Cooperative Training in Wireless Sensor and Actor Networks

Francesco Betti Sorbelli[1], Roberto Ciotti[1], Alfredo Navarra[1], Cristina M. Pinotti[1], and Vlady Ravelomanana[2]

[1] Department of Computer Science and Mathematics, University of Perugia, Italy
{navarra,pinotti}@dmi.unipg.it
[2] Laboratoire d'Informatique de Paris-Nord, University of Paris, France
vlad@lipn.univ-paris13.fr

Abstract. Exploiting features of high density wireless sensor networks represents a challenging issue. In this work, the training of a sensor network which consists of anonymous and asynchronous sensors, randomly and massively distributed in a circular area around a more powerful device, called actor, is considered. The aim is to partition the network area in concentric coronas and sectors, centered at the actor, and to bring each sensor autonomously to learn to which corona and sector belongs. The new protocol, called *Cooperative*, is the fastest training algorithm for asynchronous sensors, and it matches the running time of the fastest known training algorithm for synchronous sensors. Moreover, to be trained, each sensor stays awake only a constant number of time slots, independent of the network size, consuming very limited energy. The performances of the new protocol, measured as the number of trained sensors, the accuracy of the achieved localization, and the consumed energy, are also experimentally tested under different network density scenarios.

Keywords: wireless sensor network, training, localization, distributed algorithms.

1 Introduction

Miniaturized, low-cost, battery-operated nodes, which integrate sensing abilities, signal processing and wireless communication are well known as *sensors*. In this work, Wireless Sensor and Actor Networks (WSAN) are considered, which consist of massively and randomly deployed sensors plus few more powerful entities, called *actors*.

The random deployment results in sensors initially unaware of their spatial coordinates. Since the sensed data is of scarce utility unless related to the localization of the sensors that collect them, each actor organizes the sensors in its range of transmission (the so called *actor-zone*) in a dynamic virtual infrastructure which provides the sensors with a coarse-grained localization awareness. Specifically, the actor arranges its zone into equiangular sectors and equiwidth

N. Bartolini et al. (Eds.): QShine/AAA-IDEA 2009, LNICST 22, pp. 569–583, 2009.

concentric coronas centered at the actor itself, imposing a discretized polar coordinate system. In doing so, the actor-zone is subdivided into small regions, one for each corona-sector intersection.

The task that allows each sensor in the actor-zone to acquire its corona (sector, resp.) coordinate is known, in the literature, as the *corona (sector, resp.) training* process. The new protocol, called *cooperative*, is analytically studied under the assumption that the density of the random distributed network is sufficiently high to guarantee that each sensor is trained. The new protocol is the fastest training algorithm for asynchronous sensors, and it matches the running time of the fastest known training algorithm for synchronous sensors. Moreover, during the training, each sensor stays awake only a constant number of time slots, independent of the network size, saving thus energy.

The remainder of this paper is organized as follows. Section 2 defines the network model. Section 3 presents the *cooperative* training algorithm by specifying the actor and sensor behaviors. Assuming the network to be sufficiently dense to train all the sensors, Section 4 studies the algorithm performances measured in overall running time, drained energy per sensor, number of trained sensors and accuracy of the achieved localization. Section 5 experimentally validates the results in Section 4, and argues on the actual network density needed to train all the sensors. Finally, Section 6 offers concluding remarks and open problems.

2 The Network Model

In this section, network model and assumptions are described. At first, the virtual coordinate system to be established in the network is as follows:

1. *Coronas*: The actor-zone area is divided into k coronas $C_0, C_1, \ldots, C_{k-1}$ of fixed width $\rho > 0$, centered at the actor, determined by k concentric circles whose radii are $\rho, 2\rho, \cdots, k\rho$, respectively;
2. *Sectors*: The actor-zone area is divided into h equiangular sectors $S_0, S_1, \ldots, S_{h-1}$, originated at the actor, each having a width of $\frac{2\pi}{h}$ radians.

The actor is equipped with a long-range radio and an isotropic antenna and it is able to broadcast with variable-range R in order to reach all the sensors at distance at most $R \leq k\rho$.

The time is ruled into slots, with sensors and actor using in-phase and equally long slots. Nonetheless, since the *asynchronous* model is adopted, the sensors are not engaged in any explicit synchronization protocol and each sensor starts to count the time from when it wakes up for the first time. Thus, the same time slot corresponds to different local times for sensors which woke up at different times. The time slot when the training process starts is numbered 0 at the actor. From now on, the time slot numbering done at the actor is called *global* time, whereas that at each sensor is indicated as *local* time.

