Self-Configuring Sensors for Uncharted Environments

Norman Salazar, Juan A. Rodriguez-Aguilar, Josep Lluis Arcos
IIIA, Artificial Intelligence Research Institute
CSIC, Spanish National Research Council
Bellaterra, Spain
{*norman,jar,arcos*}*@iiia.csic.es*

***Abstract*—Sensor networks (SN) have arisen as one of the most promising monitoring technologies. The recent emergence of small and inexpensive sensors ease the development and proliferation of this kind of networks in a wide range of actual-world applications.[1] So far the majority of SN deployments have assumed that sensors can be configured prior to their deployment because the area and events to monitor are well known at design time. Nevertheless, when the purpose of an SN is to monitor the events of an environment such that the distribution and nature of its events is uncertain, we cannot longer assume that sensors can be configured at design time. Instead, sensors must be endowed with the capacity of autonomously reconfiguring and coordinating in order to maximize the amount of information they perceive over time. In this paper, we propose a low cost (in terms of energy and computation) collective distributed algorithm, the so-called *collective search diffusion* (CDS) algorithm, which allows the sensors in an SN to collaboratively search for the configurations that maximise the information that they perceive based only on their local knowledge. We empirically show that the CDS algorithm helps an SN efficiently monitor environments where various dynamic events occur while showing high degrees of resilience to sensor failures. Both features make the CDS algorithm a suitable tool for monitoring remote and/or hostile uncharted environments.**

***Keywords*-Self-Configuration; Sensor Networks; Coordination; Collective Search;**

I. Introduction

As technology continuously improves, it is becoming apparent that sensor networks (SN) are a powerful and versatile tool [3]. They have been employed by numerous applications on domains of a wide range of characteristics. Nevertheless, many of these applications rely on static sensor configurations (i.e pre-configured at design time), which can be detrimental. It has been argued that in real-world deployments the complexity, diversity, and dynamicity of the sensing requirements is a major issue that cannot be tackled through static configurations.

In particular, many of the events to sense have a dynamic nature. They may continuously expand or shrink (diffuse events), or move over the environment (moving events) [4]. Examples of these events are wildfires, glacier movements, gas plumes, and warm water currents, among others. Within such dynamic settings, it is necessary that sensors are configurable. Thus, sensors must be allowed to change their configurations to vary the content, resolution and accuracy of their observations to maximise, in an energy-efficient manner, the information gathered over time. Therefore, an SN must count on *active sensing* capabilities[3]: "the capacity of autonomously reconfiguring and coordinating its sensors in order to maximise the amount of information perceived over time". Henceforth, we consider a sensor's configuration as a schedule of (parameterized) actions the sensor must take to monitor/control/track some particular event. For instance, to monitor a wildfire, a sensor needs to measure the heat levels, humidity, look for carbon monoxide, and/or detect smoke.

Moreover, it has been noted that in large environments various distinct events are prone to occur at once. In other words, there is a spatial distribution of concurrent events. Hence, a sensor's configuration depends on the event(s) present on its geographic location (a sensor must be able to adopt as many configurations as events are possible). In these cases, it is likely that neighboring (close-by) sensors experience the same event(s), consequently making them require similar (the same) configurations. *Collective active sensing* [3] strategies are investigated for SNs whose sensors need to coordinate to collectively perform some sensing task. Regarding the settings above, a collective sensing strategy would be necessary to have sensors requiring similar configurations coordinate and cooperate towards a common goal (discovering the most useful configuration).

Collective active sensing strategies in dynamic environments have recently spurred research. On the one hand, according to the *dynamic region theory* (DRT) [5] sensors can select their configurations from a pre-defined set of configurations by identifying their spatial locations (regions) and their neighbors. Notice that this approach relies on the fact that the sensing requirements of each possible event and location are known at design time. In other words, it requires a thorough study of the deployment environment and a deep knowledge of its possible events. On the other hand, coalition formation based approaches have been also employed for active sensing. Sims et. al [6] attempt to find the coalition of sensors to perform each task out of a set of available (sub)tasks in such a manner that some utility

[1]We refer the interested reader to [1] and [2] for surveys of a wealth of applications of SNs.

is maximized. Likewise DRT, this coalition-based approach also depends on a through knowledge (at design time) of the tasks or subtasks to perform. Additionally, the approach assumes that the sensors have (near-) complete information of the other sensors (or at least of a subgroup).

To summarise, the common assumption of previous works in the literature is that the deployment environment has been well studied, and thus that the sensor designers and the sensors themselves can make use of the available domain knowledge for configuration purposes. Nevertheless, this may not always be the case. It has been argued that sensor networks can be particularly useful in remote or hostile environments that have rarely been studied due to their inaccessibility [7]. Therefore, how to employ some collective active sensing in these *uncharted environments* remains an open issue that we address in this paper.

Here we propose a collective approach to monitor uncharted environments where only (at most) partial domain knowledge is available to the sensors in an SN. In our approach, we embed in an SN a distributed algorithm that: i) has a low computational overhead and a low energy consumption; ii) employs diffusion search to collectively find/construct the most useful sensor configurations for the occurring events; iii) promptly reconfigures the sensors in response to dynamicity of the events; and iv) works when the sensor cannot rely on the available domain knowledge (uncharted environment).

This paper is organized as follows. Firstly, we describe and formalize the problem to solve. Next, we present the algorithm used to solve the problem. Finally, we empirically evaluate our algorithm and draw some conclusions.

II. Problem description

Our objective is to employ an SN to act in an uncharted environment, where acting refers to monitoring, controlling or tracking events. By uncharted environment we mean a location of which we only have partial domain knowledge. In other words, unlike in most current SN applications, we are not aware of the kind of events that may occur, nor of their possible locations in the environment.

