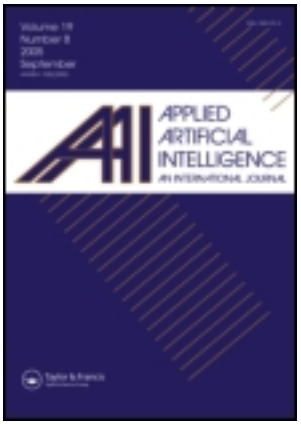


Applied Artificial Intelligence: An International Journal

Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/uaai20

A DECENTRALIZED APPROACH FOR DETECTING DYNAMICALLY CHANGING DIFFUSE EVENT SOURCES IN NOISY WSN ENVIRONMENTS

Jose Luis Fernandez-Marquez [a], Josep Lluis Arcos [a] & Giovanna Di Marzo Serugendo [b]

[a] IIIA-CSIC, Spanish National Research Institute, Campus UAB, Bellaterra, Spain

[b] Centre Universitaire d'Informatique, University of Geneva, Switzerland

Published online: 08 May 2012.

To cite this article: Jose Luis Fernandez-Marquez, Josep Lluis Arcos & Giovanna Di Marzo Serugendo (2012) A DECENTRALIZED APPROACH FOR DETECTING DYNAMICALLY CHANGING DIFFUSE EVENT SOURCES IN NOISY WSN ENVIRONMENTS, Applied Artificial Intelligence: An International Journal, 26:4, 376-397, DOI: 10.1080/08839514.2012.653659

To link to this article: http://dx.doi.org/10.1080/08839514.2012.653659

Applied Artificial Intelligence, 26:376–397, 2012

ISSN: 0883-9514 print/1087-6545 online
DOI: 10.1080/08839514.2012.653659

Taylor & Francis
Taylor & Francis Group

A DECENTRALIZED APPROACH FOR DETECTING DYNAMICALLY CHANGING DIFFUSE EVENT SOURCES IN NOISY WSN ENVIRONMENTS

Jose Luis Fernandez-Marquez[1], Josep Lluis Arcos[1], and Giovanna Di Marzo Serugendo[2]
[1]*IIIA-CSIC, Spanish National Research Institute, Campus UAB, Bellaterra, Spain*
[2]*Centre Universitaire d'Informatique, University of Geneva, Switzerland*

□ *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 sensors' measurements pose a challenge that begs a realistic solution. In this article 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 decentralized algorithm based on local interactions and local knowledge of the environment. Reported experiments show that our approach efficiently adapts in tracking the event sources as they appear, is scalable, and is robust to noise and failures.*

INTRODUCTION

The localization of diffuse event sources and plumes is a problem that appears in a wide range of real-world 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 (the wind, obstacles, etc.).

This work was funded by projects EVE (TIN2009-14702-C02-01), Agreement Technologies (CSD2007-0022), and ANERIS (CSIC-PIF08-15-2). Partially funded by Generalitat de Catalunya under grant 2009-SGR-1434. First author holds a FPI scholarship from the Spanish Government.

Address correspondence to Jose Luis Fernandez-Marquez, Battelle-Batiment A, 7 route de Drize, Carouge, CH-1227, Switzerland. E-mail: JoseLuis.Fernandez@unige.ch

In some scenarios, the source is fixed and does not vary with time, although the plume varies constantly. A recent well-known example is the eruption of the Eyjafjallajökull volcano in Iceland. The source is well known and somehow fixed, whereas the changing ash plume is the main point of concern. In other scenarios, the sources themselves vary (in location and number) over time and it is imperative to detect all of them as quickly as possible. For instance, in 2002 the tanker *Prestige* 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 disjointed 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 led to a huge ecological disaster, the oil spills stretching further 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-world example of dynamically changing diffuse event sources is the bush fires in Australia in 2009. Because of the wind, embers were blown ahead of the fire front, and new spot fires then started where the embers landed. In this particular example, the presence of smoke complicated 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 may play a key role in these kinds of scenarios.