Each sensor is *anonymous*, that is, it has no individual unique ID and works *unattended*. A sensor is called of *type* x, with $x \in [0, k-1]$, if it wakes up for

the first time at the global time x. However, each sensor is only aware of its own local time, and it has no idea of the global time. Each sensor alternates between *awake* and *sleep* periods. The sensor awake-sleep cycle has a total length of L time slots, out of which each sensor is awake for d slots and in sleep mode for $L - d$ slots. The i-th, with $i \geq 1$, awake-sleep period of a sensor of type x starts and finishes at the global time slots $x + (i-1)L$ and $x + iL - 1$, respectively. In order to save energy, a sensor which is not required to be active in an awake-sleep cycle can skip it staying in sleep mode for L time slots.

The sensors are equipped with a small-range radio and an isotropic antenna, and during an awake period they can transmit or listen to either the actor or the sensor neighbors. A sensor can transmit in two modalities: with transmission range equal to $r = \rho$ for routing purposes or equal to $r < \rho/2$ for the cooperative training algorithm. If an awake sensor receives more than one message at the same time, we assume that it correctly receives the message only if all the transmissions refer to the same message. Otherwise, the sensor hears *noise*.

3 The Cooperative Corona Training Algorithm

In this section we present the cooperative training algorithm, which localizes each individual sensor in the actor-zone. From now on, we will assume the corona width $\rho = 1$ and the awake-sleep period $L = k$.

The cooperative training consists of three stages: the first stage is deterministic and it is the only one that involves the actor. Immediately after deployment, the actor starts to transmit. Let $|a|_k$ denote the *modulo* operation, that is the nonnegative remainder of the division of a by k. At time slot t, with $0 \leq t \leq k + d - 2$, it transmits the beacon $|k - 1 - t|_k$ at a power level sufficient to reach all the sensors up to corona $C_{|k-1-t|_k}$, but not those beyond $C_{|k-1-t|_k}$.

For sensors, the protocol has three stages. The first stage deterministically trains a certain percentage of sensors. In the other two stages, in contrast, the percentage of sensors trained strictly depends on the network density.

Each sensor has its own local time τ, which is set to 0 when the sensor wakes up for the first time, and a counter j of the awake-sleep cycles passed from the beginning of the training protocol.

The pseudo-code for the sensor protocol is given in Appendix Figure 6. The first stage lasts one awake-sleep cycle for each sensor. The sensors alternate an awake period of d time slots with a sleep period of $L - d = k - d$ time slots. At time slot t of the awake period, with $0 \leq t \leq d - 1$, each sensor listens to the actor and stores in $C[t]$ either the beacon received by the actor or the mark $\emptyset$ when no beacon is received. A sensor in corona γ which is awake while the actor transmits beacon b receives such a beacon if and only if $b \geq \gamma$. A sensor becomes trained by the actor, and hence it becomes a *seed*, when one of the two following *Training Conditions* is verified.

TC 1: A sensor residing in corona 0 receives beacon 0. In fact, only sensors inside corona 0 can receive such a beacon.

TC 2: A sensor residing in corona γ receives beacon γ but not beacon $\gamma-1$ when it knows that the actor is transmitting beacon $\gamma-1$.

Since the above training condition TC2 can only be verified if $d \geq 2$, from now on, we assume $2 \leq d < k$.

In the second stage of the corona training protocol, the sensors communicate among them in order to broadcast the corona identity from the seeds to the untrained sensors. In other words, the seeds boost the cooperative process. For each sensor, the second stage lasts for at most two awake-sleep cycles. The sensors of type $d-1 \leq x \leq k-1$ enter in the second stage as soon as their 2-nd awake-sleep cycle starts. The sensors of type $0 \leq x \leq d-2$ skip their second cycle and enter in the second stage within their 3-rd awake-sleep cycle. During the second stage, the seeds broadcast their corona identity for two awake periods if they have type $d-1 \leq x \leq |2d-3|_k$, or for one awake period, otherwise. The awake untrained sensors are listening until they become either *trained* or *white-flag*.

An untrained sensor that receives all concordant messages from its neighbors becomes trained and broadcasts for the remaining time slots of its awake period. Contrary, if an untrained sensor hears noise, that is, it receives more than one message from two or more neighbors transmitting different corona identities, it becomes a white-flag. It stops to listen and it waits the third stage to eventually acquire an approximation of its location.

The third stage of the sensor training protocol is also distributed and lasts for a single awake-sleep period for each sensor. Each trained sensor, which belongs to an even corona, transmits its corona identity, whereas all the awake white-flag sensors are listening. Since a white-flag is a sensor that in the second stage has received simultaneously two consecutive corona identities, it is surely covered by a sensor in an even corona. Such a sensor trains the white-flag during the 3-rd stage. Hence, at the end of the third phase, all the white-flag sensors are trained and they learn to belong to an even corona. Thus, the white-flags that belong to an odd corona acquire a localization that differs of at most ± 1 from the actual one. As a macroscopic effect, at the end of the third stage, the even coronas of the virtual infrastructure will expand over the odd coronas. It is worth noting that this approximation has little effects on the estimate of the distance from the sensors to the actor. Indeed, recalling that the sensors uses a transmission radius $r < 1/2$ during the training protocol and a transmission radius $r = 1$ to route messages from the sensors to the actor, a wrong sensor, which believes to be in corona γ but it is indeed in corona $\gamma \pm 1$, is at most at one extra hop from the actor.