An event stands as some phenomenon (of interest) that occurs in some geographic area of the environment. Moreover, in large environments various distinct events are likely to occur at the same time in different areas. Figure 1.a illustrates an environment where four different events occur simultaneously (each color represents a distinct event). Observe that the presence of multiple, spatially-distributed events has the effect of partitioning the environment. However, in uncharted environments the events (and thus its partitioning) may not be known at design time. Therefore, sensors need to be capable of configuring themselves according to events occurring in each location in such a manner that their selected configurations allow each of them to efficiently monitor/control/track the events.

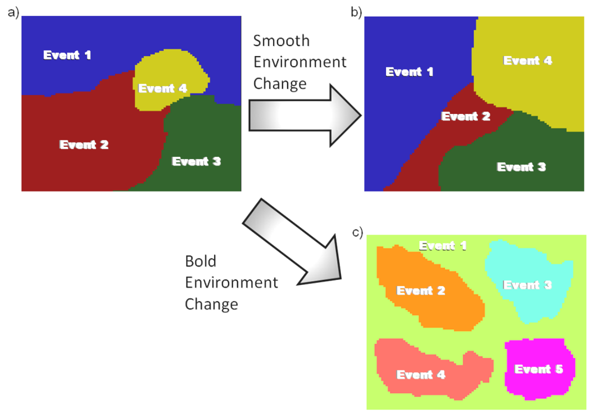

Figure 1. a) Distribution of four different events in an environment, (each color signifies a distinct event); b) state of the environment after the (diffuse) events expand and/or shrink; c) state of the environment if all previous events disappear, and new distinct ones take their place.

Furthermore, in real-world situations, events usually change over time (i.e they are dynamic, with a diffuse or moving nature). For example, figures 1.a and 1.b depict the transition of four diffuse events (by expanding or shrinking); whereas the differences between figures 1.a and 1.c show a harsher transition in which the four events disappear and five new ones take their place (the new colors in figure 1 represent the new events). Hence, the sensors also need to adapt their configuration to cope with changes.

Before formalizing our problem, we need to detail some working assumptions.

A. Assumptions

Firstly, we assume that a large number of sensors may be deployed in a non-deterministic manner (e.g they are dropped from a helicopter) over a large environment. Thus, only those events that are geographically available to the sensors' locations will be monitored [6]. We also assume that the sensors are technologically capable of monitoring many different types of events (i.e for any event there exists a sensor configuration capable of monitoring it).

Following [5], we consider that collections of neighboring sensors are likely to reside on the same environment partition. Hence, experiencing the same event for periods of time during the operation of an SN. Therefore, they are also likely to require similar configurations.

Moreover, the uncharted nature of the environment means that the possible events are not catalogued (identified). Neither are the geographic areas where events may occur. In other words, a mapping between regions (locations) and events is not available. Consequently, the sensors need to have at their disposal a large set of possible configurations to cope with a large variety of possible dynamic events (i.e the sensors are highly configurable).

Moreover, we assume that each sensor has a preference structure [8] that expresses the satisfaction of any particular configuration when faced with a choice between different alternatives. Thus, a preference structure brings together all possible alternatives and represents a sensor's preferences over the set of possible configurations. In order to value a configuration, we assume that each sensor can measure the value of the information (observations) collected. This approach is similar to the one taken in [9], and in general it is a common in the data fusion and the tracking literature.

Finally, the sensors may need to be some period of time in the environment to find their proper configurations. Therefore, even though sensors can be added or removed at any point in time, the sensor population does not fluctuate wildly [6].

B. Formalization

It is time to formalize the monitoring problem faced by sensors considering the assumptions in section II-A. At any time t there is a fixed set of sensors in the network $S = \{s_1, \ldots, s_n\}$ and a variable number of spatially-distributed events E^t. Henceforth, we shall refer to E as the set of all events over time. Let $r : E \to S$ be a function that maps each diffuse event to the sensors spatially situated in the same region. We assume that all sensors are homogeneous and there is a finite set of configurations K for them. By configuration we mean a schedule of parameterized operations (e.g. measurements: heat, water level, presence of gas; processing information, processing, sleeping). Let $u_s : K \times E \to \mathbb{R}$ be a sensor utility function that allows each sensor to assess the utility of a given configuration. Based on this definition, given some event $e_i \in E$ we can assess the utility of the configurations of the sensors situated in the very same region as e_i as $u_e(e_i) = \sum_{s \in r(e_i)} u_s(\kappa_s, e_i)$, where κ_s stands for some configuration of sensor s. Now we can readily derive the utility of the configuration of a whole sensor network, the *network utility* u_N, for a distribution of events E^t as $u_N(E^t) = \sum_{e_i \in E^t} u_e(e_i)$. Finally, we can pose the problem faced by the sensors in the network as that of finding at each time t the configuration $\mathcal{K}^*$ that maximizes the network utility such that:

$$\mathcal{K}^* = \underset{\mathcal{K} \in K^n}{\arg\max} \sum_{e_i \in E^t} \sum_{s \in r(e_i)} u_s(\kappa_s, e_i) \qquad (1)$$

where K^n stands for the n-ary Cartesian product over the set of configurations K, and hence $\mathcal{K} = \kappa_1 \times \ldots \times \kappa_n$ stands for a joint configuration of the sensors in the network.

C. Challenges

The main challenges of the problem originate from the lack of a priori information in an uncharted environment. Because neither the possible events nor the environment characteristics are known beforehand (at design time), a region identification algorithm cannot be used to select the appropriate configurations (unlike [5]). Hence, the sensors must *search* for their proper configurations in a likely large configuration space.

