A decentralized approach for detecting dynamically changing diffuse event sources in noisy WSN environments

Fernández-Márquez, José Luis; Arcos, Josep Lluis; Di Marzo Serugendo, Giovanna

How to cite

FERNÁNDEZ-MÁRQUEZ, José Luis, ARCOS, Josep Lluis, DI MARZO SERUGENDO, Giovanna. A decentralized approach for detecting dynamically changing diffuse event sources in noisy WSN environments. In: Applied artificial intelligence, 2012, vol. 26, n° 4, p. 376–397. doi: 10.1080/08839514.2012.653659

This publication URL: https://archive-ouverte.unige.ch//unige:42389
Publication DOI: 10.1080/08839514.2012.653659

A Decentralized Approach for Detecting Dynamically Changing Diffuse Event Sources in Noisy WSN Environments

Jose Luis Fernandez-Marquez, Josep Lluis Arcos
Artificial Intelligence Research Institute
CSIC, Spanish National Research Institute
Campus UAB, Bellaterra, E-08193 (SPAIN)
{fernandez,arcos}@iiia.csic.es

Giovanna Di Marzo Serugendo
Birkbeck College
University of London
Malet Street WC1E 7HX London, UK
dimarzo@dcs.bbk.ac.uk

***Abstract*—Localizing dynamically changing diffuse event sources in real environments is still an open problem in Wireless Sensor Networks (WSN). The dynamism of the environment, the energy limitations of the sensors, and the noise associated to the sensor's measurements is a challenge that a realistic solution has to deal with. In this paper we propose a decentralized approach to detect diffuse event sources in dynamic and noisy environments, using a Wireless Sensor Network infrastructure. Our approach is gradient-based and follows a distributed and decentralised algorithm based on local interactions and local knowledge of the environment. Performances are assessed in terms of messages sent and number of measures to find the sources. Results show that our approach efficiently adapts in tracking the event sources as they appear, is scalable and robust to noise and failures.**

I. Introduction

The localization of diffuse event sources and plumes is a problem that appears in a wide range of real applications such as toxic gas detection, detection of underwater leaks, or detection of acoustic and heat sources. Diffuse events are huge phenomena that can spread in a 2D or 3D space without a regular shape. A diffuse event consists of one *source* and its *plume*. The source is the focus of the event whereas the plume is the area or space the diffuse event covers. Plume sizes and shapes are constantly changing due to the environment dynamism that acts over them (i.e. the wind, obstacles, etc...).

In some scenarios, the source is fixed and does not vary with time, while the plume varies constantly. A recent example is provided by the eruption of the Eyjafjallajökull volcano in Iceland. The source is well known and somehow fixed, while the changing ash plume is the main point of concern.

In other cases, the source(s) itself varies (in location and number) over time and it is imperative to detect all of them as quickly as possible. We can mention here the following two examples. In 2002, the Prestige tanker was damaged and began losing its cargo during a storm. The Prestige was carrying approximately 81.000 tons of oil. The oil spread over the sea near the Spanish and Portuguese coasts. Due to the wind and sea currents and the way the tanker sank, the oil split into several disjoint spots. The different spots of oil moved over the sea and continued splitting into new spots, rendering the recuperation of the oil and the cleaning process difficult. Ultimately, this accident lead to a huge ecological disaster, the oil spills stretching on more than 1000 km. The detection and tracking of the spots was a difficult task that could have been simplified with the use of sensor networks. Another real example of dynamically changing diffuse event sources is provided by the bush fires in Australia in 2009. Because of the wind, embers were blown ahead of the fire front, new spot fires then started where the embers landed. The new fire sources need to be monitored and tracked, in order to predict the fire movements and mitigate them as soon as possible. In this particular example, the presence of smoke makes difficult the localization of the main fire focuses. Infrared vision sensors, as used in the project Spread[1] have been used to localize hot temperature spots and to predict fire movement, thus demonstrating the usefulness of sensors in tracking fire. In scenarios where sources are dynamically changing, localizing as soon as possible all diffuse event sources is crucial (e.g. to avoid the spreading of toxic gas and possible large disasters). We consider that the sensor network and the localization of diffuse event sources play a key role in this kind of scenarios.

So far, approaches exploiting WSN, have essentially concentrated on detecting plumes using centralized algorithms [1], on detecting a single source (global optimum) in static and noise-free environments [2], [3], or detecting multiple sources with sensors well distributed in the environment and following a centralized strategy [4]. More generally, regarding the detection of static diffuse event sources in non-noisy environments, Ruair et al. [1] demonstrated that existing algorithms for target tracking do not scale well when they are applied to the localization of diffuse event sources. These algorithms require that each sensor reports the data to the sink when it reads a sensor value higher than a threshold. Since diffuse events can cover large areas, a large number of

[1] http://www.algosystems.gr/spread/index.html

sensors would try to report the data to the sink, producing a network overload.

To the best of our knowledge, the problem of detecting dynamically changing diffuse event sources in noisy WSN environments has not been addressed before.

Our work focuses on the detection of diffuse event sources in *dynamic* and *noisy* environments. The main task is to detect not only the main event source (i.e. location of the global optimum given for instance by the highest temperature or the highest density of oil) but also any residual event sources that may become new principal events (i.e. local optima becoming global optimum). Thus, the goal is to detect *all* event sources *dynamically appearing* over time in the system. Additionally, any realistic solution to the problem has to deal with the imprecision related to sensors measurements and the noise introduced by the environmental changes (e.g. weather conditions or ocean currents).