Note that an untrained sensor that at the end of the second stage has heard nothing will not be involved in the third stage and it will remain untrained. Finally, sensors that acquire a localization that differs of more than ± 1 from the actual one are called *mistrained*.

However, as experimentally tested, if the network is sufficiently dense, very few sensors become mistrained or remain untrained.

4 Algorithm Properties

In order to analyze which sensors become seeds in the first stage in each corona, let us recall that, the sensors of type x, with $x \in [0, \ldots, k-1]$, start the first awake period at the global time slot x and stay awake up to time $x+d-1$, while the actor broadcasts beacons $|k-1-x|_k, |k-1-x-1|_k, \ldots, |k-1-x-d+1|_k$. Note that the sensors of type x receive the same beacons independent of the corona to which they belong, but they behave differently from one corona to another. In fact:

Lemma 1. *The seed in corona γ, $1 \leq \gamma \leq k-1$, are the sensors of type $x = |k-1-\gamma-w|_k$ with $w = [0, d-2]$, or equivalently:*

$$x \in \begin{cases} [|k-\gamma-d+1|_k, |k-1-\gamma|_k] & \text{if } |k-\gamma-d+1|_k \leq |k-1-\gamma|_k \\ [|k-\gamma-d+1|_k, k-1] \cup [0, |k-1-\gamma|_k] & \text{if } |k-\gamma-d+1|_k > |k-1-\gamma|_k \end{cases} \quad (1)$$

Similarly, the seeds in corona 0 are those with type $x = |k-1-w|_k$ with $w = [0, d-1]$, or:

$$x \in [|k-d|_k, |k-1|_k] \quad (2)$$

□

The second stage lasts $2k$ time slots, starting from the global time slot $k+d-1$. Recalling that a sensor of type x wakes up for the i-th awake period, with $i \geq 1$, at time slot $x+(i-1)L$, and that $L = k$, in the interval $t \in [k+d-1, 2k+d-2]$, all types of sensors enter in the second stage. In fact, at time t, the sensors of type $x = |t|_L = |t|_k$ wake up. Thus, during the interval $t \in [k+d-1, 2k-1]$ the sensors of type $x \in [d-1, k-1]$ wake up because they enter in the second stage in their 2-nd awake period, while during the period $t \in [2k, 2k+d-2]$ those of type $x \in [0, d-2]$ wake up because they enter in the second stage during their 3-rd awake period. Moreover:

Lemma 2. *In the interval $t \in [k+2d-2, 2k+2d-3]$, all the sensors of the $d-1$ types $|t-w|_k$, with $w = [0, d-2]$, are awake simultaneously.* □

While so far the results were independent of the network density, in what follows, the density plays an important role.

The cooperative process becomes operative in each corona when all the seeds are awake and broadcast. Thus, this happens for the first time, by Lemma 1, in corona $\gamma = |k-2d+2|_k$ at time slot $k+2d-3$. Since by the training condition TC2 all the seeds are awake simultaneously for two time slots, the seeds in corona $\gamma = |k-2d+2|_k$ are awake simultaneously and broadcast also at time slot $k+2d-2$. At that time, the sensors of the type $|2d-2|_k$, which are untrained in corona $|k-2d+2|_k$, wake up and, listening to their seed neighbors, they become trained. Then, the new trained sensors start to broadcast for the remaining $d-1$ time slots of their awake period, replacing the seed of type $|t-d+1|_k = d-1$ that go back to sleep. This is repeated for $k-d$ time slots up to time $2k+2d-3$, training all the type of sensors in corona $|k-2d+2|_k$.

Hence, during the cooperative process, an untrained sensor becomes trained only if it has in its neighborhood at least one trained sensor awake at the same time. This might happen or not depending on the network density. From now on, we assume that the network is sufficiently dense for the above condition to be verified. For the same reason, we consider the cooperative process to be operative starting from the time slot when all the $d-1$ seeds are simultaneously awake. We discuss the effects of density only in Section 5, where experiments with different densities are reported.

Theorem 1. *Assuming that the network is sufficiently dense, the cooperative training process becomes effective in corona* $\gamma = |k - 2d + 2 - y|_k$ *at time slot* $k + 2d - 2 + y$ *and, in such a corona, a new type of sensors is trained in each subsequent time slot* $2k + d - 2 + y$*, with* $0 \leq y \leq k - 1$*.* □

Observe that the last corona to be trained is corona $|k - 2d + 3|_k$ where the process lasts from time $2k + 2d - 3$ up to $3k + d - 3$. Moreover, note that at time slot $3k + d - 3$ the sensors of type $d - 2$, which entered as last in the second stage, have just completed their third sleep-awake cycle.

