the SeTraStream's segmentation stage. During the segmentation stage (left part of Fig. 6.2), a previously buffered batch is dequeued and compared with other batches' feature matrices in O_i's window. SeTraStream seeks both for short and long term changes in O_i's movement pattern, and identifies an episode whenever feature matrices are found to be dissimilar based on the RV-Coefficient (to be defined later) and a specified division threshold σ.

Symbol	Description
N	Number of monitored objects
T, τ	Window size and batch interval
d	Number of movement features
O_i	The i-th monitored object id
B_i	The i-th batch from a candidate div. point
$Q^{\ell s}$	Tuple including $\langle x, y, t\rangle$ triplet of an objects' raw location stream
$Q^{\ell f}$	Feature vector derived from the raw location stream at t
Q^{cf}	Complementary feature vector at timestamp t
$\delta_{outlier}, \delta_{smooth}, \sigma$	Filtering, smoothing and segmentation thresholds respectively
res	The residual between the smoothed and the true value
sed, scc	Synchronous Euclidean Distance and Correlation Coefficient
W_ℓ, W_r	A left and right workpiece respectively
e_i	The i-th episode in an object's window

Table 6.1: Notation of Chapter 6

6.3 Online Data Preparation

As already described, arriving batches involving monitored objects contain their raw location stream, as well as complementary feature vectors. In this section, we discuss the initial steps of data preparation before proceeding to episode determination (i.e. trajectory segmentation). The talk regards three steps depicted in the right part of Fig. 6.2: (1) an *online cleaning* step that deals with noisy tuples, (2) an *online compression* stage that manages to reduce both the available memory usage and the processing cost in computing trajectories, and (3) extracting movement feature vectors, including both the location stream features and complementary features. Table 6.1 summarizes the symbology utilized in the current and the upcoming sections as well.

6.3.1 Online Cleaning

The main focus of trajectory data cleaning is to remove GPS errors. Jun et al. [61] summarize two types of GPS errors: *systematic errors* (i.e. the totally different GPS positioning from the actual location which is caused by low number of satellites in view, Horizontal Dilution Of Position HDOP etc.) and *random errors* (i.e. the small errors up to ± 15 meters which can be caused by the satellite orbit, clock or receiver issues). These systematic errors are also named "outliers", where researchers usually design *filtering* methods to remove them; whilst random errors are small distortions from the true values and their influences can be decreased by *smoothing* methods. Many offline GPS data cleaning works can be found such as [61][101][117].

In the context of streaming data, *online filtering & smoothing* of streaming tuples has become a hot topic [31][43][44][62][71]. Different from the focus of prior works on data accuracy and distribution estimation, our primary concern of cleaning such streaming movement data is refining the data points that have substantial distortion of movement features for computing semantic trajectories[1].

For efficient data cleaning, we need to combine *online filtering* and *online smoothing* in a single loop. When a new batch B regarding object O_i arrives (right part of Fig.6.2), we do the following cleaning steps:

1. Build a kernel based smoothing model: $(\widehat{x}, \widehat{y}) = \frac{\sum_i k(t_i)(x_{t_i}, y_{t_i})}{\sum_i k(t_i)}$ where $k(t)$ is a function with the property $\int_0^{|B|} k(t)dt = 1$. The kernel function describes the weight distribution, with most of the weight in the area near the point. In our experiments, as in [101], we apply the Gaussian kernel $k(t_i) = e^{-\frac{(t_i - t)^2}{2ß^2}}$, where ß refers to the bandwidth of the kernel.

2. Calculate the residual between the model prediction and the true value $\langle x, y \rangle$ of the examined point $Q_p^{\ell s}$, i.e. $res = \sqrt{(\widehat{x} - x)^2 + (\widehat{y} - y)^2}$.

3. By using a speed limit v_{limit} and the speed $v_{Q_{p-1}^{\ell s}}$ at the previous point $Q_{p-1}^{\ell s}$, respectively compute the outlier bound ($\delta_{outlier} = v_{limit} \times (t_{Q_p^{\ell s}} - t_{Q_{p-1}^{\ell s}})$) and the smooth bound ($\delta_{smooth} = v_{Q_{p-1}^{\ell s}} \times (t_{Q_p^{\ell s}} - t_{Q_{p-1}^{\ell s}}) \times 120\%$[2]).

4. Filter out the point if the residual is more than the outlier bound, i.e. $res > \delta_{outlier}$, or replace the location of the point $\langle x, y \rangle$ with the smoothed value $\langle \widehat{x}, \widehat{y} \rangle$ if the residual is between the outlier bound and the smooth bound, i.e. $\delta_{smooth} < res < \delta_{outlier}$. Otherwise, we keep the original $\langle x, y \rangle$ of the point.

This cleaning method has taken both advantages of the distance based outlier removal and the local-weighted kernel smoothing method with linear memory requirements of $O(|B|)$, where $|B|$ is the size of a batch.

6.3.2 Online Compression

A primary concern when operating in a streaming setting regards the load shedding with respect to incoming tuples. In the context of semantic trajectory extraction, this happens both for limiting the available buffer usage as well as to reduce the processing cost [10][64][82][95]. In our approach, as both Definitions 6.2.2, 6.2.3 imply, the approximation quality of the mere spatiotemporal trajectories is not our only concern. Semantic trajectories will be extracted based on additional features other than those derived from spatiotemporal $\langle x, y, t \rangle$ points. On the other hand, if we overlook the spatiotemporal trajectory approximation quality, the portion of the movement features that rely on the pure location stream will later be uncontrollably distorted. To cope with the

[1] Q^{cf} values are not examined as the micro-sensory devices of vehicles usually possess self-calibrating capabilities.

[2] Here, we increase the smooth bound by 20% of the location prediction provided by the speed of the previous point.

previous requirements, we propose a method and define a *significance score* suitable to serve our purposes.

Assume that a batch regarding object O_i is processed (step. 2 at right part of Fig.6.2) and $(Q^{\ell s}_{p-1}, Q^{\ell s}_{p})$ is the last examined pair of points in it. When a new point $Q^{\ell s}_{p+1}$ is inspected, we first obtain the significance of $Q^{\ell s}_{p}$ from a spatiotemporal viewpoint by fostering the *Synchronous Euclidean Distance*, defined as [82][95]:

$$sed(Q^{\ell s}_{p}, Q^{\ell s}_{p-1}, Q^{\ell s}_{p+1}) = \sqrt{(x_{Q'^{\ell s}_p} - x_{Q^{\ell s}_p})^2 + (y_{Q'^{\ell s}_p} - y_{Q^{\ell s}_p})^2}$$

with $x_{Q'^{\ell s}_p} = x_{Q^{\ell s}_{p-1}} + v^x_{Q^{\ell s}_{p-1}Q^{\ell s}_{p+1}} \cdot (t_{Q^{\ell s}_p} - t_{Q^{\ell s}_{p-1}})$ and $y_{Q'^{\ell s}_p} = y_{Q^{\ell s}_{p-1}} + v^y_{Q^{\ell s}_{p-1}Q^{\ell s}_{p+1}} \cdot (t_{Q^{\ell s}_p} - t_{Q^{\ell s}_{p-1}})$ while v^x, v^y refer to the velocity vector (please refer to [95] for further details).

The above measure is also employed in the sampling based approach of [95]. Nevertheless, *sed* constitutes an absolute number that lacks the ability to quantify the particular significance of a point with respect to other spatiotemporal points within the current batch. In order to appropriately derive the aforementioned significance quantification, in SeTraStream's compression scheme we normalize sed and define the relative spatiotemporal significance Sig^{SP}:

$$Sig^{SP}(Q^{\ell s}_p) = \frac{sed(Q^{\ell s}_{p}, Q^{\ell s}_{p-1}, Q^{\ell s}_{p+1})}{max_{sed}} \tag{6.1}$$

with $0 \leq Sig^{SP}(Q^{\ell s}_p) \leq 1$. The denominator max_{sed} denotes the current maximum *sed* of points in the batch. Obviously, increased $Sig^{SP}(Q^{\ell s}_p)$ estimations represent points of higher spatiotemporal significance.

