CISTER
Research Center in
Real-Time & Embedded
Computing Systems

Journal Paper

Feature Extraction in Densely Sensed Environments: Extensions to Multiple Broadcast Domains

Maryam Vahabi*
Vikram Gupta
Michele Albano*
Raghu R.*
Eduardo Tovar*

*CISTER Research Center

CISTER-TR-150704

2015

Feature Extraction in Densely Sensed Environments: Extensions to Multiple Broadcast Domains

Maryam Vahabi*, Vikram Gupta, Michele Albano*, Raghu R.*, Eduardo Tovar*

*CISTER Research Center

Polytechnic Institute of Porto (ISEP-IPP)

Rua Dr. António Bernardino de Almeida, 431

4200-072 Porto

Portugal

Tel.: +351.22.8340509, Fax: +351.22.8321159

E-mail: mamvi@isep.ipp.pt, vigup@isep.ipp.pt, mialb@isep.ipp.pt, raghu@isep.ipp.pt, emt@isep.ipp.pt

http://www.cister.isep.ipp.pt

Abstract

The vision of the Internet of Things (IoT) includes large and dense deployment of interconnected smart sensing and monitoring devices. This vast deployment necessitates collection and processing of large volume of measurement data. However, collecting all the measured data from individual devices on such a scale may be impractical and time consuming. Moreover, processing these measurements requires complex algorithms to extract useful information. Thus, it becomes imperative to devise distributed information processing mechanisms that identify application-specific features in a timely manner and with a low overhead.

In this article, we present a feature extraction mechanism for dense networks that takes advantage of dominance-based medium access control (MAC) protocols to (i) efficiently obtain global extrema of the sensed quantities, (ii) extract local extrema, and (iii) detect the boundaries of events, by using simple transforms that nodes employ on their local data. We extend our results for a large dense network with multiple broadcast domains (MBD). We discuss and compare two approaches for addressing the challenges with MBD and we show through extensive evaluations that our proposed distributed MBD approach is fast and efficient at retrieving the most valuable measurements, independent of the number sensor nodes in the network.

Hindawi Publishing Corporation
International Journal of Distributed Sensor Networks
Volume 2015, Article ID 457537, 21 pages
http://dx.doi.org/10.1155/2015/457537

Research Article

Feature Extraction in Densely Sensed Environments: Extensions to Multiple Broadcast Domains

Maryam Vahabi,[1] Vikram Gupta,[1,2] Michele Albano,[1] Raghuraman Rangarajan,[1] and Eduardo Tovar[1]

[1]*CISTER/INESC-TEC, ISEP, Polytechnic Institute of Porto, 4249-015 Porto, Portugal*
[2]*Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA 15213, USA*

Correspondence should be addressed to Maryam Vahabi; mamvi@isep.ipp.pt

Received 12 May 2015; Accepted 1 July 2015

Academic Editor: Davide Brunelli

The vision of the Internet of Things (IoT) includes large and dense deployment of interconnected smart sensing and monitoring devices. This vast deployment necessitates collection and processing of large volume of measurement data. However, collecting all the measured data from individual devices on such a scale may be impractical and time-consuming. Moreover, processing these measurements requires complex algorithms to extract useful information. Thus, it becomes imperative to devise distributed information processing mechanisms that identify application-specific features in a timely manner and with low overhead. In this paper, we present a *feature extraction* mechanism for dense networks that takes advantage of dominance-based medium access control (MAC) protocols to (i) efficiently obtain global extrema of the sensed quantities, (ii) extract local extrema, and (iii) detect the boundaries of events, by using simple transforms that nodes employ on their local data. We extend our results for a large dense network with multiple broadcast domains (MBD). We discuss and compare two approaches for addressing the challenges with MBD and we show through extensive evaluations that our proposed distributed MBD approach is fast and efficient at retrieving the most valuable measurements, independent of the number sensor nodes in the network.

1. Introduction

Several technological advancements in hardware design enable IoT application scenarios with very dense deployments of sensor nodes. Researchers are designing tiny radio-on-a-chip communication devices with processing and communication capabilities. These low-power wireless devices can also be powered from the energy scavenged from ambient radio waves [1]. Several market studies project that trillions of things will soon be connected to the Internet [2]. In a large-scale dense network, the amount of collected data will therefore be enormous and it becomes necessary to extract only the information that is relevant to further processing [3]. For example, in some cases, it maybe required to alert the occurrence of certain features in a timely manner.

In this work, we tackle the problem of *feature extraction* in a dense IoT application. (An earlier version of this work was published in the proceeding of DCOSS 2014, DOI: 10.1109/DCOSS.2014.29. This work considered a single broadcast domain (SBD), where all nodes are located within the same transmission range.) We define feature extraction as *the process of determining certain features such as peaks, boundaries, and shapes in the distribution of a physical quantity*.

While *feature extraction* may not be an issue for a low density network (e.g., tens of sensor nodes), it is still a challenging problem for a high density network. In applications requiring measurements with a high spatial granularity, covering even a small area (e.g., one square meter) may require hundreds to thousands of sensor nodes. Some examples of densely deployed sensing nodes, where *features* of one or more physical quantities need to be monitored frequently, are sleep monitoring for health-care applications [4], smart surfaces for aviation systems [5], and fruit monitoring in agriculture industries [6].

Figure 1 presents an example distribution of the measurements of a physical quantity (e.g., temperature or pressure values). As shown, the distribution has three distinct regions

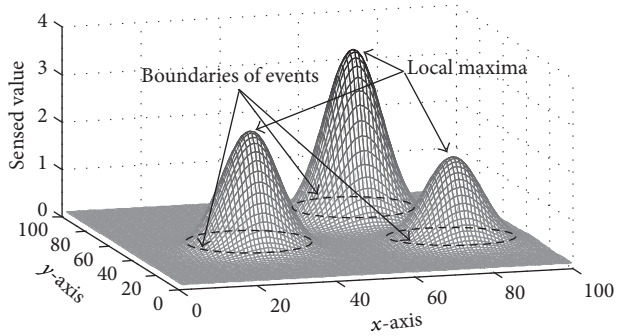

Figure 1: An example of a 2D physical quantity *field* with 10,000 sensor nodes. Each data point corresponds to a value sensed by an individual node.

of activity that require detection and evaluation. Such a dense sensor network for measuring a physical quantity, called a *field*, would involve deploying one sensor node to measure each data point in the distribution. The regions of activity, called *active regions*, are those where the values measured by sensor nodes are of interest.

One naive way to identify active regions would be to collect readings from all sensor nodes and process them centrally. This is usually inefficient since typical channel access techniques do not scale well with an increase in number of nodes. It is therefore advantageous to devise techniques that identify active regions irrespective of the density of the network.

Computing even simple aggregate quantities such as extrema (minimum or maximum) is not trivial for a dense network as it may require collecting data, in the worst case, from all nodes [7] (even if some sort of spatial subsampling is employed [8]). Dominance-based or binary-countdown MAC protocols help in finding the minimum value in constant time [9–11], provided that all nodes reside in a single broadcast domain (SBD). Furthermore, finding peaks and their boundaries in a distributed network, where each data point is measured by individual sensor nodes, is computationally expensive, is time-consuming, and typically does not even scale linearly with respect to network size.

In [3], we first proposed a set of feature extraction techniques for an SBD and established that finding the local extrema is a challenge in an SBD even after the global maximum is known. Once the global maximum is identified in constant time, we proposed a few transforms that nodes employ on local data, which helps in identifying other peaks (local maxima) and their boundaries in the spread of the physical quantity being measured. Our proposed transforms, referred to as *augmenting functions*, allow the identification of local extrema in constant time. We showed that, instead of collecting all data as in the naive case, these *augmenting functions* retrieve the most valuable sensor nodes' measurements that result in fewer number of measurements being collected.

In this paper, we present the design, implementation, and evaluation of a set of distributed algorithms to extract certain features of a physical quantity in a dense sensor network. We enhance our preliminary work published at [3] with the following new contributions:

(i) We propose a new distributed algorithm (dubbed *ripple*-based) to extend the process of obtaining global extrema in an MBD dense network. Since finding global extrema is the main building block in the proposed *augmenting functions*, this extension enables the applicability of these functions for MBD networks.

(ii) We modify an existing clustering approach [12] to obtain the global extrema for MBD networks and compare the novel ripple-based approach with the classical cluster-based approach and evaluate their efficiencies.

(iii) We design a simulation model to analyze the impact of relevant network parameters (e.g., communication range in wireless medium) on the functioning of the ripple-based approach.

This paper is organized as follows. An outline of other works related to our approach is given in Section 2 after a brief introduction on the dominance-based approach. The architecture and the system description based on an aggregate quantity function are provided in Section 3. Afterwards, we introduce various *augmenting functions* that are used to transform the sensor value such that local extrema or boundaries in various directions become the global extremum. In Section 5, a brief explanation of the clustering approach is given, followed by a novel approach, called ripple-based, to obtain the global extrema in an MBD network. This approach is fully distributed and exploits concurrent transmissions of dominance-based protocols. In Section 6, the evaluation of proposed *augmenting functions* is given together with a comprehensive comparison on the cluster-based and the ripple-based approaches for MBD networks.

2. Background and Related Work

Detection of events in sensor networks is a major application domain and a very broad topic. There are different approaches to address the problem of boundary detection in dense sensor networks. To the best of our knowledge, this is the first boundary detection technique that utilizes dominance-based MAC protocols. To this end, first we provide a detailed explanation of this MAC paradigm and explain how to compute an aggregate value (e.g., global `MIN`) utilizing the dominance-based MAC protocol, followed by an outline of related work.

2.1. Dominance-Based Approach. This work is inspired from dominance-based or binary-countdown protocols [9] implemented for wired networks in the widely used CAN bus [10] as well as its wireless version, called WiDom [11]. The major reason for using a dominance-based MAC protocol is its scalability and constant time-complexity even for very dense networks. In this work, we focus on time-complexity and we do not enrich our dominance-based MAC protocol with sleep states or other energy-saving mechanisms; thus, the current

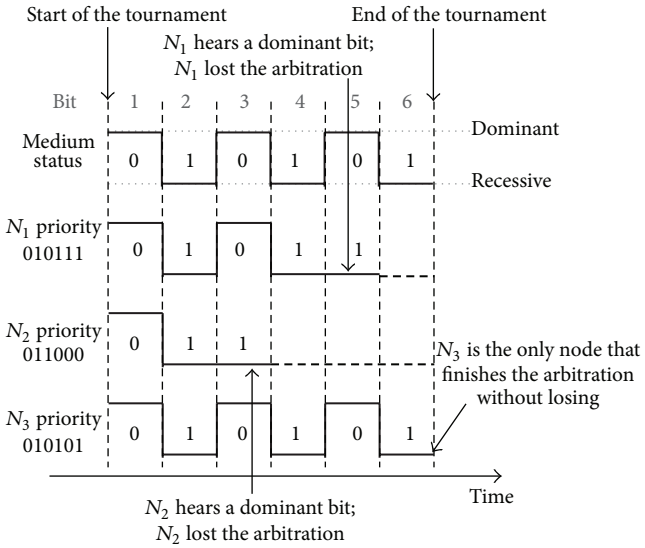

Figure 2: Tournament in dominance/binary countdown protocols.

state of the protocol is ready to be used in critical applications where energy issues are not the main focus of the scenarios.

In dominance-based MAC protocols, each node is associated with a priority value that is used to resolve the medium contention. All nodes "simultaneously" start a conflict resolution phase which we refer to as *tournament* (depicted in Figure 2), by transmitting synchronously their priority values bit-by-bit, starting with the most significant one, while simultaneously monitoring the medium. The medium must be devised in such a way that nodes will only detect a "1" value if no other nodes are transmitting a "0." Otherwise, every node detects a "0" value regardless of what the node itself is sending. For this reason, a "0" is said to be a dominant bit, while a "1" is said to be a recessive bit. Therefore, low numbers in the priority field of a message represent high priorities. If a node contends with a recessive bit but hears a dominant bit, then it will refrain from transmitting any further bits and will proceed only monitoring the medium. Since the medium acts as a logical `AND` operator, the node with the smallest priority value (winner) gains access to the medium.

By using sensed physical quantities (like temperature or acceleration) as the message priority, various aggregate quantities can be obtained in dense networks. For instance, the minimum of sensed value (`MIN`) can be found by only one tournament (i.e., in constant time) [13].

In fact, simultaneous transmissions are a key to the time-efficiency of dominance-based MAC protocols. With carrier-sense or time-division based MAC protocols, computing `MIN` depends on the number of sensor nodes. Studies show that the computation of `MIN` with standard IEEE 802.15.4 MAC linearly increases with the growth of network density (e.g., 80 ms for a 40-node network size) [7, 14]. Instead, the WiDom implementation [7] guarantees a constant time (10 ms) for calculating `MIN` regardless of network density.

The process of finding `MIN` has been leveraged in the past to find an approximate interpolation and other aggregate quantities [7, 15]. Interpolated values are computed through an iterative process by integration of local information available at each sensor node (its own location information and measured value plus the location information and the measured value of the winner received after each tournament). Each sensor node computes the difference between its measurement and the corresponding computed interpolation value, known as the error value. This error is then used as the priority value in the conflict resolution phase. After a number of iterations (defined by user), an approximate interpolation of the physical quantity is obtained. However, instead of getting the complete interpolated image, in this work, we aim at detecting certain *features* of the physical quantity *field* by selecting a set of sensor nodes that hold the most valuable information.

2.2. Boundary Detection. The problem of boundary detection and determining the extent of an event in sensor networks has been investigated in [8, 16–19]. Chintalapudi and Govindan presented localized edge detection techniques based on statistics, image processing, and classification [16]. Nowak and Mitra described a method for hierarchical boundary estimation using recursive dyadic partitioning [8]. They developed an inverse proportionality relation between energy consumption and the mean-square error in boundary detection and showed that their method is near-optimal with respect to this fundamental trade-off. Another hierarchical boundary estimation is proposed in [19] where a contiguous 2D shape of an event is found with a threshold-based boundary detection.