So far, approaches exploiting WSN have essentially concentrated on detecting plumes using centralized algorithms (Ruair and Keane 2007), on detecting a single source (global optimum) in static and noise-free environments (Blatt and Hero 2006; Ermis and Saligrama 2006), or detecting multiple sources with sensors well distributed in the environment and following a centralized strategy (Weimer, Sinopoli, and Krogh 2009). More generally, regarding the detection of static diffuse event sources in non-noisy environments, Ruair and Keane (2007) demonstrated that existing algorithms for target tracking do not scale well when they are applied to the localization of diffuse events. These algorithms require that each sensor reports the data to the sink when it reads a sensor value higher than a threshold. Because diffuse events can cover large areas, a large number of sensors would try to report the data to the sink, producing a network overload. Figure 1 shows a diffuse event and a sensor network used to monitor it. The network overload is produced when all "positive sensors,"

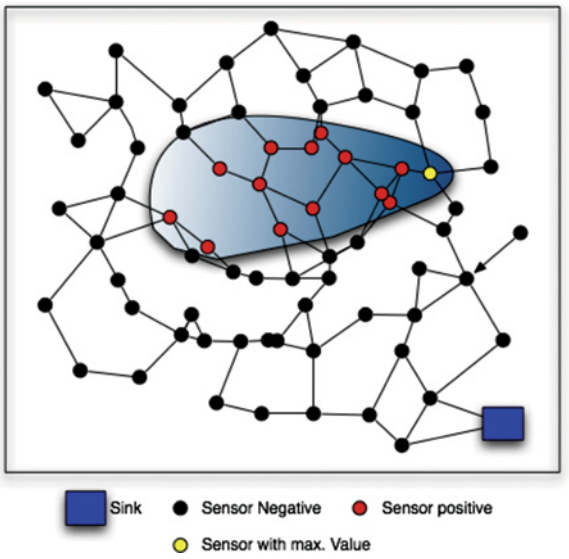


FIGURE 1 Diffuse event example. (Figure is provided in color online.)

sensors that are inside the diffuse event, try to send the information to the sink. To avoid this problem, our proposal is to find the sensor with higher value, and then only one sensor will send the information to the sink.

To the best of our knowledge, with the exception of our previous preliminary work (Fernandez-Marquez, Arcos, and Serugendo 2010), the problem of detecting dynamically changing diffuse event sources in noisy WSN environments has not been addressed. 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 (Vinyals, Rodriguez-Aguilar, and Cerquides 2011).

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

The article is organized as follows. First, we discuss related works. Then, we briefly explain the lower-power listening mode assumed in this article 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 sensors' measurements, resilience to noise and failures. We also performed a study on the impact of the parameters used. Finally, we present conclusions and future work.

RELATED WORK

Localization of diffuse event sources differs from target tracking (Yang and Feng 2006) and environment monitoring (Corkill, Holzhauer, and Koziarz 2007). 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. Because of the dynamicity of the environment, diffuse events are phenomena whose behavior and appearance are unpredictable or difficult to model. The use of WSN further complicates the situations, because it involves a high latency when tracking objects.

The problem of localizing diffuse event plumes in a WSN has been addressed in Ruair and Keane (2007), proposing a multiagent system (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 their roles by following a gradient-based strategy with the aim of covering 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 and Hero (2006) and Ermis and Saligrama (2006) proposed different algorithms to detect and localize sources that emit acoustic waves. They consider static and noise-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 good results (Weimer, Sinopoli, and Krogh 2009). When the data sampling periods are much longer than the communication time, a centralized approach for detection and localization is feasible. Indeed, the time required to coordinate the nodes is shorter than the sampling time. This solution, however, does not scale to a large number of nonexpensive sensors spread randomly over the space, because we cannot assume that all nodes are sampling at each period.

Finally, as the main studies in dynamic multimodal optimization have demonstrated (Blackwell 2007; Lung and Dumitrescu 2007), in highly dynamic environments, detecting only the global optima is not sufficient (the diversity of the exploration is a required feature). A current trend in dynamic multimodal optimization is to localize most of the best local optima in order to guarantee a fast adaptation to environmental changes (Fernandez-Marquez and Arcos 2009).