Carefully inspecting sed's formula, we can conceive that the intuition behind its definition is to measure the amount of distortion that can be caused by pruning the spatiotemporal point $Q^{\ell s}_p$. That is, having omitted $Q^{\ell s}_p$ we could virtually infer the respective data point at timepoint $t_{Q^{\ell s}_p}$ using the preceding and succeeding ones $(Q^{\ell s}_{p-1}, Q^{\ell s}_{p+1})$. And calculating $Q'^{\ell s}_p$, $sed(Q^{\ell s}_{p}, Q^{\ell s}_{p-1}, Q^{\ell s}_{p+1})$ measures the incorporated distortion.

Thus, as regards the complementary feature vectors of O_i we choose to base the measure of their significance on the *Correlation Coefficient (corr)* metric. First, fostering an attitude similar to that in sed's calculation as explained in the previous paragraph, we estimate the value at the i-th position of vector Q'^{cf}_p as: $[Q'^{cf}_p]_i = [Q^{cf}_{p-1}]_i + \frac{[Q^{cf}_{p+1}]_i - [Q^{cf}_{p-1}]_i}{t_{Q^{cf}_{p+1}} - t_{Q^{cf}_{p-1}}}(t_{Q^{cf}_p} - t_{Q^{cf}_{p-1}})$. Then, based on $corr$ we define the *Synchronized Correlation Coefficient (scc)* between (Q'^{cf}_p, Q^{cf}_p) of complementary feature vectors:

$$scc(Q'^{cf}_p, Q^{cf}_p) = \frac{E(Q'^{cf}_p Q^{cf}_p) - E(Q'^{cf}_p)E(Q^{cf}_p)}{\sqrt{(E((Q'^{cf}_p)^2) - E^2(Q'^{cf}_p))(E((Q^{cf}_p)^2) - E^2(Q^{cf}_p))}} \tag{6.2}$$

where $E()$ refers to the mean and $-1 \leq scc(Q_p'^{cf}, Q_p^{cf}) \leq 1$.

The choice of scc is motivated by the fact that, as we pointed out in Chapter 5, its stem, $corr$, possesses the ability to indicate the similarity of the trends that are profound in the examined vectors rather than relying on their absolute values [31][43][44][71]. Hence, it provides an appropriate way to identify (dis)similar patterns in the complementary vectors and can be generalized in order to detect similar patterns between movement feature vectors in their entirety. Values of scc that are close to -1 exhibit high dissimilarity between $(Q_p'^{cf}, Q_p^{cf})$, indicating that omitting Q_p^{cf} results in higher pattern distortion. Calculating $1 - scc$ enables higher measurements to account for more dissimilar patterns and taking one step further, min-max normalization on $1 - scc$ allows (dis)similarity values lie within $[0, 1]$. Thus, we eventually compute the relative significance of the complementary feature vector:

$$Sig^C(Q_p^{cf}) = \frac{1 - scc(Q_p'^{cf}, Q_p^{cf})}{2max\{(1 - scc)\}} \tag{6.3}$$

In the context of our compression scheme, the more dissimilar $(Q_p'^{cf}, Q_p^{cf})$ are, the higher the probability to be included in the window should be. As a result, the overall significance $Sig(Q_p)$ of Q_p can be estimated by the combination of both the location stream feature $Sig^{SP}(Q_p^{\ell s})$ and the complementary feature $Sig^C(Q_p^{cf})$. The weight balance between them is application dependent, though we choose to treat them equally important [74]:

$$Sig(Q_p) = \frac{1}{2}(Sig^{SP}(Q_p^{\ell s}) + Sig^C(Q_p^{cf})) \tag{6.4}$$

Eventually, for a threshold $0 \leq Sig_{thres} \leq 1$, Q_p remains in the batch when $Sig(Q_p) \geq Sig_{thres}$, or it is removed for compression purposes otherwise.

6.4 Semantic Trajectory Extraction

We now describe the core of SeTraStream, the online trajectory segmentation stage. This stage comes after data cleaning and compression utilizing the *extracted feature vectors* of a batch (step. 3 at right part of Fig.6.2).

6.4.1 Online Episode Determination - Trajectory Segmentation

Upon deciding the data points of a batch that are to be included in the window as devised in the previous subsection, SeTraStream proceeds by examining episode existence in T. To start with, we assume the simple case of the current window consisting of a couple of τ-sized batches (i.e. $T = 2\tau$). We will henceforth refer to each part of the window composed of a number of compressed batches as *workpiece*. Intuitively, distinguishing episodes is equivalent to finding a *division point*, where the movement feature vectors on its left and right sides are uncorrelated and thus correspond to different movement patterns. In our simple scenario, a *candidate division point* is placed in the middle of the available workpieces.

Hence, we subsequently need to dictate a suitable measure in order to determine movement pattern change existence. We already noted the particular utility of the correlation coefficient on the discovery of trends [31][43][44][71], and thus (in our context) patterns in the movement data. In this processing phase movement feature vectors composing each workpiece essentially form a pair of matrices for which correlation computation needs to be conducted. As a result, we will reside to the *RV-coefficient* which constitutes a generalization of the correlation coefficient for matrix data. We organize W_ℓ into a $d \times m$ matrix, where d is the number of movement features and m represents a number of vectors (at different timestamps) that are the columns of the matrix. Similarly, W_r is organized in a $d \times n$ matrix i.e. n columns exist. The *RV-Coefficient* between $\langle W_\ell, W_r \rangle$ is defined as:

$$RV(W_\ell, W_r) = \frac{Tr(W_\ell W_\ell' W_r W_r')}{\sqrt{Tr([W_\ell W_\ell']^2) Tr([W_r W_r']^2)}} \tag{6.5}$$

where W_ℓ', W_r' refer to the transpose matrices, $Tr()$ denotes the trace of a matrix and $0 \leq RV \leq 1$. RV values closer to zero are indicative of uncorrelated movement patterns.

Based on a division point threshold σ workpieces W_ℓ, W_r can be assigned to a pair of different episodes $e_\ell = (0, T-\tau, geometry_{bound})$, $e_r = (T-\tau+1, T, geometry_{bound})$ when:

$$RV(W_\ell, W_r) \leq \sigma \tag{6.6}$$

or to a single episode $e = (0, T, geometry_{bound})$ otherwise.

An interesting observation is that the RV coefficient can be equivalently expressed as a cosine similarity, which was set as the similarity measure - monitored function on which we focused in Chapter 5. Let Vec be an isomorphism such that $Vec : R^{d \times n} \rightarrow R^{d \cdot n}$. This essentially is a linear transformation that renders the matrix of a batch to vectors by stacking its columns. Then $Tr(W_\ell W_\ell' W_r W_r') = Vec(W_\ell W_\ell') \cdot Vec(W_r W_r')$. Moreover, $\sqrt{Tr([W_\ell W_\ell']^2)} = ||Vec(W_\ell W_\ell')||$, where $||.||$ again refers to the L_2 norm of the formed vector and similarly for $W_r W_r'$. Overall, we can manage to express the RV coefficient as a cosine similarity:

$$RV(W_\ell, W_r) = \frac{Tr(W_\ell W_\ell' W_r W_r')}{\sqrt{Tr([W_\ell W_\ell']^2) Tr([W_r W_r']^2)}} =$$

$$= cos(\frac{Vec(W_\ell W_\ell') \cdot Vec(W_r W_r')}{||Vec(W_\ell W_\ell')|| \cdot ||Vec(W_r W_r')||}) \tag{6.7}$$

However, as we are going to present in the distributed version of our framework in Section 6.6, the LSH reduction scheme cannot be applied for this type of vectors.