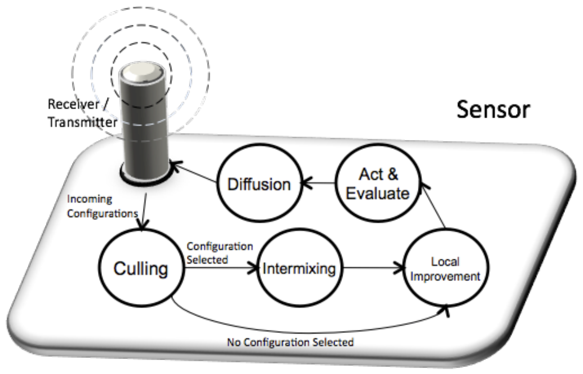

Figure 2. Diffusion state transition.

Additionally, although we assume that neighboring sensors are likely to experience the same event (thus requiring similar configurations), this may not always be the case. Sensors close to the frontier of two (or more) different events can have neighbors that require completely different configurations.

Moreover, there are also challenges related to the (technological) capabilities of sensors. Even though sensors are becoming more and more computationally powerful they are still somewhat limited. Therefore, any approach used to solve the problem should not take a significant amount of processing power away from the sensor duties. In other words, a low computational overhead is desired. Furthermore, it is well known that sensors are quite restricted energy-wise. Hence, the algorithms employed by the sensors must be as energy efficient as possible.

III. A Collective Approach

In this section we propose an approach to solve the problem described in section II, namely for sensors to dynamically find the most useful configurations required to monitor the dynamic events occurring in their geographical location. The proposed approach is designed to function in uncharted environments (where only partial domain knowledge is available) using the sensors' local feedback.

As stated before, it has been argued [5] that in environments with spatial events (e.g diffuse events) it is safe to assume that neighboring sensors may require the same (or a similar) configuration since they are prone to be monitoring the same event. Therefore, it is reasonable for sensors to collectively coordinate and cooperate to discover the most useful configurations, hence solving the problem in equation 1.

In situations where the sensors are deployed to an uncharted environment, it may be the case that the only available useful information is the one provided by the sensors' own feedback function. Under such circumstances, cooperation becomes necessary since sensors can improve their partial domain knowledge (regarding the configurations) by sharing their local experiences. Moreover, the number of possible configurations may be very large, thus it may be unfeasible for the sensors to individually search for their configurations. Furthermore, cooperation and coordination are also useful computationally speaking, because even though sensors are becoming more powerful, their resources (e.g CPU) are still constrained. Hence, if multiple sensors search for the same configuration, they can save time and power by searching together. Once a sensor finds a good configuration, it can be promptly shared with its searching peers.

To that aim, we designed the *collective diffusion search* (CDS) as an algorithm based on the collective sharing of configurations amongst neighboring sensors. The state machine of the CDS (as implemented by each sensor) is shown in figure 2. In what follows we describe the main components of CDS and their rationale.

A. Diffusion

We opt to employ diffusion as the component in charge of sharing the configurations since we regard it as an efficient (computation-wise) mechanism. In a sensor, diffusion consists in a broadcast (to its close-range neighbors) of its configuration. However, sending a broadcast (message) requires energy consumption. Hence, if a sensor's priority was to save as much energy as possible, then it is in its best interest to reduce its number of broadcasts. With that aim, a sensor's likelihood of sending a broadcast can be regulated through a *probability of diffusion* ($p_{diffusion}$). The higher the value of this probability, the most likely a sensor is to broadcast its configuration. From a computational perspective, diffusion has a low overhead on the transmitting (sensor) side, since it amounts to sending a message without caring who will receive it. Nonetheless, receiving various broadcasts raises an issue, because a receiving sensor needs to decide what to do with these received configurations

B. Culling

Attaching in each broadcast the utility of a configuration effectively provides a receiving sensor with the means to decide how to deal with multiple incoming configurations. This new information allows each receiving sensor to implement a culling component to dismiss useless (received) configurations. For instance, we implement this through a filter that selects the best received configuration and only if it is better than the sensors own.

Through diffusion and culling, groups of sensors that are close by and that experience the same event will adopt the same configuration. What is more, this has the effect of emerging of a common configuration per event partition, since a configuration is only diffused to where it is useful. For example, imagine an environment partitioned by two different events (e_1 and e_2) and consider a sensor, s_1, located on the e_1 partition but with at least a neighbor, s_2, on the e_2 side. Now assume that sensor s_1 knows the best configuration for its event (it has a high utility), which it will broadcast to its neighbors. Sensor s_2 on the e_2 partition receives the broadcast containing this highly valuated configuration, thus its culling will make the sensor adopt it. Nonetheless, since this configuration is not useful for event e_2, when employed by s_2 it will be valuated poorly. Therefore, even though sensor s_2 will still broadcast it to its neighbors in the e_2 partition, their culling filter will dispose it, halting the diffusion of the configuration on that side.

C. Intermixing

Notice that diffusion and culling do not have searching capabilities, at most they will establish the best configuration known by any of the sensors (per event). Thus, some searching needs to be incorporated since its unlikely to expect that some sensor knows its most useful configuration a priori. A low-overhead search method, consists in intermixing (combining) two configurations (the selected through culling and the sensor's current one) to create a new one. This can be regarded as using someone else's experience without completely forgetting your own. For instance, we implement this by combining a part of selected configuration with part of the sensor's current one (the parts are randomly decided), in such a manner that new configuration is constructed. Nevertheless, sensors cannot always depend on the usefulness of their neighbors configurations (e.g if all the neighboring sensors have the same configuration and it is not useful).