Other schemes represent the boundary of an event or the signal landscape of a sensor network compactly using in-network aggregation [20–24]. Gandhi et al. studied the problem of monitoring the events of sensor networks using sparse sampling [21]. However, their algorithm requires the prior knowledge of the event geometry (e.g. circle, ellipse, or rectangle) for computational efficiency. Similar method has been explored in [23], which utilized a regression-based spatial estimation technique to determine discrete points on the boundary. Ham and Rodriguez present a distributed boundary detection based on in-network aggregation in which only sensor nodes that identify a boundary transmit their observation to the remote station [24]. To that end, they first applied a Delaunay triangulation to determine the neighbors of each node and then generate boundary segments between neighboring sensors. However, this algorithm cannot be known as a fully decentralized approach since the two steps mentioned above should be done centrally through a remote station.

There are also contour-based methods for deciding the type of an event [25, 26]. They consider energy-efficient techniques to construct and incrementally update a number of 2D contour maps in a sensor network. Another field of research involving detecting holes and topological features in a sensor network is presented in [27, 28]. In these approaches, local connectivity graphs are used to infer static features of an event and require the involvement of all the nodes in the network. By contrast, our approach is quite different from the above-mentioned techniques, as we exploit an underlying

dominance-based MAC protocol for very sparse spatial sampling through a strategic selection of sensors.

3. System Model

We consider a sensor network where each sensor node has a unique *identifier*, i, and measures a particular physical quantity, s_i, using a sensing unit. Each sensor node knows its 2D coordinates (x, y) in the plane of deployment. We assume that the *feature extraction* mechanism can be either carried out periodically as a part of a *sense-process-actuate* control-loop or sporadically initiated by an external controller, like a data sink or a master node.

The collection of all the sensor values across the total sensed area is referred to as a *field* (Figure 1). Each data point in the *field* corresponds to a true (or nonfaulty) sensor reading value, sensed by an individual sensor node at its physical location. The spatial granularity and the size of the *field* are directly correlated to the distribution of the nodes and their spread. We also define *active region* as a physical area populated by sensor nodes that sense some *activity*. The overarching goal of our approach is to find location, boundaries, and shape of an *active region*, which we referred to as *features*, in the physical environment. For illustrating our approach and its evaluation, we generated various sample *fields* by a summation of 2D Gaussian functions (explained in the appendix).

Function $\mathscr{M}(v_i)$ is defined as the process of finding `MIN` over values v_i published by independent sensor nodes in a broadcast medium where $i \in \{1, 2, \ldots, N\}$ and N is the number of sensor nodes in a broadcast domain. In fact, $\mathscr{M}(v_i)$ represents one execution of the tournament in an SBD (in Section 5.2 we discuss the details of function $\mathscr{M}$ for MBD, where several tournaments need to be run). We exploit the property of simultaneous transmissions in dominance-based protocols to devise this function. v_i is the scaled value that each sensor node computes based on its measured value s_i and the global maximum $s_{\max}$ measured in the *field*. Each application of $\mathscr{M}(v_i)$ is referred to as a *round*. After each *round*, all the other sensor nodes know the *identifier* and the location of the sensor node with `MIN` value. This sensor node is known as the winner of the *round*. We use $\widehat{i}$ to denote the *identifier* of the winner. Hence, function $\mathscr{M}$ can be formally represented as

$$\left\{\widehat{v}, \widehat{i}, \widehat{x}, \widehat{y}\right\} = \mathscr{M}\left(v_i\right) \quad \forall i \in \{1, 2, \ldots, N\}, \tag{1}$$

where $\widehat{x}$ and $\widehat{y}$ are the x and y coordinates of the sensor node with the global minimum value $\widehat{v}$.

The choice of v_i values used by an ith sensor node in the application of function $\mathscr{M}$ depends on the requirements of the application. It should be noted that a sensor node can only use its local information (such as *identifier*, sensor value, and physical location) and other global data available from previous iterations of $\mathscr{M}$. For our goal of identifying various features, we augment (or transform) these input values in such a way that the global `MIN` returned by $\mathscr{M}$ corresponds to one of the local minima or an edge of an *active region*.

Function $\mathscr{M}$ can be applied to the value computed by ϕ_i, which is a function of the sensor values and their location. The codomain of function ϕ_i denotes the set of values it can take. We assume that the cardinality of the codomain of ϕ_i is large enough that the probability of computing the same ϕ_i by two sensor nodes is negligible and thus a unique sensor node $\widehat{i}$ is chosen. The time-complexity of $\mathscr{M}$ is proportional to the number of bits used to encode ϕ_i and hence it is proportional to the logarithm of the cardinality of the codomain. However, as all sensor nodes transmit simultaneously in a dominance-based MAC protocol, the time required for the application of $\mathscr{M}$ over a network is independent of the number of sensor nodes. The abbreviations section summarizes the notations and symbols that we use in the following sections where we define a set of functions to extract different *features* of the *field*.

4. Feature Extraction Using Augmenting Functions

In this section, we describe in detail a set of *augmenting functions* that can extract an approximate but faithful representation of various *features* in the *field* by applying simple transforms on the sensor values.

We show that this process can be done with a limited number of broadcast messages or flooding in case of SBD or MBD, respectively.

4.1. $\mathscr{A}_\beta$ Distance Augmenting Function. As described earlier, a global maximum value in a sensor *field* can be easily found applying function $\mathscr{M}$. However, finding the spread or boundaries of this peak (local maximum) is not trivial. If we modify the MAC protocol such that the sensor node with the global maximum does not participate in the next *round*, there is still a high chance that one of the adjacent sensor nodes to the peak will become the next local maximum. On the other hand, we might have to predefine a neighborhood range around the peak that should be excluded in the next cycle to make sure that another local maximum will be the new global maximum. In this case, choosing the size and shape of the neighborhood is also a challenge.

We demonstrate the process of finding an adjacent local minimum with a simple 1D Gaussian *field* example, shown in Figure 3. Using this example we show that, by utilizing the *augmenting function* over the input signal, the adjacent local minimum becomes the global minimum.

In this example, the *field* consists of three peaks and the highest peak (the global maximum) lies in the center. Finding the spread of this global maximum is not trivial as the global minimum point can be one of those sensor nodes near the borders of the *field*. It is important to suitably modify the process of identifying the extrema such that a local minimum adjacent to a peak (*adjacent valley*) can be found. For this purpose, we observe that an *adjacent valley* should have a low value and its distance from the peak should also be small. Hence, each value in the *field* is transformed (multiplied) with the distance from the peak. With this transformation, the points located farther from the peak are associated with

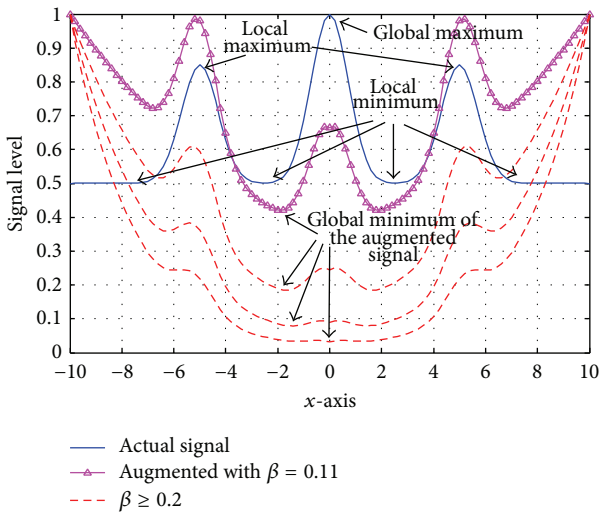

FIGURE 3: An example of *augmented function* $\mathscr{A}_\beta$ of the normalized input signal with various β in range of [0.11, 0.5) in 1D domain.

higher values (compared with its sensed value) and only points with lowest sensed value and smallest distance from the peak can become the global minimum in the augmented *field*. It is possible that this global minimum is a point in an *adjacent valley*. The boundary of this peak is found with just two *rounds* of executing $\mathscr{M}$ function: (i) finding the global maximum in the original *field* and (ii) identifying the global minimum in the augmented *field*.

For 2D Gaussian *fields*, the 1D approach described above can be directly applied. Different *active regions* of the *field* are found by excluding the sensor nodes lying inside the identified *active region* from participating in the next *rounds*. The process of finding *active regions* is shown in Algorithm 1. Initially, function $\mathscr{M}$ is used to find the global maximum $s_{\max}$, and then a circular area around the identified peak is filtered out. The radius of this filtering circle is set as the distance of an *adjacent valley* from the peak. The value and the location of the *adjacent valley* ($S_{\text{adj-valley}}$ and $d_{\text{adj-valley}}$) are found by one application of $\mathscr{M}$ function (line 9 in Algorithm 1).

Each sensor node uses function ϕ_β as an input to $\mathscr{M}$. Function ϕ_β takes into account two properties of the *field*, sensed value s and sensor's proximity to the peak d, and is defined as follows:

$$\phi_\beta = \nu \times \mathscr{A}_\beta(d), \tag{2}$$

where ν is the scaled value and is defined as

$$\nu = \frac{1+s}{1+s_{\max}} \tag{3}$$

and $s_{\max}$ is the value of the global maximum, found at the beginning of the algorithm. $\mathscr{A}_\beta(d)$ is the *augmenting function*, which is formulated as follows:

$$\mathscr{A}_\beta(d) = e^{\beta(d/d_{\max})^2}, \tag{4}$$

where $d_{\max}$ is the diameter of the monitored area and it is used to normalize the distance from the peak and β is a parameter to control the impact of distance on the augmented *field* with respect to the scaled value ν. To ensure that the winning sensor node is located at the adjacent local minimum of the *field*, the priority function ϕ_β is computed so that the distance is exponentially penalized.

Finally, by finding the global minimum over ϕ_β values at all the sensor nodes, we can find a point that lies in the *adjacent valley* with high probability. The distance between the *adjacent valley* point and the peak determines the filtering radius, R_f (line 10 in Algorithm 1). Sensor nodes that are located within the filtering circle refrain from participating in the next iterations. Repeating this procedure helps in finding all the peaks in the region. The algorithm stops when the value of new peak ($s_{\text{new-peak}}$) is lower than a certain user defined threshold π_s, which is a fraction of the global maximum. By finding the peaks and their spread, this approach helps in identifying the location and the number of circular *active regions* in a *field*.

4.2. $\mathscr{A}_\gamma$ Vector Augmenting Function. Our second *augmenting function* is used for cases where we are interested in finding a boundary around the whole *active region*. This approach can be used for a range of applications such as crowd monitoring for smart cities or sleep monitoring for health-care. In this approach, if we assign larger values of ϕ to sensor nodes that lie on the boundary of an *active region*, then the result of applying the $\mathscr{M}$ function over the negation of ϕ value corresponds to the boundary of the *active region*. This is implemented by augmenting the sensor values with a function that grows in a particular direction.

The *vector augmenting function*, $\mathscr{A}_\gamma$, is designed to work with binary *fields*, where the input signal is not smoothly distributed, and two neighboring sensor nodes may have very close or very different measurements. By applying $\mathscr{A}_\gamma$, each sensor node multiplies its measurement with a vector $\vec{u}$. The rationale behind using a direction is to find sensor nodes that sense a high value and are located as far as possible in the direction given by $\vec{u}$, which corresponds to the edge of an *active region* in that direction. To compute function ϕ, sensor nodes transform their locations with a direction as

$$\phi_\gamma = \nu_p \times \mathscr{A}_\gamma(x, y, \theta), \tag{5}$$

where ν_p is a *participation value* which is either 0 or 1 depending on the sensed value being below or above a threshold. Sensor nodes with $\nu_p = 1$ are part of the *active region*. The *augmenting function* is defined as

$$\mathscr{A}_\gamma(x, y, \theta) = e^{\gamma(x\cdot\cos(\theta)+y\cdot\sin(\theta))}, \tag{6}$$

where x and y are the coordinates of the sensor location and θ is the direction given by vector $\vec{u}$.

By using $\mathscr{A}_\gamma$, the value is "projected" in a direction given by the vector $\vec{u}$. The sensor node that has the largest value has a high probability of being located on the border of an *active region* in the direction of $\vec{u}$.

The working of this approach is outlined in Algorithm 2. The algorithm explores the region by choosing random directions defined in a set $\{\overline{v_\theta}\}$. We assume that the seed

```
(1) begin
(2)   Silent ← 0;
(3)   s_max ← M(−φ_s); // find global MAX
(4)   π_s ← a fraction of s_max; // termination condition setting
(5)   while s_new-peak > π_s do
(6)     if Silent ≠ 1 then // not filtered out
(7)       s_new-peak ← M(−φ_s); // find the new peak
(8)       Compute φ based on (2);
(9)       ⟨s_adj-valley, d_adj-valley⟩ ← M(φ_β);
(10)      R_f ← d_adj-valley;
(11)      d_i ← distance between new-peak and node n_i;
(12)      if d_i < R_f then
(13)        Silent ← 1;
```

Algorithm 1: Distance augmenting function $\mathscr{A}_\beta$, executed on each sensor node n_i.

```
(1) begin
(2)   s_max ← M(−φ_s); // find global MAX
(3)   π_s ← a fraction of s_max; // termination condition setting
(4)   {f_θ} ← ∅; // filtered direction set
(5)   if s < π_s then
(6)     v_p ← 1;
(7)   else
(8)     v_p ← 0;
(9)   {v̄_θ} ← a set of θ directions;
(10)  foreach θ ∈ {v̄_θ} do
(11)    if θ ∉ {f_θ} then
(12)      Compute φ_γ based on (5);
(13)      ⟨s_edge, (x, y)_edge, θ_edge⟩ ← M(−φ_γ);
(14)      if (x, y)_edge was found with other direction, θ_x
          then
(15)        {f_θ} ⟵ {f_θ} ∪ θ_x θ + 2ε;
```

Algorithm 2: Vector augmenting function $\mathscr{A}_\gamma$, executed on each sensor node n_i.

for generating these pseudorandom directions is generated by the initiator of the boundary detection process (thus, all the sensor nodes will use the same $\{\overline{v_\theta}\}$ in each iteration). Repeating this procedure in different directions makes it possible to find the boundary of an *active region* by computing the convex hull of the collected locations.

