Sensors **2008**, *8*, 4821-4850; DOI: 10.3390/s8084821

OPEN ACCESS

sensors

ISSN 1424-8220

www.mdpi.org/sensors

Article

Distributed Principal Component Analysis for Wireless Sensor Networks

Yann-Aël Le Borgne [1,⋆], Sylvain Raybaud[2], and Gianluca Bontempi[1]

[1] Machine Learning Group, Département d'Informatique, Faculté des Sciences, Université Libre de Bruxelles, Boulevard du Triomphe, 1050 Brussels, Belgium

[2] École Normale Supérieure de Cachan, 61, Avenue du Président Wilson, 94235 Cachan Cedex, France

E-mails: yleborgn@ulb.ac.be; sraybaud@dptmaths.ens-cachan.fr; gbonte@ulb.ac.be

⋆ Author to whom correspondence should be addressed.

Received: 27 May 2008; in revised form: 29 July 2008 / Accepted: 4 August 2008 / Published: 11 August 2008

Abstract: The Principal Component Analysis (PCA) is a data dimensionality reduction technique well-suited for processing data from sensor networks. It can be applied to tasks like compression, event detection, and event recognition. This technique is based on a linear transform where the sensor measurements are projected on a set of *principal components*. When sensor measurements are correlated, a small set of principal components can explain most of the measurements variability. This allows to significantly decrease the amount of radio communication and of energy consumption. In this paper, we show that the power iteration method can be distributed in a sensor network in order to compute an approximation of the principal components. The proposed implementation relies on an aggregation service, which has recently been shown to provide a suitable framework for distributing the computation of a linear transform within a sensor network. We also extend this previous work by providing a detailed analysis of the computational, memory, and communication costs involved. A compression experiment involving real data validates the algorithm and illustrates the tradeoffs between accuracy and communication costs.

Keywords: Wireless sensor networks, distributed principal component analysis, in-network aggregation, power iteration method.

---

1 Introduction

Efficient in-network data processing is a key factor for enabling wireless sensor networks (WSN) to extract useful information and an increasing amount of research has been devoted to the development of data processing techniques [1–4]. Wireless sensors have limited resource constraints in terms of energy, network data throughput and computational power. In particular, the radio communication is an energy consuming task and is identified in many deployments as the primary factor of sensor node's battery exhaustion [5]. Emitting or receiving a packet is indeed orders of magnitude more energy consuming than elementary computational operations. The reduction of the amount of data transmissions has therefore been recognized as a central issue in the design of wireless sensor networks data gathering schemes [6]. Data compression is often acceptable in real settings since raw data collected by sensors typically contain a high degree of spatio-temporal redundancies [5, 7–9]. In fact, most applications only require approximated or high-level information, such as the average temperature in a room, the humidity levels in a field with a $\pm 10\%$ accuracy, or the detection and position of a fire in a forest.

An attractive framework for processing data within a sensor network is provided by the data aggregation services such as those developed at UC Berkeley (TinyDB and TAG projects) [10, 11], Cornell University (Cougar) [12], or EPFL (Dozer)[13]. These services aim at aggregating data within a network in a time- and energy-efficient manner. They are suitable when the network is connected to a base station from which queries on sensor measurements are issued. In TAG or TinyDB, for example, queries are entered by means of an SQL-like syntax which tasks the network to send raw data or aggregates at regular time intervals. These services make possible to compute "within the network" common operators like *average*, *min*, *max*, or *count*, thereby greatly decreasing the amount of data to be transmitted. Services typically rely on synchronized routing trees along which data is processed and aggregated along the way from the leaves to the root [10, 11].

Recently, we have shown that a data aggregation service can be used to represent sensor measurements in a different space [14]. We suggested that the space defined by the principal component basis, which makes data samples uncorrelated, is of particular interest for sensor networks. This basis is returned by the Principal Component Analysis (PCA) [15], a well-known technique in multivariate data analysis. The design of an aggregation scheme which distributes the computation of the principal component scores (i.e., the transformed data in the PCA space) has three major benefits. First, the PCA provides varying levels of compression accuracies, ranging from constant approximations to full recovery of original data. Second, simple adaptive protocols can leverage this flexibility by trading network resources for data accuracy. Third, principal component scores contain sufficient information for a variety of WSN applications like approximate monitoring [16], feature prediction [17, 18] and event detection [19, 20].

The approach we proposed in [14] exclusively addresses the distribution of the computation of the principal component scores and requires the component basis to be computed beforehand in a centralized manner. This limits the applicability of the PCA to small networks, as the centralized computation of the principal component basis does not scale with the network size.

The main contribution of this article is to provide a distributed implementation of the principal com-

ponent basis computation. The algorithm is based on the Power Iteration Method [21, 22], an iterative technique for computing the principal component basis, which can be implemented in a distributed manner by means of an aggregation service. In particular we show that this algorithm can properly compute the principal component basis under the mild hypothesis that the sensor measurements collected by distant sensors are not significantly correlated. We also extend the previous work of [14] by an in-depth discussion of the network tradeoffs and scalability issues.

The article is structured as follows. Section 2 reviews previous work in the domain by describing the principles of data aggregation services in WSN and detailing how principal component scores can be computed by an aggregation service. This section also provides a brief overview of PCA together with potential WSN applications and related tradeoffs. Section 3 presents the main issues in the distribution of the PC basis and proposes an implementation based on a data aggregation service. Experimental results illustrating the tradeoffs between accuracy and communication costs are discussed in Section 4. Additional related work and future research tracks are addressed in Section 5.