Our work proposes a similar approach to that of Ruair and Keane (2007). However, we focus on the diffuse event sources location instead of mapping the contours of diffuse events. Moreover, we adopt some ideas from dynamic optimization to improve the adaptability of our approach in dynamic environments.

SLEEP/WAKE MODES

The required lifetime 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 is taking measurements, is computing, and while it is communicating (sending or listening for data). Communication is the most energy-consuming activity of the sensor (Croce, Marcelloni, and Vecchio 2008). The energy used in the communication device, even in idle listening, is three orders of magnitude higher than when the node is in the sleep mode.

Different proposals for dealing with energy efficiency at the Media Access Control (MAC) layer in sensor networks communication have been presented. Two main approaches can be identified (Na, Lim, and Kim 2008). On the one hand, the synchronized listening (SL) approach causes sensors to turn on and off their radio at regular intervals; sensors must be synchronized to communicate with each other. The synchronization 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 send information when they want. The only requirement is for the sender to send a large preamble data in order to synchronise with other sensors in the communication range.

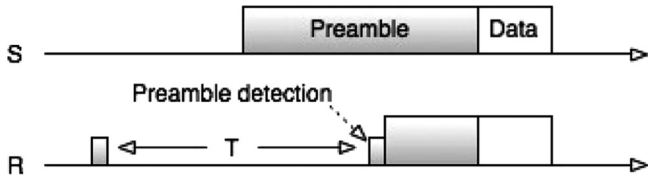


FIGURE 2 Low-power listening (from Na, Lim, and Kim 2008).

Potential receiving sensors wake up asynchronously to detect and synchronize with any detected preamble.

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

The 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 2 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*. LPL can be applied to those devices for which switching the radio on/off takes little time. Recently, further improvements have been realized in both approaches (SL, LPL; Na, Lim, and Kim 2008).

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.

OUR APPROACH

The aim of our approach is to localize the diffuse event sources as soon as possible, minimizing sensors' measurements and communication. Diffuse events appear and disappear over time. Basically, the idea is to find those sensors closest to diffuse events. 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 proposal produces a better scalability when diffuse events are spread over a significant 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 access only to local information. Communication between agents is allowed only when they reside in adjacent sensors, that is, a hop-by-hop communication protocol is not assumed. Sensors communicate with other sensors only 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 over 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 fewer 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 with which our main goals are to reduce the number of sensors' measurements and the bandwidth used. The GPS-free approach reduces WSN cost (Savvides, Chan, and Srivastava 2001) and works 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, in this article we use 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. The mobile agent

approach has the advantage of providing a simpler design of the system's behavior. In addition, we could also consider extending the functionalities of mobile agents, such as monitoring both the plume and the source by applying flocking techniques. In these cases, a token-based approach would not appropriately capture that swarm behavior.

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. Sensors do not know their positions (i.e., no global position system is assumed). Sensors are identical and they run the same software. Transmission collisions are handled by lower MAC-layer protocols and are not considered in this article. Sensors follow the LPL mode described earlier and no multihop protocol is assumed. Sensors are reactive to agents' requests. No proactive behavior is assumed from the sensor side. With every T_w tick, 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 implies a high global exploration and also a higher cost (i.e., an increment on the sensors' measurements and on the number of messages sent). Moreover, sensors send data measurements 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 data.

Algorithm 1 The Sensor Algorithm

```
if (timeElapsed(Tw)) then
    if (Random() <Pa) then
        CreateAgent()
    end
end
if (sensorReadRequestEvent()) then
    sendSensorData()
end
```

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 using the sensors' communication devices. 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 neighbor 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, and 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 closest 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. 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).