If two angles lead to the same point, it means that any angle over the arc confined by these two angles would result in that point. Thus, this arc can be filtered out from further investigation. Figure 4 shows an example where two angles of $\theta_1 = 31^\circ$ and $\theta_2 = 89^\circ$ lead to find the same location P_2. In this case, there is no need to examine more directions in the region of $31^\circ \le \theta \le 89^\circ$. To further limit the redundant directions from the set $\{\overline{v_\theta}\}$, we introduce a *marginal extension* factor, ϵ. Doing so, the redundant arc will be further confined by $\theta_i \pm \epsilon$. We discuss the impact of ϵ in Section 6.

More iterations of the algorithm leads to more accurate boundaries, but at the cost of more resource consumption. We show in Section 6 that, with the above filtering strategy, we are able to reduce the number of dominance *rounds*, while still building a good description of the *active region*. The worst case for our algorithm happens when the event boundary looks like a perfect circle where new directions will always give new points (considering a very dense deployment of sensors and a marginal extension of $\epsilon = 1^\circ$, in this case, up to 359 individual readings will be collected). However, the results show that the number of readings is usually much less in practice.

4.3. $\mathscr{A}_\delta$ Joint Augmenting Function. As described earlier, the *distance augmenting function* $\mathscr{A}_\beta$ identifies circular *active regions*. For complex *fields*, $\mathscr{A}_\beta$ may need several circular *active regions* to cover a noncircular shape. On the other hand, *vector augmenting function* $\mathscr{A}_\gamma$ only provides a convex hull of all the *active regions*. So a *field* with several isolated *active regions* is identified as one large shape, which may not provide enough insights regarding the structure of the *active regions*.

To find the boundary of an *active region* with noncircular distribution, we devised a new *augmenting function*, $\mathscr{A}_\delta$, that is a composition of $\mathscr{A}_\beta$ and $\mathscr{A}_\gamma$. By applying $\mathscr{A}_\delta$ sensor nodes

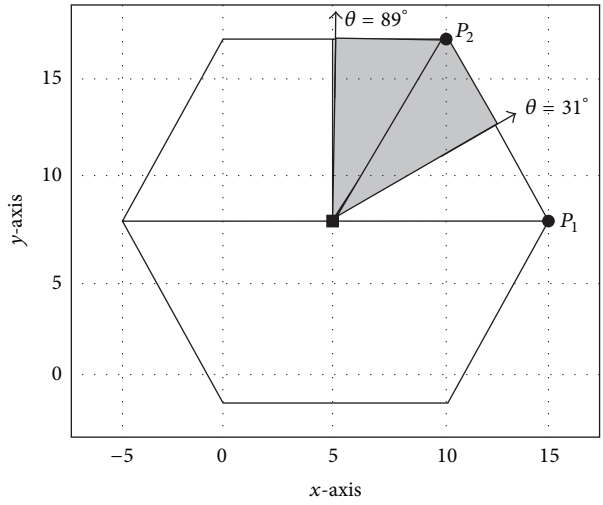

FIGURE 4: An example of boundary detection for a hexagonal-shape event.

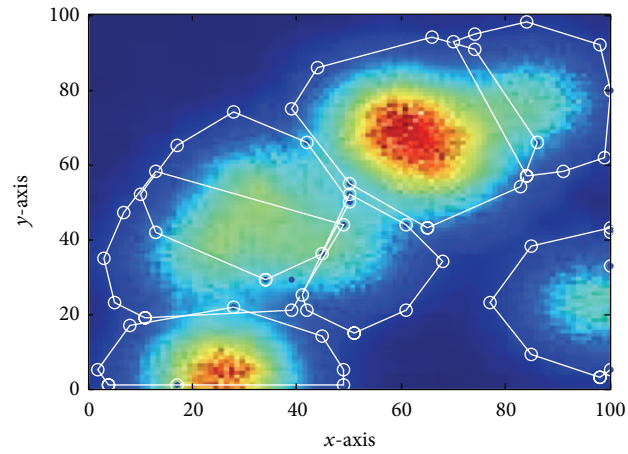

FIGURE 5: An example of boundary detection around the peaks with nonuniform distribution.

that identified with $\mathscr{M}$ function should have the following properties: (i) lying close to the peak and (ii) being located on the edge of the local boundary in a given direction, the value used by each sensor node is then defined as follows:

$$\phi_\delta = \mathscr{A}_\delta \times \frac{\mathscr{A}_\beta(d)}{\mathscr{A}_\gamma(x, y, \theta)}, \tag{7}$$

where $\mathscr{A}_\delta$ is given by

$$\mathscr{A}_\delta = \nu^{-\delta}. \tag{8}$$

$\mathscr{A}_\delta$ is an inverse polynomial of degree δ of the scaled sensed value, ν, and δ is a parameter to guarantee that the low values lying far away from a peak bring a stronger contribution to the ϕ values. As a consequence, for a given value of θ, the point that has the minimum value of ϕ_δ is more likely to lie on the boundary in the θ direction. We sweep the area around a peak with different values of θ. In our evaluation, for θ, we used equal intervals of $\pi/4(\theta \in \{0, \pi/4, \ldots, 2\pi\})$ (smaller intervals will result in better accuracy of boundary at the cost of higher number of broadcast messages.). Thus, after finding a new peak, the locations of the nearest *adjacent valleys* in eight directions are found and the convex hull of all these readings represents the area around that *active region*. This enables us to find complex geometric shapes according to the shape of *active regions* instead of only circles, as shown with an example in Figure 5.

5. Computing Global Extrema in MBD Networks

In this section, we address the challenge of computing `MIN` in a dense network, where nodes are not necessarily confined in an SBD. We consider two different approaches to extend the functionality of $\mathscr{M}$ function for MBD dense networks: (i) clustering the network into several SBDs as proposed in [12] and (ii) a novel approach using concurrent transmissions by taking advantage of the dominance-based protocol.

5.1. Cluster-Based Approach. In the clustering approach, proposed in [12], a topology discovery algorithm is first executed to partition the network such that all nodes in each partition are in the same broadcast domain (*Setup phase*). Then, at runtime, nodes within the same cluster find the minimum sensor reading and communicate these values to their cluster leaders. The cluster leaders form a collection tree with root at the leader node where the query for computing `MIN` was initiated.

The high-level pseudocode of the cluster-based approach is given in Algorithm 3. A minimum virtual dominating set (MVDS), as introduced in [29], has been considered during the setup phase. With MVDS, it is possible to divide the network into several clusters where the nodes in each cluster form an SBD. The cluster leaders (known as black-nodes in MVDS) are responsible for collecting `MIN` in each cluster and forwarding it to the leader node through a collection tree graph known as black-node tree (`B`). The virtual range is used to guarantee that all nodes in a cluster are located in the same broadcast domain. In fact, the MVDS is a distributed algorithm with two main phases. In the propagation phase, it forms the clusters; and, in the response phase, the topology information is delivered to the leader node (the node initiating the aggregation query). Figure 6(a) shows an example of cluster construction using MVDS.

The leader node uses this topology information to schedule the activation time of each cluster to avoid any collision between neighboring nodes that reside in different clusters. To make this approach more efficient, we utilize a heuristic that was proposed for the register interfering graph (RIG) problem [30] to find the *chromatic number* Δ for concurrent active clusters. However, unlike the RIG problem, in our case the number of available colors is not known in advance. This is the number of registers in RIG problem, k.

Function `time slot-assignment(G)` is performed by the leader node to find the value of Δ as shown in Algorithm 4. From basic graph theory it is known that, for an n-node graph $G(V_n, E)$, if the maximum degree of nodes in the graph is d, then it is possible to color the graph with $d + 1$ different

(1) **begin**
(2) **run** MVDS(R_{co}, r) [29] to partition the network (setup phase); // `execute on each sensor node` n_i
(3) **construct** cluster interfering graph $G_i(V, E)$ and black-node tree `B` (response phase); // `execute on leader node`
(4) $\Delta \leftarrow$ `timeslot-assignment(G)`; // `executed on leader node`
(5) **for** $i = 1$ **to** Δ **do**
(6) **execute** one dominance *round* according to assigned timeslot; // `execute on nodes in each active cluster`
(7) **collect data** from black-node tree `B`; // `execute on leader node`
(8) **disseminate** global `MIN` to the network; // `execute on leader node`

Algorithm 3: Cluster-based approach.

(1) **begin**
(2) $\{C\} \leftarrow \emptyset$; // `possible chromatic numbers set`
(3) $k_{min} = 0, k_{max} = d + 1, continue = 1, k\text{-}colorable = 0$;
(4) **while** $continue == 1$ **do**
(5) $k = \left\lceil \frac{k_{max} - k_{min}}{2} \right\rceil + k_{min}$;
(6) **if** $k\text{-}colorable == 1$ **then** // `feasible to color the graph with` k `number based on the heuristic in` [30]
(7) $k_{max} = k$;
(8) **if** $\{C\} \neq \emptyset$ **then**
(9) **foreach** $k_i \in \{C\}$ **do**
(10) **if** $k \geq k_i$ **then**
(11) $continue = 0$;
(12) $\{C\} \leftarrow k$;
(13) **else**
(14) $k_{min} = k$;
(15) $\Delta \leftarrow \min\{C\}$;

Algorithm 4: Function `timeslot-assignment(`G_i`)`, executed on the leader node.

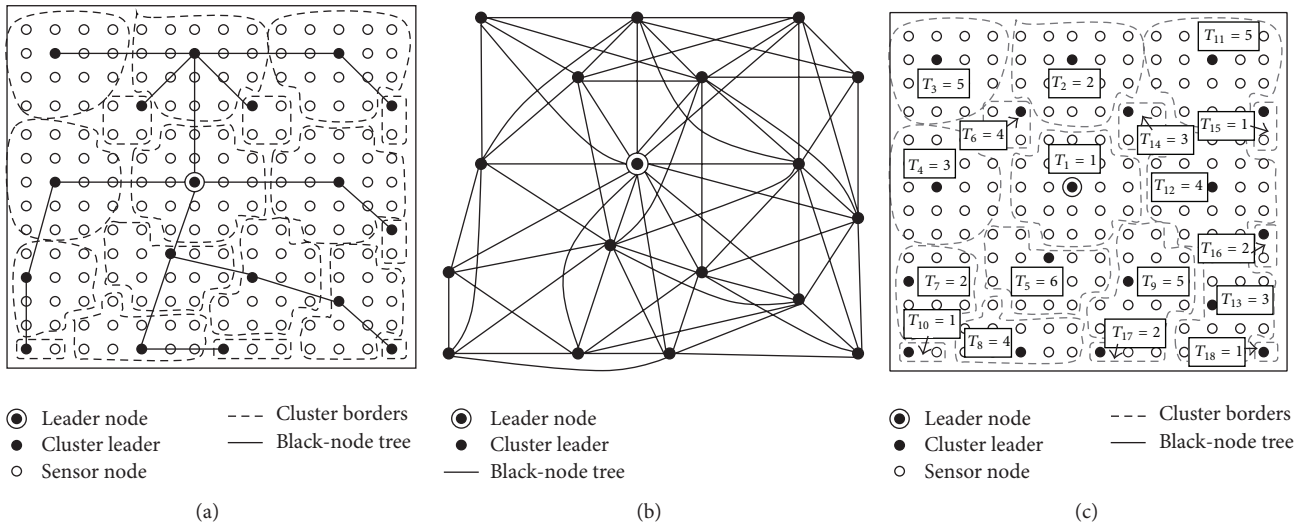

Figure 6: Cluster-based approach: (a) cluster formation and black-node tree (`B`) construction. $R_{co} = 5$ and we assume that $r = R_{co}/2$. With the given virtual range r, 18 clusters are constructed; (b) cluster interfering graph G_i (with $d = 14$). This graph is used by the leader node to compute the activation time slot for each cluster. Two clusters are assumed to be interfering if the distance between the cluster leaders is less than $3 \times r$; (c) time slot assignments (the chromatic number for the interfering graph is 6).