2 Principal component aggregation in wireless sensor networks

Let us consider a static network of size p whose task is to collect at regular time instants sensor measurements or aggregates thereof, and to route them to a destination node. This scheme is standard for plenty of real-time WSN applications like surveillance, actuator control or event detection. The destination node is commonly referred to as the *sink* or the *base station* and is often assumed to benefit from higher resources (e.g., a desktop computer). Let $t \in \mathbb{N}$ refer to a discretized time domain and $x_i[t]$ be the measurement collected by sensor i, $1 \leq i \leq p$, at time t. At each time instant t, the p resulting measurements can be seen as components of a vector $\mathbf{x}[t] \in \mathbb{R}^p$. The sampling period is also referred to as *epoch*.

Since the communication range of a single node is limited, sensors which are not in communication range of the sink have their measurements relayed by intermediate sensors. A standard approach consists in setting up a multi-hop network by means of a routing tree whose root is connected to the sink (Figure 1). Different metrics such as hop count, latency, energy consumption and network load may be taken into account during the design of the routing tree [23].

2.1 Data aggregation

2.1.1 Aggregation service

An aggregation service allows to aggregate data within a network in a time- and energy-efficient manner. A well-known example of aggregation service is the TAG system, developed at the University of Berkeley, California [10, 11]. TAG stands for Tiny AGgregation and is an aggregation service for sensor networks which has been implemented in TinyOS, an operating system with a low memory footprint specifically designed for wireless sensors [24]. In TAG, an epoch is divided into time slots so that sensors' activities are synchronized according to their depth in the routing tree. Any algorithm can be

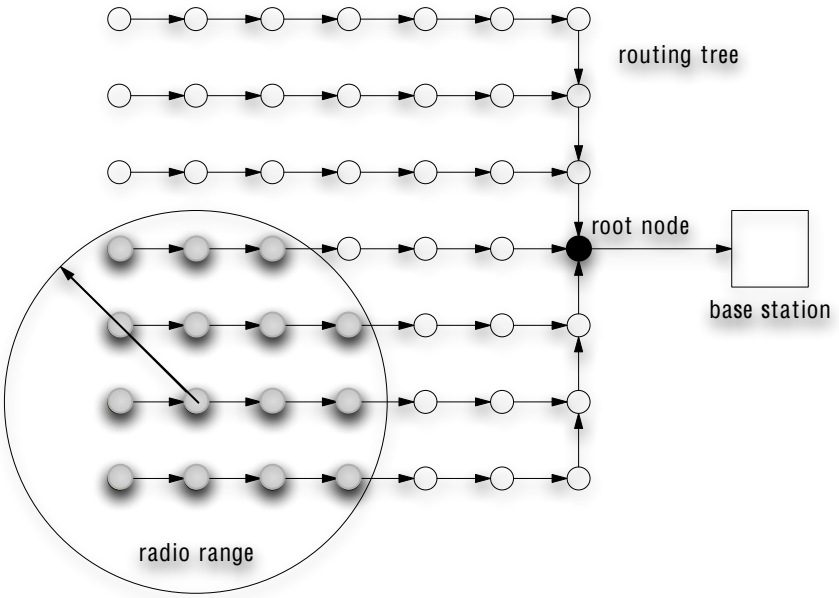

Figure 1. A multi-hop network where all the sensor nodes are connected to the base station by means of a routing tree. Note that as the radio range usually allows a node to communicate with several others, different routing trees can be obtained. The black node represents the root node.

relied on to create the routing tree, as long as it allows data to flow in both directions of the tree and does not send duplicates [10].

The TAG service focuses on low-rate data collection tasks which permits loose synchronization of the sensor nodes. The overhead implied by the synchronization is therefore assumed to be low. The goal of synchronization is to minimize the amount of time spent by sensors in powering their different components and to maximize the time spent in the idle mode, in which all electronic components are off except the clock. Since the energy consumption is several orders of magnitude lower in the idle mode than when the CPU or the radio is active, synchronization significantly extends the wireless sensors' lifetime. An illustration of the sensors' activities during an epoch is given in Figure 2 for a network of four nodes with a routing tree of depth three. Note that the synchronization is maintained at the transport layer of the network stack, and does not require precise synchronization constraints as in TDMA. Rather, nodes are synchronized by timing information included in data packets, and a CSMA-like MAC protocol with a random backoff scheme is used at the link layer to resolve multiple access collisions.

2.1.2 Aggregation primitives

Once the routing tree is set up and the nodes synchronized, data can be aggregated from the leaves to the root. Each node adds its contribution to a *partial state record* X which is propagated along the routing tree. Partial state records are merged when two (or more) of them arrive at the same node. When the partial state record is eventually delivered by the root node to the base station, the desired result is