So far, it has been assumed that during the second stage each untrained sensor receives concordant and correct corona identities. Nonetheless, since all the sensors of the same type are always awake simultaneously independent of the corona to which they belong, but their status (i.e., seed, untrained, trained) depend on their corona, it may happen that the sensors in the corona borders listen to sensors of the same type but in different status. Consider, for example, the sensors of type $\overline{x} = |k - \gamma - d + 1|_k$, with $0 \leq \gamma \leq k - 1$ during the second stage. When such sensors wake up, they start to broadcast in corona γ where they are seed, whereas they listen in corona $\gamma - 1$ where they are untrained. A sensor of type $\overline{x}$ on the border of corona $\gamma - 1$ can receive only the corona identity γ and thus it acquires a wrong localization. In this case, however, the correct localization may still be derived exploiting the fact that the sensor in corona $\gamma - 1$ has received beacon $\gamma - 1$ in the first stage.

Unfortunately, this is not always the case that the right corona information can be retrieved. There are cases when the sensors cannot acquire the exact localization or no localization can be derived during the second stage of the training protocol. For example, consider an untrained sensor in the corona border that hears two trained sensors of the same type belonging to two different coronas. Such a sensor can only hear noise and it cannot be localized. It will be a white-flag sensor, and it has to wait the third stage to be trained.

Theorem 2. *At the end of the third stage, all the white-flag sensors in the even coronas are turned into trained sensors, while those on the odd coronas become* ± 1*-trained.* □

Finally, in order to evaluate the power consumption per sensor during the cooperative algorithm, observe that, when a sensor is awake, its micro-controller is active and its radio is listening, receiving, or transmitting. Contrary, when a sensor is sleeping, its micro-controller is not active, its timer is on, and its

radio is off. Let p_{awake}, p_{TX}, and p_{sleep} be the power consumption by a sensor when it is listening/receiving, transmitting, or sleeping, respectively. Since the radio startup and shutdown require a not negligible overhead, let p_{trans} denote the power consumption for a sleep/wake transition followed by a wake/sleep transition.

Observing that a sensor wakes up at most for 4 times, that it transmits at most for 3 awake periods if it is a seed, and that the entire algorithm lasts 4 awake periods for each sensor, the maximum power consumed $p_{\max}$ per sensor can be upper bounded as:

$$p_{\max} < 4p_{\text{trans}} + dp_{\text{awake}} + 3dp_{\text{TX}} + 4(k-d)p_{\text{sleep}} \tag{3}$$

In conclusion, recalling that $2 \leq d \leq L = k$, one has:

Theorem 3. *The cooperative training process terminates in $O(k)$ time slots. During the training, each sensor is awake for $O(d)$ time slots and consumes at most $4p_{\text{trans}} + dp_{\text{awake}} + 3dp_{\text{TX}} + 4(k-d)p_{\text{sleep}}$ power.* □

Comparing the above results with the literature, the new cooperative training process is as fast as the best synchronized training algorithm and it is the fastest asynchronous training algorithm [2,4]. Moreover, in this protocol, since d is independent of k, each sensor can stay awake for a very short interval of time, almost constant. Instead, each sensor is awake for $O(\log k)$ and up to $O(k)$ time slots in the synchronous and asynchronous protocols, respectively. However, one cannot forget that this new protocol is probabilistic (i.e., we are not sure that all the sensors will be trained), while the previous algorithms are deterministic (i.e., all the sensors are trained).

5 Experimental Tests

In this section, the performances of the cooperative training algorithm, shortly denoted with *Coop*, are experimentally evaluated when the network density varies with respect to the accuracy of the localization, and the power consumption per sensor. In the simulation, each corona has a unit width and N sensors are uniformly distributed within a circle of radius k, centered at the actor. Moreover, each sensor generates its type x, as an integer uniformly distributed in the range $[0, k-1]$.

By varying the total number of sensors N, the number of coronas k, and the sensor radius r, we consider three different settings for our simulations. For each setting, let $\mathbb{E}(N_x) = \mathcal{O}(\frac{N}{k})$ be the expected number of sensors of the same type $x \in [0, k-1]$ and $\delta = \mathcal{O}(\frac{N}{\pi k^2})$ be the network density, that is the expected number of sensors that belong to a unit area of the actor zone.