Now, consider the general case of T covering an arbitrary number of batches. It can easily be conceived that in a larger time window an alteration in the movement pattern may happen: (a) instantly as a sharp change, or (b) in a more smooth manner as time passes. As a result, upon the arrival of a new workpiece W_r, we initially check for short-term changes in the patterns of movement. We thus place a candidate division point between the newly received workpiece and the last of the existing ones. Then the correlation between the movement feature vectors present in $\langle W_{1\ell}, W_r \rangle$ is computed. Notice that $W_{1\ell}$ this time possesses an additional subscript which denotes the step of the procedure, as will be shortly explained. Similarly to our discussion in the previous paragraphs, when $RV_1(W_{1\ell}, W_r)$ is lower than the specified division threshold, a division point exists and signals the end of the previous episode e_ℓ and starts a new one e_r.

No short-term change existence triggers our algorithm to proceed by seeking long-term dis-correlations. For this purpose, we first examine $RV_2(W_{2\ell}, W_r)$ doubling the time scale of the left workpiece by going 2τ units back in the window from the candidate division point. In case RV_2 does not satisfy Inequality 6.6, this procedure continuous by *exponentially expanding* the time scale of the left workpiece in a way such that at the i-th step of the algorithm the size of $W_{i\ell}$ is $2^{(i-1)}\tau$ units and $RV_i(W_{i\ell}, W_r)$ is calculated. When Inequality 6.6 is satisfied the candidate division point is a true division point which bounds the previous episode $e_i = (time_{from}, time_{to}, geometry_{bound})$ and constitutes the onset of a new. Otherwise, W_r is rendered the current bound of the last episode by being appended to it. If no long-term change is detected, the aforementioned expansion ceases when either the beginning of the last episode or the start of T (in case all previous batches have been attributed to the same episode) is reached, i.e. no data points of the penultimate episode are considered since its extend has already been determined.

The exponential workpiece expansion fostered here is inspired by the *tilted time window* definition [40] as a general and rational way to seek movement pattern changes in different time granularities. Other expansion choices can also be applied. All of these options are orthogonal to our approaches and do not affect the generic function of SeTraStream. Our approach manages to effectively handle sliding windows as a slide of τ time units results in: (1) the expiration of the initial batch of the first episode e_{first} of O_i which affects its $(time_{from}, geometry_{bound})$ attributes and (2) the appendage of a newly received batch that either extends the last episode e_{last} (when no division point is detected) or starts a new episode. The outcome of the online segmentation consists of tuples $\mathcal{T}_{O_i} = \{e_{first}, \ldots, e_{last}\}$ representing objects' semantic trajectories.

6.4.2 Time and Space Complexity

The introduced trajectory segmentation procedure, premises that a newly appended batch will be compared with left workpieces that may be (depending on whether a division point is detected) exponentially expanded until either the previous episode end or the start of the window is reached. Based on this observation, the lemma below elaborates on the complexity of the checks required during candidate division point examination.

Lemma 6.4.1. *The time complexity of SeTraStream's online segmentation procedure, for N monitored objects, under exponential $W_{i\ell}$ expansion is $O(Nlog_2(\frac{T}{\tau}))$ per candidate division point.*

Proof. For a single monitored object, the current window is composed of $\frac{T}{\tau} - 1$ batches (excluding the one belonging to W_r). The worst case scenario appears when no previous episode exists in the window and the candidate division point is not proven to be an actual division point. By considering the exponential workpiece expansion, comparisons (i.e., σ checks) may reach a number of $k = min\{i \in \mathbb{N}^* : \frac{\frac{T}{\tau}-1}{2^{(i-1)}} \geq 1\}$ at most. Adopting logarithms on the previous expression and summing for N objects completes the proof. □

Now, recalling the definition of the *RV-Coefficient* measure, it can easily be observed that its computation relies on the multiplication of the bipartite matrices with their transpose. Assume that the number of d-dimensional movement feature vectors in a cleaned and compressed batch are n. Based on the above observation we can see that instead of maintaining the original form of the vectors which requires $O(d \cdot n)$ memory space, we can reduce the space requirements during episode determination by computing the product of the $d \times n$ matrix of the batch with its transpose. This reduces the space requirements to $O(d^2)$ per batch since in practice $d \ll n$. So, to check a short-term change in the movement patterns we do not need to store the full matrices of $W_{1\ell}, W_r$ which in this case are composed of one batch each, but only the matrix products as described above.

However, this point may not be of particular utility since left workpieces are expanded during the long-term pattern alteration checks. A natural question that arises regards whether or not the product $W_{i\ell}W'_{i\ell}$ can be expressed by means of the multiplication of single batch matrices, with their transposes.

Lemma 6.4.2. *$W_{i\ell}W'_{i\ell}$ is the sum of batch matrix products with their transposes: $W_{i\ell}W'_{i\ell} = \sum_{j=1}^{2^{(i-1)}} B_j B'_j$, where B_j is used to notate the matrix formed by the vectors in the j-th batch (from a candidate division point to the end of $W_{i\ell}$).*

Proof. Let $W_{i\ell} = [B_1|B_2|\cdots|B_{2^{(i-1)}}]$ the matrix of the (i-th) left workpiece during the current division point check. B_js are used to denote sub-matrices belonging to individual batches that were appended to the workpiece. It is easy to see that the transpose matrix can be produced by transposing these submatrices: $W'_{i\ell} = [B'_1|B'_2|\cdots|B'_{2^{(i-1)}}]$. And then $W_{i\ell}W'_{i\ell}$ can be decomposed into $B_jB'_j$ products: $W_{i\ell}W'_{i\ell} = B_1B'_1+B_2B'_2+\cdots+B_{2^{(i-1)}}B'_{2^{(i-1)}} = \sum_{j=1}^{2^{(i-1)}} B_jB'_j$ □

Thus, for each batch we only need to store a square $d\times d$ matrix[3], which determines the space complexity of online segmentation leading to Lemma 3.

Lemma 6.4.3. *During the online episode determination stage of SeTraStream, the memory requirements per object O_i are $O(d^2\frac{T}{\tau})$ and assuming N objects are being monitored the total space utilization is $O(d^2N\frac{T}{\tau})$.*

6.4.3 Episode Tagging

Having detected an episode e_i, SeTraStream manages to specify in an online fashion the triplet $(time_{from}, time_{to}, geometry_{bound})$ describing its spatio-temporal extend. The final piece of information associated with an episode regards its tag as it was described in Section 6.2.1. Given application's context, possible tag instances form a set of movement pattern classes and notice that the instances of the classes are predetermined for the applications we consider (Section 6.1). Hence, the problem of episode tag assignment can be smelted to a trivial classification task, where the classifier can be trained in advance based on the collected episodes (with features like segment *distance*, *duration*, *density*, *avg. speed*, *avg. acceleration*, *avg. heading* etc.) and the detected episode e_i can be timely classified based on the trained model and the episode features. Suitable techniques include decision trees, boosting, SVM, neural or Bayesian networks [36]. Additional Hidden Markov Model based trajectory annotation can be referred to [115].

6.5 Experiments

In this section, we present our experimental results in real-time extraction of semantic trajectories from streaming movement data.

Experimental Setup. We utilize two different datasets: *Taxi Data* - this dataset includes taxi trajectory data for 5 months with more than 3M GPS records, which do not have any complementary features. We mainly use taxi data to validate compression. It is non-trivial to get real-life on-hand dataset with both complementary features and

[3]We also keep the geometry bound of the batch that is utilized in the final episode geometry bound determination as well as some additional aggregate statistics, of minor storage cost, for classification and tag assignment in the next step.