D. Local improvement

This component makes each sensor capable of searching for new configurations without depending on its neighbors. Moreover, this not only helps to improve the existing configurations, it is particularly necessary in environments with dynamic events since the events can disappear or appear unexpectedly. Local improvement can be accomplished (without expending much processing power) by introducing a random change to a sensor's configuration with some probability ($p_{improvement}$). Various disciplines have shown this to be effective [10].

E. Collective Diffusion Search

Altogether, in collective diffusion search each sensor continuously attempts to propagate its configuration while trying to improve it at the same time. The sensor receives some broadcasts which are then filtered through culling in an attempt to determine if there is a better configuration. In case there is, the sensor's configuration and the selected one are combined in an attempt to create a new (and ideally better) configuration. Afterwards (or if culling fails to select a configuration), local improvement can be used to continue the search for the best configuration. Once this is over, the sensors configuration is used and its utility valuated (act and evaluate in figure 2) through the feedback generated by the performed actions. Lastly (is a matter of perspective) the sensor wraps its configuration along with its utility into a message for broadcasting. An execution of the state machine shall hereafter be referred as a *communication cycle*.

Overall, this approach can be regarded as a distributed evolutionary process, since configurations evolve through time as a consequence of the constant application of diffusion, combination and local improvements. Once the configurations cannot evolve anymore, the end result is a stabilized set of useful configurations.

To summarize, collective diffusion search is a low overhead, but powerful distributed algorithm that when embedded in each sensor empowers them to dynamically find the most useful (utility-wise) configurations even when only partial domain knowledge is available (uncharted environments). The algorithm can be easily implemented in a sensor, since its formed of four *lightweight/low overhead* components: `diffusion`, `culling`, `intermixing` and `local improvement`.

IV. Experimental Results

The aim of our experiments is to verify two hypotheses, if through collective diffusion search: i) the sensors can find the configurations that maximize their utility according to the events that occur in their location; and ii) the sensors can reconfigure themselves in the response to dynamic events.

To that end, we designed three types of experiments: 1) event recognition (figure 1.a): recently deployed sensors must find the configurations needed to monitor the events occurring in the environment; 2) sensor reconfiguration against *smooth event changes* (transition from figure 1.a to 1.b): the existing events' area of covering changes over time (diffuse events); and 3) sensor reconfiguration against *bold event changes* (transition from figure 1.a to 1.c): existing events disappear and completely new events (different covering and features) take their place.

It is important to understand that in real-like situations dynamic events may change slowly through time. However, in our experiments we opt to make it harder for the sensors by introducing the changes in a more abrupt manner. In a real deployment such abruptness may actually occur from a sensor's perspective, since the sensors may sleep for long intervals. Thus, every time a sensor awakens the events may have different features.

A. Empirical settings

Each of our *experiments* consists of 50 discrete event simulations, each one up to 5000 ticks. Our simulation environment is formed by a 100 x 100 grid initially covered by four distinct events. Figure 1.a depicts with different

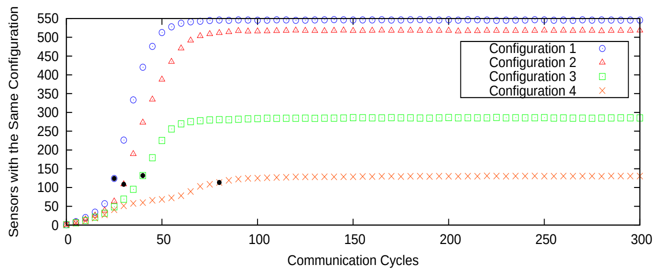

Figure 3. Results of convergence after the initial deployment. The black dots mark when the best configurations were found.

colors the form of each event at start up. Observe that all events are different from each other in area, shape, and borders. During a simulation a set of 1500 sensors is randomly deployed unto this environment. However, because we are evaluating their self-configuration capability in an uncharted environment, none of the sensors is aware of the environment partitioning nor of the possible dynamic events. Hence, each sensor starts with a random configuration.

A sensor configuration is given by an ordered sequence of 5 actions selected from a pool of 20 possible actions (i.e $\|K\| = 20^5$). Following the assumption in section II-A, our environmental feedback function valuates if the actions in the configuration are useful or not.

B. CDS settings

The parameters for the diffusion component were set to (unless otherwise indicated): a broadcast range of 4 cell and $\sim 20\%$ likelihood of broadcast per sensors at a given point in time ($p_{diffusion} = 0.20$). Whereas in the local improvement, each action in the configuration has a 0.0008 probability of randomly changing.

C. Metrics

To measure the usefulness of the CDS approach described in section III, we counted the number of sensors that found the configuration needed to monitor the event in their location. The counts of each simulation in the experiment were then aggregated using the inter-quartile mean. From here on, *configuration 1,2,3...* refers to configuration employed by the majority of the sensors located in the area of *event 1,2,3...*. Furthermore we used communication cycles (also defined in section III) as a time scale for the measurements.

D. Initial Deployment

The event recognition experiments aim to verify if collective diffusion search allows recently deployed sensors to find and adopt their most useful (utility-wise) configurations required by their location. Moreover, we are interested in measuring: i) how fast these configurations are found and adopted by sensors; and ii) how much energy (as a consequence of the transmitted messages) it requires.

Figure 3 shows that CDS is quite effective in finding the most useful configurations for almost all the sensors. Observe that once the configurations are found (black dots in the figure) the sensors promptly adopt them. These configurations are found at $\sim 20, \sim 30, \sim 40$ and ~ 60 communication cycles for each of the four events respectively. Figure 4 shows in more detail (for a single simulation and for the sensors in event 3) the search and adoption of the most useful configuration. Firstly, there is an initial variety of different configurations (initial spike in the standard deviation). Next, as a result of the diffusion and culling components collectives of sensors adopt similar (not so bad) configurations. The adopted configuration is then improved (through intermixing and local improvement) at each communication cycle, and with each improvement (observable in the average utility) more sensors start to adopt it (shown by the decrease in the deviation). This continues until the configuration that provides the highest utility is found and consequently adopted by most of the sensors.