To track diffuse event sources, we consider sensor networks covering large areas created by a vast number of connected devices spread randomly in the environment. Despite the improvement in the technology, which has made possible the development of ultra-small fully autonomous and communicating sensors, characterized by small size, low power consumption, low cost and low computation power, one of the most important requirements in a WSN remains the design of energy-efficient algorithms able to extend the network lifetime [5].

Therefore, a quick detection of dynamically changing diffuse event sources in large sensing areas calls for decentralized self-organising approaches able to adapt to the dynamicity of the environment, robust to noise and that scale without being greedy on energy consumption.

This paper proposes a decentralized multi-agent approach, following a gradient-based strategy and exploiting local interactions among sensors. It detects all the diffuse event *sources* as soon as they appear and has the additional advantage of limiting the energy consumption of the sensors.

This paper is organized as follows. First, we discuss related works. Then, we briefly explain the lower power listening mode assumed in this paper for the sensors. Next, we describe our model and approach. Then, we report on simulations and discuss the performance of our approach in terms of messages sent, number of measures, resilience to noise and failures. We also performed a study on the impact of the parameters used. Finally, we present conclusions and future work.

II. Related Work

Localization of diffuse event sources differs from target tracking [6] and environment monitoring [7]. These related problems are concerned either with the prediction of object movements or with the creation of a model to monitor the changes in a specific area. We assume that diffuse events are phenomena whose behavior is unpredictable because of two main reasons: the environment dynamism and the high latency that WSN require to track objects. Moreover, the appearance of diffuse events cannot be predicted by any model.

The problem of localizing diffuse event plumes in a WSN has been addressed by Ruair et al. [1] who propose a MAS approach to map the contours of large diffuse events. Agents are distributed over a WSN playing different roles: an agent playing the *leader* role and operating on one sensor, and multiple agents playing the *member* role and operating on sensors adjacent to the location of the leader agent. Agents change role by following a gradient-based strategy, the aim is to cover an event's contour (plume). The proposed mechanism can be adapted to deal with multiple sources, but it has not been demonstrated to be enough for dynamic and noisy environments.

Blatt et al. [2] and Ermis et al. [3] proposed different algorithms to detect and localize sources that emit acoustic waves. They consider static and noisy-free environments, and their goal is to assess the global optimum value avoiding the local optima of the acoustic signals.

When the cost of the sensors is expensive, sensors are allocated strategically and a centralized solution produces really good result, [4]. When the data sampling periods are much larger the communication time, a centralized approach for detection and localization is feasible. Indeed, the time required to coordinate the nodes is smaller than the sampling time. This solution however does not scale to a large number of non expensive sensors spread randomly over the space, since we cannot assume that every node executes a sample in every period.

Finally, as the main studies in dynamic multi-modal optimization have demonstrated [8], [9], in highly dynamic environments detecting the global optima only is not sufficient (the diversity of the exploration is a required feature). A current trend in dynamic multi-modal optimization is to localize most of the best local optima to guarantee a fast adaptation to environmental changes [10].

III. Sleep/Wake Modes

The required life time of sensors for environment monitoring can reach several years. In order to achieve this requirement, a sensor must be in sleep mode most of the time. A sensor consumes energy while it takes measurements, is computing and while it is communicating (sending or listening for data). Communication is the most energy consuming activity of the sensor [11]. The energy used in the communication device, even in idle listening is three orders of magnitude higher than when the node in the sleep mode.

Different proposals for dealing with energy efficiency at the MAC layer in sensor networks communication have been presented. Two main approaches can be identified [12]. On the one hand, the synchronized listening (SL) approach

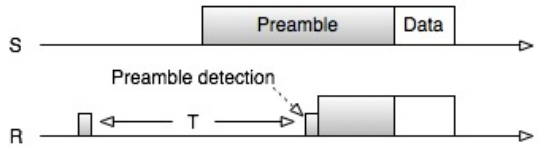

Figure 1. Low Power Listening (taken from [12])

causes sensors to turn on and off their radio at regular intervals; sensors must be synchronised to communicate with each other. This algorithm presents the problem of the synchronization and coupling between the sensors. The synchronisation has an extra cost and sensors cannot send data when they need to, they have to wait for the wake up events to do so. On the other hand, the low power listening (LPL) approach allows sensors to be decoupled, that means they can send information when they want. The only requirement is, for the sender, to send a large preamble data in order to synchronize with other sensors in communication range. Potential receiving sensors wake up asynchronously to detect and synchronize with any detected preamble. The main drawback of this algorithm is that the preamble data does not only wake up the sensor that must receive the information, but any sensor in communication range.

We consider that in emergency scenario like forest fires, or a gas leaks, a sensor should not wait until the next wake up period, but the sensor must be able to send the information in a short period of time. Therefore in this paper we assume the LPL approach.

A. Low Power Listening

The Low Power Listening (LPL) approach reduces the idle listening time, by incorporating a duty cycle in the physical layer. This approach is motivated by the idea that most of the time sensors do not need to communicate, because interesting events rarely occur. Basically, LPL increments the size of the data sent by the transmitter and reduces the cost from the receiver. Figure 1 shows how the receiver wakes up asynchronously and checks whether there is a preamble or not. If the preamble is detected, the receiver continues listening until it receives the data, otherwise it turns off the radio until the next cycle T.

Halkes et al. [13] have demonstrated that LPL reduces the idle listening overhead by a factor of ten, using a sample time of $30\mu s$ for detecting any preamble and a wake up interval $T = 300\mu s$. LPL can be applied to those devices where switching the radio on/off takes little time. Recently, further improvements have been realized in both approaches (SL, LPL) [12].