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 (neighbours)
    if (bestSensor.data > sensorData) then
        moveToSensor(bestSensor)
        if (existAgentInSensor ()) then
            exit()
        end
    end
end
```

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 (i.e., it sends the information to the sink) until an environmental 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.

EXPERIMENTS

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

The simulation has been implemented using REPAST (Samuelson and Macal 2006) for modeling sensors and agents, and the moving peaks

benchmark (MPB) (Branke) for modeling the environmental changes (diffuse events). MPB is a benchmark created to compare dynamic function optimization algorithms, providing a fitness function changing over time. The function is composed of different peaks (cones) that change in width, height, and position. These peaks are used as diffuse events in our simulation. Notice that, in the absence of noise, the plume follows a monotonous increment of concentration to the source. 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:

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

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 environmental change occurs at each $t_c = 200$ ticks. That is, a simulation holds 1000 environment changes. In each environmental change the diffuse events change the position, intensity, and size. The results reported are the averages of these 1000 changes. Simulations take place in a rectangular space of $10^3 \times 10^3$ square meters where 1000 sensors are distributed randomly. The number of diffuse events varies from 1 to 3 with a radius of the plume ranging from 30 to 5000 meters, and the frequency of an agent creation event is $T_w = 20$ ticks. From results reported later, the probability of creating an agent $P_a = 0.5\%$, the number of nearby sensors receiving a data request from an agent $n_s = 3$, and the sensors' communication range is 80 meters (Table 1 summarizes the configuration of MPB).

Figure 3(a) shows an example of a simulated scenario, in which the sensors are spread over the space and three 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 mobile agents. White-filled points represent sensors with a hosted mobile agent. Circles represent communication ranges of sensors hosting an agent that has detected an event; n_s sensors within the circle will receive the data

TABLE 1 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

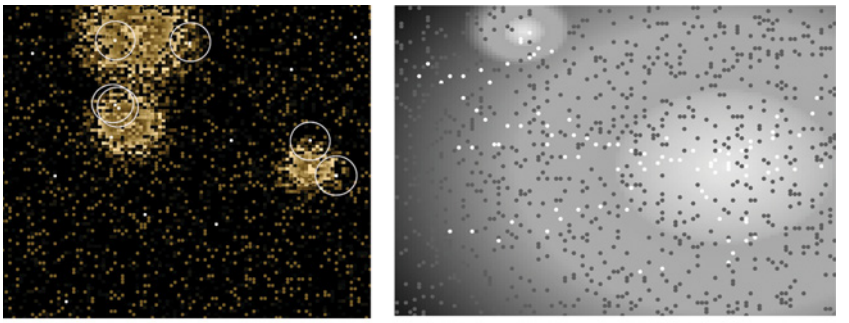


FIGURE 3 Scenario examples. (Figure is provided in color online.)

requests. Figure 3(b) shows an example of a scenario with two diffuse events without noise where the optima have been found. Dark nodes are those that have not been visited, and white nodes are those that have been visited by agents. White nodes describe the trajectories followed by different agents from the agent's creation to the diffuse event sources. Notice that in most of the cases, one source is found by more than one agent.

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; i.e., up to 3 at each environmental change). These values are measured for each environment change. 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), in other words, 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 agents are ready to send the sensor data to the sink. The cost of sending the information to the sink depends on the routing algorithm 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 agents reach event sources. We performed an additional experiment for measuring the number of sensor data reads and messages when no diffuse events are present (i.e., for measuring the cost of the global exploration). For all the following experiments, we consider the parameters described above, unless otherwise specified.

Varying the Number of Sensors in WSN

These first experiments 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 (Intanagonwiwat et al. 2002). In these experiments, the number of sensors varies from 500 to 8000, and noise is not applied to the sensor data reads.