However, observe that even though the average utility stabilizes ($\sim$ 1.0), the standard deviation still indicates that not all sensors adopt same configuration. On the one hand, this occurs because at any point in time the local improvement component causes a small number of sensors to try new configurations. On the other hand, because a sensor broadcasts its configuration to all its neighbors, the sensors in the frontiers (near the geographic border of two or more different events) may receive conflicting configurations from their neighbors (similar to the example in section III).

Moreover, notice that depending on features of the event, some configurations require more time to be adopted by sensors. This appears to be related to the dimension of the area occupied by the event, and thus by the number of sensors that require the same configuration. *Event 4* is a particularly pronounced example of this effect (sensors localized in the region of this event take the longest to find the best configuration and thus to adopt it: $\sim$ 60). Event 4 encompasses the smallest area and has the lowest number of sensors by far, which seem to give empirically credence to the idea that the number of sensors influence how fast a configuration can be found and adopted. From the algorithmic point of view, this is reasonable in a collective approach because fewer sensors are looking for the same configuration. Although there may be another factor to consider, the location of the event. Observe that event 4 is completely surrounded by the other events, which means that sensors in that area are constantly receiving conflicting configurations from their neighbors. Additionally, because of its small dimensions a broadcast originated in its frontiers may cover a significant area of the event. In other words, sensors as far as the center of the event may receive conflicting configurations.

To summarize, from these experiments we conclude that: i) through CDS sensors can find the most useful configuration for the events in their locations; and ii) the dimensions and location of the events affect how promptly such configuration is found and adopted.

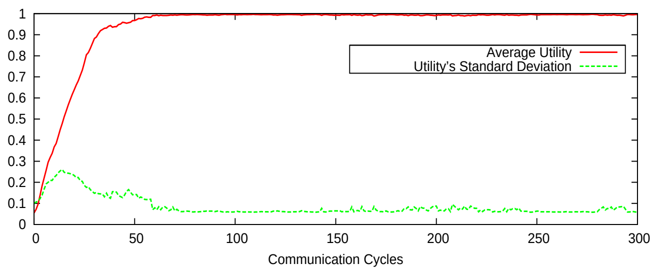

Figure 4. Normalized average and standard deviation utility (in a single simulation) of the sensors in the region of event 3.

E. Collective vs Individual Search

The results of the previous section have shown that collective search is an effective self-configuration approach. However, it is not clear if our approach provides an advantage against individual search. Therefore, in this section we experiment with sensors that employ an individual search mechanism instead of a collective one.

It is not hard to see that the CDS shares various similarities with evolutionary algorithms (EAs)[11], to the point that it could be considered a distributed EA. Hence, it makes sense to employ a classic EA as the individual search approach to compare with.

To that end, for the experiments in this section we endowed each sensor with an EA to help it find its best configuration. The chosen EA was a classic genetic algorithm [12] with a population of 20 configuration (per sensor). In other words, to complete an iteration of the algorithm each sensor needs to evaluate 20 configurations. This is significantly higher from the CDS which only requires one evaluation per iteration.

As to the actual results of the experiments, each sensor needs around 2000 iterations to individually find its best configuration. In other words, 4×10^5 configurations are evaluated per sensor. This is extremely high when compared to the CDS, which needs around 60 (in the worst case) to find the best configuration for *all* sensors.

Hence, we can conclude that collective search can be considered as the more appropriate choice for sensors self-configuration.

F. The Role of Diffusion

In section III, it was stated that a sensor can save energy by reducing its number of broadcasts, and that this is regulated by the probability of diffusion. Moreover, the experiments in the previous subsection showed that even though at any given time only $\sim 20\%$ of the sensors were broadcasting the best configuration was still found and

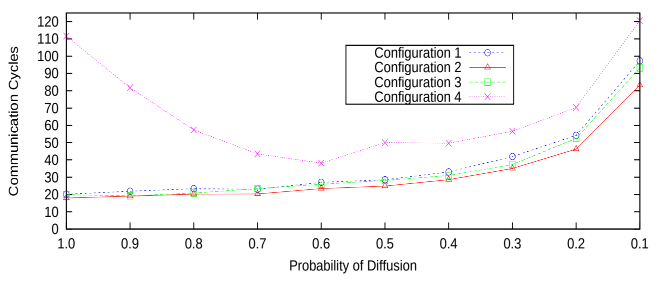

Figure 5. Communication cycles required by sensors to adopt the best configuration using different diffusion probabilities.

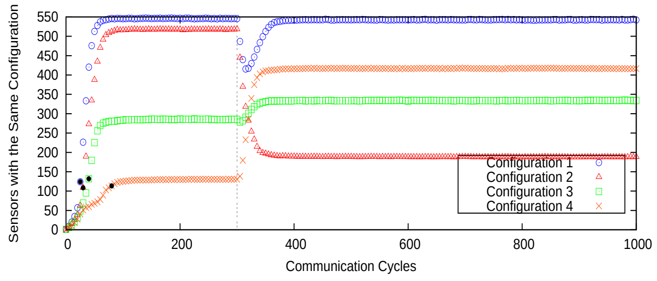

Figure 6. Reconfiguration after smooth environmental change. The vertical dashed line (300 cycles) marks the environment change.

shared in a reasonable amount of time ($\sim$ 70 communication cycles). Therefore, next we aim to understand how different probability values affect the CDS.