Our work does not focus on the different MAC protocols proposed in order to save energy in WSN. This brief introduction is presented only to justify the assumption that the network can work in an asynchronous mode and that every sensor is constantly in a sleep mode (has its communication device off) unless it is awaken by another sensor sending some data. As soon as a sensor has performed its duty (answering a request or transmitting information) it turns off its communication device again.

IV. The Model

Let A be the geographical area of interest. Let $Sys^t = \{S^t, Ag^t, D^t, h\}$ be the system at time t. $S^t = \{s_1, s_2, \ldots, s_n\}$ is the set of all sensors present in the system at time t (including those potentially off). $Ag^t = \{ag_1, ag_2, \ldots, ag_k\}$ is the set of all mobile agents present in the system between time 0 and time t (including those that may have stopped their execution). $D^t = \{d_1, d_2, \ldots, d_m\}$ the set of all diffuse events in the system at time t (including those that may have disappeared or appeared after time 0). $h \in A$ is the location of the sink where the information must be sent. Each sensor at time t is a triple $s_i = (state, position, sampleValue)$, where the $state$ is either *off*, *sleep* or *awake* (corresponding to the LPL modes), the $position \in A$ is the location of the sensor, and $sampleValue \in \mathcal{R}$ is the sample that the sensor has measured at time t. The sample value can be 0 if the sensor does not take a measure at time t. A diffuse event at time t is a pair $d_i = (pos, radius)$ with $pos \in A$ the position of the source and $radius \in \mathcal{R}$ the radius of the plume centered at pos. For our experiments, we assume that two diffuse events do not have the same pos, and that in the absence of noise, the plume has the shape of a circle.

For the sensors, we consider the following:

- Sensors are randomly and uniformly spread over the environment A.
- Sensors don't know their position, no global position system (GPS) is assumed.
- Sensors are identical and they run the same software.
- We do not assume any multi-hop protocol. Communication happens only locally with sensors within communication range.
- Sensors know neighbour sensors in communication range.
- Transmission collisions are handled by lower MAC layer protocols and are not considered in this paper.
- Sensors follow the Low Power Listening (LPL) mode described in Section III.
- Sensors are reactive to agents request. No proactive behavior is assumed from the sensor side.

For the mobile agents, we consider the following:

- Mobile agents can communicate with other agents within communication range.
- Agents use the sensor communication devices to communicate with other agents.
- Agents are proactive (send requests to other agents, move) and sensors are reactive (respond to agents requests, take measurements).

For the diffuse events we assume that:

- Diffuse events appear and disappear over time.
- Each diffuse event has one source and one plume.
- When the plume is not subject to noise, the plume represents a circle centered around the source, with the maximum intensity in the center and the minimum intensity at the edge of the circle. In the presence of noise the shape varies and the gradient from source to edge is no longer perfect .
- Over time the size of the plume, the position of the source and the intensity of the source may vary.

The performance of our approach is evaluated in terms of the number of messages sent and the number of samples values read by the sensors between time 0 and time t.

Definition 1 (Number of messages sent at time t): Let $msg(ag, t)$ be the number of messages sent by $ag \in Ag^t$ between 0 and t, the number of messages sent at time t is given by:

$$MSG(t) = \sum_{ag \in Ag^t} msg(ag, t) \quad (1)$$

Definition 2 (Number of reads at time t): Let $read(ag, t)$ be the number of samples taken by $ag \in Ag^t$ between 0 and t, the number of messages sent at time t is given by:

$$READ(t) = \sum_{ag \in Ag^t} read(ag, t) \quad (2)$$

V. Our Approach

The aim of our approach is to localize the diffuse event sources as soon as possible, minimizing sensor measurements and communication. Basically, the idea is to find those sensors closest to the diffuse event. One of the contributions of this algorithm is that the search of the diffuse event sources is executed in a decentralized way, by collaboration. This produces a better scalability when the diffuse events are spread over a huge number of sensors. Once we find the sources the number of sensors that report the information about the diffuse event sources localization is very low compared with the traditional tracking algorithms used in sensor networks, where every sensor that samples a value higher than a fixed threshold sends the information to the sink.

We assume a WSN where the sensors are spread randomly over a 2-dimensional space. All sensors are identical and reactive. Over the WSN there is a middleware that permits a set of agents to move from one sensor to another and have access to the sensor data and sensor communication devices. All agents run the same algorithm and agents have only access to local information. Communication between agents is only allowed when they reside in adjacent sensors, that is, a hop by hop communication protocol is not assumed. Sensors only communicate with other sensors when an agent hosted in some sensor requires information.

We propose a distributed and decentralized approach based on a mobile MAS where agents move freely over the sensor network to localize the sources of diffuse events that are randomly appearing and disappearing along the time. Moreover, agents are responsible for monitoring the localized events once the source is reached. They are responsible for requiring measures from the sensors.

Our approach pursues a number of active agents lower than the number of sensors, as we show later on. As a consequence, a low number of environment measurements are performed. Because we cannot control the number of active diffuse events, we include a mechanism to control the number of mobile agents that live in the WSN. This mechanism controls the number of agents in the WSN in a decentralized way and without additional communication cost.

In order to deal with energy constraints we use a GPS-free algorithm where our main goals are to reduce the number of data sensor measures and the used bandwidth. The GPS-free approach can reduce WSN cost [14] and can work either in indoor or underwater environments with high energy constraints.

Our approach performs two different explorations: (1) a *global exploration* thanks to the random generation of new agents on the WSN; and (2) a *local exploration* that drives agents to the sources. Global exploration is required to continuously monitor new diffuse events as they appear. We consider that the system *converges* when, for each active event, there is an agent located at the sensor nearest its source (i.e. *all* event sources are monitored).