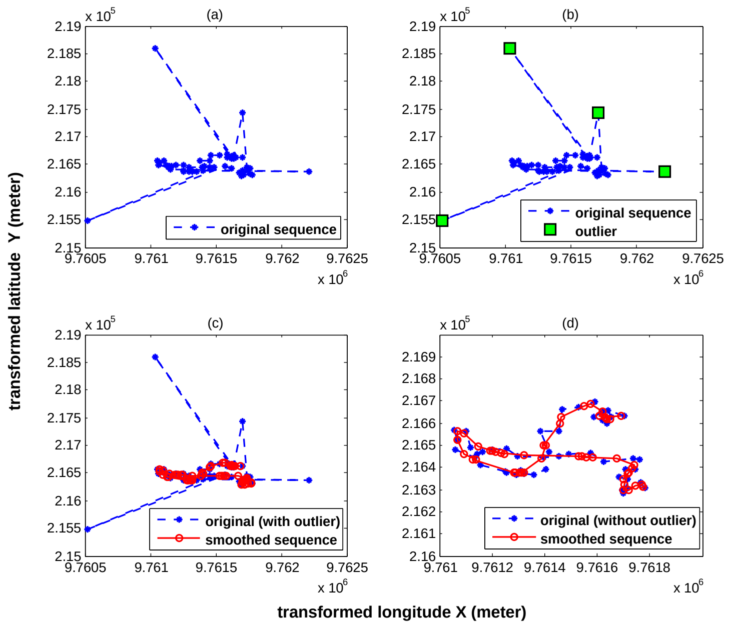


Figure 6.3: Data cleaning (outlier removal and smoothing)

the underlying segment ground-truth tags. Therefore, we collect our own trajectory data by developing Python S60 scripts deployed in a Nokia N95 smartphone, which can generate both GPS data and accelerometer data from the embedded sensors. We calculate GPS features (e.g. *transformed longitude, latitude, speed, direction*) as the location stream vectors ($Q^{\ell f}$) and accelerometer features (e.g. *mean, variance, magnitude, covariance of the 3 accelerometer axis*) as the complementary feature vectors (Q^{cf}). We term the latter dataset as *Phone Data* within which, we also provide our own real segment tags (e.g. *standing, jogging, walking*) to validate the online segmentation accuracy. For *Phone Data*, we also work on the GPS data from the data campaign organized by Nokia Research Center - Lausanne, which has collected 185 users' phone data with about 7M records in total [69][115].

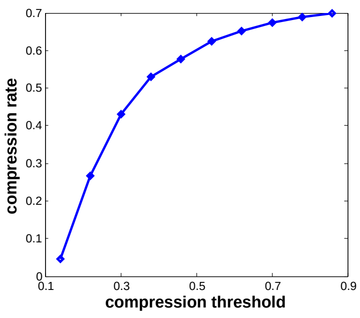


Figure 6.4: Data compression rate w.r.t. different thresholds $Sig^{SP}(Q_p^{\ell s})$

Data Cleaning. As described previously, our online data cleaning needs to consider two types of GPS data errors, i.e. filtering outliers as systematic errors and smoothing the random errors. The experimental cleaning results are shown in Fig. 6.3: (a) sketches the original trajectory data; (b) identifies the outliers during the online cleaning process; (c) and (d) present the original movement sequences together with the final smoothed trajectories, where (c) includes the outliers in the original sequences, whilst (d) removes them for better visualization.

Online Compression. Technically, compression makes sense when dealing with large data sets, however both *Taxi Data* and the big part of *Phone Data* have no complementary features (Q_p^{cf}) available but only the GPS features ($Q_p^{\ell s}$). Thus, our current experiment validates the sensitivity of data compression rate with respect to the spatiotemporal significance $Sig^{SP}(Q_p^{\ell s})$ on location streams, without considering the significance of the complementary features $Sig^{C}(Q_p^{cf})$. As shown in Fig. 6.4, we plot the compression rate sensitivity when applying different thresholds on $Sig^{SP}(Q_p^{\ell s})$. The results are proportional when using the *Phone Data* with respective $Sig(Q_p)$ thresholds.

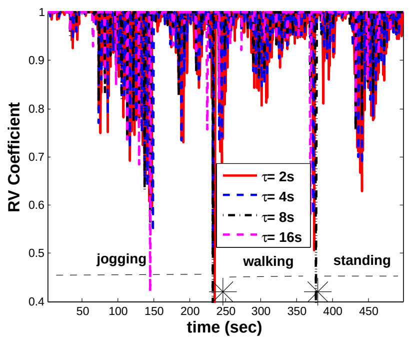


Figure 6.5: Episode identification varying the batch size, for $\sigma = 0.6$

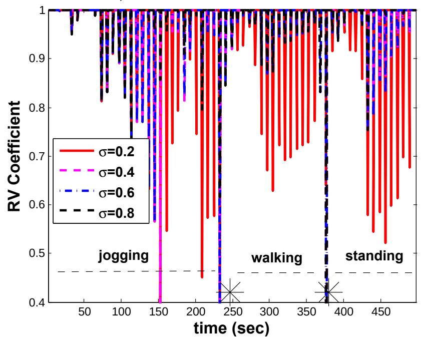


Figure 6.6: Sensitivity of RV w.r.t. different σ at $\tau = 8s$

Online Segmentation. SeTraStream's procedure in online trajectory segmentation relates to (1) initially computing the RV-coefficient between two workpieces $RV(W_\ell, W_r)$ and (2) expanding W_ℓ if $RV(W_\ell, W_r)$ is bigger than the given threshold σ (otherwise, we identify a division point between two episodes). Results are shown in Fig. 6.5, where for $T = 60s$ we can discover two main division points (with RV-coefficient< 0.6 and batch size $\tau < 16$), which is consistent with the underlying ground-truth tags. The stars in the figures are the real division points in the streaming data, which indicate when user changes their movement behaviors e.g. from jogging to walking and finally to standing.

Fig. 6.5 analyzes the sensitivity of using different batch sizes, where the best outcome (i.e accurate episode extend determination) is $\tau = 8s$; when $\tau = 16s$, we actually identify three division points, which is partially correct, since as we can see there are only two real division points in the stream. Similarly, we also investigate the segmentation sensitivity regarding different division thresholds σ in Fig. 6.6. The best segmentation result is achieved when $\sigma = 0.6$. Finally, we evaluate the time performance

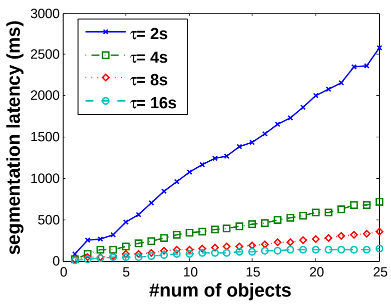


Figure 6.7: Segmentation latency with different τ sizes (σ=0.6)

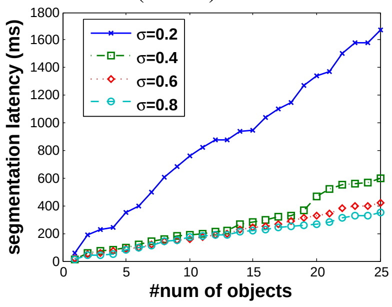


Figure 6.8: Segmentation latency with different σ thresholds ($\tau = 8s$)

of SeTraStream's trajectory segmentation module. We measure the segmentation latency with 25 users in the *Phone Data*. In the experiments, we used a laptop with 2.2 Ghz CPU and 4 Gb of memory. From Fig.6.7 and Fig. 6.8, we can see the segmentation time is almost linear, in both situations with different batch sizes (τ) and different division thresholds (σ), which is quite consistent with Lemma 1.

6.6 Distributed Semantic Trajectory Extraction

In our discussion so far, we presented a complete framework for online semantic trajectory extraction. In the current section we comment on how the developed techniques can be adapted so as to perform in a distributed streaming setting. Our focus is again in coming up with efficient ways to perform the distributed semantic trajectory extraction being primarily concerned with communication load reduction.