Moreover, we consider the constant $q = \frac{\frac{N}{k}}{\log\left(\frac{N}{k}\right)} \frac{r^2}{k^2}$ which is approximately the ratio between the number of the sensors of the same type x in a circle of radius r and the logarithm of the overall number of the sensors of the same type $log(\frac{N}{k})$. Roughly speaking, q is a measure of the connectivity of the network. Very

Table 1. Experiment settings

	N	k	r	$\mathbb{E}(N_x)$	δ	q
S_1	310000	8	$\frac{1}{5}$	38750	1541.8	2.29
S_2	700000	12	$\frac{1}{7}$	58333	1547.3	0.75
S_3	819200	32	$\frac{1}{4}$	25600	254.65	0.15

Table 2. Estimate of sensor power consumption

Sensor Mode	**Power consumption**
μC sleep with timer on	60 μW
μC switch on, radio startup	30 mW
μC switch off, radio shutdown	30 mW
μC active, radio idle listening	60 mW
μC active, radio TX	80 mW

informally, if $q > 1$, it means that in a circle of radius r there are $\log(\frac{N}{k})$ sensors of the same type, and hence the sensors of such a type have a minimum degree of about $\log(\frac{N}{k})$ and therefore the network of such nodes is $\log(\frac{N}{k})$-connected [8].

Table 1 reports, the parameters N, k, r, as well as $\mathbb{E}(N_x)$, δ, and q for the three settings S_1, S_2 and S_3 used in the experiments.

In the settings S_1, S_2 and S_3, assuming the corona width $\rho = 100$ meters, there are 0.15, 0.15, and 0.025 sensors per square meter, respectively. At the present state of the technology, small sensors, supporting communications in a range varying from 10 to 100 meters, like TinyNode 584 produced by Shockfish S.A. or T-node developed by SOWNet Technologies can be used for built such massive networks [5,9].

Moreover, in order to evaluate the power consumption per sensor during the Coop algorithm, Table 2 reports the power consumption, measured in the field, of a T-node in different operational modes [9] to have a realistic setting. The data refer to the power consumed operating using 10 dBm transmission power, and hence attaining a transmission range around twenty meters, at a low bandwidth of 75 Kbit/s. Note that such a bandwidth is sufficient because the sensors have to transmit just their corona identity, which consists of $O(\log k)$ bits.

The Coop algorithm has been tested on each setting fixing $L = k$ and varying the sensor awake period d between 2 and 10. In fact, as proved by Lemma 1 and Theorem 1, the awake period d influences the number of seeds in the first stage as well as the number of trained sensors that are awake and broadcast in each time slot of the second stage.

In order to evaluate the quality of the localization, observe that, at the end of the Coop algorithm, a sensor can be in one of the following *status*:

- `trained`, if it has learnt the actual corona to which it belongs
- ± 1-`trained`, if it has learnt to belong to a corona which differs of ± 1 from the correct one

- `mistrained`, if it has acquired a corona which arbitrarily differs from the correct one
- `white-flag`, if it cannot decide to which corona it belongs although it is in the neighborhood of trained sensors
- `untrained` if it does not belong to the neighborhood of any trained sensor

In our experiments, statistics are taken on how many sensors are in each status at the end of the Coop algorithm. Specifically, Figures 1 and 2 report the results of the experiments on settings S_1 and S_2 along with S_3, respectively, when the awake period d varies, with $d \geq 2$. The results are averaged over 3 independent experiments, which differ in the deployment distribution of the sensors and in the sensor type generation.

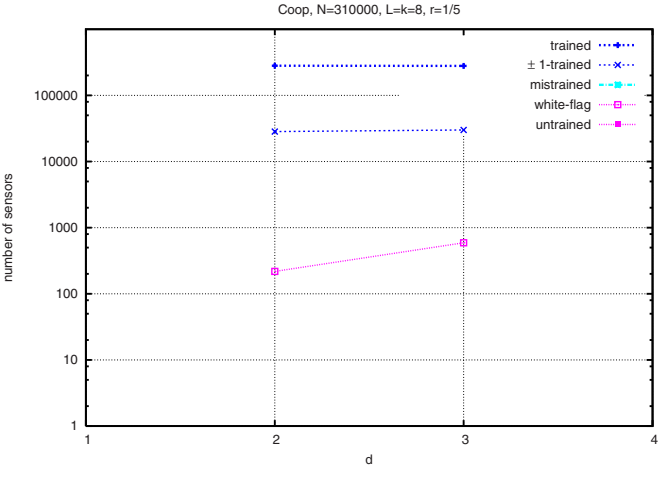

Fig. 1. Sensor statistics at the end of the Coop algorithm on setting S_1

At first, it is worth noting that there are no untrained sensors for all the experiments on the settings S_1 and for the experiments on S_2 with $d \geq 3$. Moreover, on S_3, the number of untrained sensors rapidly decreases when d increases. This confirms the first setting has enough sensors to satisfy the density assumed in Theorem 1. Settings S_2 and S_3 are not sufficiently dense when $d = 2$. Increasing d, however, the number of untrained sensors decreases up to 0. We can say that the density assumed in Theorem 1 is achieved for S_2 and S_3 when $d = 3$ and $d = 9$, respectively. Not surprisingly, since S_3 has a value of q smaller than the one of S_2, a greater value of d is needed. Indeed, in all the experiments when $(d-1)q > 1$, at the end of the Coop algorithm, more than 98% of the sensors acquire a satisfying localization (i.e. trained or $\pm$1-trained), and the 89% are correctly trained. In Table 3, for a detailed analysis of the trained sensors, the number of seeds, sensor trained, $\pm$1-trained, and white-flag sensors are reported