To ease the discussion, we use in this paper the notion of mobile agents. However, to further reduce computation and communication costs, the actual movement of the agent can be replaced by moving a token (instead of a whole agent). In that case, each sensor hosts a stationary agent and the movement would consist in sending a token among the sensors until the token reaches the diffuse event source.

A. Sensors

Sensors are responsible for creating agents. Sensors provide an infrastructure to host agents allowing the agents to access their data and communication devices. Sensors are most of the time in the sleep state, that is, with the wireless communication turned off and using low energy. Every T_w ticks, a sensor creates an agent with probability P_a. It is important to note that the creation of an agent does not change the communication state, if the sensor is in the sleep state, it will stay so until it switches to the awake state because of a communication request (i.e. data received from a nearby sensor or sent on request of the agent). The P_a parameter controls the number of agents that are created across the whole environment. A high P_a value

Algorithm 1 The Sensor Algorithm

```
if (createAgentEvent()) then
    if (Random() < P_a ) then
        CreateAgent()
    end
end
if (sensorReadRequestEvent()) then
    sendSensorData()
end
```

implies a high global exploration and also a higher cost, i.e an increment on the sensor measures and on the number of messages sent. Moreover, sensors send data measures when they receive *data requests*. These are sent by an agent to a neighbor sensor when it performs local exploration. The sensor algorithm is sketched in Algorithm 1.

For simplicity purposes, we do not show the change of communication state (sleep to awake to sleep again). The sensor is always in the sleep mode, except when it sends or receives a data.

B. Mobile Agents

Mobile Agents are responsible for actively tracking diffuse event sources and monitoring them once they have reached the source. Mobile Agents use a WSN as an infrastructure that enables them to move over the space, to obtain sensor data, and to communicate with other sensors or agents. The agent procedure has to deal with uncertain data (mistaken measurements) and with a weak infrastructure that can fail at any time (sensors can break down, sensor data may contain noise, and communications can fail).

The goal is to design a robust agent algorithm that allows agents to monitor diffuse events with a high performance. The agents decide when a sensor must read a sensor data or when a sensor must communicate its sensor data to a neighbour sensor. Sensors are managed by the agents, i.e they are not proactive.

In order to deal with the requirements (low number of sensor reads and low number of communication messages), the number of active agents must be considerably lower than the number of sensors. We consider the following policies: (1) when an agent is created, it first checks whether another agent exists in another sensor within its communication range, the agent with a higher creation timestamp finishes its execution; and (2) when two different agents reach the same sensor, only one of them continues its execution (i.e. two agents cannot coexist at the same sensor).

The intuition is that when agents are created, they try to reach the closer diffuse event source by following the shortest path according to a gradient-based strategy. Specifically, each agent uses the sensor data of the neighboring sensors in order to guide its movements and finally find the source.

Algorithm 2 The Agent Algorithm

```
if (agentsInNeighborhood()) then
    exit()
end
while (true) do
    sensorData = readSensor()
    if (sensorData <= 0) then
        exit()
    end
    neighbors = selectAdjNodes (n_s)
    requestReads (neighbors)
    bestSensor = selectBestSensor (neighbors)
    if (bestSensor.data > sensorData) then
        moveToSensor(bestSensor)
        if (existAgentInSensor () ) then
            exit()
        end
    end
end
```

Following Algorithm 2, when an agent is created, it first checks if there is another agent placed in one of the adjacent sensors. If that is the case, the most recent agent finishes its execution. Otherwise, it reads the sensor data and checks if a given event plume is detected. If nothing is detected (the measured value is too low), it finishes its execution. When an event is detected, the execution continues by choosing n_s adjacent sensors and sending a sensor data request to the selected n_s sensors. When all the answers are received, the agent selects the best sensor. That is, the sensor providing the highest sensor data read (e.g. highest gas concentration, highest temperature). If the data of the best neighbor sensor is higher than the data the agent has measured on its host sensor, the agent migrates to the selected sensor. After migrating, if another agent is already hosted at that sensor, the migrating agent finishes its execution. Otherwise, the main loop starts again (reading the sensor data of the host sensor).

When an agent reaches the source of a diffuse event (i.e. when it does not move between consecutive reads), it continuously monitors the event until an environment change occurs. An event source may disappear or change its location. When it disappears, the data obtained from the sensor becomes zero and the agent finishes its execution. When an event source changes its position (i.e. the event moves slightly), the requests to the neighbor sensors will guide the agent to the new source location.

VI. Experiments

The goal of this section is to demonstrate the performance of our approach in simulated scenarios and to perform a

Table I
SIMULATION SETTINGS

Params	values	Params	values
Space A	$10^3 \times 10^3\ m^2$	S_N	1000
T_S	2×10^5 ticks	T_w	20 ticks
t_c	200 ticks	P_a	0.5%
Sensor Rng	80 m	n_s	3

Table II
STANDARD SETTINGS FOR MPB

Params	values	Params	values
movrand	random	num. of peak	1-3
num. of dimensions	2	minheight	30
maxheight	100	stdheight	50
minwidth	0.1	maxwidth	5.0
stdwidth	0.0	mincoordinate	0
maxcoordinate	100	peak_function	cone

study of the impact of the parameters of our proposal. Specifically, we analyze the performance of our approach when the number, i.e. density, of the sensors changes; when local and global exploration vary; or when the system is subject to different noise levels. Moreover, we measure the exploration cost and we study the robustness of our approach in front of network failures.