Initially, we assume that the GPS enabled devices that have been attached on the monitored objects possess adequate self calibrating capabilities to perform the necessary data preparation (data cleaning and compression) steps for each batch separately. Otherwise, it is easy to see that the formed batches need to be transmitted to the central server as are and assign the data preparation load to it. On the other hand, the memory capacity of those devices suffices to hold only the information (the movement feature vectors $Q^{\ell f}$ and Q^{cf}) of a small number of batches. Thus, in the general case, objects are not allowed to individually perform the episode determination phase, as the $W_{i\ell}$ workpiece can grow as large as $\frac{T}{\tau}$ batches and cannot be accommodated in the local memory.

Hence, together with the compression scheme of Section 6.3.2, we focus on further reducing the total size of the batch B that is to be transmitted to the central server. This happens because the aforementioned size is equivalent to the bandwidth a monitored object consumes every τ time units. Thus, we aim at inventing data reduction techniques that can alleviate the communication burden with appropriate accuracy guarantees.

Having Expression 6.7 in mind and based on Chapter 5, where the main monitored function - similarity measure in our discussion was the cosine similarity, our first, deliberate reaction is to try applying the locality sensitive hashing scheme on individual matrices of batches that have previously been vectorized. Unfortunately, in this particular case we cannot do so. This is because if each device installed on a moving object transforms its original feature vectors to bitmaps, the left workpiece expansion while attempting to detect long term pattern alterations cannot be carried out utilizing separate batch bitmaps.

In order to perform the distributed semantic trajectory extraction in an efficient manner, we take advantage of the ascertainment of Lemma 6.4.2. Recall that according to the aforementioned lemma, during the desired RV coefficient checks, the element $W_{i\ell}W'_{i\ell}$ is the sum of batch matrix products with their transposes and the same holds for $W_rW'_r$ which is composed of a single batch. Consequently, it suffices for every object O_i to transmit the current product $B \cdot B$ for the lastly acquired batch. This essentially constitutes the right workpiece that is to be compared for similarity with left workpieces as dictated in Section 6.4. Section 6.4 also notes that the size of this product is fixed d^2 for d features of interest. Given the above, the upcoming corollary establishes the communication cost of the distributed extraction process for a system that operates for U time units.

Corollary 6.6.1. *The communication cost of the distributed, semantic trajectory extraction process for U time units is $O(d^2 \frac{U}{\tau})$ per monitored object. Assuming N objects are being monitored, the total communication load in the network links is $O(d^2 \frac{U}{\tau} N)$.*

Note that a second attempt to reduce the communication cost would be the application of the prediction - based geometric monitoring ideas introduced in Chapter 3. More precisely, our interest in this scenario is the application of the geometric approach between coordinator - moving object pairs. However, in this particular case, the monitored function of the RV coefficient is derived based on the vectorized batches obtained by a single object O_i i.e. no averaging operation is needed. Furthermore, the coordinator does not produce movement feature vectors itself. Thus the whole procedure is reduced to simple, bipartite predictors' installation and maintenance. Based on the predictions regarding $Vec(W_{i\ell}W'_{i\ell})^p$ and $Vec(W_{ir}W'_{ir})^p$, the coordinator will be able to derive estimations of whether $RV_i(W_{i\ell}, W_r) \leq \sigma$ or not. On the other hand, each object O_i, which knows the information about the predictor it transmitted the last time it contacted the coordinating source, will be able to see if a false (negative or positive) estimation of the σ-similarity check is performed. Overall, predictors, as those presented in Table 3.2, may find themselves useful even in the current scenario for further reducing the bandwidth consumption. Nevertheless, then the monitoring task does not raise any additional demand for the adoption of the geometric monitoring framework.

A plausible observation is that Corollary 6.6.1 achieves to reduce the amount of communicated data without compromising the accuracy of the extraction process. In other words, the pinpointed division points will be exactly the same with those that would be identified should the framework performed in a centralized setting where batches are directly gathered to the server. Furthermore, it worths observing that the communication cost of the distributed process is independent from both the sampling rate of the used devices, i.e. the number of the movement feature vectors in a batch and the chosen window size T. In contrast, the bandwidth consumption is determined by the total time of semantic trajectory extraction, the number of monitored features and the amount of monitored objects. Hence, given the number of monitored objects N, users are enabled to predetermine the characteristics of the hardware infrastructure that needs to be utilized.

6.7 Synopsis

In this chapter, we proposed a novel and complete online framework, namely *SeTraStream* that enables semantic trajectory extraction over streaming movement data. This is the first method proposed in the literature tackling with this problem in real-time streaming environments. Moreover, we considered challenges occurring in real world applications including data cleaning and load shedding procedures before accurately identifying trajectory episodes in objects' streaming movement data. Eventually, we devised extensions of the basic framework to distributed streaming settings by pointing out smart communication reduction techniques that manage to leave the accuracy of the framework unaffected.

Chapter 7

Conclusions and Outlook

This thesis elaborated on algorithms for monitoring as well as mining distributed data streams. Our algorithms were derived based on a couple of different rationales while performing the monitoring and the mining process. Having pointed out the need of communication load reduction, in the distributed monitoring part of our study we decomposed the problem of monitoring non-linear functions into local constraints that can be independently checked by each site in the network and call for central data collection only if these constraints were violated. In the mining part of our study we managed to dictate efficient, distributed algorithms by data reduction techniques that could provide guarantees on their accuracy.

In Chapter 3 we generalized the geometric monitoring framework of [103, 105] by incorporating the notion of predictors [42]. We thus introduced efficient algorithms for prediction - based geometric monitoring of complex threshold functions. According to our experimental analysis using a variety real datasets, functions and parameter settings our methods are capable of providing significant communication load reduction during the tracking procedure. Our future work on this issue is directed towards the monitoring of thresholded functions with relative thresholds such as the common case [23, 22] where the coordinator is supposed to keep an approximation $f(e^p(t)) \in (1 \pm \epsilon)f(v(t))$ of the true value of the pinpointed f function, for some constant $0 < \epsilon < 1$. Furthermore, we concentrate our efforts in the choice of optimal reference points, as in [106], that could perhaps enable "looser" conditions for strict convex hull containment(Section 3.4.1).

In Chapter 4 we provided a case study of the prediction-based geometric monitoring framework of Chapter 3 and devised alternative geometric approaches for the specific scenario of distributed representative trajectory monitoring. As with any averaging operation, our ReTra computation is sensitive to the dispersion of the values around the mean movement pattern. Consequently, apart from performing the monitoring task, users need to check the amount of variance the current ReTra holds. For instance, imagine that we wish to monitor the objects commuting inside a rectangular area of interest. If half of the objects move at the north half of the rectangular area, while the rest choose the south counterpart the extracted ReTra will lie in the middle of the area. Obviously, such a representative trajectory does not provide e.g., accurate results regarding the concentration of traffic. Evidently, traffic experts need to be aware of that fact so as to avoid rush conclusions regarding traffic wardens' placement in the previous example. The incorporation of motion variance tracking as well as the adaptive presentation of more than one ReTras depending on this variation constitute future work considerations.

Our TACO framework [44, 45] was discussed in Chapter 5. TACO was the first in the literature to formulate an in-network, continuous outlier identification procedure where the reduction of the communication load simultaneously provides the means to predict the accuracy of the similarity tests between motes upon using those squeezed representations. Our ongoing work extends TACO to render it capable of accomondating: a) multidimensional outlier definitions with tumble (disjoint windows) operation, b) unidimensional outlier definition and sliding window operation and finally (c) multidimensional outlier definition and sliding window operation.