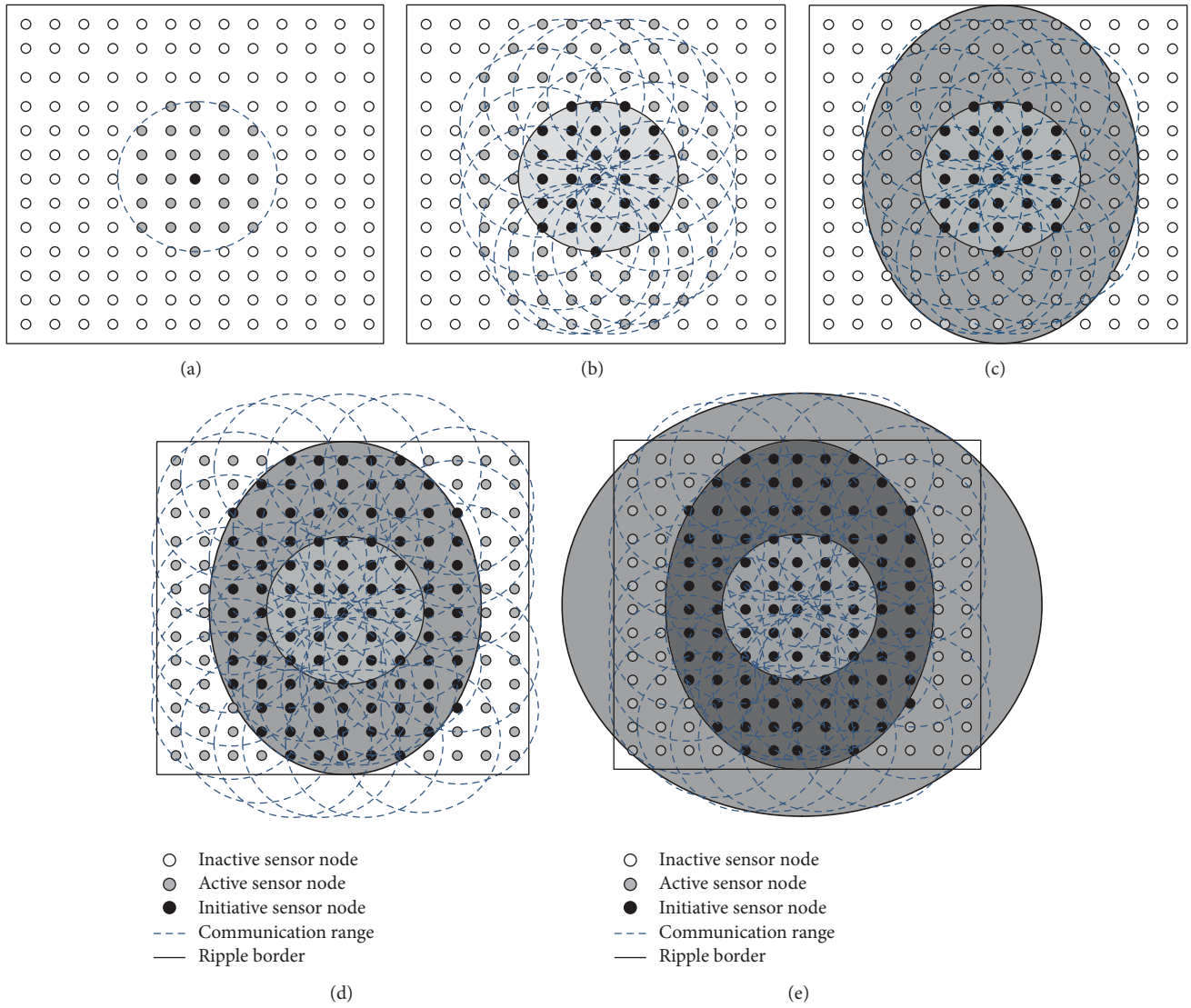

Figure 7: Ripple propagation throughout the network: (a) an initiator sensor node that signals start of a tournament. Sensor nodes within the communication range of this node then perform one tournament; (b) nodes participating in the previous tournament become initiator nodes in this round; (c) and (d) ripple moves toward the border of the network; and (e) all sensor nodes are activated.

colors [31]. In our modified heuristic, the leader node initiates the process of finding Δ by setting k to $\lceil (d+1)/2 \rceil$ and then restricts the search area $[k_{\min}, k_{\max}]$. If the current k is able to color the graph (line 6 in Algorithm 4), the algorithm will add k into the set of possible chromatic numbers $\{C\}$ and updates $k_{\max}$; otherwise, it updates $k_{\min}$. The algorithm terminates when the current k is larger than an element in the set $\{C\}$. For the example given in Figure 6, with $d = 14$ the algorithm finds $\Delta = 6$ after four assignments of k.

5.2. Ripple-Based Approach. The ripple-based approach aims at eliminating the setup overhead in the cluster-based approach and mitigating the initialization cost by using a distributed algorithm to compute the global extrema. It uses the concurrent transmission property of the dominance-based protocol to find the `MIN` in an MBD network. The details of the ripple-based approach are provided in Algorithm 5. A good analogy for this algorithm is a ripple's propagation on a water surface and Figure 7 illustrates this approach with an example. Initially all sensor nodes are inactive (n_i-*active* = 0);

```
(1) begin
(2)   n_i-active = 0, i = 0;
(3)   while i ≤ 2 × ⌈D/R_co⌉ do
(4)     i++;
(5)     if n_i-active == 1 then broadcast the query signal;
(6)     else if n_i receives a query signal then
(7)       n_i-active = 1;
(8)     execute one dominance round;
```

Algorithm 5: Ripple-based approach, executed on each sensor node n_i.

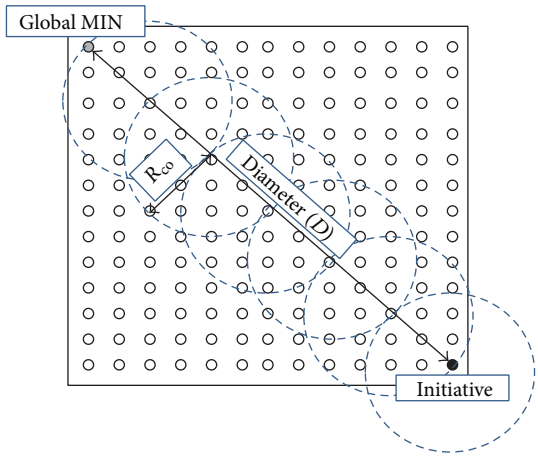

Figure 8: An example of the worst case scenario to determine the required number of tournaments ($\mathcal{N}$). Assuming $D = 12\sqrt{2}$ and $R_{co} = 2\sqrt{2}$, the number of tournaments will be $\mathcal{N} = 2 \times \lceil D/R_{co} \rceil = 12$, that is, two times the number of circles depicted in this figure. In fact, it takes six tournaments to activate the node with MIN and another six tournaments to receive back the global MIN by the initiative sensor node.

that is, they do not participate in any tournament. The process is initiated by a sensor node broadcasting a query for computing the global MIN. We assume that the query signal is a modulated wave which is used for both synchronization and starting a new tournament. Sensor nodes that receive this signal (i.e., in the initiator node's range) perform one tournament of dominance protocol to find a MIN value. Since only a subset of all nodes are participating in this stage, the resulting value is a local MIN. Immediately after finishing the first tournament, all sensor nodes, which had participated in the last tournament, initiate another tournament of the dominance protocol by sending a new query signal (with the new MIN value). This tournament in turn activates those sensor nodes, which are situated within the communication range, to change their status from idle to tournament state. This procedure continues until all nodes get activated.

To compute the global MIN in an MBD network, the tournament needs to be run in worst case $\mathcal{N} = 2 \times \lceil D/R_{co} \rceil$ times where D is the longest distance between two nodes belonging to the network (e.g., network diameter of the given square shape network) in number of hops and R_{co} is the communication range of sensor nodes. Note that this is an upper bound on the number of tournaments that needs to be performed till all nodes in the network become aware of MIN. Figure 8 reveals the rationale behind this process with an example. This bounding event occurs when the sensor node with the lowest value and the initiator node are located at opposite ends of the network. Hence, it takes twice the length of D/R_{co} to assure that the MIN value is propagated to all sensor nodes.

The $\mathcal{N}$ times execution of the tournament is corresponding to one application of $\mathcal{M}$ function or simply one *round* in MBD network. At the end of each *round*, the winner node broadcasts a packet which includes its location information and its identifier. This message is then flooded throughout the network, using the ripple-based approach, such that all nodes have the same knowledge about the properties of the winning node (note that, in SBD, this information is broadcasted once by the winner).

Figure 9 shows an example of computing MIN in an MBD network, where the place of each node is represented by its measured value. In this example, we assume that the node located at the center of the network initiates the query for computing MIN (see Figure 9(a)). Sensor nodes in the communication range of the initiator execute one tournament. Since nodes that are located in the border of the communication range of the initiator are not reachable by each other, border sensor nodes find different MIN values according to their location.

Figure 9(b) shows the result of finding MIN after execution of the first tournament. The two local MIN values are propagated in the first tournament (1 and 2). In the next iteration, all activated nodes perform the tournament and update their MIN values (see Figure 9(c)). Finally, after five tournaments, the global MIN is known throughout the network.

However, the multihop protocol described above can suffer from the hidden terminal problem. As dominance-based protocols allow nodes to listen to MIN value even from simultaneous receptions, hidden terminals in this case can cause a device in the middle to learn a spurious MIN value.

To elaborate, consider three sensor nodes N_1, N_2, and N_3 (with measured values 1, 2, and 3, resp.) as depicted in Figure 10. According to dominance-based protocols [9], during the tournament a sensor node transmits its measured value bit-by-bit. Upon knowing the existence of other nodes with a smaller measured value, it stops sending the rest of the bits. Now consider the scenario given in Figure 10(b). Node N_3 loses the tournament after receiving the smallest value from node N_1. Since nodes N_1 and N_2 are not able to communicate directly, node N_2 proceeds sending its last bit, which is 0 and node N_3 mistakenly perceives MIN in the network as 0.

To avoid this problem, we consider two slots for each bit transmission as proposed in [32]. The first slot is dedicated for the bit transmission itself and the second slot is dedicated for the retransmission of the bit detected during the first slot. If the current bit of a sensor node is dominant, there is no need to retransmit this bit for the second time, but if the bit is recessive and the sensor node detects a dominant bit during the transmission slot, it retransmits a dominant bit in the following retransmission slot (see Figure 10(c)). This bit-level redundancy solves the problem and leads to the correct perception of MIN in the network. Obviously, it will increase the tournament duration.

6. Evaluation

We evaluated our proposed approaches in different simulated scenarios by considering the following three metrics.

Number of Rounds. A sensor node gets channel-access permit to broadcast a message in one *round* of execution of $\mathcal{M}$

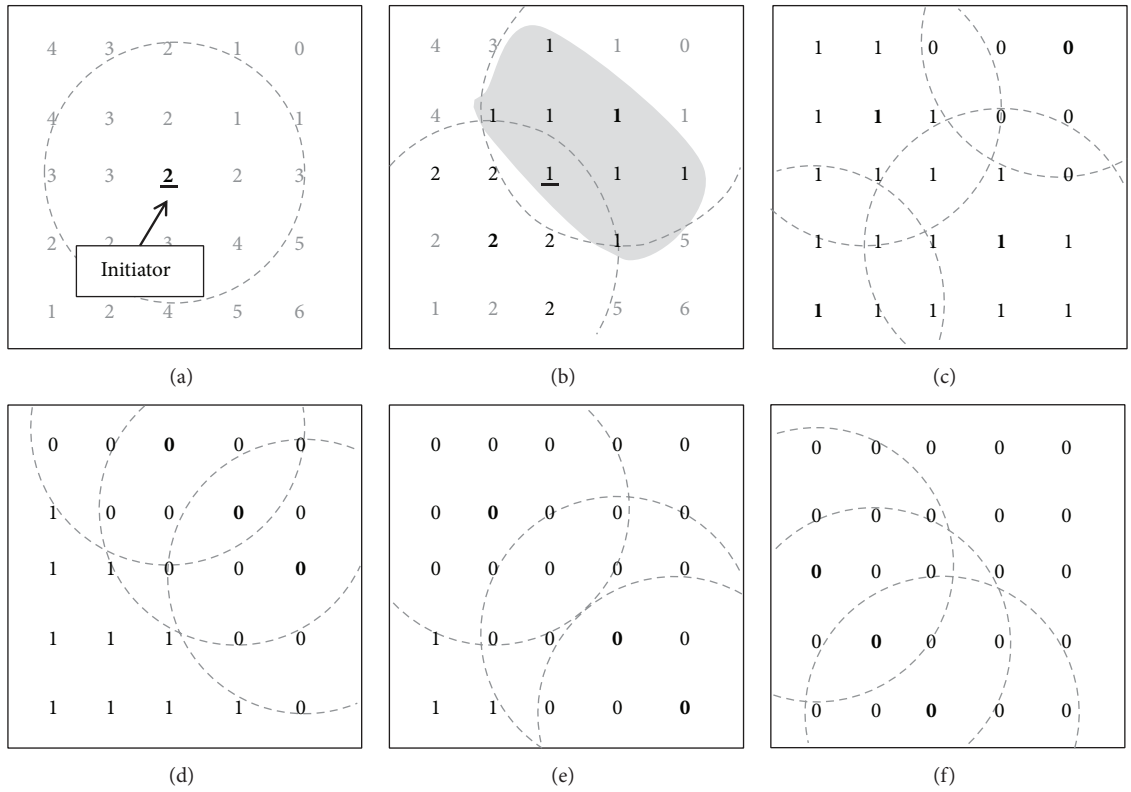

Figure 9: Compute `MIN` in the MBD network. The evolution of determining the `MIN` value is shown in (a)–(f). The network diameter is $D = 5$ and communication range is $R_{co} = 2$. Pale numbers show inactive sensor nodes; dotted circles show the communication range in an SBD; the effective sensor nodes are shown in bold.

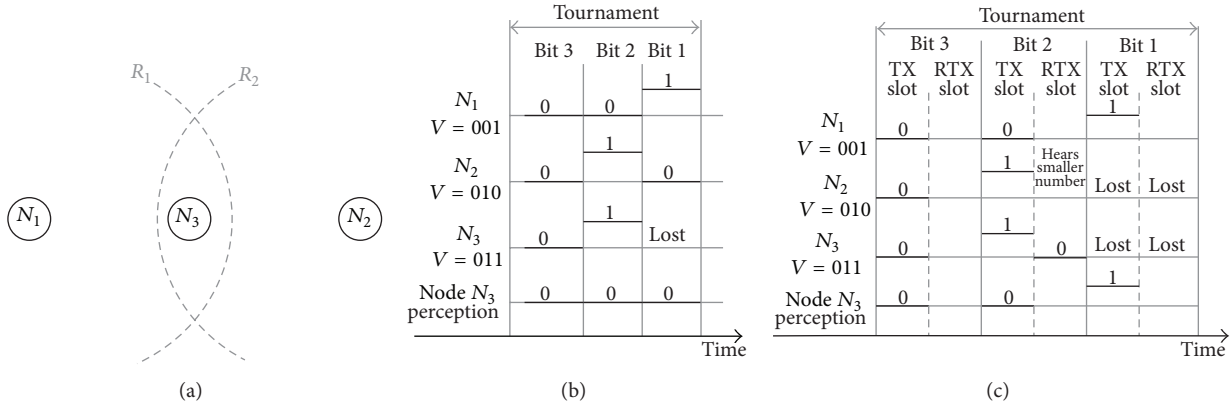

Figure 10: Example of hidden terminal problem during a tournament execution (we assume three priority bits): (a) network topology, (b) execution of tournament that leads to an error in node N_3, and (c) execution of tournament, including an extra time slot for bit retransmission.

function if its value is the global minimum. Hence, the number of *rounds* corresponds to the number of broadcast messages or flooding.