The simulation has been implemented using REPAST [15] for modeling sensors and agents, and the Moving Peaks Benchmark (MPB) [16] for modeling the environment changes (diffuse events). MPB is a benchmark created to compare dynamic function optimization algorithms, providing a fitness function changing along the time. The function is composed by different peaks (cones) that change in width, height and position. These peaks are used as diffuse events in our simulation. Figure 2 shows an example of environment change. In the upper part of the figure, the environment presents two different diffuse events with small plumes. The lower part of the figure, shows the situation after an environment change, we can see three diffuse events with different width, height and position. Each (different) situation is called *scenario* in this paper.

In order to aggregate noise to the sensor reads, we modified MPB such as the fitness function incorporates a noise factor γ in the following way:

$$SensorValue(\vec{p}) = MPBValue(\vec{p}) + (2 * \theta - 1) * \gamma \quad (3)$$

where θ generates a uniform random number between [0..1] and γ, the noise factor, varies between 0 and 10 depending on the experiment.

A simulation is a run of $T_S = 2 \times 10^5$ ticks, where an environment change occurs at each $t_c = 200$ ticks. That is, a simulation holds 1000 environment changes (similar to those shown on Figure 2). The results reported are the averages of these 1000 changes. Simulations take place in

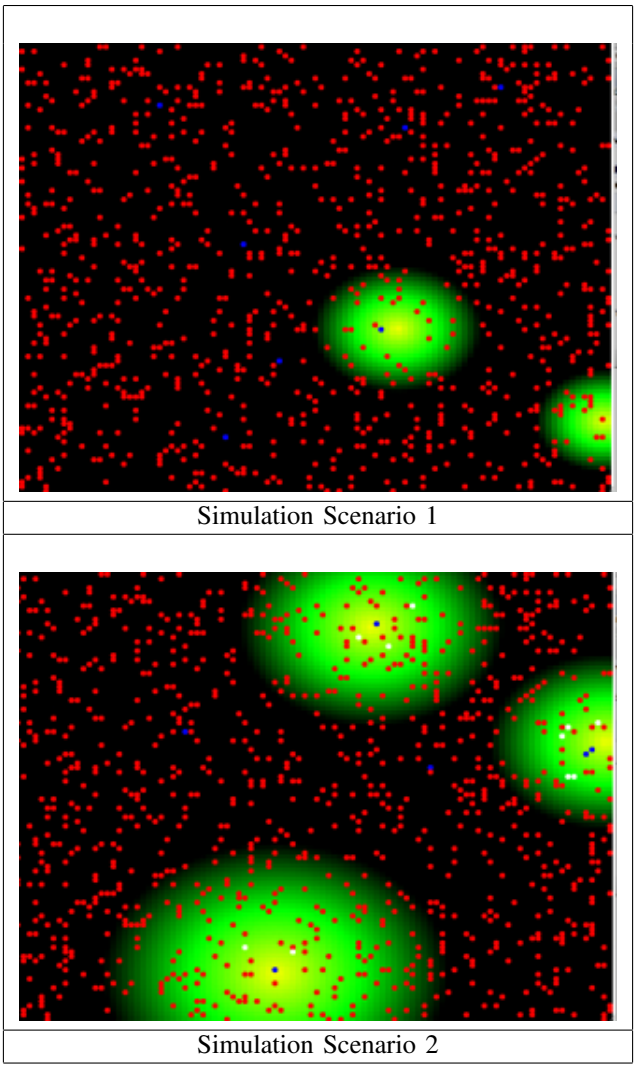

Figure 2. Environment change example

a rectangular space A of 1000×1000 square meters where $S_N = 1000$ sensors are distributed randomly. The parameter settings used in the simulation are summarized in Table I, where T_s is the simulation time, t_c the frequency of the environment changes, S_N the number of sensors, T_w the frequency of the agent creation event, P_a the probability of creating an agent and n_s the number of nearby sensors receiving a data request from an agent. Table II summarizes the configuration of MPB. Mainly, the number of diffuse events vary from 1 to 3 with a radius of the plume ranging from 30 to 5000 meters.

Figure 3 shows an example of a simulated scenario, where the sensors are spread over the space and 3 diffuse events are active. Gray blurred regions represent the diffuse events perceived with noise, i.e. event plumes do not form a continuous space. Small filled points represent the sensors. Gray filled points represent sensors not hosting agents. White filled points represent sensors with a hosted agent.

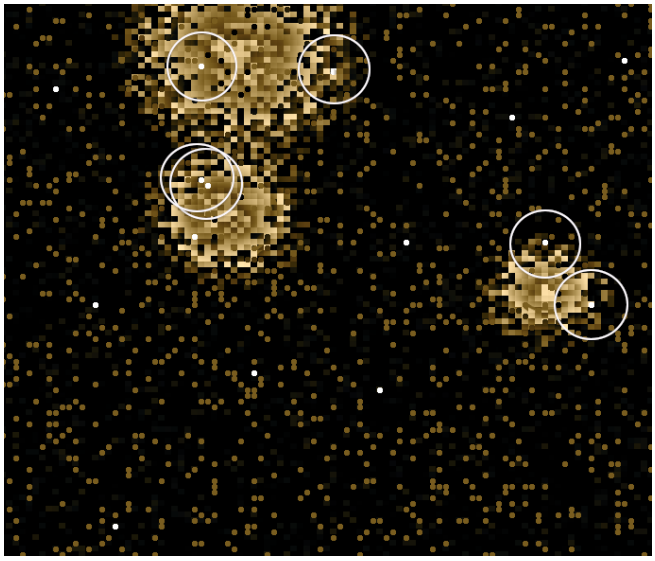

Figure 3. Scenario with noise

Circles represent communication range of sensors hosting an agent that has detected an event; n_s sensors within the circle will receive the data requests.