The first observation is that, when the density of sensors increases, the number of failures decreases, in other words, agents are able to find better paths to navigate toward event sources (see Table 2). Notice that the number of failures reaches 35% only when the number of sensors is low (500). This percentage of failures 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 difficult scenarios require more than 1000 reads. Notice that difficult scenarios are those in which the diffuse events have overlapping areas in which at least one of the diffuse events is covered by a low number of sensors (small diffuse event). Notice that we consider a convergence only when *all* the event sources of a given scenario are located. Moreover, detecting a local optimum, in addiction to detecting *all* event sources (i.e., global optima), is not considered a failure. If a high concentration is measured the sink should be notified even when it is not the real source. Analogously to dynamic optimization problems, local optima are likely to become global optima when environmental changes occur.

The results achieved in this first simulation show that our approach is able to find all the diffuse event sources with a probability of 80% when the number of reads is ∼40% of the number of sensors, and the number of messages is ∼60% of the number of sensors (line 2 of Table 2). The number of messages and reads grows linearly with the number of sensors, wheareas the number of failures decreases (good scalability).

Varying the Communication Range

In these experiments, the goal is to analyze the algorithm's behavior with different sensor communication ranges. We use 4000 nodes and we

TABLE 2 Varying Sensor Number Without Noise

Sensor Number	Reads	Msgs	Failures	Adj. avg.
500	326.10	543.42	35.3%	9.2
1000	422.82	695.23	15.9%	18.76
2000	676.08	1106.77	4.9%	37.09
4000	1158.56	1887.11	2.9%	74.94
8000	2334.98	3805.85	1.9%	149.7175

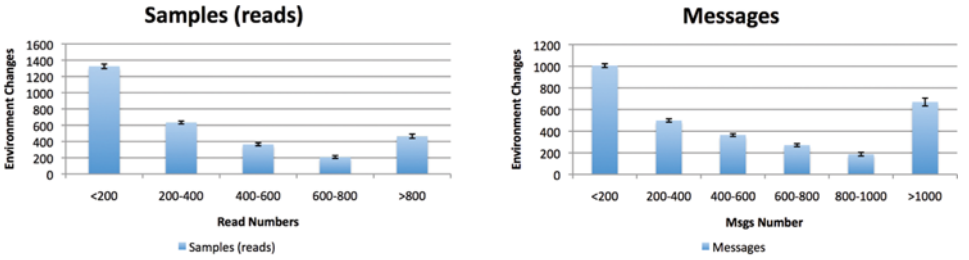
TABLE 3 Varying Communication Range Without Noise

Comm. Rng	Reads	Msgs	Failures	Adj. avg.
20	4236.44	7081.07	32.8%	4.88
40	1078.08	1755.21	0.8%	19.42
60	1108.58	1804.63	0.7%	42.92
80	1158.56	1887.11	2.9%	75.38
100	1380.94	2263.52	6.3%	115.60

vary the communication range from 20 to 100 meters. Table 3 shows the number of reads, number of messages, and the percentage of failures for each communication range. Similarly to the experiment, "Varying the Number of Sensors," the algorithm presents better results for an average number of adjacent neighbors between 18 and 75. The algorithm achieved the worst result when the average of adjacent neighbors is lower than 9 or higher than 120. This experiment shows that when the communication range is higher than 60 meters, the percentage of failures increases. Intuitively, we could think that long sensors' communication ranges induce better performances than short communication ranges, because the agents reach the source in a fewer number of hops (i.e., longer communication ranges allow the agents to cross long distances in a single hop). However, once the agents come close to the diffuse event source, long communication ranges make it difficult to precisely locate the sources. Agents tend to stay within the communication range of the sensor emitting the highest value, but are not able to approach further. Thus, when the source is in the communication range of one agent, the probability to find the source is equal to n_s (number of nearby sensors receiving the data request) divided by the number of sensors in the communication range.

Quality of Convergence

In these experiments we analyze the average of the number of reads and the average of the number of messages the approach needs to reach the convergence. Figure 4(a) shows how, for most of the scenarios, our approach is able to reach the convergence in fewer than 200 reads. The black line on the top of the bars shows the standard deviation over five runs with each run having 3000 environmental changes. More precisely, 1300 convergences of a total of 3000 are assessed with fewer than 200 reads, while 450 scenarios require more than 800 reads or do not converge at all. Figure 4(b) shows that similar results are obtained for the number of messages: 30% of the convergences are reached with fewer than 200 messages.



(a) Reads histogram (b) Msgs histogram

FIGURE 4 Performance results. (Figure is provided in color online.)

Varying the Noise Factor

So far, all the experiments considered a noise-free environment. 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. Notice that when the environment is subjected to noise, the plume does not follow a monotonous decrease when moving away from the source. Table 4 shows 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. This result is achieved because noise introduces a stochastic behavior that increases the exploration in the search. This higher exploration increases the number of reads and messages, but produces a better convergence (lower percentage of failures). We can also note that our algorithm 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.