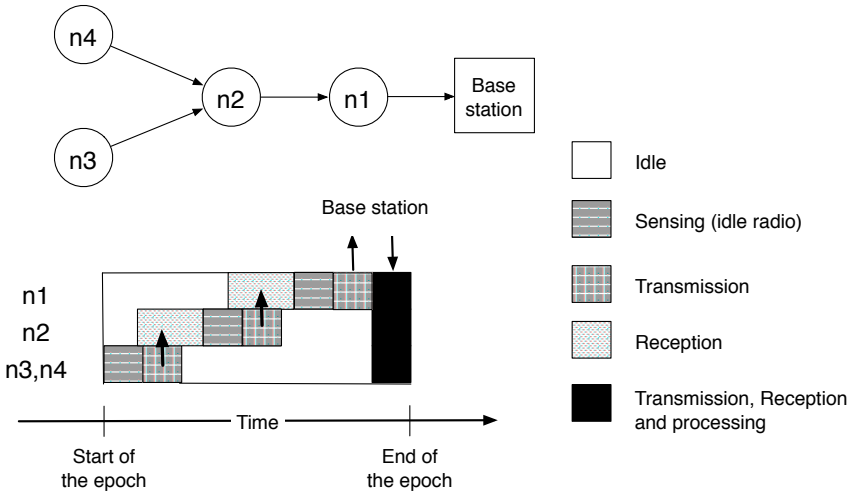

Figure 2. Activities carried out by sensors in the routing tree *(adapted from [13])*. Transmissions are synchronized for optimizing energy savings. The last stage involves all sensors and allows unsynchronized operations (for sensor discovery e.g.).

returned by means of an evaluator function. An aggregation service requires then the definition of three primitives [10, 11]:

- an initializer $init$ which transforms a sensor measurement into a partial state record,
- an aggregation operator f which merges partial state records, and
- an evaluator e which returns, on the basis of the root partial state record, the result required by the application.

Note that when partial state records are scalars or vectors, the three operators defined above may be seen as functions. Partial state records may however be any data structure which, following the notations of [10], are represented using the symbols $\langle . \rangle$.

We illustrate the aggregation principle by the following example. Suppose that we are interested in computing the Euclidean norm of the vector containing the WSN measurements at a given epoch. Rather than by directly sending all the p measurements to the base station for computation, an aggregation service (Figure 3) can obtain the same result in an online manner once the following primitives are implemented :

$$\begin{aligned} init(x) &= \langle x^2 \rangle \\ f(\langle X \rangle, \langle Y \rangle) &= \langle X + Y \rangle \\ e(\langle X \rangle) &= \sqrt{X} \end{aligned}$$

In this example the partial state records X and Y are scalars of the form

$$X = \sum_{i \in I} x_i^2$$

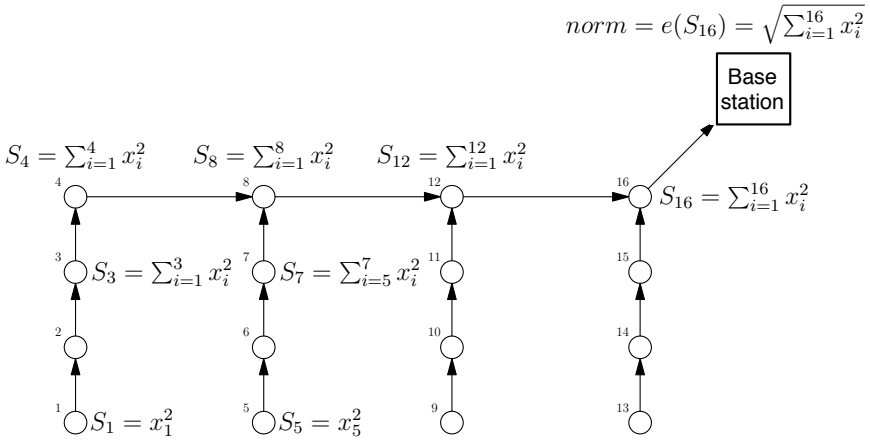

Figure 3. Aggregation service at work for computing the norm of the vector measured.

where $I \subset \{1, \ldots, p\}$.

It is worth noting that in-network aggregation does not only reduce the amount of packets transmitted but also increases the network lifetime. The energy consumption of the processing unit of a sensor node is at least one order of magnitude lower than the one of the wireless radio [6, 25, 26]. According to the analysis [27] of the main wireless sensor platforms available today on the market, sending one bit of data is equivalent to 2000 CPU cycles in terms of energy. For a packet length of 30 bytes (average TinyOS packet length), this corresponds to 480000 CPU cycles. It follows that, with respect to transmission costs, the energy consumption of simple aggregation operators is negligible. There is therefore a large variety of in-network processing algorithms that can lead to energy savings by reducing the number of packet transmissions.

2.1.3 Communication costs

This section details the communication costs caused by three types of network operation. The first is the *default* data collection operation in which all measurement are routed to the sink without any aggregation. This is referred to as the D operation. The second is the *aggregation* operation, referred to as A operation, which consists in tasking the network to retrieve an aggregate by means of the aggregation service. Finally, we denote by F the *feedback* operation which consists in flooding the aggregate obtained at the sink back to the whole set of sensors.

Let q be the size of a partial state record, C_i be the number of direct children of node i in the routing tree, RT_i be the size of the subtree whose root is the i-th node and $i_C^* = \arg\max_i C_i$ the node whose number of children is the highest.

The following analysis compares the orders of magnitude of the communication costs caused by the D, A and F operations, respectively. For this reason we consider the number of packets processed by each node in an ideal case where overhearing, collisions or retransmissions are ignored.

D operation Without aggregation, all the measurements are routed to the base station by means of the routing tree. As mentioned before, the network load at the sensor nodes, i.e., the sum of received and transmitted packets, is ill-balanced. The load is the lowest at by leaf nodes, which only send one packet per epoch, while the load is the highest at the root node which processes $2p-1$ packets ($p-1$ receptions and p transmissions) per epoch. The load at a generic i-th sensor node depends on the routing tree, and amounts to $2RT_i - 1$ packets per epoch.