In the simulations we use the number of data sensor reads and the number of messages sent as an estimation of the cost to reach the convergence, i.e. when *all* diffuse event sources of a given scenario have been detected. These values are measured for each environment change: from the moment a new scenario is in place until convergence is reached (all events sources detected). We consider a failure of the system if the system cannot reach the convergence before a new change in the environment (i.e. 200 ticks), that means at least one of the sources has not been detected. Once the system has reached the convergence, the agents continue exploring and monitoring the events. At that moment the agent are ready to send the sensor data to the sink. The cost of sending the information to the sink depends on the routing algorithm that is used and it is not addressed in this work. Thus, the monitoring reads and routing messages are not counted here, because they depend on the routing algorithm and on external parameters such as the desired monitoring frequency. Our counting of reads and messages stops when we reach the event source. We performed an additional experiment for measuring the number of sensor data reads and messages when no diffuse events are present, i.e. the cost of the global exploration.

A. Varying the number of sensors in WSN

This first experiment had two goals: (1) to demonstrate that the complexity of our approach grows linearly with the WSN size (i.e. our approach is scalable) and (2) to demonstrate the adaptability or our approach to different WSN densities. The different densities used in this simulation have been established following [17]. In this experiment the number of sensors varies from 500 to 8000 and noise is not applied to the sensor data reads.

Table III
VARYING SENSOR NUMBER WITHOUT NOISE

Sensor Number	Reads	Msgs	Failures	Adj. avg.
500	311.30	518.82	35.5%	9.2
1000	397.38	651.64	15.2%	18.76
2000	681.67	1115.48	5.8%	37.09
4000	1222.37	1998.11	4.6%	74.94
8000	2339.69	3816.25	3.2%	149.7175

The first observation is that, when the density of sensors increases, the number of failures decreases, i.e. agents are able to find better paths to navigate toward event sources (see Table III). Notice that the number of failures reaches a 35% only when the number of sensors is low (500). This percentage of failure could be reduced by incrementing the P_a probability or by reducing the T_w interval, as we will present in the next experiment. The number of consumed resources varies according to the size and location of the diffuse events. Fast convergences are reached with only 15 sensor reads whereas hard scenarios require more than 1000 reads. Notice that the difficult scenarios are those where the diffuse events have overlapping areas or at least one of the diffuse event is present in a low number of sensor (small diffuse event). Notice that we consider a convergence only when all the event sources of a scenario are located.

The results achieved in this first simulation show that our approach is able to find all the diffuse event sources with a probability of 85% when the number of reads is approx. 40% of the number of sensors, and the number of messages is approx. 60% of the the number of sensors (line 2 of Table III). The number of messages and reads grows linearly with the number of sensors, while the number of failures decreases (good scalability).

B. Quality of Convergence

In the experiments we are showing the average of the number of reads and the average of the number of messages. However, the number or reads and messages that the approach needs to reach the convergence, that is, to find *all* the diffuse event resources in one scenario do not follow a uniform distribution. Figure 4 shows how, for most of the scenarios, our approach is able to reach the convergence in less than 200 reads. The black line on the top of the bars shows the standard deviation over 5 runs where each runs has 3000 environment changes. More precisely, 1300 convergences of a total of 3000 are assessed with less than 200 reads, while 450 scenarios require more than 800 reads or do not converge at all.

Figure 5 shows that similar results are obtained for the number of messages. 30% of the convergences are reached

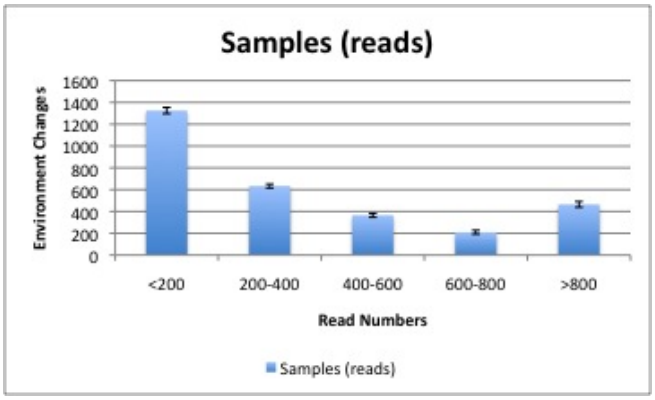

Figure 4. Reads histogram

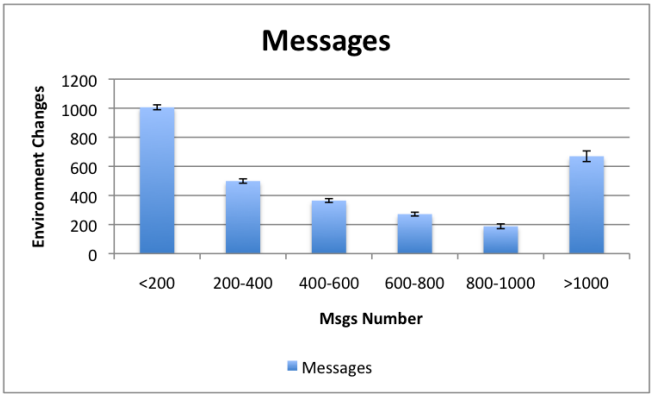

Figure 5. Msgs histogram

with less than 200 messages.

C. Varying the Noise Factor

The goal of these experiments was to evaluate the performance of our proposal in the presence of different noise levels. Specifically, the noise factor γ was varied from 0 to 10.