Varying Local Exploration Without Noise

In these experiments we studied the performance of the algorithm when varying the local exploration in a noise-free environment. Local

TABLE 4 Varying the Noise Factor γ

γ	Reads	Msgs	Failures
0%	422.82	695.23	15.9%
$\pm 2\%$	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 5 Varying the n_s Parameter without Noise

n_s	Reads	Msgs	Failures
1	467.45	656.24	17.3%
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%

exploration is controlled by the number of sensors that an agent uses to decide its next location (n_s). In Table 5, we observe that even when we increase to 10 the number of requested sensors, the number of failures does not significantly decrease. The reason behind this result is that increasing local exploration is not enough to detect all of the diffuse event sources. Specifically, global exploration is the main factor of failures. As expected, the number of messages and sensor reads increases when local exploration is higher. From the results of these experiments (see Table 5), we set the parameter $n_s = 3$.

Varying Local Exploration With Noise

In these experiments we analyzed the performance of the algorithm changing the local exploration (i.e., the n_s parameter) for a noise factor of $\pm 4\%$ (Table 4). Similarly to the experiment "Varying the Local Exploration Without Noise," increasing the number of nearby sensors receiving a data request does not reduce dramatically the number of failures (i.e., even with n_s equal to 20, the approach cannot reduce the percentage of failures to less than 12%). Moreover, the increment of local exploration produces an increment of the number of reads and number of messages, but it does not produce any significant reduction of the number of failures. The best results, for the number of reads, number of messages, and percentage of failures, are assessed for n_s values between 3 and 5.

Varying Global Exploration Without Noise

In the previous experiments we observed that, even increasing the local exploration, the number of failures is not significantly reduced. Thus, the goal of the current 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

TABLE 6 Varying the n_s Parameter with Noise $\gamma = \pm 4\%$

n_s	Reads	Msgs	Failures
1	624.27	879.90	23.9%
2	655.37	1029.82	18.9%
3	698.31	1160.37	15.1%
4	728.97	1254.22	13.8%
5	756.40	1331.84	13.9%
6	788.16	1411.77	13.3%
10	954.97	1778.99	12.3%
20	1417.07	2725.35	12.0%

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 7 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 8). 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 approach, depending on the search priority. This trade-off between the quality of the results and the cost can be used to control the priority of the search process. Emergency situations will tend to increase the exploration cost. Notice that even when we reduce the percentage of failure to 0.3%, 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.

Varying Global Exploration With Noise

The goal of these experiments is to demonstrate that the rise of failures produced by the noise can be reduced by increasing the global exploration.

TABLE 7 Varying Agent Creation, P_a

P_a	T_w	Reads	Msgs	Failures
0.2%	20	322.30	538.29	37.7%
0.5%	20	422.82	695.23	15.9%
1%	20	484.01	783.83	4.5%
5%	20	654.21	1024.92	0.3%

TABLE 8 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	422.82	695.23	15.9%
0.5%	50	307.75	513.11	38.5%

In these experiments we fixed the noise level to $\gamma = \pm 4\%$ and we varied the P_a parameter (i.e., creation probability) from 0.2% to 5%. As it is reported in Table 9, by increasing incrementally the global exploration (i.e., P_a parameter), the approach reduces the percentage of failures while keeping a reasonable number of reads and messages.

Experimental results have demonstrated that, even in the presence of a high noise level, the number of failures is reduced by incrementally increasing 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 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.

The Exploration Cost

The goal of these experiments is to measure the exploration cost when no diffuse events are present in the system (the most frequent case). Specifically, we tested our approach when different noise levels are applied. Notice that noise is acting as false plumes that temporarily drive agents through the WSN. Table 10 shows that when the noise level increases from 0% to ±2%, the exploration cost increases by 50%. Thus, we may conclude that noise increases the exploration cost. However, this increment remains constant, even when we increase the noise to ±5%, or even to ±10%. Thereby, our approach does not depend on the noise level.