To that end we repeated the experiment from the previous scenario for different probabilities of diffusion ($p_{diffusion} \in \{1.0, 0.9, 0.8, 0.6, 0.5, 0, 4, 0.3, 0.2, 0.1\}$). Figure 5 summarizes the experimental results by showing the number of communication cycles required for the sensors (at least 90%) to find and adopt the most useful configuration (according to their localization). Observe that for most events the higher the diffusion probability (the larger the number of sensors broadcasting) the faster the most useful configurations are adopted.

However, small (area-wise) events do not follow this trend. With a high diffusion probability, sensors also require more time to find and adopt the most useful configuration. Such effect occurs because (almost) constant diffusion diminishes the search capabilities of the CDS in small areas, which were already reduced because of the low number of sensors in the event.

Although, at first glance it may appear desirable to use a somewhat high probability of diffusion, one needs to consider what such probability means energy-wise. For instance, when using the highest probability of diffusion a sensor broadcasts its configuration at every communication cycle, which requires considerable energy consumption. Overall, the number of transmitted messages (broadcast) used by the sensors to reach their best configurations (for the four events) is of $\sim$ 167500 (or $\sim$ 110 messages per sensor). Whereas using a low value (0.2) may slightly increase the number of communication cycles needed by most sensors to adopt their configuration but the reduction in the number of messages is quite significative ($\sim$ 18000 overall, or $\sim$ 12 per sensor).

Finally, we can conclude that the probability of diffusion represents a straightforward parameter that controls the trade off between the time needed to find the proper configurations and the energetic consumption (as a product of the transmitted messages).

G. *Dynamic Events*

The previous experiments have shown that CDS allows recently deployed sensors to configure themselves according to the existing environmental events. However, as stated in section II events are usually dynamic, i.e they change over time (e.g expand, shrink, appear, disappear). Therefore, the purpose of this section is to verify if sensors with CDS can reconfigure themselves in response to dynamic events. In what follows, we present the two different types of dynamic events against which we tested the CDS.

Smooth event changes

In these experiments some time (300 cycles) after the initial deployment, the area covered by each of the events expands or shrinks (diffuse events), but the most useful configuration required by each event stays the same (transition between figures 1.a and 1.b). Therefore, in such situations the sensors should reconfigure themselves by redistributing the existing configurations amongst themselves (i.e there is no need to search for new configurations). The CDS can accomplish this purely through the diffusion and culling components. Figure 6 shows that through CDS sensors respond promptly to changes in the environment (marked by the vertical line). The sensors, for which the event in their location changes, smoothly transition to their newly required configurations by adopting them from their stable neighbors. For instance, the sensors monitoring *event 1* (the blue circle line of configuration 1) suffer the most complicated transition since the region of the event shrinks from one side and expands from another. The configuration redistribution for these events is observable in figure 6 (between the $300-400$ communication cycles) by the sudden decline of the sensors with *configuration 1* followed by a sudden increment.

Moreover, events may be very dynamic and thus change continuously. Therefore, to validate the CDS we constructed a scenario that continuously transition back and forth between figures 1.a and 1.b. The results of the experiment are illustrated on figure 7 (only the configurations of the event 2 and 4 are shown since they are most affected). Notice that every time the smooth changes occurs the sensors

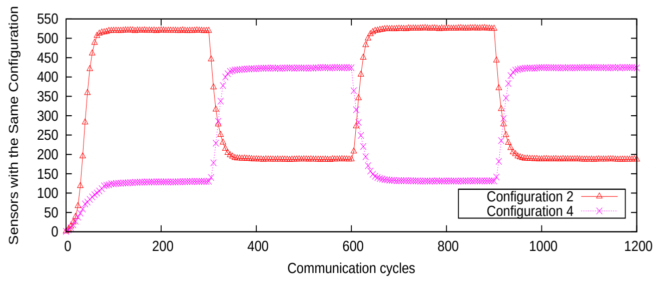

Figure 7. Reconfiguration when smooth changes occur continuously.

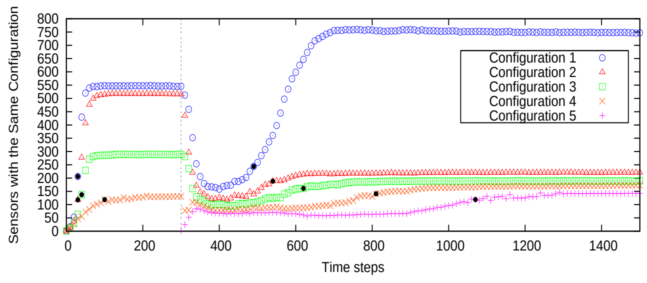

Figure 8. Reconfiguration after a bold environmental change. The vertical dashed line (300 cycles) marks the environment change.

react promptly to adopt their new configuration ($\sim$ 30 communication cycles).

Bold event changes

For these experiments we model a more extreme dynamic event: the sudden disappearance of the existing events and the sudden appearance of completely new ones (transition between figures 1. a and 1.c). Thus, unlike in the smooth change, the sensor need to search (again) for the most useful configurations required by the new events. Furthermore, since most of the sensors had (previously) stable configurations (that now are completely useless) there is an overall lack of configurations diversity (most sensors have one of the 4 previous configurations). Hence, during this stage diffusion and intermixing does not provide much help, and its up to the local improvement component to increase the configuration diversity (once there is some diversity intermixing speeds up the search).