Accuracy. It represents the fraction of sensor nodes that declare themselves to be located in the *active region*(*s*) N_{det} to the number of sensor nodes that truly lie in the *active region*(*s*) N_{true}:

$$\text{Accuracy} = \frac{N_{det}}{N_{true}} \times 100\,(\%)\,. \tag{9}$$

In cases where the detected area is larger than the actual *active region*, ($N_{det} > N_{true}$), accuracy is more than 100%, which signifies the overestimation of the *active region*.

Execution Time. It is the time needed to extract different features according to the proposed *augmenting functions* and is computed as

$$\text{Exection Time} = \text{Number of Rounds} \times T_{\mathscr{M}}, \tag{10}$$

where $T_{\mathscr{M}}$ is the execution time of $\mathscr{M}$ function. In SBD networks, $T_{\mathscr{M}}$ is equal to one tournament duration. However,

in MBD networks, $T_{\mathscr{M}}$ depends on the approach used for computing the global extrema (as discussed in Section 5).

We first show the process of computing $T_{\mathscr{M}}$ in MBD networks and then evaluate the feature extraction techniques for various scenarios.

6.1. Computing the Execution Time of $\mathscr{M}$ Function, $T_{\mathscr{M}}$. As stated earlier, the application of $\mathscr{M}$ function results in identifying the global MIN over values of sensor nodes in a broadcast medium. We considered a network of size 100×100 sensor nodes in a shared medium. We change the communication range of sensor nodes R_{co} to study the performance of the new ripple-based and the classical cluster-based approaches in scenarios where the communication range of sensor nodes would divide the network into multiple broadcast domains. For the cluster-based approach, where clusters are built based on a virtual range, we set the range of virtual communication, r, to be always half of the actual communication range R_{co} of sensor nodes.

To have a precise computation of a tournament round, we consider the timeouts given in [33]. As illustrated in Figure 10, to solve the hidden terminal problem in the ripple-based approach, it is needed to send each bit twice during the tournament. Considering 110 μs as the length of one bit exchange [33], the tournament takes $110 \times n_{prio} + \eta\,\mu$s for the cluster-based approach and $2 \times 110 \times n_{prio} + \eta\,\mu$s for the ripple-based approach where n_{prio} is the number of priority bits and η is the time overhead imposed by the MAC protocol during the tournament (this extra time overhead is mainly due to synchronization and hardware shortcomings). For $n_{prio} = 15$, the value of extra time overhead is $\eta = 3099\,\mu$s [33]. Accordingly, the tournament takes 4749 μs and 6399 μs for cluster-based and ripple-based approaches, respectively.

Table 1 shows the number of referenced tournaments along with the time needed to find the global extremum in an MBD network. In this computation, we took into account the time needed to aggregate data in case of cluster-based approach, as well as the time required for flooding the local information of the winner in the ripple-based approach. However, in both cases we assume the best case scenario. For data aggregation through the black-node tree, we first compute the maximum degree d of the cluster interfering graph (see Figure 6(a)). To prevent any collision, it is considered that we need to have at least d time slots to collect data in the leader node. The duration of the time slot is considered to be the time needed to transmit a 128 byte-size packet. Assuming the IEEE 802.15.4-compliant MICAz's radio [34] with data rate of 250 Kb/s, the time-slot duration (T_s) is then 4096 μs.

For the ripple-based approach, we compute the flooding time according to the theoretical lower latency bound given in [35]. Given that nodes transmit concurrently, the theoretical lower latency bound in a network with size h hops is $h \cdot (T_s + T_d)$. The number of hops is computed as $h = \lceil D/R_{co} \rceil$, where D is the network diameter and R_{co} is the communication range of sensor nodes. T_d is the radio processing delay which is in the order of few microseconds and is determined by the radio; for simplicity, we ignore T_d in our computation.

Table 1: Computation time of MIN value in cluster-based versus ripple-based approach.

Communication range	Cluster-based (R_{co}, r)		Ripple-based (R_{co})	
	Tnmt[a]	Time (μs)	Tnmt[a]	Time (μs)
$R_{co} = 14$, $r = 7$	12	159388	8	96248
$R_{co} = 20$, $r = 10$	10	137602	5	64763
$R_{co} = 28$, $r = 14$	10	141698	4	50172
$R_{co} = 34$, $r = 17$	11	150543	3	39677
$R_{co} = 40$, $r = 20$	10	137602	3	35581

[a]Number of referenced tournaments.

As expected, increasing the communication range results in faster computation of MIN with the ripple-based approach. Also this method needs a smaller number of tournaments compared to the cluster-based approach. However, in the cluster-based approach, we do not observe the same continuous descending trend in the number of referenced tournaments. The reason is that the placement of the clusters plays an important role in the construction of the black-node tree and cluster interfering graph that affects the d value.

It should be mentioned that the given time for both approaches is slightly optimistic. In cluster-based approach, we do not consider the extra overhead imposed by the setup phase for cluster formation and similarly in ripple-based approach; we have used the lower bound of the flooding delay as in [35]. Note that, in the cluster-based approach, not all sensor nodes become aware of MIN, while, in the ripple-based approach, which is a fully distributed algorithm, all nodes would know MIN.

6.2. Identifying the Active Regions. We considered the same network settings as described in the previous subsection, with size of 100×100 sensor nodes. First, we show the performance of *distance augmenting function* to identify circular *active regions*. For evaluating our approach, we generated various scenarios with several *active regions*. The *active regions* may or may not overlap resulting in complex *fields*. The details of the scenarios are provided in Figure 24.

In each *round* of execution, after finding the global maximum, all the sensor nodes compute local values of ϕ_β according to (2). Figure 11 illustrates the number of circular *active regions* with different termination rules for scenario sc1. Increasing the threshold level helps the algorithm to converge faster in smaller number of *rounds*, but at the cost of reduced accuracy.

Figure 12 shows number of *rounds* and the corresponding obtained accuracy for different threshold values. We believe that the property of the *field* plays an important role in the number of required *rounds* and the accuracy of the detection. In some scenarios, increasing the termination threshold reduces the number of *rounds* as well as the overestimated area. This happens for scenarios sc1, sc3, and sc6, where the *active regions* can overlap. However, for scenarios sc2, sc4, and sc5, where either few *active regions* exist or the *active regions* are far apart, the number of *rounds* and accuracy remain almost the same for all termination conditions. The existence

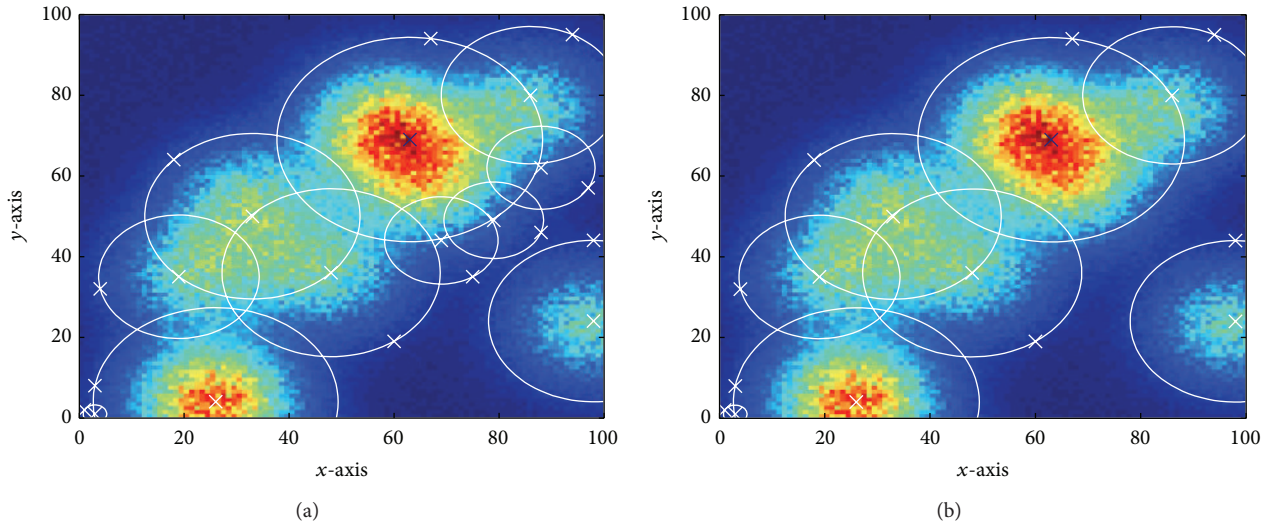

Figure 11: The effect of termination threshold, π_s, on the detection of *active regions*. The algorithm terminates when a new detected peak is (a) 20% and (b) 30% of the global maximum value.

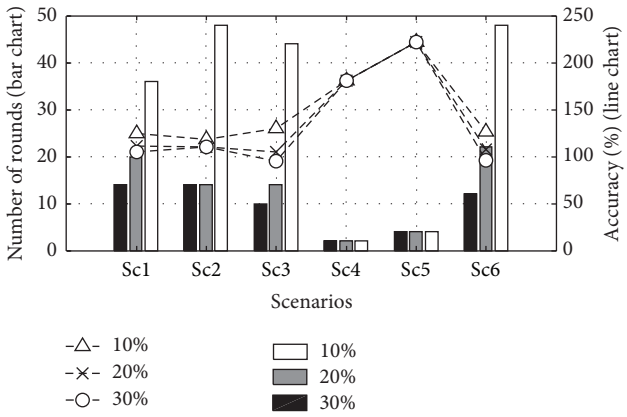

Figure 12: The number of *rounds* and the accuracy of *active region* estimation in different scenarios for various $\pi_s = \{10\%, 20\%, 30\%\}$ of the global maximum.

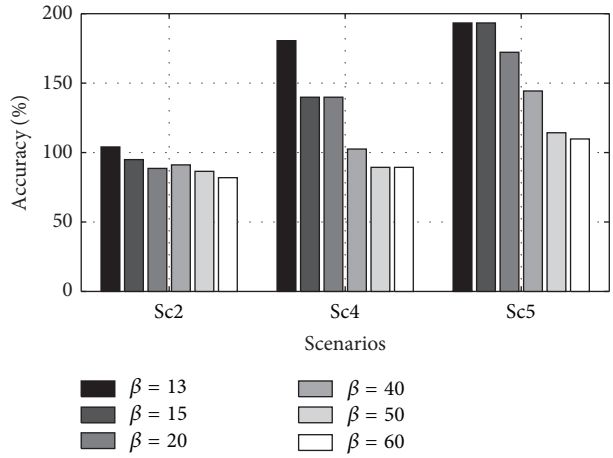

Figure 13: The effect of β on the accuracy.

of isolated peaks in the *field* results in identifying all *active regions* by a fixed number of circular sections. The high value of overestimation in sc4 and sc5 is due to the steepness of the peak in the *field*. This steepness causes an overestimation in the filtering radius which in turn leads to much larger (squared) overestimation of the circle's area.

We found that the choice of the exponential coefficient β in our given heuristic (4) affects the overestimation of *active regions*. In fact, coefficient β impacts the relative weight of distance from the peak with respect to the value of the *field* at a given point. Choosing an optimal value of β is not possible without the prior knowledge of the *field*, but the order of β can be chosen based on the size of the *field* such that a proper trade-off between the sensor value at a given sensor node and its distance from the peak is maintained. Particularly, β should be chosen such that the impact of distance can be pronounced compared to the distance normalization ($d_{\max}$ in (4)). For an area of 100×100, we found that suitable values of β are in the order of 10. Specific results on the accuracy for three scenarios sc2, sc4, and sc5 are shown in Figure 13. The main goal of this experiment was to investigate the level of overestimation given by $\mathcal{A}_\beta$. It is observed that, for scenarios with isolated peaks, higher values of β improve the coverage estimation accuracy ($\beta = 40$ for scenario sc4 and $\beta = 60$ for scenario sc5). But for scenarios where the spread of peaks overlap, the effect of changing β is less pronounced, as in the case of scenario sc2.

Figure 14 shows the time it takes to detect the location of *active regions* by $\mathcal{A}_\beta$. We examined the performance of cluster-based and ripple-based approaches for different communication ranges $R_{\text{co}} = \{14, 40\}$, which are the extreme ranges (shortest and longest) in Table 1. Interestingly, the execution time of $\mathcal{A}_\beta$ under ripple-based approach is at least 58% faster than that of the cluster-based approach. Increasing

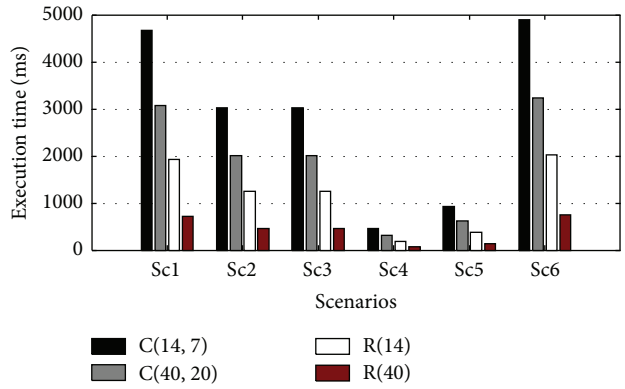

FIGURE 14: Execution time of $\mathscr{A}_\beta$ ($\beta = 10$, $\pi_s = 20\%$) in an MBD network with different communication ranges for cluster-based approach: {C(14, 7), C(40, 20)}, and ripple-based approach: {R(14), R(40)}.