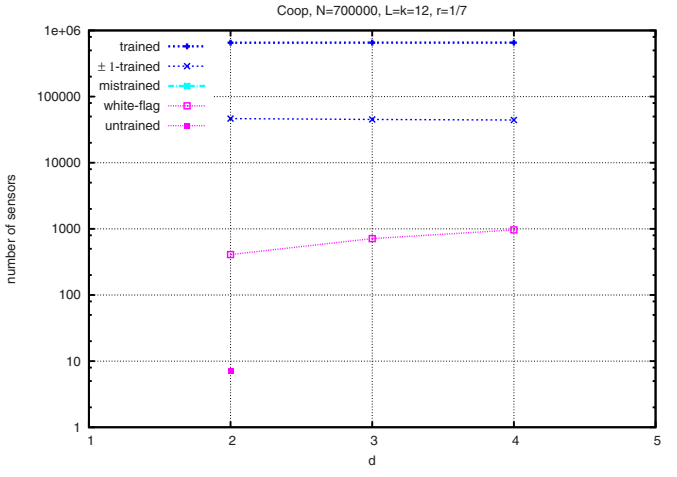

Fig. 2. Sensor statistics at the end of the Coop algorithm on setting S_2

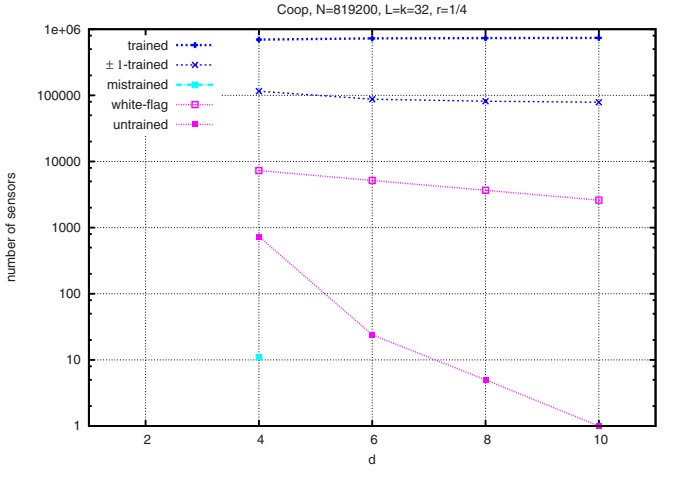

Fig. 3. Sensor statistics at the end of the Coop algorithm on setting S_3

at the end of the second stage (Coop2) of the training algorithm as well as at the end of the third stage (Coop). Moreover, half of the sensors which are white-flag at the end of the second stage become trained at the end of the third stage and half become ± 1-trained.

The white-flag and ± 1-trained sensors are placed at the borders of the coronas. When d increases, since more sensors are awake simultaneously the number of white-flag sensors increases, while that of the ± 1-trained sensors decreases.

Table 3. The acquired localization in S_2 and S_3 after Coop2 and Coop

		S_2, $d=2$	S_2, $d=3$	S_2, $d=4$	S_3, $d=4$	S_3, $d=6$	S_3, $d=8$	S_3, $d=10$
Coop2	trained	624787	620148	618647	639778	662189	666610	669426
	seed	58731	117131	175377	76934	127828	179162	230158
	±1-trained	15610	8176	5677	62218	24771	15213	10337
	white-flag	59596	71676	75676	116470	132216	137372	139436
Coop	trained	653052	654238	654735	695667	726559	733865	737996
	seed	58731	117131	175377	76934	127828	179162	230158
	±1-trained	46532	45050	44302	115506	87460	81664	78598
	white-flag	409	712	963	7293	5157	3664	2605

Table 4. Acquired localization in A-Seed and Coop2

	$S_1, d=2$		$S_2, d=2$	
	A-Seed	Coop2	A-Seed	Coop2
trained	260535	263877	623201	624787
±1-trained	823	6657	5751	15610
white-flag	48642	39466	71036	59596
untrained	0	0	12	7

It is worthy to note that we do not report the mistrained sensors because they are zero in all experiments but one. Thus, as expected, the network density guarantees that the corona identity propagation is confined in each corona.