Finally, Chapter 6 presented our SeTraStream [116] framework which was the first in the literature to provide semantic trajectory extraction over streaming movement data. We further elaborated on extensions of the basic, centralized framework to distributed settings. An interesting subject that we left as future work regards the combination of the concepts of Chapter 4 with those of Chapter 6 so as to distributively perform semantic aware representative trajectory extraction. This imposes new challenges to the basic methods of semantic trajectory extraction and ReTra monitoring as it combines the monitoring (of the ReTra) and the mining (semantic trajectory extraction) functions in a single operational module.

Bibliography

[1] L. O. Alvares, V. Bogorny, B. Kuijpers, J. Macedo, B. Moelans, and A. Vaisman. A Model for Enriching Trajectories with Semantic Geographical Information. In *GIS*, 2007.

[2] V. Athitsos, M. Potamias, P. Papapetrou, and G. Kollios. Nearest Neighbor Retrieval Using Distance-Based Hashing. In *ICDE*, 2008.

[3] B. Babcock and C. Olston. Distributed top-k monitoring. In *SIGMOD*, 2003.

[4] M. Bawa, H. Garcia-Molina, A. Gionis, and R. Motwani. Estimating Aggregates on a Peer-to-Peer Network. Technical report, Stanford, 2003.

[5] S. D. Bay and M. Schwabacher. Mining distance-based outliers in near linear time with randomization and a simple pruning rule. In *KDD*, 2003.

[6] S. Brakatsoulas, D. Pfoser, R. Salas, and C. Wenk. On map-matching vehicle tracking data. In *VLDB*, 2005.

[7] J. Branch, B. Szymanski, C. Giannella, R. Wolff, and H. Kargupta. In-network outlier detection in wireless sensor networks. In *ICDCS* , 2006.

[8] M. Buchin, A. Driemel, M. V. Kreveld, and V. Sacristan. An Algorithmic Framework for Segmenting Trajectories based on Spatio-Temporal Criteria. In *GIS*, 2010.

[9] S. Burdakis and A. Deligiannakis. Detecting Outliers in Sensor Networks using the Geometric Approach. In *ICDE*, 2012.

[10] H. Cao, O. Wolfson, and G. Trajcevski. Spatio-Temporal Data Reduction With Deterministic Error Bounds. *The VLDB Journal*, 15(3), 2006.

[11] D. Carney, U. Çetintemel, M. Cherniack, C. Convey, S. Lee, G. Seidman, M. Stonebraker, N. Tatbul, and S. Zdonik. Monitoring streams: a new class of data management applications. In *VLDB*, pages 215–226, 2002.

[12] M. Charikar. Similarity estimation techniques from rounding algorithms. In *STOC*, 2002.

[13] M. Chatterjee, S. Das, and D. Turgut. WCA: A Weighted Clustering Algorithm for Mobile Ad hoc Networks. *Journal of Cluster Computing(Special Issue on Mobile Ad hoc Networks)*, 5, 2002.

[14] J. Chen, S. Kher, and A. Somani. Distributed Fault Detection of Wireless Sensor Networks. In *DIWANS*, 2006.

[15] F. Chu, Y. Wang, D. S. Parker., and C. Zaniolo. Data cleaning using belief propagation. In *IQIS*, 2005.

[16] E. Cohen. Size-estimation framework with applications to transitive closure and reachability. *J. Comput. Syst. Sci.*, 55(3):441–453, 1997.

[17] E. Cohen, M. Datar, S. Fujiwara, A. Gionis, P. Indyk, R. Motwani, J. Ullman, and C. Yang. Finding Interesting Associations without Support Pruning. In *ICDE*, 2000.

[18] E. Cohen, M. Datar, S. Fujiwara, A. Gionis, P. Indyk, R. Motwani, J. D. Ullman, and C. Yang. Finding interesting associations without support pruning. In *ICDE*, 2000.

[19] J. Considine, M. Hadjieleftheriou, F. Li, J. Byers, and G. Kollios. Robust approximate aggregation in sensor data management systems. *ACM Trans. Database Syst.*, 34(1):1–35, 2009.

[20] G. Cormode and M. Garofalakis. Sketching streams through the net: Distributed approximate query tracking. In *VLDB*, 2005.

[21] G. Cormode and M. Garofalakis. Streaming in a connected world: querying and tracking distributed data streams. In *SIGMOD*, 2007.

[22] G. Cormode and M. Garofalakis. Approximate continuous querying over distributed streams. *ACM Transactions on Database Systems*, 33(2), 2008.

[23] G. Cormode, M. Garofalakis, S. Muthukrishnan, and R. Rastogi. Holistic aggregates in a networked world: distributed tracking of approximate quantiles. In *SIGMOD*, 2005.

[24] G. Cormode, S. Muthukrishnan, and K. Yi. Algorithms for distributed functional monitoring. *ACM Trans. Algorithms*, 7:21:1–21:20, 2011.

[25] G. Cormode, S. Muthukrishnan, and W. Zhuang. Conquering the divide: Continuous clustering of distributed data streams. In *ICDE*, 2007.

[26] A. Das, S. Ganguly, M. Garofalakis, and R. Rastogi. Distributed set-expression cardinality estimation. In *VLDB*, 2004.

[27] A. Deligiannakis, Y. Kotidis, and N. Roussopoulos. Compressing Historical Information in Sensor Networks. In *ACM SIGMOD*, 2004.

[28] A. Deligiannakis, Y. Kotidis, and N. Roussopoulos. Hierarchical In-Network Data Aggregation with Quality Guarantees. In *EDBT*, 2004.

[29] A. Deligiannakis, Y. Kotidis, and N. Roussopoulos. Bandwidth Constrained Queries in Sensor Networks. *The VLDB Journal*, 2007.

[30] A. Deligiannakis, Y. Kotidis, V. Vassalos, V. Stoumpos, and A. Delis. Another Outlier Bites the Dust: Computing Meaningful Aggregates in Sensor Networks. In *ICDE*, 2009.

[31] A. Deligiannakis, Y. Kotidis, V. Vassalos, V. Stoumpos, and A. Delis. Another Outlier Bites the Dust: Computing Meaningful Aggregates in Sensor Networks. In *ICDE*, 2009.

[32] A. Deshpande, C. Guestrin, S. Madden, J. Hellerstein, and W. Hong. Model-Driven Data Acquisition in Sensor Networks. In *VLDB*, 2004.

[33] W. Dong, M. Charikar, and K. Li. Asymmetric distance estimation with sketches for similarity search in high-dimensional spaces. In *SIGIR*, 2008.

[34] D. Douglas and T. Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *The Canadian Cartographer*, 10(2), 1973.

[35] E. Elnahrawy and B.Nath. Cleaning and querying noisy sensors. In *WSNA*, 2003.

[36] U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors. *Advances in Knowledge Discovery and Data Mining*. AAAI/MIT Press, 1996.

[37] M. Garofalakis, J. Gehrke, and R. Rastogi. Querying and mining data streams: you only get one look a tutorial. In *SIGMOD*, 2002.

[38] K. Georgoulas and Y. Kotidis. Random Hyperplane Projection using Derived Dimensions. In *MobiDE*, 2010.

[39] A. Ghoting, S. Parthasarathy, and M. Otey. Fast Mining of Distance-Based Outliers in High-Dimensional Datasets. In *SIAM*, 2006.

[40] C. Giannella, J. Han, J. Pei, X. Yan, and P. S. Yu. *Mining Frequent Patterns in Data Streams at Multiple Time Granularities*. MIT Press, 2002.

[41] F. Giannotti, M. Nanni, F. Pinelli, and D. Pedreschi. Trajectory Pattern Mining. In *KDD*, 2007.

[42] N. Giatrakos, A. Deligiannakis, M. Garofalakis, I. Sharfman, and A. Schuster. Prediction - based geometric monitoring over distributed data streams. In *SIGMOD*, 2012.