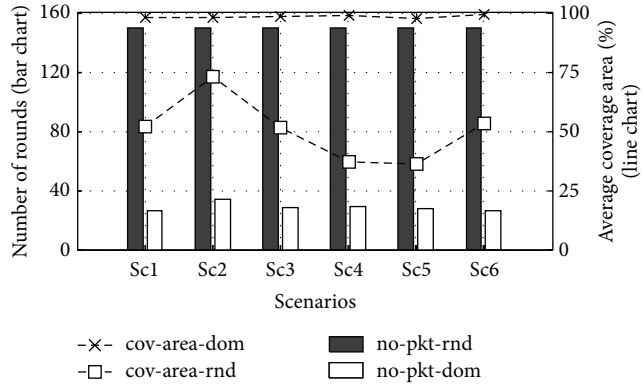

FIGURE 15: The number of *rounds* for $\mathscr{A}_\gamma$ (no-pkt-dom) and random approach (no-pkt-rnd) and the average coverage area estimation for $\mathscr{A}_\gamma$ (cov-area-dom) and random approach (cov-area-rnd) in different scenarios.

the communication range from $R_{co} = 14$ to $R_{co} = 40$ converges the algorithm 63% faster under the ripple-based approach while it improves the cluster-based approach by 34%.

6.3. Convex-Hull around Active Regions. For evaluating the convex-hull technique described in Section 4.2, we convert the *field* to a binary *field* by thresholding, such that a sensor node's value is 1 if the sensed value is greater than 10% of the maximum value, and 0 otherwise. The details of the scenarios are also provided in Figure 25. We set the marginal extension angle to $\epsilon = 1^\circ$ and $\gamma = 1$. We compared our technique with a TDMA approach, where a fixed number of randomly chosen sensor nodes send their measurements. For the random approach, the number of readings was set to 150.

Figure 15 shows the accuracy of our second technique in terms of average percentage of coverage area by running the simulation over 100 iterations. As shown, our technique covers more than 97% of the area in all the scenarios through transmitting 26 to 33 broadcast messages compared with 36% to 72% coverage by 150 randomly chosen packets. Hence, we acquire a more accurate boundary estimation with 77% less broadcast messages.

Increasing the marginal extension angle, ϵ, still provides a satisfactory coverage area estimation, while reducing the number of messages. We tested the performance of our proposed algorithm with different marginal extension angles in all the mentioned scenarios. As shown in Figure 16, by increasing ϵ, the number of *rounds* decreases, since by enlarging the angle, more search space is filtered out and consequently less packets are needed to be broadcast.

However, increasing ϵ might diminish the performance of the algorithm in terms of estimated coverage area. Figure 17 shows the performance degradation by increasing the ϵ up to 90°. Under the latter setting, the proposed algorithm is able to cover around 70% of the *active region*. In addition to the shown value of ϵ in the graph, we also ran the simulation for $\epsilon = \{5^\circ, 10^\circ\}$ and took the standard deviation of the average

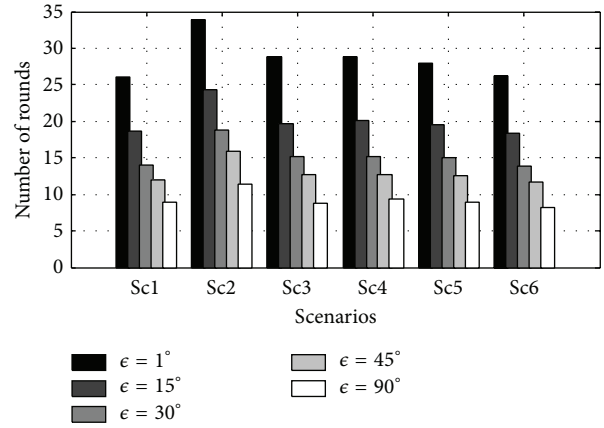

FIGURE 16: The number of *rounds* in $\mathscr{A}_\gamma$ with different marginal extension angle, ϵ.

coverage area for $\epsilon = (1^\circ : \epsilon)$. Since the results for $\epsilon = \{5^\circ, 10^\circ\}$ were very close to the case where $\epsilon = 1^\circ$, the coverage area computed by these values is not shown in the figure. As can be seen in Figure 17, $\epsilon = 15^\circ$ has the smallest standard deviation (smaller than 0.65), which suggests that the coverage area computed by $\epsilon = 15^\circ$ leads to the same coverage area as given by $\epsilon = 1^\circ$, while at the same time using $\epsilon = 15^\circ$ requires much smaller number of *rounds* compared with $\epsilon = 1^\circ$ setting, as shown in Figure 16.

Figure 18 shows the execution time of *vector augmenting function* $\mathscr{A}_\gamma$ in an MBD network. In this experiment we set $\epsilon = 15^\circ$. As expected, the proposed ripple-based approach provides faster detection than the cluster-based approach for any communication range setting. Comparing Figures 14 and 18 reveals that while there are severe fluctuations in the execution time of $\mathscr{A}_\beta$ for discussed scenarios, the execution time of $\mathscr{A}_\gamma$ is almost the same in all scenarios. The reason is that, unlike $\mathscr{A}_\gamma$, the number of required *rounds* in $\mathscr{A}_\beta$ depends on the number and the location of *active regions*. The same reasoning is also applied to Figures 12 and 16.

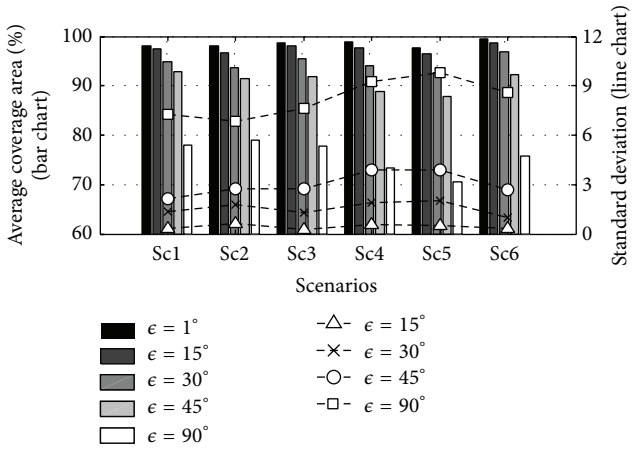

FIGURE 17: The average of estimated coverage area with different marginal extension angle, ϵ, and its standard deviation.

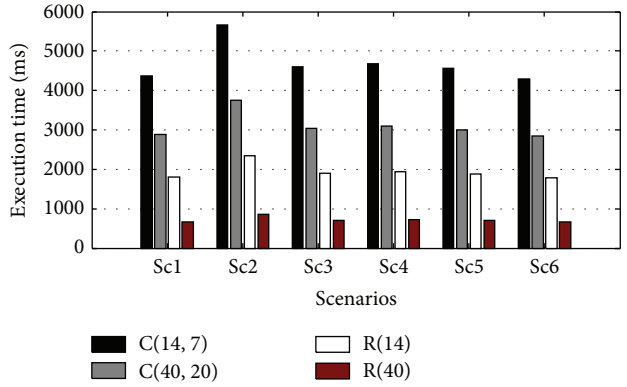

FIGURE 18: Execution time of $\mathscr{A}_\gamma$ ($\gamma = 1, \epsilon = 15^\circ$) in an MBD network with different communication ranges for cluster-based approach: {C(14, 7), C(40, 20)}, and ripple-based approach: {R(14), R(40)}.

6.4. Noncircular Active Regions. As discussed in Section 4.3, $\mathscr{A}_\delta$ helps in finding the boundary of a noncircular *active region*, instead of identifying circular *active regions* from a complex shape.

Figure 19 shows the comparison of the performance of *distance augmenting function* with the *joint augmenting function*. For scenarios where a number of *active regions* lie very close to each other, the number of rounds required by $\mathscr{A}_\beta$ is 20% more than that for $\mathscr{A}_\delta$. For more sparse events, the number of rounds required by $\mathscr{A}_\beta$ is 50% less compared with $\mathscr{A}_\delta$. This happens due to the higher number of iterations that are needed to be performed by $\mathscr{A}_\delta$ to find the boundary of more than one complex shape. It should be noted that $\mathscr{A}_\delta$ is able to find the boundary of a noncircular *active region* and detects it as one region instead of a group of several close *active regions*.

Comparison of *vector augmenting function* and joint augmentation is shown in Figure 20. It is evident that the combined approach helps in demarcating different *active regions* while $\mathscr{A}_\gamma$ detects the overall outer boundary. Figure 21 shows the execution time of the *augmenting functions* $\mathscr{A}_\beta$, $\mathscr{A}_\gamma$, and $\mathscr{A}_\delta$ in an MBD network for cluster-based and ripple-based approaches with different communication range for the scenarios given in Figures 20(a) and 20(b). We observe that for scenarios with overlapping events $\mathscr{A}_\delta$ converge faster, while $\mathscr{A}_\beta$ outperforms other *augmenting functions* for scenarios with sparse events.

We further examined the cluster-based and ripple-based approaches to compare the time needed to extract different features according to the augmenting functions given in Section 4 for a single scenario. Figure 22 shows the elapsed time for different augmenting functions ($\mathscr{A}_\beta, \mathscr{A}_\gamma, \mathscr{A}_\delta$) for the sample scenario sc2 over a 100×100 network. We chose this scenario because, according to the results in Section 6, this scenario includes the most complex *field*. It is evident that for all communication range settings and for all augmenting functions the proposed ripple-based approach outperforms the classical cluster-based approach.

6.5. The Impact of Network Density. Finally, we compared our techniques under various network densities. As our techniques only depend on collecting the global extrema of various *augmenting functions*, the results indicate that increasing the network density has almost no impact on the number of rounds. Figure 23 shows the number of *rounds* required in each technique with respect to the network size. The slight variation in the number of *rounds* is due to the effect of termination condition in $\mathscr{A}_\beta$ and the randomness in the choice of θ in $\mathscr{A}_\gamma$.

6.6. Alternative Flooding Approaches. Many emerging cyber physical systems need mechanisms to share and process data among all devices in the network. The ripple-based approach, proposed in this work, is one of such mechanisms, in the sense that it computes an aggregate quantity over all the devices. Chaos [36] and Low-Power Wireless Bus (LWB) [37] are two other examples of such mechanisms that build their data collection or dissemination capabilities based on the Glossy protocol [35].

Since we aim at obtaining an efficient and scalable algorithm with low time-complexity, we compare the performance of different flooding approaches based on execution time and scalability. Chaos outperforms LWB in terms of completion time [36]. In fact, Chaos is able to collect small amounts of data, such as a 1-byte payload, from a 100-node network within less than 100 ms. We believe that there are two main advantages of our proposed ripple-based approach over Chaos. Firstly, it provides a faster execution time; considering the tournament duration 6399 μs (see Section 6.1), in the ripple-based approach, all nodes will know `MIN` in at most 52 ms within a seven-hop network deployment. Secondly, it offers better scalability; in fact while in Chaos the number of nodes is limited to 856 in a SBD (unless some data compression mechanism is applied), our ripple-based approach does not limit the number of involved devices.

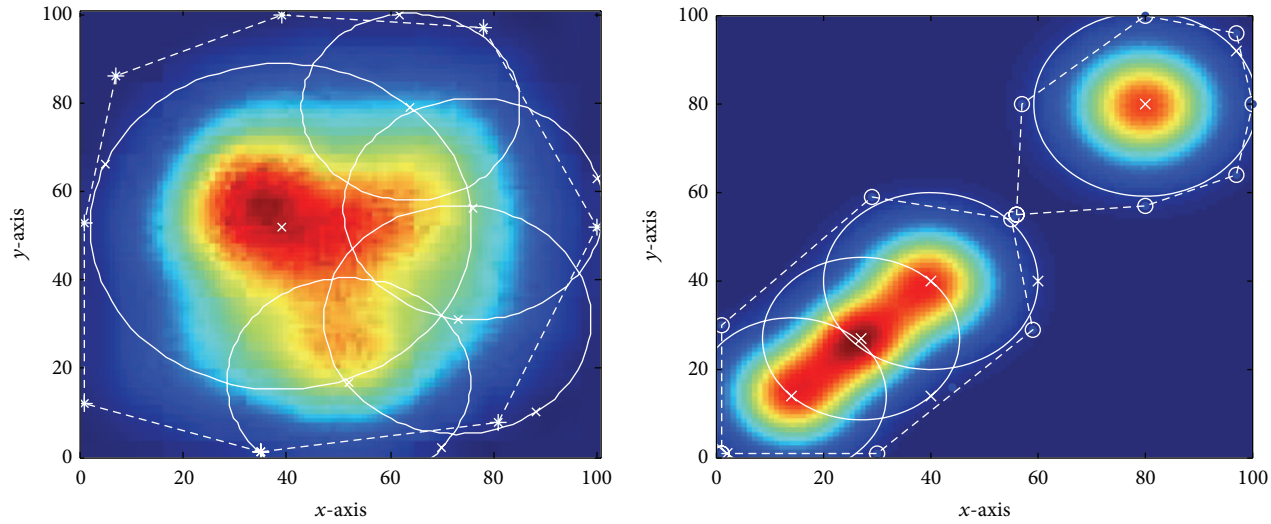

FIGURE 19: $\mathscr{A}_\beta$ versus $\mathscr{A}_\delta$, dash lines show the boundaries provided by $\mathscr{A}_\delta$ and solid lines refer to *active regions* detected by $\mathscr{A}_\beta$: (a) $\beta = 30$, $\gamma = 0.3$; (b) $\beta = 30$, $\gamma = 0.2$, with $\beta = 10$ for $\mathscr{A}_\beta$.