A operation During the aggregation, the i-th node sends q packets and receives a number of packets which depends on its number C_i of children. The total number of packets processed is therefore $q(C_i+1)$ per epoch. The load is the lowest at leaf nodes, which only have q packets to send, while the load is the highest at the node whose number of children is the highest.

F operation The feedback operation consists in propagating the aggregated value back from the root down to the all leaves of the tree. This operation can be used, for instance, to get all sensor nodes acquainted with the overall norm of their measurements or with the approximation evaluated at the sink. The feedback of a packet containing the result of the evaluation generates a network load of two packets for all non-leaf nodes (one reception and one transmission for forwarding the packet to the children) and of one packet for the leaves (one reception only) .

2.2 Principal component analysis

This section describes the Principal Component Analysis (PCA), a well-known dimensionality reduction technique in statistical data analysis, data compression, and image processing [28, 29]. The technique is based on a basis change which reduces the number of coordinates used to represent the data while maximizing the variance retained. Its use is particularly adapted to correlated data where the dimensionality of the data can be strongly reduced while retaining most of the measurement variations. Given a set of N centered multivariate measurements $\mathbf{x}[t] \in \mathbb{R}^p$ used as training samples,* $1 \leq t \leq N$, the PCA basis is obtained by minimizing the following optimization function

$$\begin{array}{rcl} J_q(\mathbf{x}[t], \mathbf{w_k}) & = & \frac{1}{N}\sum_{t=1}^{N} ||\mathbf{x}[t] - \sum_{k=1}^{q} \mathbf{w_k}\mathbf{w_k^T}\mathbf{x}[t]||^2 \\ & = & \frac{1}{N}\sum_{t=1}^{N} ||\mathbf{x}[t] - \hat{\mathbf{x}}[\mathbf{t}]||^2 \end{array} \quad (1)$$

where the set of $q \leq p$ vectors $\{\mathbf{w_k}\}_{1\leq k\leq q}$ of $\mathbb{R}^p$ form the PCA basis. The linear combinations $\hat{\mathbf{x}}[\mathbf{t}] = \sum_{k=1}^{q} \mathbf{w_k}{\mathbf{w_k}}^T\mathbf{x}[t]$ represent the projections of $\mathbf{x}[t]$ on the PCA basis, and Eq. (1) therefore quantifies the mean squared error the original measurements $\mathbf{x}[t]$ and their projections. Note that the optimization function equivalently maximizes the variance retained by the family of vectors $\{\mathbf{w_k}\}_{1\leq k\leq q}$. Setting to

*Measurements are centered so that the origin of the coordinate system coincides with the centroid of the set of measurements. This translation is desirable to avoid a biased estimation of the basis $\{\mathbf{w_k}\}_{1\leq k\leq q}$ of $\mathbb{R}^p$ towards the centroid of the set of measurements.

zero the derivative of $J_q(\mathbf{x}[t], \mathbf{w_k})$ with respect to $\mathbf{w_k}$ gives

$$\mathbf{C}\mathbf{w_k} = \lambda_k \mathbf{w_k} \tag{2}$$

where $\mathbf{C} = \frac{1}{N}\sum_{t=1}^{N} \mathbf{x}[t]\mathbf{x}[t]^T$ is the sample covariance matrix of the sensor measurements $\mathbf{x}[t]$. The minimizer of Equation (1) is therefore given by the set of the first q eigenvectors $\{\mathbf{w_k}\}$ of the sample covariance matrix $\mathbf{C}$ [28]. These eigenvectors are called the *principal components* and form the PCA basis. The corresponding eigenvalues λ_k quantify the amount of variance conserved by the eigenvectors. Indeed, left-multiplying Equation (2) by $\mathbf{w_k}^T$ gives

$$\mathbf{w_k}^T \mathbf{C}\mathbf{w_k} = \mathbf{w_k}^T \lambda_k \mathbf{w_k} \tag{3}$$

and as $\mathbf{w_k}^T \mathbf{C}\mathbf{w_k} = \mathbf{w_k}^T(\frac{1}{N}\sum_{t=1}^{N} \mathbf{x}[t]\mathbf{x}[t]^T)\mathbf{w_k} = \sum_{t=1}^{N} \hat{\mathbf{x}}[\mathbf{t}]^T \hat{\mathbf{x}}[\mathbf{t}]$ and $\mathbf{w_k}^T \mathbf{w_k} = 1$, each eigenvalue λ_k quantifies the variance of the projections of the measurements $\mathbf{x}[t]$ on the corresponding k-th eigenvector. The sum of the eigenvalues therefore equals the total variance of the original set of observations X, i.e.:

$$\sum_{k=1}^{p} \lambda_k = \frac{1}{N}\sum_{t=1}^{N} ||\mathbf{x}[t]||^2$$

It is convenient to order the vectors $\mathbf{w_k}$ by decreasing order of the eigenvalues, so that the proportion P of retained variance by the first q principal components can be expressed by:

$$P(q) = \frac{\sum_{k=1}^{q} \lambda_k}{\sum_{k=1}^{p} \lambda_k} \tag{4}$$

Storing columnwise the set of vectors $\{\mathbf{w_k}\}_{1 \leq k \leq q}$ in a $W_{p \times q}$ matrix, approximations $\hat{\mathbf{x}}[\mathbf{t}]$ to $\mathbf{x}[t]$ in $\mathbb{R}^p$ are obtained by:

$$\hat{\mathbf{x}}[\mathbf{t}] = WW^T\mathbf{x}[t] = W\mathbf{z}[\mathbf{t}] \tag{5}$$

where

$$\mathbf{z}[\mathbf{t}] = W^T\mathbf{x}[t] = \begin{pmatrix} \sum_{i=1}^{p} w_{i1}x_i \\ ... \\ \sum_{i=1}^{p} w_{iq}x_i \end{pmatrix} = \sum_{i=1}^{p} \begin{pmatrix} w_{i1}x_i \\ ... \\ w_{iq}x_i \end{pmatrix} \tag{6}$$

denotes the column vector of coordinates of $\hat{\mathbf{x}}[\mathbf{t}]$ in $\{\mathbf{w_k}\}_{1 \leq k \leq q}$, also referred to as the *principal component scores*.