About the ±1-trained sensors, their number also depends on the fact that the cooperative process does not start simultaneously in all the coronas because different coronas have different seeds. In fact, sensors at the border that wake up earlier than the seeds on their own corona might be ±1-trained. This behavior has been tested in Table 4, where a new algorithm, called *A-Seed*, is introduced. The A-Seed algorithm performs only the second stage of the cooperative training algorithm and assumes that the seeds are the sensors of a given type $\overline{x} \in [0, k-1]$, selected initially at random. Clearly, during the A-Seed algorithm, the cooperative process becomes effective at the same time slot in all the coronas and the number of white-flag sensors increases at the expenses of that of the ±1 trained.

With respect to the power consumption, substituting the values in Table 2 in Equation 3, the maximum power consumed $p_{\max}$ per sensor can be upper bounded as:

$$p_{\max} < 4 * 2 * 30 + d * 60 + 3 * d * 80 + 4 * (k - d) * 0.060mW \qquad (4)$$

The results of the experiments on settings S_2 and S_3 are reported in Figures 4.

The behavior of the Coop algorithm is compared with that of the asynchronous training protocol Flat [2]. The Flat and Coop algorithms assume the same parameter values, except that Flat uses an awake-sleep cycle of length $L = k + 1$ instead of k. Indeed, when $L = k$, Flat cannot complete the training process [2], and thus the smallest value of L for which Flat trains all the sensors has been used.

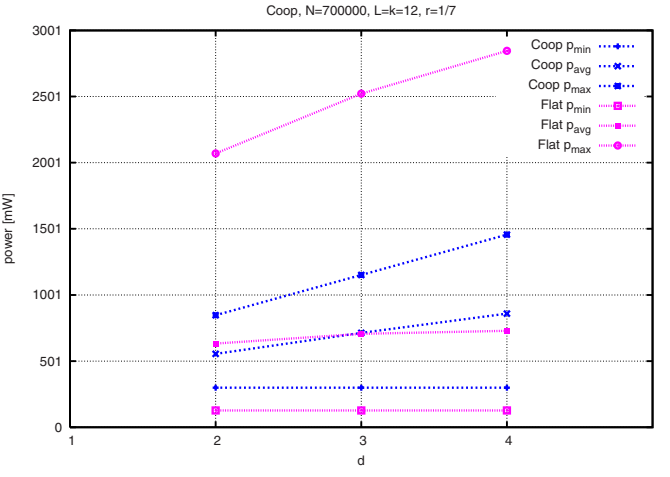

Fig. 4. Power consumption per sensor during the Coop algorithm on setting S_2

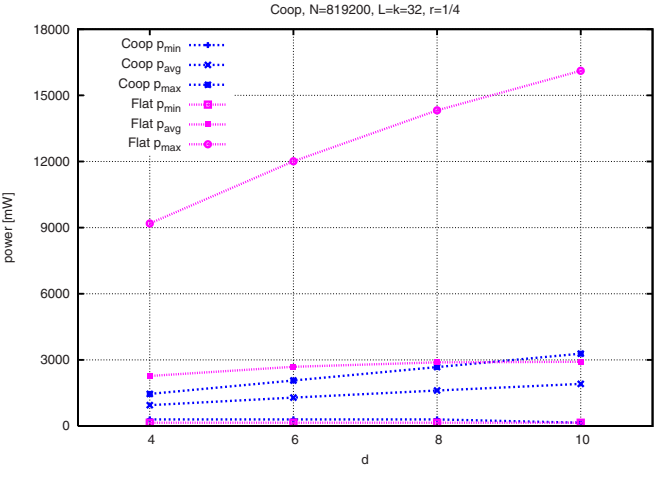

Fig. 5. Power consumption per sensor during the Coop algorithm on setting S_3

For each algorithm, it has been measured the maximum $p_{\max}$ (minimum $p_{\min}$, resp.) power consumed by each sensor along with the average p_{avg} power.

One can note that although Coop and Flat consume overall almost the same power as shown in Figure 4, the difference between the sensor maximum and minimum power consumption in the Coop algorithm is much less than that measured for Flat. In other words, the Coop training algorithm drain the sensors in a balanced way, and therefore it works in favor of the network lifespan.

When k increases, like in Figure 4 scenario S_3, the power effectiveness of the Coop algorithm is neat. The power $p_{\max}$ of Coop is smaller than or equal to the p_{avg} of the Flat Algorithm for any value of d.

6 Conclusion

In the context of anonymous, asynchronous and randomly distributed sensor and actor networks, we have proposed a new cooperative training algorithm which exploits the high density features of the considered kind of network. After describing the phases of the algorithm, we have provided analytical and experimental results with respect to the accuracy for the localization and the consumed energy. The new training algorithm is particularly suitable in large and dynamic networks, that need to frequently and quick raise up the network, for instance in presence of an actor moving to track an intruder. Moreover, once the proposed course-grain localization has been performed, easy routing algorithms can be applied with respect to the obtained virtual infrastructure induced by our algorithm.