[43] N. Giatrakos, Y. Kotidis, and A. Deligiannakis. PAO: Power-efficient Attribution of Outliers in Wireless Sensor Networks. In *DMSN*, 2010.

[44] N. Giatrakos, Y. Kotidis, A. Deligiannakis, V. Vassalos, and Y. Theodoridis. TACO: Tunable Approximate Computation of Outliers in Wireless Sensor Networks. In *SIGMOD*, 2010.

[45] N. Giatrakos, Y. Kotidis, A. Deligiannakis, V. Vassalos, and Y. Theodoridis. In-Network Approximate Computation of Outliers with Quality Guarantees. *Information Systems*, 2011. http://dx.doi.org/10.1016/j.is.2011.08.005.

[46] A. Gionis, D. Gunopulos, and N. Koudas. Efficient and tunable similar set retrieval. In *SIGMOD*, 2001.

[47] M. Goemans and D. Williamson. Improved Approximation Algorithms for Maximum Cut and Satisfiability Problems Using Semidefinite Programming. *J. ACM*, 42(6), 1995.

[48] G. Gutin, A. Yeo, and A. Zverovich. Traveling salesman should not be greedy: domination analysis of greedy-type heuristics for the TSP. *Discrete Applied Mathematics*, 117:81–86, 2002.

[49] R. Güting and M. Schneider. *Moving Objects Databases*. Morgan Kaufmann, 2005.

[50] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-Efficient Communication Protocol for Wireless Microsensor Networks. In *HICSS*, 2000.

[51] L. Huang, M. Garofalakis, J. Hellerstein, A. Joseph, and N. Taft. Toward sophisticated detection with distributed triggers. In *MineNet*, 2006.

[52] L. Huang, X. Nguyen, M. Garofalakis, and J. M. Hellerstein. Communication-efficient online detection of network-wide anomalies. In *INFOCOM*, 2007.

[53] P. Indyk. Dimensionality reduction techniques for proximity problems. In *SODA*, 2000.

[54] P. Indyk and R. Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *STOC*, 1998.

[55] A. Jain, J. M. Hellestein, S. Ratnasamy, and D. Wetherall. A wakeup call for internet monitoring systems: The case for distributed triggers. In *HotNets*, 2004.

[56] S. Jeffery, G. Alonso, M. J. Franklin, W. Hong, and J. Widom. Declarative Support for Sensor Data Cleaning. In *Pervasive*, 2006.

[57] S. Jeffery, G. Alonso, M. J. Franklin, W. Hong, and J. Widom. A pipelined framework for online cleaning of sensor data streams. In *ICDE*, 2006.

[58] S. Jeffery, M. Garofalakis, and M. Franklin. Adaptive Cleaning for RFID Data Streams. In *VLDB*, 2006.

[59] H. Jeung, Q. Liu, H. T. Shen, and X. Zhou. A hybrid prediction model for moving objects. In *ICDE*, 2008.

[60] H. Jeung, M. L. Yiu, X. Zhou, C. S. Jensen, and H. T. Shen. Discovery of Convoys in Trajectory Databases. In *VLDB*, 2008.

[61] J. Jun, R. Guensler, and J. Ogle. Smoothing Methods to Minimize Impact of Global Positioning System Random Error on Travel Distance, Speed, and Acceleration Profile Estimates. *Transportation Research Record: Journal of the Transportation Research Board*, 1972(1), jan 2006.

[62] B. Kanagal and A. Deshpande. Online Filtering, Smoothing and Probabilistic Modeling of Streaming data. In *ICDE*, 2008.

[63] B. Karp and H. Kung. GPSR: Greedy Perimeter Stateless Routing for Wireless Networks. In *MOBICOM*, 2000.

[64] G. Kellaris, N. Pelekis, and Y. Theodoridis. Trajectory Compression under Network Constraints. In *SSTD*, 2009.

[65] D. Kempe, A. Dobra, and J. Gehrke. Gossip-Based Computation of Aggregate Information. In *FOCS*, 2003.

[66] E. Keogh, S. Chu, D. Hart, and M. Pazzani. An Online Algorithm for Segmenting Time Series. In *ICDM*, 2001.

[67] R. Keralapura, G. Cormode, and J. Ramamirtham. Communication-efficient distributed monitoring of thresholded counts. In *SIGMOD*, 2006.

[68] N. Khoussainova, M. Balazinska, and D. Suciu. Towards Correcting Input Data Errors Probabilistically using Integrity Constraints. In *MobiDE*, 2006.

[69] N. Kiukkoneny, J. Blom, O. Dousse, D. Gatica-Perez, and J. Laurila. Towards Rich Mobile Phone Datasets: Lausanne Data Collection Campaign. In *ICPS*, 2010.

[70] Y. Kotidis. Snapshot Queries: Towards Data-Centric Sensor Networks. In *ICDE*, 2005.

[71] Y. Kotidis, V. Vassalos, A. Deligiannakis, V. Stoumpos, and A. Delis. Robust management of outliers in sensor network aggregate queries. In *MobiDE*, 2007.

[72] T. Kristensen. Transforming tanimoto queries on real valued vectors to range queries in euclidian space. *Journal of Mathematical Chemistry*, 48:287–289, 2010.

[73] J.-G. Lee, J. Han, and K.-Y. Whang. Trajectory clustering: a partition-and-group framework. In *SIGMOD*, 2007.

[74] J.-G. Lee, J. Han, and K.-Y. Whang. Trajectory Clustering: a Partition-and-Group Framework. In *SIGMOD*, 2007.

[75] P. Levis, N. Lee, M. Welsh, and D. Culler. TOSSIM: Accurate and scalable simulation of entire TinyOS applications. In *SenSys*, 2004.

[76] D. D. Lewis, Y. Yang, T. G. Rose, and F. Li. Rcv1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5(Apr):361–397, 2004.

[77] Z. Li, B. Ding, J. Han, R. Kays, and P. Nye. Mining Periodic Behaviors for Moving Objects. In *KDD*, 2010.

[78] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. TAG: A Tiny Aggregation Service for ad hoc Sensor Networks. In *OSDI Conf.*, 2002.

[79] S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. Tinydb: an acquisitional query processing system for sensor networks. *ACM Trans. Database Syst.*, 30:122–173, March 2005.

[80] G. Marketos, E. Frentzos, I. Ntoutsi, N. Pelekis, A. Raffaetà, and Y. Theodoridis. Building real-world trajectory warehouses. In *MobiDE*, 2008.

[81] S. Meng, K. Xie, G. Chen, X. Ma, and G. Song. A kalman filter based approach for outlier detection in sensor networks. In *CSSE* , 2008.

[82] N. Meratnia and R. A. de By. Spatiotemporal Compression Techniques for Moving Point Objects. In *EDBT*, 2004.

[83] A. Monreale, F. Pinelli, R. Trasarti, and F. Giannotti. Wherenext: a location predictor on trajectory pattern mining. In *KDD*, 2009.

[84] C. Olston, J. Jiang, and J. Widom. Adaptive filters for continuous queries over distributed data streams. In *SIGMOD*, 2003.

[85] M. Otey, A. Ghoting, and S. Parthasarathy. Fast distributed outlier detection in mixed-attribute data sets. *Data Min. Knowl. Discov.*, 12(2-3), 2006.

[86] A. T. Palma, V. Bogorny, B. Kuijpers, and L. O. Alvares. A Clustering-based Approach for Discovering Interesting Places in Trajectories. In *SAC*, 2008.

[87] C. Panagiotakis, N. Pelekis, and I. Kopanakis. Trajectory voting and classification based on spatiotemporal similarity in moving object databases. In *IDA*, 2009.

[88] C. Panagiotakis, N. Pelekis, I. Kopanakis, E. Ramasso, and Y. Theodoridis. Segmentation and sampling of moving object trajectories based on representativeness. *TKDE*, 2011.