2.3 Principal component aggregation

The computation of the principal component scores $\mathbf{z}[\mathbf{t}]$ can be performed within the network if each node i is initialized with the q scalar components $\mathbf{w}_{i1}, .., \mathbf{w}_{iq}$ of the principal component basis [14]. As proposed in [14], this initialization can done during a training stage where a set of measurements is first collected from the network and the related covariance matrix is estimated. This set of measurements is

supposed to be large enough to provide an accurate estimate of the covariance matrix. Once the first q principal components are computed, they are sent back to the network and stored in the respective nodes (e.g., the i-th node stores only the elements $\mathbf{w}_{i1}, .., \mathbf{w}_{iq}$). While our proposition for distributing the computation of the principal components will be introduced in Section 3, here we limit to review the aggregation mechanism to compute the PC scores in a distributed manner.

The principal component scores can be computed by summing along the routing tree the vectors $(\mathbf{w}_{i1}x_i[t], .., \mathbf{w}_{iq}x_i[t])$ available at each node. The aggregation primitives take then the following form :

$$\begin{aligned} init(x_i[t]) &= \langle \mathbf{w}_{i1}x_i[t]; ..; \mathbf{w}_{iq}x_i[t] \rangle \\ f(\langle x_1; ..; x_q \rangle, \langle y_1; ..; y_q \rangle) &= \langle x_1 + y_1; ..; x_q + y_q \rangle \end{aligned}$$

where partial state records are vectors of size q.

2.4 Applications

Once the transformed data $\mathbf{z}[t]$ reaches the base station, it can be used for multiple purposes and applications, such as approximate monitoring, feature extraction or event detection.

2.4.1 Approximate monitoring

In approximate monitoring, the state of the process sensed by the WSN at time t is summarized by the vector

$$\begin{aligned} \hat{x}_t = e(z_1[t], ..., z_q[t]) &= (\hat{x}_1[t], ..., \hat{x}_n[t]) \\ &= W^T z[t] \end{aligned}$$

returned by the evaluator function at the base station.

If the number of components is p, the exact set of measurements collected by sensors is retrieved at the base station. Otherwise, for a number of components less than p, the vector $\hat{x}_t$ provides an optimal approximation to the real measurement vector in a mean square sense (Equation (4)). Note that the transformation in a principal component basis allows the design of simple policies to manage congestion issues, like discarding scores associated to the components with the lowest variance.

It is also interesting to note that a simple additional procedure can be set up to assess the accuracy of approximations against a user defined threshold. We know that the approximation $\hat{x}_i[t]$ for the i-th sensor, $1 \leq i \leq p$, is given by (5) by:

$$\hat{x}_i[t] = \sum_{k=1}^{q} z_k[t] * w_{ik}$$

Since the elements $\{w_{ik}\}$ are already available at each node, by sending back to the routing tree the k principal component scores we can enable these two additional functionalities: (i) each sensor is able to

compute the approximation retrieved at the sink, and (ii) each sensor is able to send a notification if the approximation error is greater than some user defined threshold ϵ. This scheme, referred to as *supervised compression* in [14], guarantees that all data eventually obtained at the sink are within $\pm\epsilon$ of their actual measurements.

2.4.2 Dimensionality reduction

An important class of WSN applications consists in inferring, on the basis of sensor measurements, some *high level* information such as the type or number of occurred events. For instance, a WSN monitoring the vibrations of a bridge could be used to answer queries concerning the type or number of vehicles that pass over the bridge. This class of problems is typically tackled by supervised learning techniques such as Naive Bayes classifiers, decision trees, or support vector machines [30, 31]. The PCA is in this context an effective preprocessing technique which simplifies the learning process by reducing the dimensionality of the problem [28, 31].

2.4.3 Event detection

The use of PCA for identifying at the network scale unexpected events which are not detectable at the node level has been discussed in the literature [19, 20]. Detection of such events can be achieved by focusing on the low variance components. In fact, such components are expected to bear coordinates close to zero, as they typically account for sensor equipment noise. In this context, the principal component aggregation scheme can be used to deliver the value of low variance components, and the evaluator function takes the form of a statistical test checking if the coordinates on one or more low variance components is different from zero.

2.5 Tradeoffs

In this section, we will use the term *default* to denote the scheme that forwards all the measurements to the sink at each epoch, and the term *PCAg* to denote the scheme based on the principal component aggregation. Let us first consider the distribution of the network load among sensor nodes, with a particular focus on its upper bound. The upper bound, henceforth referred to as *highest network load*, determines the network capacity, that is the amount of traffic that the network can handle at one time [6]. The highest network load is also related to the network lifetime in terms of time to first failure[†] as the radio is known to be the most energy consuming task of a wireless sensor [5, 25].