As an open problem, it remains to study analytically the minimum sensor network density which guarantees that the algorithm trains, with high probability, all the sensors. Moreover, as future works, one can investigate larger networks where the actors move. Even more challenging would be the comparisons of Flat and Coop in a test-bed wireless sensor network.

References

1. Akyildiz, I.F., Kasimoglu, I.: Wireless sensor and actor networks: research challenges. Ad Hoc Networks 2, 351–367 (2004)
2. Barsi, F., Bertossi, A.A., Betti Sorbelli, F., Ciotti, R., Olariu, S., Pinotti, M.C.: Asynchronous Corona Training Protocols in Wireless Sensor and Actor Networks. IEEE Transactions on Parallel and Distributed Systems (to appear)
3. Baryshnikov, Y.: Connectivity in Geometric Graphs: Beyond the Standard Model. Private Communications
4. Bertossi, A.A., Olariu, S., Pinotti, M.C.: Efficient corona training protocols for sensor networks. Theoretical Computer Science 402(1), 2–15 (2008)
5. Burri, N., von Rickenbach, P., Wattenhofer, R.: Dozer: Ultra-low power data gathering in sensor networks. In: Proc. IPSN 2007, Cambridge, MA (April 2007)
6. Gautschi, W.: The Incomplete Gamma Functions Since Tricomi. In Tricomi's Ideas and Contemporary Applied Mathematics. Atti dei Convegni Lincei, Accademia Nazionale dei Lincei, Roma 147, 203–237 (1998)
7. Olariu, S., Waada, A., Wilson, L., Eltoweissy, M.: Wireless sensor networks leveraging the virtual infrastructure. IEEE Network 18(4), 51–56 (2004)
8. Penrose, M.D.: Random Geometric Graphs. Oxford Studies in Probability (2003)
9. The Sensor Network Museum Project, `http://www.snm.ethz.ch/Main/HomePage`
10. Temme, N.: Uniform asymptotic expansions of the incomplete gamma functions and the incomplete beta functions. Math. Comput. 29, 1109–1114 (1975)
11. Temme, N.: The asymptotic expansion of the incomplete gamma function. SIAM J. Math. Anal. 10, 757–766 (1979)

12. Waada, A., Olariu, S., Wilson, L., Eltoweissy, M., Jones, K.: Training a wireless sensor network. Mobile Networks and Applications 10(1), 151–168 (2005)
13. Xu, Q., Ishak, R., Olariu, S., Salleh, S.: On asynchronous training in sensor networks. In: Lumpur, K. (ed.) Proc. 3rd Intl. Conf. on Advances in Mobile Multimedia (September 2005)
14. Xue, F., Kumar, P.R.: The number of neighbors needed for connectivity of wireless networks. Wireless Networks 10, 169–181 (2004)

Appendix

```
Procedure Sensor
     Input: x, d, L, k, r, j;
1.    case j :
        j = 1 :
2.          for  t := 0 to d − 1                                          {Initialize}
3.          C[t] := −1;
4.      trained := white-flag := seed := false;
5.      corona := −∞; τ := −1;
6.      for t := 0 to d − 1                                               {First stage}
7.              C[t] := listen-actor(γ);
8.              if C[t] = 0
9.                   then trained:= seed := true; corona:= 0;
10.                  else if (t ≥ 1 and C[t] = ∅ and C[t − 1] = γ)
11.                            then trained:= seed:=true, corona:= C[t − 1];
12.             τ := τ + 1;
13.     if x ≥ d − 2
14.             then j := 2; set-alarm-clock(τ + L − d);
15.             else j := 3; set-alarm-clock(τ + 2L − d);
        j = 2, 3 :
16.     for i := j to 3                                                   {Second stage}
17.         for t := 0 to d − 1
18.             if trained
19.               then broadcast(corona)
20.               else if ¬ white-flag
21.                         then listen-sensor(corona);
22.                              if corona ≠ ∅ then corona := compatible(corona); trained := true;
23.                              if corona =noise then white-flag:=true;
24.             τ := τ + 1;
25.         if (white-flag or (trained and ¬ seed) or (seed and x ∉ [d − 1, |2d − 3|_k]))
26.                then j := 4; set-alarm-clock (τ + (3 − i)L + L − d);
27.                else j := j + 1; set-alarm-clock (τ + L − d);
        j = 4 :
28.     for t := 0 to d − 1                                               {Third stage}
29.       if trained and |corona|_2 = 0
30.             then broadcast(corona)
31.             else if white-flag
32.                       then listen-sensor(corona);
33.                            if corona ≠ ∅ then trained:=true;
34.       τ := τ + 1;
35.     set-alarm-clock(τ + L − d);
```

Fig. 6. The corona training protocol for the sensor