In Table IV, we may observe how, when the noise factor increases, the performance of the system decreases (in terms of reads). However, when the noise level is equal to or lower than 4, the percentage of failures decreases. That is because the noise introduces a stochastic behaviour that increases the exploration in the search. This increment in the exploration increases the number of reads and messages, but produces a better convergence (less percentage of failures). We can also note that our algorithms is robust to noise. Indeed, even when the noise factor is $\pm 10\%$ the algorithm is able to reach the convergence, that is to detect the optimum sensor for all the diffuse events in 75% of the scenarios.

D. Varying the Local Exploration

In this simulation we studied the performance of the algorithm when we vary the local exploration in a noise-free environment. Local exploration is controlled by the number of sensors that an agent uses to decide its next location (n_s).

Table IV
VARYING THE NOISE FACTOR γ

γ	Reads	Msgs	Failures
0%	397.38	651.64	15.2%
$\pm 2\%$	547.70	907.64	13.4%
$\pm 4\%$	698.31	1160.37	15.1%
$\pm 6\%$	776.21	1291.31	18.6%
$\pm 10\%$	878.73	1461.43	25.1%

Table V
VARYING THE n_s PARAMETER

n_s	Reads	Msgs	Failures
1	446.41	625.68	15.4%
2	425.96	663.72	16.7%
3	397.38	651.64	15.2%
4	435.79	740.80	12.6%
5	517.21	903.14	14.1%
6	525.74	934.45	14.4%
10	752.18	1391.42	13.1%

In Table V, we observe that even when we increase to 10 the number of requested sensors, the number of failures is not significantly decreasing. The reason behind this result is that the so increased local exploration is not enough to detect all the diffuse event sources. Specifically, the global exploration (and not the local explorations) is the main factor of failures. As expected, the number of messages and sensor reads increases when the local exploration is higher. From the results of this experiment, we set the parameter $n_s = 3$ (see Table V).

E. Varying Global Exploration

In the previous experiment we observed that, even increasing the local exploration, the number of failures is not reduced. Thus, the goal of this experiment is to reduce the system failures by increasing the global exploration and to measure the cost associated to this strategy. The global exploration is controlled by the frequency (T_w) of the sensors to create agents and the probability (P_a) to actually do so. Both parameters can increase or decrease the number of agents that are exploring the space at the same time. We performed a study assessing the contribution of these parameters to the global exploration ratio, the relation between the global exploration ratio and system failures, and the cost of the exploration when reducing system failures.

Table VI shows how, when the exploration rate increases due to an increased probability P_a of creating an agent, the number of failures decreases. However, the price is an increment of the number of reads and messages. Similar results are found when the frequency T_w is increased (see Table VII). In both experiments we are increasing the number of agents that explore the WSN. As a conclusion of the results, P_a and T_w can be used to customize our

Table VI
VARYING AGENT CREATION PROBABILITY, P_a

P_a	T_w	Reads	Msgs	Failures
0.2	20	322.30	538.29	37.7%
0.5	20	397.38	651.64	15.2%
1	20	484.01	783.83	4.5%
5	20	654.21	1024.92	5.0%

Table VII
VARYING FREQUENCY, T_w

P_a	T_w	Reads	Msgs	Failures
0.5	5	576.26	920.21	1.8%
0.5	10	488.36	790.16	4.4%
0.5	20	397.38	651.64	15.2%
0.5	50	307.75	513.11	38.5%

Table VIII
THE EXPLORATION COST

Noise	Reads	Msgs
0%	49.28	0±0
±2%	100.22	108.74
±5%	99.46	107.87
±10%	101.78	111.05

Table IX
FAILURE TOLERANCE

Failure Prob.	Reads	Msgs	Failures
0%	449.91	740.33	13.7%
5%	455.76	751.13	15.1%
10%	409.97	675.38	16.9%
20%	422.26	697.46	19.6%
40%	463.55	772.60	30.7%

approach depending of the search priority. This trade-off between the quality of the results and the cost can be used in order to control the priority of the search process. Emergency situation will tend to increase the exploration cost. We consider that even when we reduce the percentage of failure to 5%, the number of reads and messages present good results. Indeed, the algorithm is able to find the sensor closest to the event with 654 reads in an environment with 1000 sensors. We consider this a good number of reads, because first not all sensors have performed a read (there is clearly less than 1000 reads), and second there will be *only one* sensor that will send the result to the sink. In approaches where all sensors perform a read (1000 reads), the system still does not know which sensor is the closest to the event. In such a case, the sensors must still decide which of them must send the information to the sink, thus increasing the communication cost.

Experimental results have demonstrated that, even in the presence of a high noise level, the number of failures is reduced by incrementing the global exploration. For instance, increasing P_a to 2% and the noise level to 10% the number of reads is 1190 and the number of messages is 1947 whereas the number of failures number is 162 (16.2%)(i.e. same number of failures achieved without noise). Thus, the global exploration level can reduce the number of failures produced by the lack of sensors in the WSN or by the presence of noise.

F. The Exploration Cost

The goal of these experiments was to measure the exploration cost when no diffuse events are present in the system (most frequent case). Specifically, we tested our approach when different noise levels are applied. Notice that noise is acting as false plumes that temporarily drives agents through the WSN.

Table VIII shows how when the noise level increases from 0% to ±2%, the exploration cost increases by 50%. Thus, we may conclude that noise increments the exploration cost. However this increment remains constant, even when we increment the noise to ±5%, or even to ±10%. Thereby, our approach does not dependent on the noise level.