In the *default scheme*, the root of the routing tree is the node that sustains the highest network load as it forwards to the base station the measurements from all other sensors. Its radio throughput therefore causes the root node to have its energy exhausted first, which may lead the rest of the network to be disconnected from the base station.

[†]The time to first failure is the amount of time at which the first node in the network runs out of energy.

In the *PCAg scheme* (Section 2.1.3), the network load is distributed in a more homogeneous manner. The node $i_{\mathcal{C}}^* = \arg\max_i C_i$ whose number of children is the highest becomes the limiting node in terms of highest network load. The network traffic at this node amounts to $q(C_{i_{\mathcal{C}}^*}+1)$ packets per epoch, where q is the number of components retained. A PCAg scheme therefore reduces the highest network load in a configuration where

$$q(C_{i_{\mathcal{C}}^*}+1) \leq 2p-1. \tag{7}$$

We illustrate the tradeoff by considering two extreme configurations. If only the first principal component is retained, the PCAg reduces the highest network load as $C_{i_{\mathcal{C}}^*} < p$. On the contrary, if all the components are retained, i.e. $q = p$, the equation (7) simplifies into $pC_{i_{\mathcal{C}}^*} \leq p-1$, and the PCAg incurs a higher network load than the default scheme.

The main parameter of the approach is therefore the number of principal components to retain, which trades network load for sensing task accuracy. The amount of information retained by a set of q principal components depends on the degree of correlation among the data sources. Whenever a set of sensors collects correlated measurements, a small proportion of principal components is likely to support most of the variations observed by the network. The number of principal components to retain can be specified by the application, for example by means of a cross validation procedure on the accuracy of the sensing task at the base station [14]. It can be also be driven by network constraints or network congestion issues. In the latter case, a policy prioritizing packets containing the component scores of the principal components would allow to retain a maximum amount of information under network load constraints.

Finally, given a number of principal components, note that the highest network load may be further reduced if the algorithm that builds the routing tree uses $C_{i_{\mathcal{C}}^*}$ as a parameter. The network load indeed depends on the number of children a node has. Although we de not address this aspect, routing strategies that aim at lowering $C_{i_{\mathcal{C}}^*}$ can therefore further improve the efficiency of the PCAg scheme.

3 Computation of the principal components

3.1 Outline

This section addresses the initialization procedure whereby each node i can be initialized with the elements $\mathbf{w}_{i1}, .., \mathbf{w}_{iq}$ of the principal components $\mathbf{w}_1, .., \mathbf{w}_q$. In [14], we proposed a centralized approach for performing this initialization stage. This approach is first briefly recalled in 3.2. We then extend this work by showing that a fully distributed implementation can perform this initialization process. The proposed approach relies on the power iteration method (PIM), a classic iterative technique for computing the eigenvectors of a matrix [21], and on a simplifying assumption on the covariance structure which allows to estimate the covariance matrix in a distributed way. Section 3.3 presents and discusses the distribution of the covariance matrix computation and storage. We then show in Section 3.4 that by means of the distributed covariance matrix computation, the PIM can be implemented in an aggregation service in a distributed manner.

Each of these sections is concluded by a complexity analysis of the communication, computational,

and memory costs. We finally discuss in Section 3.5 three methods for performing the initialization stage.

3.2 Centralized approach

Let us suppose that t vectors of sensor measurements $\mathbf{x}[\tau] \in \mathbb{R}^p$, $\tau \in \{1, .., t\}$ are retrieved at the base station. Let $\mathbf{X}[t]$ denote the $p \times t$ matrix whose τ-th column-vector is the vector $\mathbf{x}[\tau] \in \mathbb{R}^p$. An estimate of the covariance matrix $\mathbf{C}[t] = (c_{ij}[t])_{1 \leq i,j \leq p}$ is obtained at time t by computing

$$\begin{aligned} \mathbf{C}[t] &= \frac{1}{t} \sum_{\tau=1}^{t} (\mathbf{x}[\tau] - \bar{\mathbf{x}}[t])(\mathbf{x}[\tau] - \bar{\mathbf{x}}[t])^T \\ &= \frac{1}{t} \mathbf{X}[t]\mathbf{X}[t]^T - \bar{\mathbf{x}}[t]\bar{\mathbf{x}}[t]^T \end{aligned} \tag{8}$$

where $\bar{\mathbf{x}}[t] = \frac{1}{t} \sum_{\tau=1}^{t} \mathbf{x}[\tau]$ is the average vector.

The covariance matrix can be updated recursively as new vectors of observations $\mathbf{x}[t+1]$ are made available at the base station. From Equation (8) it follows that

$$c_{ij}[t] = \frac{1}{t} S_{ij}[t] - \frac{1}{t^2} S_i[t] S_j[t] \tag{9}$$

where

$$\begin{aligned} S_i[t] &= \sum_{\tau=1}^{t} x_i[\tau] = S_i[t-1] + x_i[t] \\ S_{ij}[t] &= \sum_{\tau=1}^{t} x_i[\tau] x_j[\tau] = S_{ij}[t-1] + x_i[t] x_j[t] \end{aligned} \tag{10}$$

This means that the storage of the quantities t, S_j and S_{ij}, $1 \leq i, j \leq p$, in the base station makes possible the update of the covariance matrix estimation $\mathbf{C}[t]$ at time t. Once the covariance matrix is estimated, the principal components can be computed using a standard eigendecomposition method [21] and its elements communicated to the sensor nodes.