Figure 8 shows the results of the simulations. Observe that after the bold change (the vertical line at 300 cycles) the sensors start to diversify their configurations (the number of sensors with the same configuration declines). Once some sensor(s) find some useful configurations they begin to collectively improve it until the most useful ones are found. However, unlike during the initial deployment, it takes longer ($\sim$ 600 cycles) to find the most useful configurations for all the five events. On the one hand this is caused by the above-mentioned lack of diversity at the moment of the change. On the other hand, like the experiments in subsection IV-D the dimensions and frontiers of the events affect how fast the configurations are found and adopted. For instance, after the bold change four of the five regions are considerably small (consequently there is a low number of sensors located in the event), which slows down the search. Furthermore, notice that even though the new event 1 covers a large area and its most useful configuration is (relatively) promptly found ($\sim$ 100 cycles) it take some time for most sensors located on the event to adopt it ($\sim$ 200 cycles). The reason behind this, is that the region of event 1 has a lot of frontiers which as we conclude before (subsection IV-D) affect the effectiveness of diffusion.

To summarize, from these experiments we conclude that: i) through CDS sensors can reconfigure themselves in response to both smooth and bold environmental changes; ii) the diffusion component makes the CDS particularly effective against smooth changes; iii) local improvement is a key component to accomplish reconfiguration if a sudden event where to appear; and iv) the speed of the search and adoption of configurations are indeed affected by the dimensions and frontiers of the events.

H. Fault Resilience

The experimental results so far have shown that CDS allows a collective of sensors to successfully (re)configure themselves over uncharted environments. However, our experiments have failed to consider that hazardous conditions are frequently a reason why such environments are uncharted. Thus, sensors may be more prone to failures, e.g because of the deployment impact (sensors may be thrown from an helicopter), fire or extreme heat and animal or vehicular accidents [13]. Therefore, in this section we show the *fault-resilience* capabilities of the CDS.

One could naively assume that the worst failure case would be if some sensors cease to function completely. Nonetheless, this type of failure is meaningless for CDS, since it is a collective approach that relies on the existence of multiple sensors. The actual worst failure cases are malfunctioning reception and/or malfunctioning sensing (measuring). The former, mainly refers to a lack of capability in receiving communications from other sensors, whereas the latter refers to measuring errors resulting from either damaged or badly calibrated equipment. Therefore, we can say that the CDS is fault resilient if it is capable of dealing with such kind of (worst case scenario) failures.

To that end we designed two sets of failure experiments to evaluate the CDS :

- a malfunctioning reception scenario; and
- a malfunctioning reception and sensing scenario.

Malfunctioning Reception

These experiments model a scenario where during the initial deployment some sensors (with probability $p_{failure}$) become incapable of receiving communications, thus making

	Event 3		Event 1		Event 2		Event 4	
Failure probability	Best configuration	# Non-failing sensors	Best configuration	# Non-failing sensors	Best configuration	# Non-failing sensors	Best configuration	# Non-failing sensors
None	98.54 %	291	99.42 %	540.432	98.28 %	535	98.47 %	133
Medium	96.66 %	236	99.54 %	429	97.57 %	428	94.03 %	106
High	85.84 %	147	84.29 %	270	91.87 %	271	62.43 %	66
Very High	19.64 %	87	22.05 %	165	19.74 %	159	19.37 %	39

Table I
PERCENTAGE OF WORKING (NON-FAILING) SENSORS WITH THE BEST CONFIGURATION FOR DIFFERENT RECEPTION FAILURE PROBABILITIES.

	Event 3		Event 1		Event 2		Event 4	
Failure probability	Best configuration	# Non-failing sensors	Best configuration	# Non-failing sensors	Best configuration	# Non-failing sensors	Best configuration	# Non-failing sensors
None	98.54 %	291	99.42 %	540	98.28 %	536	98.47 %	132
Low	97.68 %	262	97.81 %	486	97.81 %	481	90.33 %	120
Medium	93.33 %	235	90.71 %	429	96.589 %	428	25.13 %	106
High	03.43 %	202	11.90 %	378	04.26 %2	375	05.57 %	93

Table III
PERCENTAGE OF WORKING (NON-FAILING) SENSORS WITH THE BEST CONFIGURATION FOR DIFFERENT RECEPTION/MEASURING PROBABILITIES.

Failure probability	Event 3	Event 1	Event 2	Event 4
None	52	54	46	70
Medium	79	79	66	148
High	321	347	154	>350

Table II
COMMUNICATION CYCLES NEEDED FOR DIFFERENT RECEPTION FAILURE PROBABILITIES.

them incapable of actually adopting their proper configuration. However, since they are still capable of transmitting they will constantly send useless communications (their initial random configuration) to their neighboring sensors. We ran experiments where the failure probability was medium ($\sim$ 20% of the sensors fail), high medium ($\sim$ 50% of the sensors fail) and very high ($\sim$ 70% of the sensors fail).

Table I shows the percentage of the functioning sensors that found the best configuration for their event. Observe that even when around 20% of the sensors fail, almost all of the remaining functional ones are able to find and adopt the best configuration. However, as the number of sensors that fail increases the number of sensors that can establish the best configuration decreases. For instance, in the unrealistic case of 70% failure probability, very few of the remaining functional sensors can establish their needed configuration. This is to be expected since the number of failing sensors sending useless configurations is very high. Moreover, from the event 4 results we can once again observe that the size of the event (specifically the number of sensors in the event) can affect the CDS. This is reasonable since failing sensors means even less sensors searching for the solution. Nevertheless, overall the level of resilience shown by the CDS for this type of failure is very high.

Regarding how failure affects the convergence time, table II shows the number of communication cycles needed so that 90% of the sensors establish the best configuration. As expected, the higher the failure probability the more communication cycles that are needed. In particular, the sensors over the smallest event region (event 4) are the most affected.