TABLE 9 Varying Global Exploration with Noise $\gamma = \pm 4\%$

P_a	T_w	Reads	Msgs	Failures
0.2%	20	525.94	881.25	39.8%
0.5%	20	622.29	1031.82	14.7%
1%	20	819.022	1351.79	5.0%
5%	20	1093.08	1787.67	1.0%

TABLE 10 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

Tolerance to WSN Failures

Finally, we analyzed the robustness of our approach when sensors fail. We consider two kinds of failures: (1) when the sensors turn off unexpectedly, and (2) when sensors are providing a wrong measure (i.e., maximum value).

Simulation Failures when Sensors are Turned Off

In these experiments, we consider sensor failures corresponding to the case of some sensors turning off unexpectedly (i.e., the most usual failure). 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 (i.e., their state is off). Then, those sensors cannot be used until the next T_w interval, when the sensors may continue to be turned off or have become fixed. Analyzing Table 11, we may observe that the increase in the sensor failures involves a decrease of system convergences that is significant only when the probability of failures reaches 40%. This result is achieved because the algorithm is tolerant to sensors' failures. Indeed, each agent chooses at random three neighboring sensors that are turned on; if one of the neighboring sensors is broken or turned off, it simply doesn't appear among the available neighbors. The worst case would be that no node in the neighborhood is turned on. So, this type of failure corresponds to having fewer nodes in the system. When the probability of failure is high (i.e., 40%), the algorithm decreases the performance. This behavior is not caused by the WSN failures but by the lack of sensors in the system. Thus, when the global exploration increases

TABLE 11 Failure Tolerance

Failure Prob.	Reads	Msgs	Failures
0%	422.82	695.23	15.9%
5%	430.56	708.39	16.2%
10%	409.97	675.38	16.9%
20%	422.26	697.46	19.6%
40%	463.55	772.60	30.7%

(e.g., by increasing the probability of an agent's creation from 0.5 to 2.0), the system is able to decrease the failures to 4.7% (with an average of 705 reads and 1145 messages), in the same way that the algorithm can reduce the number of failures when the number of sensors in the WSN decreases. Thus, we may conclude that our approach recovers from the failures and reaches the convergence even with a high probability of sensor failures.

Failure Simulation when Sensors are Providing Wrong Measures

In these experiments, the sensors that fail provide a maximum value (instead of the actual value). We analyze the performance of our proposal in terms of number of reads, number of messages, and percentage of convergence failures. When a sensor provides a maximum value, it is true that it can act as a false positive and attract agents that are exploring the sensor network in its neighborhood. However, it will not attract all agents, because our approach is based only on local interactions. A single agent would be actually attracted only if both: (1) the failing sensor is in the communication range of that agent; *and* (2) the failing sensor is among the three randomly chosen nodes. Furthermore, the failing node scope is limited to its communication range and it does not prevent the system at the global level from finding other real sources.

Table 12 shows the different performances achieved when we vary the sensor failures probability. We may observe that when the probability of failure is lower than 5%, the number of reads, messages, and the convergence failures increases but not significantly. When the probability of failures increases to 10%, the performance decreases dramatically. We can compensate for this behavior by increasing global exploration from $P_a = 0.5$ to $P_a = 2.0$. The algorithm then reaches the convergence with an average of reads = 1747.07, msgs = 2969.01, and a percentage of failures equals 13.6%. Thus, we demonstrate that our proposal is tolerant also to this kind of failure. The overall result of getting such a false positive affects performances only in number of reads or number of messages, but does not affect at all the detection of the real sources.

TABLE 12 Wrong Measure Tolerance

Failure Prob.	Reads	Msgs	Failures
0%	422.82	695.23	15.9%
1%	551.01	919.66	17.8%