[89] N. Pelekis, E. Frentzos, N. Giatrakos, and Y. Theodoridis. Hermes: aggregative lbs via a trajectory db engine. In *SIGMOD*, 2008.

[90] N. Pelekis, E. Frentzos, N. Giatrakos, and Y. Theodoridis. HERMES: Aggregative LBS via a Trajectory DB Engine. In *SIGMOD*, 2008.

[91] N. Pelekis, A. Gkoulalas-Divanis, M. Vodas, D. Kopanaki, and Y. Theodoridis. Privacy-aware querying over sensitive trajectory data. In *CIKM*, 2011.

[92] N. Pelekis, I. Kopanakis, E. Kotsifakos, E. Frentzos, and Y. Theodoridis. Clustering trajectories of moving objects in an uncertain world. In *ICDM*, 2009.

[93] N. Pelekis, C. Panagiotakis, I. Kopanakis, and Y. Theodoridis. Unsupervised trajectory sampling. In *ECML-PKDD*, 2010.

[94] J. Polastre, R. Szewczyk, A. Mainwaring, D. Culler, and J. Anderson. *Analysis of wireless sensor networks for habitat monitoring*, pages 399–423. Kluwer Academic Publishers, 2004.

[95] M. Potamias, K. Patroumpas, and T. Sellis. Sampling Trajectory Streams with Spatiotemporal Criteria. In *SSDBM*, 2006.

[96] M. Qin and R. Zimmermann. VCA: An Energy-Efficient Voting-Based Clustering Algorithm for Sensor Networks. *J.UCS*, 13(1), 2007.

[97] D. Ravichandran, P. Pantel, and E. Hovy. Randomized algorithms and NLP: using locality sensitive hash function for high speed noun clustering. In *ACL*, 2005.

[98] J. A. M. R. Rocha, V. C. Times, G. Oliveira, L. O. Alvares, and V. Bogorny. Db-Smot: a Direction-Based Spatio-Temporal Clustering Method. In *Intelligent Systems*, 2010.

[99] D. Sacharidis, K. Patroumpas, M. Terrovitis, V. Kantere, M. Potamias, K. Mouratidis, and T. Sellis. On-line discovery of hot motion paths. In *EDBT*, 2008.

[100] G. Sagy, D. Keren, I. Sharfman, and A. Schuster. Distributed threshold querying of general functions by a difference of monotonic representation. *Proc. VLDB Endow.*, 4:46–57, 2010.

[101] N. Schüssler and K. W. Axhausen. Processing GPS Raw Data Without Additional Information. *Transportation Research Record: Journal of the Transportation Research Board*, 8, 2009.

[102] M. A. Sharaf, J. Beaver, A. Labrinidis, and P. K. Chrysanthis. TiNA: A Scheme for Temporal Coherency-Aware in-Network Aggregation. In *MobiDE*, 2003.

[103] I. Sharfman, A. Schuster, and D. Keren. A geometric approach to monitoring threshold functions over distributed data streams. In *SIGMOD*, 2006.

[104] I. Sharfman, A. Schuster, and D. Keren. Aggregate threshold queries in sensor networks. In *IPDPS*, 2007.

[105] I. Sharfman, A. Schuster, and D. Keren. A geometric approach to monitoring threshold functions over distributed data streams. *ACM Transactions on Database Systems*, 32(4), 2007.

[106] I. Sharfman, A. Schuster, and D. Keren. Shape sensitive geometric monitoring. In *PODS*, 2008.

[107] B. Sheng, Q. Li, W. Mao, and W. Jin. Outlier detection in sensor networks. In *MobiHoc*, 2007.

[108] V. Shnayder, M. Hempstead, B. Chen, G. W. Allen, and M. Welsh. Simulating the Power Consumption of Large-Scale Sensor Network Applications. In *Sensys*, 2004.

[109] S. Singh, M. Woo, and C. S. Raghavendra. Power-aware Routing in Mobile Ad Hoc Networks. In *MobiCom*, 1998.

[110] S. Spaccapietra, C. Parent, M. L. Damiani, J. A. de Macedo, F. Porto, and C. Vangenot. A Conceptual View on Trajectories. *Data and Knowledge Engineering*, 65(1), 2008.

[111] S. Subramaniam, T. Palpanas, D. Papadopoulos, V. Kalogeraki, and D. Gunopulos. Online Outlier Detection in Sensor Data Using Non-Parametric Models. In *VLDB*, 2006.

[112] Y.-J. Wen, A. M. Agogino, and K.Goebel. Fuzzy Validation and Fusion for Wireless Sensor Networks. In *ASME*, 2004.

[113] X. Xiao, W. Peng, C. Hung, and W. Lee. Using SensorRanks for In-Network Detection of Faulty Readings in Wireless Sensor Networks. In *MobiDE*, 2007.

[114] G. Xue, Y. Jiang, Y. You, and M. Li. A topology-aware hierarchical structured overlay network based on locality sensitive hashing scheme. In *UPGRADE*, 2007.

[115] Z. Yan, D. Chakraborty, C. Parent, S. Spaccapietra, and A. Karl. SeMiTri: A Framework for Semantic Annotation of Heterogeneous Trajectories. In *EDBT*, 2011.

[116] Z. Yan, N. Giatrakos, V. Katsikaros, N. Pelekis, and Y. Theodoridis. SeTraStream: Semantic Aware Trajectory Construction over Streaming Movement Data. In *SSTD*, 2011.

[117] Z. Yan, C. Parent, S. Spaccapietra, and D. Chakraborty. A Hybrid Model and Computing Platform for Spatio-Semantic Trajectories. In *ESWC*, 2010.

[118] Z. Yan, L. Spremic, D. Chakraborty, C. Parent, S. Spaccapietra, and A. Karl. Automatic Construction and Multi-level Visualization of Semantic Trajectories. In *GIS*, 2010.

[119] Y. Yao and J. Gehrke. The Cougar Approach to In-Network Query Processing in Sensor Networks. *SIGMOD Record*, 31(3), 2002.

[120] G. Yavas, D. Katsaros, O. Ulusoy, and Y. Manolopoulos. A data mining approach for location prediction in mobile environments. *Data Knowl. Eng.*, 54(2), 2005.

[121] K. Yi and Q. Zhang. Optimal tracking of distributed heavy hitters and quantiles. In *PODS*, 2009.

[122] O. Younis and S. Fahmy. Distributed Clustering in Ad-hoc Sensor Networks: A Hybrid, Energy-Efficient Approach. In *INFOCOM*, 2004.

[123] D. Zeinalipour, P. Andreou, P. Chrysanthis, G. Samaras, and A. Pitsillides. The Micropulse Framework for Adaptive Waking Windows in Sensor Networks. In *MDM*, 2007.

[124] K. Zhang, S. Shi, H. Gao, and J. Li. Unsupervised outlier detection in sensor networks using aggregation tree. In *ADMA* , 2007.

[125] Q. Zhang, J. Liu, and W. Wang. Approximate clustering on distributed data streams. In *ICDE*, 2008.

[126] Y. Zhang, N. Meratnia, and P. Havinga. Outlier detection techniques for wireless sensor networks: A survey. *International Journal of IEEE Communications Surveys and Tutorials*, 12(2), 2010.

[127] Y. Zheng, Y. Chen, Q. Li, X. Xie, and W.-Y. Ma. Understanding transportation modes based on GPS data for web applications. *Transactions on the Web (TWEB)*, 4(1), 2010.

[128] Y. Zhuang and L. Chen. In-network Outlier Cleaning for Data Collection in Sensor Networks. In *Clean DB Workshop*, 2006.

[129] Y. Zhuang, L. Chen, S. Wang, and J. Lian. A Weighted Moving Average-based Approach for Cleaning Sensor Data. In *ICDCS*, 2007.