G. Tolerance to WSN failures

In these experiments the goal was to analyze the robustness of our approach when sensors fail. To that purpose, a probability of failure was added to each sensor. Sensor failures are simulated as follows: just before T_w a percentage of sensors are declared broken down (state is off). Then, those sensors cannot be used until the next T_w interval, where the sensors may continue to be broken or have become fixed.

In Table IX we observe that the increment in the sensor failures involves a decrease of system convergences. However, when exploration is increased (for instance increasing the probability of agent creation from 0.5 to 2.0) the system is able to decrease the failures to 47 (with an average of reads of 705 and messages of 1145). Thus, we may conclude that our approach reaches the convergence even with a high probability of sensor failures.

VII. CONCLUSIONS

In this paper we have proposed a new approach, based on a Mobile Multi-Agent technology, to detect diffuse event sources in dynamic and noisy environments using a wireless sensor network infrastructure. To our knowledge, this problem has not been addressed before. Our approach proposes a distributed and decentralized algorithm based on local interactions and local knowledge of the environment. Different strategies have been designed to guarantee a low number of agents maintaining the performance of the system.

We studied the performance of our proposal on different scenarios: changing the density of the sensors; varying local and global exploration ratios; applying noise to the data that sensors gather; and subjecting sensors to failures. Experimental results have shown that the presence of noise, sensor failures, and the lack of sensors diminishes the performance

of our approach. However, it has been detailed how this degradation can be alleviated by increasing the exploration level. The increase of the exploration level involves a reasonable rise on the cost to reach the convergence. Importantly, in our approach the cost of global exploration is not dependent of the noise level.

Because our approach is not introducing any assumption on the sensor positions, we plan to explore its capabilities in scenarios like underwater applications or 3-Dimension spaces.

ACKNOWLEDGMENTS

This work was funded by EVE (TIN2009-14702-C02-01), Agreement Technologies (CONSOLIDER CSD2007-0022, INGENIO 2010), and ANERIS (CSIC-PIF08-15-2) projects. The first author holds a FPI scholarship from the Spanish Government.

REFERENCES

[1] R. M. Ruair and M. Keane, "An energy-efficient, multi-agent sensor network for detecting diffuse events," in *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-07)*, 2007, pp. 1390–1395.

[2] D. Blatt and A. O. Hero, "Energy based sensor network source localization via projection onto convex sets (POCS)," *IEEE Transactions on Signal Processing*, pp. 3614–3619, 2006.

[3] E. Ermis and V. Saligrama, "Detection and localization in sensor networks using distributed FDR," in *Proceedings of the Conference on Information Sciences and Systems (CISS-06)*, 2006.

[4] J. Weimer, B. Sinopoli, and B. H. Krogh, "Multiple source detection and localization in advection-diffusion processes using wireless sensor networks," in *30th IEEE Real-Time Systems Symposium (RTSS)*. IEEE, 2009, pp. 333–342.

[5] M. Vinyals, J. A. Rodriguez-Aguilar, and J. Cerquides, "A survey on sensor networks from a multiagent perspective," *The Computer Journal. (In press)*, 2010.

[6] L. Yang and C. Feng, "Adaptive tracking in distributed wireless sensor networks," in *Proceedings of 13th Annual IEEE International Symposium and Workshop on Engineering of Computer Based Systems*, 2006, pp. 103–111.

[7] D. D. Corkill, D. Holzhauer, and W. Koziarz, "Turn off your radios! environmental monitoring using power-constrained sensor agents," in *In AAMAS Workshop on Agent Technology for Sensor Networks*, 2007, pp. 31–38.

[8] T. Blackwell, "Particle swarm optimization in dynamic environments," in *Evolutionary Computation in Dynamic and Uncertain Environments*, ser. Studies in Computational Intelligence, S. Yang, Y.-S. Ong, and Y. Jin, Eds., 2007, vol. 51, pp. 29–49.

[9] R. I. Lung and D. Dumitrescu, "A collaborative model for tracking optima in dynamic environments," *IEEE Congress on Evolutionary Computation (CEC)*, pp. 564–567, 2007.

[10] J. Fernandez-Marquez and J. Arcos, "An evaporation mechanism for dynamic and noisy multimodal optimization," in *In Proceedings of the 10th Genetic and Evolutionary Computation Conference (GECCO)*, 2009, pp. 17–24.

[11] S. Croce, F. Marcelloni, and M. Vecchio, "Reducing power consumption in wireless sensor networks using a novel approach to data aggregation," *The Computer Journal*, no. 2, pp. 227–239, 2008.

[12] J. Na, S. Lim, and C.-K. Kim, "Dual wake-up low power listening for duty cycled wireless sensor networks," in *EURASIP Journal on Wireless Communications and Networking*, 2008.

[13] G. Halkes, T. van Dam, and K. Langendoen, "Comparing energy-saving mac protocols for wireless sensor networks," in *MONET Special Issue on WLAN Optimization at the MAC and Network Levels*, 2005, pp. 783–791.

[14] A. Savvides, C. CHan, and M. Srivastava, "Dynamic fine-grained localization in ad-hoc networks of sensors," in *Proceedings of the Seventh ACM Annual International Conference on Mobile Computing and Networking (MobiCom)*, 2001, pp. 166–179.

[15] D. Samuelson and C. Macal, "Agent-based simulation comes of age," *OR/MS Today*, no. 4, pp. 34–38, 2006.

[16] J. Branke, "The moving peaks benchmark website. www.aifb.uni-karlsruhe.de/ jbr/movpeaks/."

[17] C. Intanagonwiwat, D. Estrin, R. Govindan, and J. Heidemann, "Impact of network density on data aggregation in wireless sensor networks," in *In Proceedings of the International Conference on Distributed Computing Systems*, 2002.