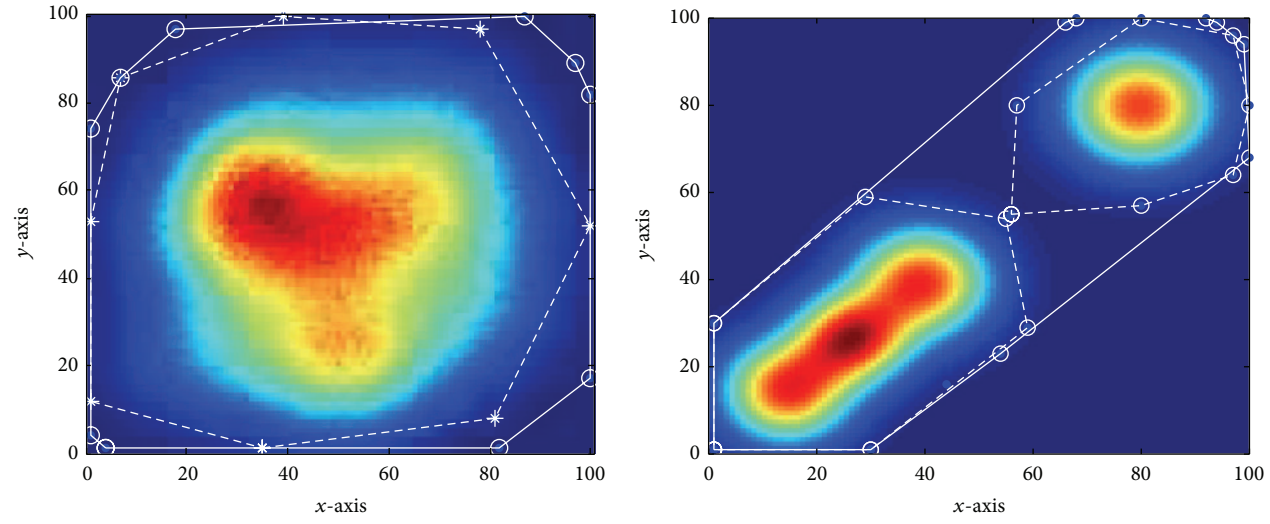

FIGURE 20: $\mathscr{A}_\gamma$ versus $\mathscr{A}_\delta$, dash lines show the boundaries provided by $\mathscr{A}_\delta$ and solid lines refer to that given by $\mathscr{A}_\gamma$: (a) $\beta = 30$, $\gamma = 0.3$; (b) $\beta = 30$, $\gamma = 0.2$, with $\epsilon = 15^\circ$ for $\mathscr{A}_\gamma$.

7. Conclusions

Low-Power Wireless Sensor networks enable many emerging applications that need thousands of sensor nodes for control and monitoring. In this paper, we presented a set of techniques that identify various *features* in the distribution of sensed physical quantities over a dense deployment of sensor nodes. With such simple yet effective modifications, we can obtain the location and shapes of *active regions* with a number of messages proportional to the properties of the *field*. Our feature extraction techniques are efficient in the sense that they collect data from just the sensors that have more effective information for revealing the status of the monitored area.

We then proposed a fully distributed approach to make our feature extraction scale with large dense networks, where sensor nodes lie in a multiple broadcast area, and compared our method with the traditional clustering approach.

There are two main directions for our future work. The first involves investigating the temporal behavior of the extracted features, that is, developing efficient algorithms to detect and predict an evolving spatial *active region*. This may require detection of change dynamics and implement them within the feature extraction algorithm. The second direction

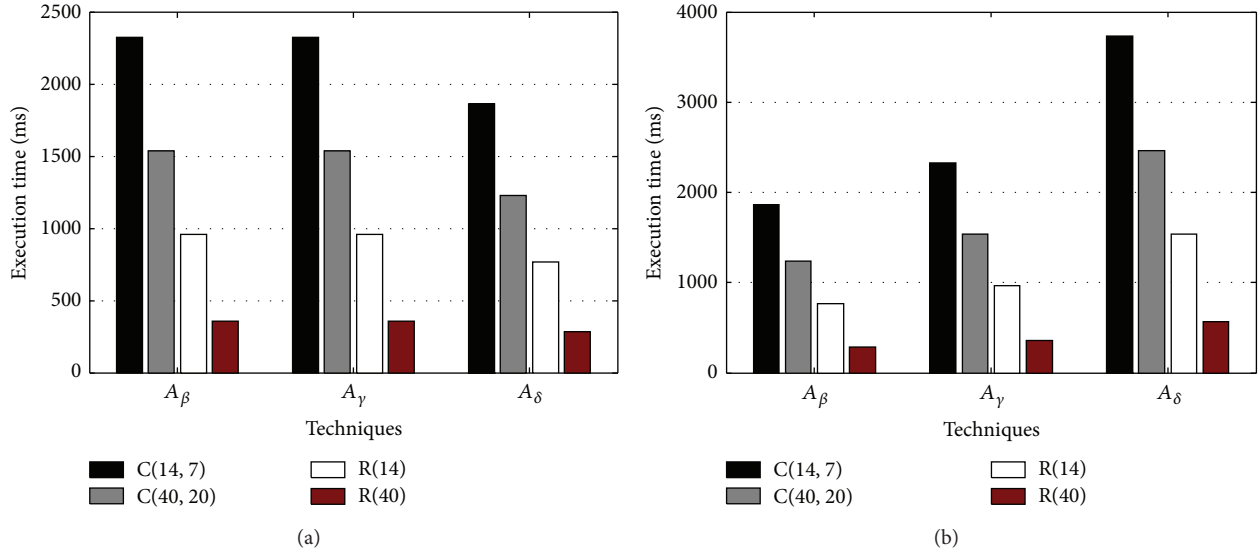

Figure 21: Execution time of different *augmenting functions* in an MBD network with different communication ranges for cluster-based approach {C(14, 7), C(40, 20)} and ripple-based approach {R(14), R(40)}: (a) considering scenarios depicted in Figure 20(a) and (b) considering scenarios depicted in Figure 20(b).

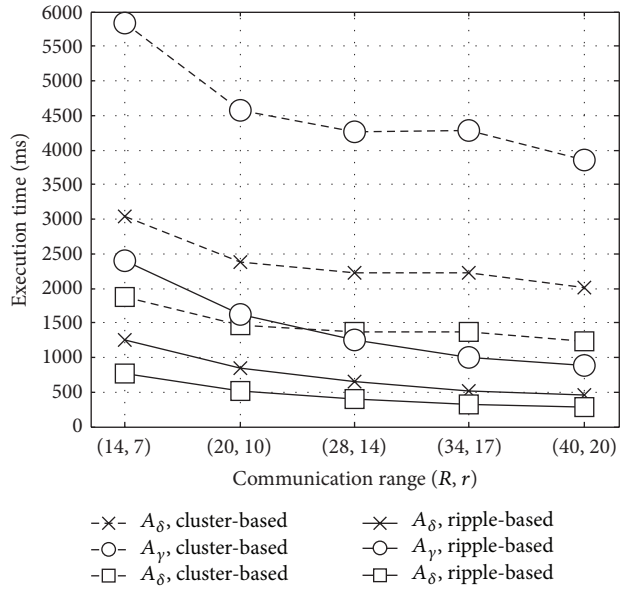

Figure 22: The computation time of various feature extraction techniques ($\mathscr{A}_\beta$, $\mathscr{A}_\gamma$, and $\mathscr{A}_\delta$) for scenario sc2 with $\pi_s = 20\%$, $\epsilon = 15^\circ$, $\gamma = 1$, and $\theta = \pi/4$.

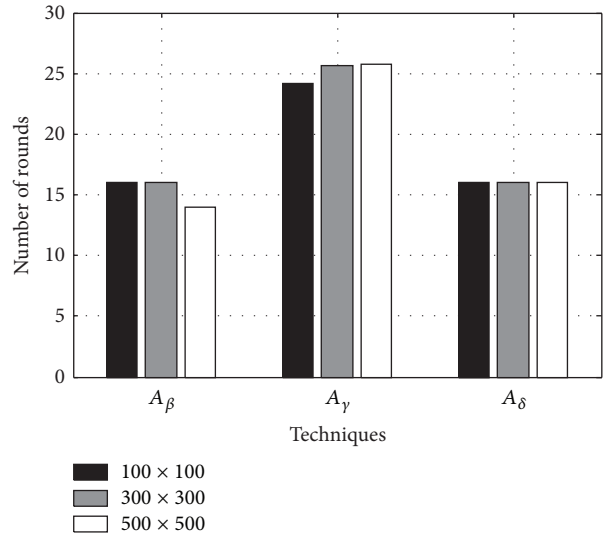

Figure 23: The impact of density on the performance of each technique for scenario sc2.

leads towards the implementation of the algorithm itself. An experiment is required, using a test bed, to implement a dense sensor network to observe a phenomena and verify the results here. Along with this line, improvements to the proposed feature extraction algorithm would be required, by developing a proper synchronization algorithm for the MBD network, in order to increase its reliability. Furthermore, the limitations imposed by communication medium may also have to be considered.

Appendix

Example Scenarios

The *field* is assumed to be sum of the signals generated by a set of active sources (like a heater or car exhaust). The spread from each source is assumed to be Gaussian with the following representation:

$$f(x_i) = k_i e^{-\alpha(x-x_i)^2+(y-y_i)^2}. \quad \text{(A.1)}$$

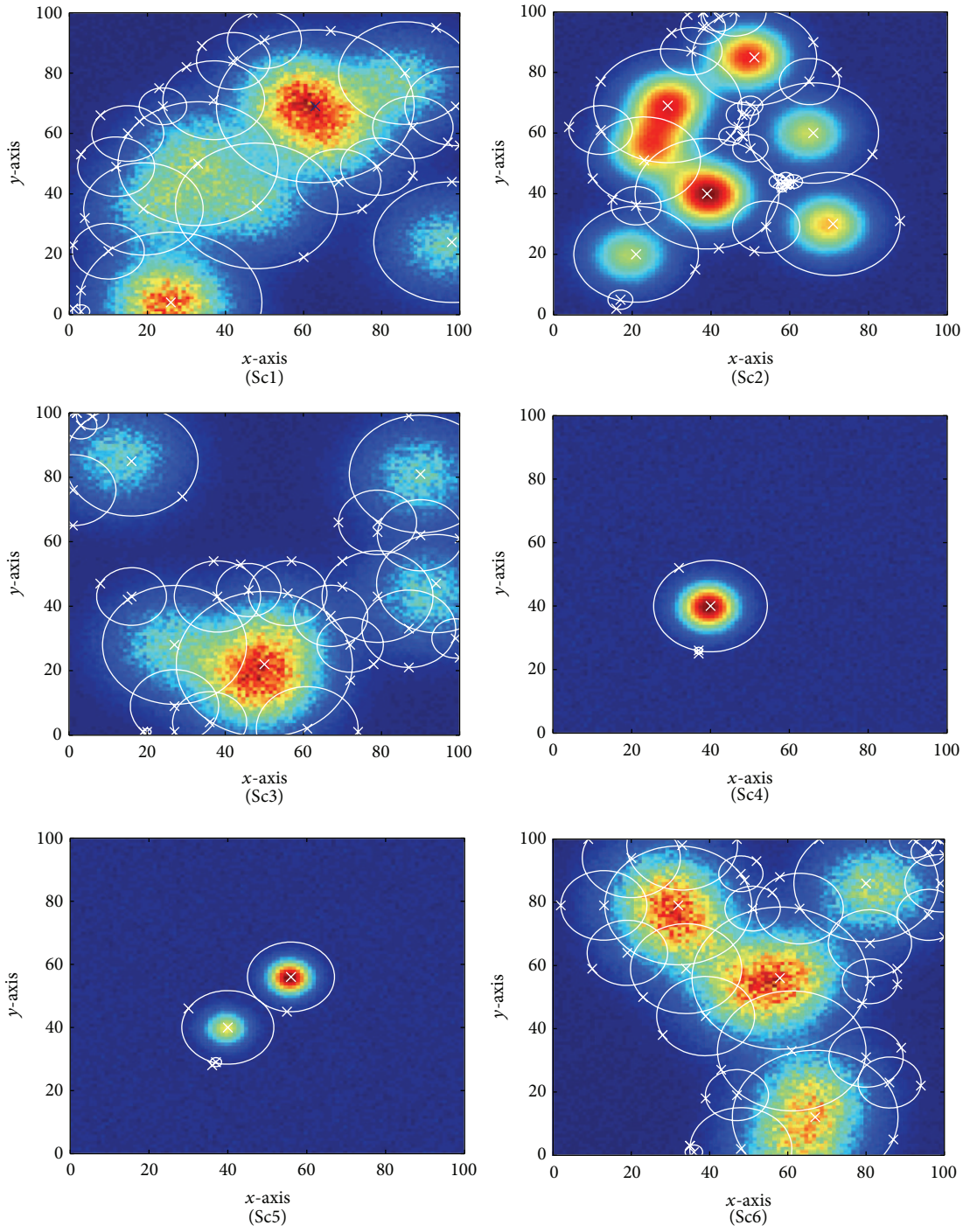

FIGURE 24: Six scenarios with different *active regions*; each sensor node sets the value of $\beta = 10$ to compute its priority—see (2)–(4). Each circle represents one filtering zone that is computed by two readings from the sensor network and excludes sensor nodes located inside the circle from participation in the future iteration(s) of the algorithm. π_s is set to 10% of the global maximum value.

The *field* is then represented by

$$f(x) = \sum_{i=1}^{n} f(x_i) + \mathcal{N}(0, \sigma^2). \quad \text{(A.2)}$$

An example of a 2D *field* is shown in Figure 1, where the z-axis corresponds to sensor values and the spread of the *field* is assumed to be the sum of several 2D Gaussian signals. The scenarios for evaluation are generated by varying the parameters n and k_i in the signal model given by (A.1) and (A.2).