2%	589.79	990.09	18.0%
5%	870.43	1480.42	22.3%
10%	1292.06	2218.34	34.8%

CONCLUSIONS

In this paper we have proposed a new approach, based on a mobile multiagent technology, to detect diffuse event sources in dynamic and noisy environments using a wireless sensor network infrastructure. To the best of our knowledge, this problem has not been addressed previously. Our approach proposes a distributed and decentralized algorithm based on local interactions and local knowledge of the environment. Different strategies have been designed to keep a low number of agents while 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. We also showed that this degradation can be alleviated by increasing the exploration level, with a reasonable rise in the cost to reach the convergence. Importantly, in our approach the cost of global exploration does not depend on the noise level. Because our approach is not introducing any assumption on the sensor positions, we plan to explore its capabilities in scenarios such as underwater applications or 3-dimensional spaces.

NOTE

1. http://www.algosystems.gr/spread/index.html (accessed February 16, 2012).

REFERENCES

Blackwell, T. 2007. Particle swarm optimization in dynamic environments. In *Evolutionary computation in dynamic and uncertain environments*, ed. S. Yang, Y.-S. Ong, and Y. Jin, 29–49. vol. 51 of *Studies in Computational Intelligence*. Heidelberg, Germany: Springer.

Blatt, D., and A. O. Hero 2006. Energy based sensor network source localization via projection onto convex sets (POCS). *IEEE Transactions on Signal Processing* 54(9), 3614–3619.

Branke, J. The moving peaks benchmark website. www.aifb.unikarlsruhe.de/jbr/movpeaks/ (accessed February 3, 2008).

Corkill, D. D., D. Holzhauer, and W. Koziarz 2007. Turn off your radios! Environmental monitoring using power-constrained sensor agents. In *AAMAS workshop on agent technology for sensor networks*, 31–38. Toronto, Canada.

Croce, S., F. Marcelloni, and M. Vecchio 2008. Reducing power consumption in wireless sensor networks using a novel approach to data aggregation. *The Computer Journal*, 2:227–239.

Ermis, E. B., and V. Saligrama 2006. Detection and localization in sensor networks using distributed FDR. In *Proceedings of the conference on information sciences and systems (CISS-06)*. Princeton, NJ.

Fernandez-Marquez, J. L., and J. L. Arcos 2009. An evaporation mechanism for dynamic and noisy multimodal optimization. In *Proceedings of the 10th genetic and evolutionary computation conference (GECCO)*, 17–24. New York, NY.

Fernandez-Marquez, J. L., J. L. Arcos, and G. D. M. Serugendo 2010. A decentralized approach for detecting dynamically changing diffuse event sources in noisy wsn environments. In *SASO*, 257–258.

Intanagonwiwat, C., D. Estrin, R. Govindan, and J. Heidemann 2002. Impact of network density on data aggregation in wireless sensor networks. In *Proceedings of the international conference on distributed computing systems*, 457–458. Washington, DC: IEEE Computer Society.

Lung, R. I., and D. Dumitrescu 2007. A collaborative model for tracking optima in dynamic environments. *IEEE congress on evolutionary computation*, 564–567. Los Alimitos, CA: IEEE.

Na, J., Lim, S., and C.-K. Kim 2008. Dual wake-up low power listening for duty cycled wireless sensor networks. In *EURASIP Journal on Wireless Communications and Networking*. New York, NY: Hindawi Publishing.

Ruair, R. M., and M. T. Keane 2007. An energy-efficient, multi-agent sensor network for detecting diffuse events. In *Proceedings of the international joint conference on artificial intelligence (IJCAI-07)*, 1390–1395. San Francisco, CA: Morgan Kaufmann.

Samuelson, D., and C. Macal 2006. Agent-based simulation comes of age. *OR/MS Today* 4:34–38.

Savvides, A., C. Chan, and M. Srivastava 2001. 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)*, 166–179. New York, NY.

Vinyals, M., J. A. Rodriguez-Aguilar, and J. Cerquides 2011. A survey on sensor networks from a multi-agent perspective. *Comput. J.* 54(3):455–470. Oxford, UK: Oxford University Press.

Weimer, J., B. Sinopoli, and B. H. Krogh 2009. Multiple source detection and localization in advection-diffusion processes using wireless sensor networks. In *30th IEEE real-time systems symposium (RTSS)*, 333–342. IEEE. Washington, DC: IEEE.

Yang, L., and C. Feng 2006. Adaptive tracking in distributed wireless sensor networks. In *Proceedings of 13th annual IEEE international symposium and workshop on engineering of computer based systems*, 103–111. Washington, DC: IEEE.