3.2.1 Scalability analysis

Highest network load: The centralized estimation of the covariance matrix from a set of t vectors of observations requires t operations of type D as defined in Section 2.1.3. The root node supports the highest network load, which is therefore in $O(tp)$. Once the eigenvector decomposition is performed at the base station, the root node then transmits the principal components to the sensor nodes, which requires qp operations of type F. Overall, the communication cost amounts to $O(tp + qp)$.

Computational and memory costs: From Equation (10), the computational cost related to the covariance matrix at the base station is $O(tp^2)$. The memory cost for storing the matrix of observations and

the covariances is $O(tp + p^2)$ or $O(p^2)$ if recursive updates are relied on. As far as the estimation of the principal components is concerned, the cost of standard eigendecomposition algorithm is $O(p^3)$ in terms of computation and $O(p^2)$ in terms of memory [21].

3.3 Distributed estimation of the covariance matrix

This section presents an alternative methodology to distribute the computation and storage of the covariance matrix within the network. The methodology relies on the observation that sensors that are geographically distant often present low correlation values. This hypothesis often holds in real-world situations as shown in [32, 33]. This leads us to make the simplifying hypothesis that covariances between sensors which are out of reach is equal to zero. Note that the radio range can usually be tuned on wireless sensor platforms by increasing or reducing the power transmission level [25]. If available, this feature could be used to set the radio range such that nearby correlated sensors are within radio range.

Let $\mathcal{N}_i$ be the neighborhood of the sensor i, i.e., the set of sensors which are in the communication range of the i-th node. An instance of neighborhood is illustrated in Figure 1. Our simplifying hypothesis assumes that $c_{ij}[t] = 0$ for all $j \notin \mathcal{N}_i$. This makes the computation of the covariance matrix scalable to large networks. The covariances between sensors i and j can indeed be computed recursively by Equation (10) once the sensor i keeps over time a record of t, S_j and S_{ij} for $j \in \mathcal{N}_i$. This hypothesis will be referred to in the following as the *local covariance hypothesis*.

3.3.1 Positive semi-definiteness criterion

If the local covariance hypothesis is not verified, the distributed sparse matrix obtained by the local covariance hypothesis may lead to a globally non positive definite matrix[‡] The positive semi-definiteness is however a necessary (and sufficient) condition for a symmetric matrix to be a covariance matrix. To face this issue, the approach adopted in this article is to discard eigenvectors whose eigenvalues are found to be negative. This approach, according to [34], is the simplest way to transform a non positive definite matrix in a semi positive definite matrix. The detection of a negative eigenvalue will be detailed in Section 3.4.2, and used as a stopping criterion in the last line of Algorithm 3.4.2.

A good review of existing methods for transforming the approximated covariance matrix can be found in [34]. Our opinion is that it seems however difficult to implement these method in a distributed manner. Note that despite its simplicity, the method that consists in discarding negative eigenvectors was observed by authors in [34] to provide good approximations to the covariance matrix.

3.3.2 Scalability analysis

Highest network load: Each update requires a node i to send one packet (its measurement) and to receive $|\mathcal{N}_i|$ packets (other sensors' measurements). Let $i^*_{\mathcal{N}}$ be the node that has the largest number of

[‡]that contains negative eigenvalues

neighbors, i.e., $i^*_{\mathcal{N}} = \arg\max_i |\mathcal{N}_i|$. The highest network load is therefore $O(t|\mathcal{N}_{i^*_{\mathcal{N}}}|)$.

Computational and memory costs: From Equation (10), the updates of S_j and S_{ij} demand $3|\mathcal{N}_i|$ floating points additions/subtractions, $2|\mathcal{N}_i|$ floating points multiplications and $2|\mathcal{N}_i|$ floating points divisions. The highest computational cost therefore scales in $O(|\mathcal{N}_{i^*_{\mathcal{N}}}|)$. The number of variables to maintain at a sensor node is one integer and $2|\mathcal{N}_i|$ floating points numbers, so the highest memory cost amounts to $O(|\mathcal{N}_{i^*_{\mathcal{N}}}|)$.

3.4 Distributed estimation of the principal components

The distribution of eigendecomposition algorithms is a fairly known research area [21]. However, the distribution sought in our problem should comply with the specificity that each node only has access to a single dimension of the problem and needs a specific entry of each principal eigenvector.

We propose in the following an approach that leverages the aggregation service to implement in a rather simple way the power iteration method (PIM), a well-known solution to the eigenvalue decomposition problem [21]. The resulting technique complies with the specific features mentioned above, and lets each node i locally identify the subset $\{w_{ik}\}$ required by the principal component aggregation.

After describing the PIM in Sections 3.4.1 and 3.4.2, we detail in Sections 3.4.3 and 3.4.4 its implementation in the aggregation service. The analysis of complexity of the method is provided in Section 3.4.5.

3.4.1 Power iteration method

Let $\mathbf{C}$ be a covariance matrix. The power iteration method starts with a random vector $\mathbf{v}_0$. At the t-th iteration, the vector $\mathbf{v}_{t+1}$ is obtained by multiplying $\mathbf{C}$ by $\mathbf{v}_t$, and by normalizing it. It can be easily shown [21] that $\mathbf{v}_t$ converges to the principal eigenvector $\mathbf{w}_1$ of $\mathbf{C}$, under the mild condition that $\mathbf{v}_0$ is not strictly orthogonal to the principal eigenvector. The convergence rate is exponential, its base being the squared ratio of the two principal eigenvalues. The convergence criteria can be defined either as a minimum variation δ for $\mathbf{v}_t$, or as a highest number of iterations $t_{\max}$ [21, 22]. Algorithm 3.4.1 outlines the different steps.