Malfunctioning Sensing and Reception

Next we model a more extreme failure situation. Besides being incapable of receiving incoming communications, failing sensors also suffer from sensing errors. Moreover, for our experiments we specifically model the worst kind of sensing error. That is to say, that a failing sensor (for some reason or another) will always valuate its configuration as the best, regardless of its actual usefulness In other words, these sensors will constantly mislead their non-failing neighbors by broadcasting their configurations as if it were the best. The experiments were ran with a duration of 400 communication cycles and the following failure probabilities where employed: low ($\sim$ 10% of the sensors fail), medium ($\sim$ 20% of the sensors fail) and high ($\sim$ 30% of the sensors fail). Notice that these probabilities are lower that those in the previous section because this type of failure is considerably worst than the one employed there.

We observe from table III that the CDS has a good resilience against a failure probability of 10%. For almost all events more than 97% of the sensors reach their best configurations, and it is only event 4 (the event with the smallest region) the one that reaches just 90%. Furthermore, observe that sensors sensing event 4 are those that suffer the most when the failure rate increases. This is not surprising since its low number of sensors makes the non-failing sensors more prone to succumb to the misleading configurations broadcasted from the failing sensors. Nevertheless,

Failure probability	Event 3	Event 1	Event 2	Event 4
none	52	54	46	70
low	92	103	89	291
medium	266	299	215	0

Table IV
CONVERGENCE TIME TO THE BEST CONFIGURATION FOR DIFFERENT RECEPTION/MEASURING PROBABILITIES.

the resilience shown by the CDS for the other event regions is very good ($\sim$ 90 %), specially for this kind of failure.

As for the convergence time (see table IV) it is not surprising to observe that more communication cycles are needed for this type of failure. However, the number of cycles is still low enough to make the CDS practical (in particular when compared to an individual search IV-E).

Overall, the experiments show that the collective diffusion search mechanism has a considerable good resilience against sensor failures. These results are encouraging since failures are more prone to occur in uncharted environments.

V. CONCLUSIONS

In this paper we presented collective diffusion search (CDS) as a low overhead distributed algorithm that empowers sensors in a sensor network to collectively find the configurations needed to monitor the events occurring in an uncharted environment. Moreover, our empirical experiments showed that through this algorithm sensors not only are capable of configuring themselves, they can also reconfigure themselves in response to various levels of dynamic changes in the events. Furthermore, our results indicate that CDS is quite efficient energy-wise since the number of message transmissions required by the sensors' self-configuration is quite low. Additionally, the CDS has shown to be considerably resilient against sensors failures. These are all highly desirable properties for any sensor's algorithm.

Finally, even though CDS accomplished its purpose in all of our experimental scenarios, we observed that the number of sensors, and the dimensions and locations (frontiers) of the event affect the speed of the sensors' re-configurations.

ACKNOWLEDGMENTS

N. Salazar thanks CONACyT. Work partially funded by projects EVE (TIN2009-14702-C02-01), AT (CONSOLIDER CSD2007-0022), and the Generalitat of Catalunya grant 2009-SGR-1434.

REFERENCES

[1] N. Xu, *A survey of Sensor Network Applications*, http://courses.cs.tamu.edu/rabi/cpsc617/resources/sensorf.

[2] D. J. Nagel, *Wireless Sensor Systems and Networks: Technologies, Applications, Implications and Impacts*, http://intranet.daiict.ac.in/ ranjan/isn2005/papers/APP/wireless.pdf.

[3] M. Vinyals, J. A. Rodriguez-aguilar, and J. Cerquides, "A survey on sensor networks from a multi-agent perspective," *The Computer Journal*, 2010.

[4] J. Yin, D. H. Hu, and Q. Yang, "Spatio-temporal event detection using dynamic conditional random fields," in *In Proceedings of IJCAI'09*, 2009.

[5] R. M. Ruairí and M. T. Keane, "An energy-efficient, multi-agent sensor network for detecting diffuse events," in *In Proceedings of IJCAI'07*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2007, pp. 1390–1395.

[6] M. Sims, C. Goldman, and V. Lesser, "Self-Organization through Bottom-up Coalition Formation," in *Proceedings of AAMAS 2003*. Melbourne, AUS: ACM Press, July 2003, pp. 867–874. [Online]. Available: http://mas.cs.umass.edu/paper/238

[7] K. Martinez, J. K. Hart, and R. Ong, "Environmental sensor networks," *Computer*, vol. 37, pp. 50–56, 2004.

[8] Y. Chevaleyre, P. E. Dunne, U. Endriss, J. Lang, M. Lemaître, N. Maudet, J. A. Padget, S. Phelps, J. A. Rodríguez-Aguilar, and P. Sousa, "Issues in multiagent resource allocation," *Informatica (Slovenia)*, vol. 30, no. 1, pp. 3–31, 2006.

[9] J. Kho, A. Rogers, and N. R. Jennings, "Decentralized control of adaptive sampling in wireless sensor networks," *ACM Trans. Sen. Netw.*, vol. 5, no. 3, pp. 1–35, 2009.

[10] M. S. San Miguel, V. M. Eguiluz, R. Toral, and K. Klemm, "Binary and multivariate stochastic models of consensus formation," *Computing in Science and Eng.*, vol. 7, no. 6, pp. 67–73, 2005.

[11] T. Bäck, *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. Oxford University Press, 1996.

[12] Z. Michalewicz, *Genetic Algorithms+Data Structures=Evolution Programs*, 3rd ed. Springer Verlag, 1996.

[13] J. L. Bredin, E. D. Demaine, M. Hajiaghayi, and D. Rus, "Deploying sensor networks with guaranteed capacity and fault tolerance," in *MobiHoc '05: Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing*. New York, NY, USA: ACM, 2005, pp. 309–319.