The scenarios which are used in the simulation experiment for evaluating *distance augmenting function* and *vector*

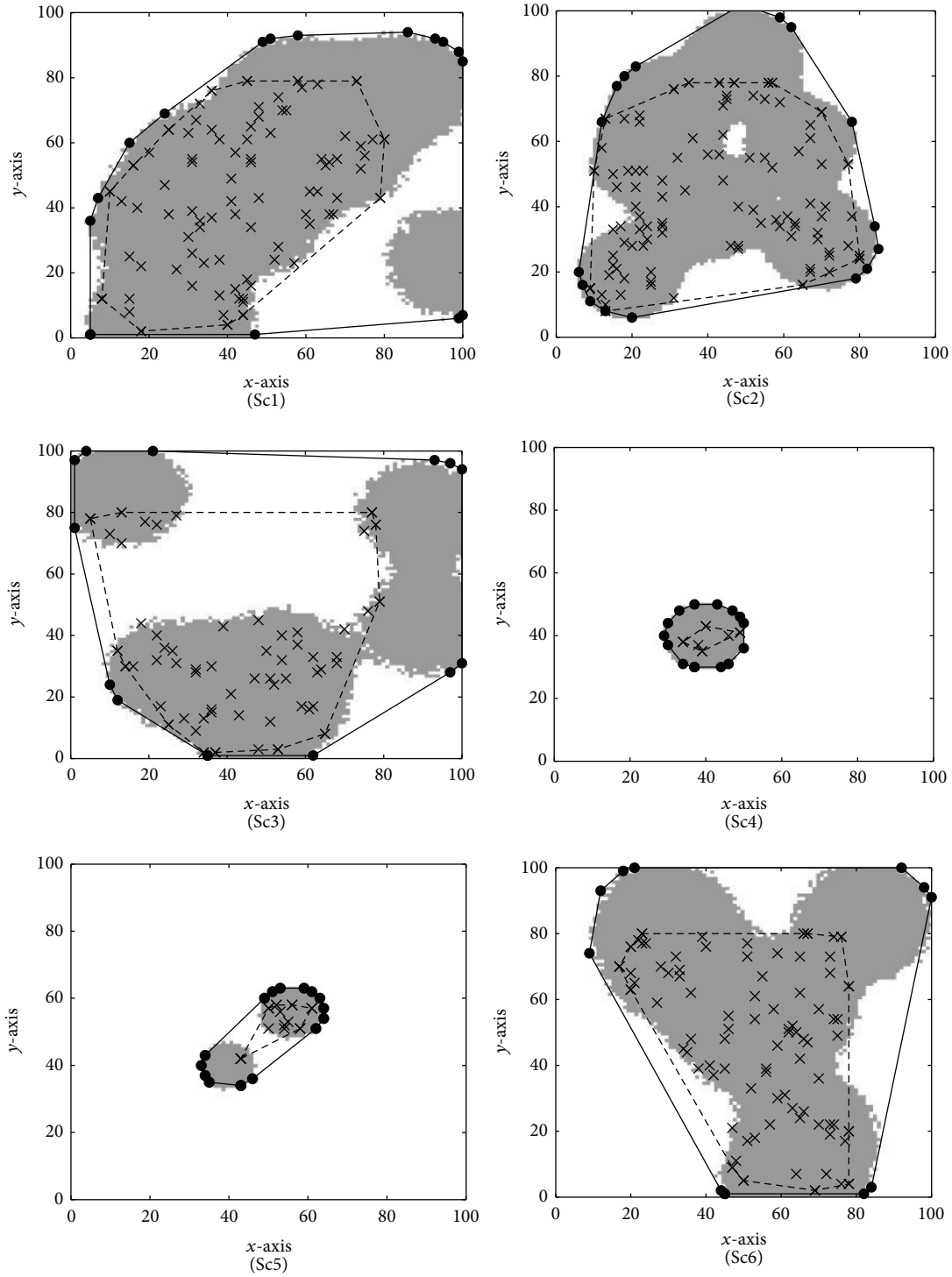

FIGURE 25: Six scenarios with different *active regions*; solid lines show the boundary computed by our algorithm with $\epsilon = 1^{\circ}$ and 40 iterations of the algorithm and dash lines show the boundary computed by the random algorithm with 150 random readings.

augmenting function are given in Figures 24 and 25, respectively.

Abbreviations

Summary of the Symbols and Notations

s_i:	Sensor value measured by sensor node n_i
$s_{\max}$:	Maximum value collected by sensor nodes
$v_i = (1 + s_i)/s_{\max}$:	Scaled sensor value with respect to the maximum value over all sensor nodes
$A_\beta, A_\gamma, A_\delta$:	*Augmenting functions* used for different *feature extraction* techniques
$\mathscr{M}(v_i)$:	Finding `MIN` over all v_i values
$\mathscr{M}(-v_i)$:	Finding `MAX` over all v_i values
$\phi(v_i, x_i, y_i)$:	Function computed by sensor nodes
π_s:	Termination condition.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work was partially supported by the North Portugal Regional Operational Program (ON.2-O Novo Norte), under the National Strategic Reference Framework (NSRF), through the European Regional Development Fund (ERDF), and by National Funds through FCT (Portuguese Foundation for Science and Technology), within Project no. NORTE-07-0124-FEDER-000063 (BEST-CASE, New Frontiers); by National Funds through FCT and by ERDF (European Regional Development Fund) through COMPETE (Operational Programme "Thematic Factors of Competitiveness"), within Projects nos. FCOMP-01-0124-FEDER-037281 (CISTER), FCOMP-01-0124-FEDER-020312 (SMARTSKIN), and FCOMP-01-0124-FEDER-028990 (PATTERN); by FCT and the EU ARTEMIS JU under Grant no. 621353 (DEWI); and also by FCT and ESF (European Social Fund) through POPH (Portuguese Human Potential Operational Program) under PhD Grant no. SFRH/BD/67096/2009.

References

[1] M. Tabesh, M. Rangwala, A. M. Niknejad, and A. Arbabian, "A power-harvesting pad-less mm-sized 24/60GHz passive radio with on-chip antennas," in *Proceedings of the 28th IEEE Symposium on VLSI Circuits Digest of Technical Papers (VLSIC '14)*, pp. 1–2, IEEE, Honolulu, Hawaii, USA, June 2014.

[2] J. A. Stankovic, "Research directions for the internet of things," *IEEE Internet of Things Journal*, vol. 1, no. 1, pp. 3–9, 2014.

[3] M. Vahabi, V. Gupta, M. Albano, and E. Tovar, "Feature extraction in densely sensed environments," in *Proceedings of the 9th IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS '14)*, pp. 143–151, May 2014.

[4] J. J. Liu, W. Xu, M.-C. Huang et al., "A dense pressure sensitive bedsheet design for unobtrusive sleep posture monitoring," in *Proceedings of the 11th IEEE International Conference on Pervasive Computing and Communications (PerCom '13)*, pp. 207–215, March 2013.

[5] J. A. Paradiso, J. Lifton, and M. Broxton, "Sensate media—multimodal electronic skins as dense sensor networks," *BT Technology Journal*, vol. 22, no. 4, pp. 32–44, 2004.

[6] M. Connolly and F. O'Reilly, "Sensor networks and the food industry," in *Proceedings of the 1st Workshop on Real-World Wireless Sensor Networks (REALWSN '05)*, Stockholm, Sweden, June 2005.

[7] N. Pereira, R. Gomes, B. Andersson, and E. Tovar, "Efficient aggregate computations in large-scale dense WSN," in *Proceedings of the 15th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS '09)*, pp. 317–326, IEEE, San Francisco, Calif, USA, April 2009.

[8] R. Nowak and U. Mitra, "Boundary estimation in sensor networks: theory and methods," in *Information Processing in Sensor Networks*, vol. 2634 of *Lecture Notes in Computer Science*, pp. 80–95, Springer, Berlin, Germany, 2003.

[9] A. K. Mok and S. A. Ward, "Distributed broadcast channel access," *Computer Networks*, vol. 3, no. 5, pp. 327–335, 1979.

[10] *Can Specification Version 2.0*, Bosch, Stuttgart, Germany, 1991.

[11] N. Pereira, B. Andersson, and E. Tovar, "Widom: a dominance protocol for wireless medium access," *IEEE Transactions on Industrial Informatics*, vol. 3, no. 2, pp. 120–130, 2007.

[12] B. Andersson, N. Pereira, and E. Tovar, "Exploiting a prioritized MAC protocol to efficiently compute min and max in multihop networks," in *Proceedings of the 5th International Workshop on Intelligent Solutions in Embedded Systems (WISES '07)*, pp. 239–249, June 2007.

[13] B. Andersson, N. Pereira, W. Elmenreich, E. Tovar, F. Pacheco, and N. Cruz, "A scalable and efficient approach for obtaining measurements in CAN-based control systems," *IEEE Transactions on Industrial Informatics*, vol. 4, no. 2, pp. 80–91, 2008.

[14] N. Pereira and B. Andersson, "Widom vs ieee 802.15.4 for computing min in a single broadcast domain," Tech. Rep., IPP Hurray, 2008.

[15] A. Ehyaei, E. Tovar, N. Pereira, and B. Andersson, "Scalable data acquisition for densely instrumented cyber-physical systems," in *Proceedings of the IEEE/ACM International Conference on Cyber-Physical Systems (ICCPS '11)*, pp. 174–183, IEEE, Chicago, Ill, USA, April 2011.

[16] K. K. Chintalapudi and R. Govindan, "Localized edge detection in sensor fields," *Ad Hoc Networks*, vol. 1, no. 2-3, pp. 273–291, 2003.

[17] Y. Wang, J. Gao, and J. S. B. Mitchell, "Boundary recognition in sensor networks by topological methods," in *Proceedings of the 12th Annual International Conference on Mobile Computing and Networking (MobiCom '06)*, pp. 122–133, ACM, September 2006.

[18] M. Singh, A. Bakshi, and V. K. Prasanna, "Constructing topographic maps in networked sensor systems," in *Proceedings of the IEEE Workshop on Algorithms for Wireless and Mobile Networks (ASWAN '04)*, August 2004.

[19] B. Avci, G. Trajcevski, and P. Scheuermann, "Managing evolving shapes in sensor networks," in *Proceedings of the 26th International Conference on Scientific and Statistical Database Management, (SSDBM '14)*, ACM, July 2014.

[20] C. Buragohain, S. Gandhi, J. Hershberger, and S. Suri, "Contour approximation in sensor networks," in *Distributed Computing in Sensor Systems*, vol. 4026 of *Lecture Notes in Computer Science*, pp. 356–371, Springer, Berlin, Germany, 2006.

[21] S. Gandhi, S. Suri, and E. Welzl, "Catching elephants with mice: sparse sampling for monitoring sensor networks," *ACM Transactions on Sensor Networks*, vol. 6, no. 1, article 1, 2009.

[22] J. M. Hellerstein, W. Hong, S. Madden, and K. Stanek, "Beyond average: toward sophisticated sensing with queries," in *Information Processing in Sensor Networks*, vol. 2634 of *Lecture Notes in Computer Science*, pp. 63–79, Springer, Berlin, Germany, 2003.

[23] S. Duttagupta, K. Ramamritham, and P. Kulkarni, "Tracking dynamic boundaries using sensor network," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, no. 10, pp. 1766–1774, 2011.

[24] M. I. Ham and M. A. Rodriguez, "A boundary approximation algorithm for distributed sensor networks," *International Journal of Sensor Networks*, vol. 8, no. 1, pp. 41–46, 2010.

[25] W. Xue, Q. Luo, L. Chen, and Y. Liu, "Contour map matching for event detection in sensor networks," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 145–156, June 2006.

[26] M. Li and Y. Liu, "Iso-map: energy-efficient contour mapping in wireless sensor networks," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 5, pp. 699–710, 2010.

[27] S. Funke, "Topological hole detection in wireless sensor networks and its applications," in *Proceedings of the Joint Workshop on Foundations of Mobile Computing (DIALM-POMC '05)*, pp. 44–53, ACM, 2005.

[28] A. Kröller, S. P. Fekete, D. Pfisterer, and S. Fischer, "Deterministic boundary recognition and topology extraction for large sensor networks," in *Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithm (SODA '06)*, pp. 1000–1009, 2006.

[29] B. Deb, S. Bhatnagar, and B. Nath, "Multi-resolution state retrieval in sensor networks," in *Proceedings of the 1st IEEE International Workshop on Sensor Network Protocols and Applications (SNPA '03)*, pp. 19–29, IEEE, Anchorage, Alaska, USA, May 2003.

[30] F. J. Kurdahi and A. C. Parker, "Real: a program for register allocation," in *Proceedings of the 24th ACM/IEEE Design Automation Conference*, pp. 210–215, ACM, 1987.

[31] T. R. Jensen and B. Toft, *Graph Coloring Problems*, vol. 39, John Wiley & Sons, 2011.

[32] N. Pereira, B. Andersson, E. Tovar, and A. Rowe, "Static-priority scheduling over wireless networks with multiple broadcast domains," in *Proceedings of the 28th IEEE International Real-Time Systems Symposium (RTSS '07)*, pp. 447–456, IEEE, December 2007.

[33] M. Vahabi, S. Tennina, E. Tovar, and B. Andersson, "Response time analysis of slotted widom in noisy wireless channels," in *Proceedings of the 20th International Conference on Emerging Technologies and Factory Automation (ETFA '15)*, IEEE, 2015.

[34] CROSSBOW, *Datasheet: MICAz*, Crossbow Technology, San Jose, Calif, USA, 2004.

[35] F. Ferrari, M. Zimmerling, L. Thiele, and O. Saukh, "Efficient network flooding and time synchronization with Glossy," in *Proceedings of the 10th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN '11)*, pp. 73–84, April 2011.

[36] O. Landsiedel, F. Ferrari, and M. Zimmerling, "Chaos: versatile and efficient all-to-all data sharing and in-network processing at scale," in *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems (SenSys '13)*, ACM, November 2013.

[37] F. Ferrari, M. Zimmerling, L. Mottola, and L. Thiele, "Low-power wireless bus," in *Proceedings of the 10th ACM Conference on Embedded Network Sensor Systems*, pp. 1–14, ACM, November 2012.