Standard power iteration algorithm $\mathbf{v}_0 \leftarrow$ arbitrary initialization $t \leftarrow 0$ $\mathbf{v}_t \leftarrow \mathbf{C}\mathbf{v}_t$ $\mathbf{v}_{t+1} \leftarrow \frac{\mathbf{v}_t}{\|\mathbf{v}_t\|}$ $t \leftarrow t+1$ $t > t_{max}$ and/or $\|\mathbf{v}_{t+1} - \mathbf{v}_t\| \leq \delta$ return $\mathbf{v}_t$

Note that as the method converges to the principal eigenvector $\mathbf{w}_1$, the normalizing factor $\|\mathbf{v}_t\|$ converges to the associated eigenvalue λ_1, as by definition :

$$\mathbf{C}\mathbf{w}_1 = \lambda_1 \mathbf{w}_1 \tag{11}$$

In practical settings, the power method quickly converges to a linear combination of eigenvectors whose eigenvalues are close, or to the eigenvector whose eigenvalue is the highest if eigenvalues are well separated. As our purpose here is to find the subspace that contains the signal, eigenvectors with

close eigenvalues can be thought of as generating a subspace where similar amount of information are retained along any vector of the subspace. The convergence to a linear combination of eigenvectors with close eigenvalues is therefore deemed acceptable as far as principal component aggregation is concerned.

3.4.2 Computation of subsequent eigenvectors

The standard way to employ PIM in order to find the other eigenvectors (up to the number q) is the deflation method which consists in applying the PIM to the covariance matrix from which we have removed the contribution of the k principal eigenvector already computed [21]. We obtain

$$\begin{aligned} \mathbf{v}_{t+1} &= (\mathbf{C} - \sum_{l=1}^{k} \mathbf{w}_l \lambda_l \mathbf{w}_l^T)\mathbf{v}_t \\ &= \mathbf{C}\mathbf{v}_t - \sum_{l=1}^{k} \mathbf{w}_l \lambda_l \mathbf{w}_l^T \mathbf{v}_t \end{aligned}$$

which boils down to orthogonalize $\mathbf{v}_t$ with respect to all previously identified eigenvectors $\{\mathbf{w}_l\}_{1 \leq l < k+1}$.

A verification step must be added to the algorithm in order to detect negative eigenvalues. This can be achieved by comparing the sign of elements of $\mathbf{v}_t$ and $\mathbf{v}_{t+1}$ after convergence is assumed. From Equation (11), a negative eigenvalue implies that elements of $\mathbf{v}_t$ and $\mathbf{v}_{t+1}$ have pairwise different signs. The criterion we define for determining the sign of an eigenvalue is

$$sign(\sum_{i=1}^{p} sign(v_t[i] v_{t+1}[i])$$

as taking the average sign of the sum of the signs over all pairwise products of $\mathbf{v}_t$ and $\mathbf{v}_{t+1}$ makes a more robust estimate of the sign of the eigenvalue in case of numerical errors. We rely on this additional criterion for stopping the computation of the principal components. The resulting process is given by Algorithm 3.4.2.

Modified power iteration algorithm for q eigenvectors $k \leftarrow 0$ $k \leftarrow k+1$ $t \leftarrow 0$ $\mathbf{v}_0 \leftarrow$ arbitrary initialization $\mathbf{v}_t \leftarrow \mathbf{C}\mathbf{v}_t$ $\mathbf{v}_t \leftarrow \mathbf{v}_t - \sum_{l=1}^{k-1} \langle \mathbf{v}_t, \mathbf{w}_l \rangle \mathbf{w}_l$ $\mathbf{v}_{t+1} \leftarrow \frac{\mathbf{v}_t}{\|\mathbf{v}_t\|}$

$t \leftarrow t+1$ $t > t_{\max}$ or $\|\mathbf{v}_{t+1} - \mathbf{v}_t\| \leq \delta$ $\lambda_k \leftarrow \pm \|\mathbf{v}_t\|$ $\mathbf{w}_k \leftarrow \mathbf{v}_{t+1}$ $k = q$ or $\lambda_k < 0$

3.4.3 Implementation in the aggregation service

A fully distributed implementation of the algorithm is possible if the covariance matrix has been estimated as proposed in 3.3. The computation of one iteration therefore goes in two stages :

Computation of $\mathbf{Cv}$

$$\forall i \in \{1, .., p\}\ ,\ (\mathbf{C}\mathbf{v})[i] = \sum_{j=1}^{p} c_{ij} v[j]$$

As we made the assumption that:

$$\forall i \in \{1,..,p\}\ ,\ \forall j \notin \mathcal{N}_i\ ,\ c_{ij} = 0$$

The sum can be simplified :

$$\forall i \in \{1,..,p\}\ ,\ \left(\mathbf{Cv}\right)[i] = \sum_{j \in \mathcal{N}_i} c_{ij} v[j]$$

Recall that each node i has available the set of covariances (c_{ij}), $j \in \mathcal{N}_i$. Therefore at each step t, node i only has to broadcast $v_t[i]$ and to receive the $v_t[j]$, $j \in \mathcal{N}_i$ from its neighbors for updating locally its $\sum_{j \in \mathcal{N}_i} c_{ij} v[j] = \left(\mathbf{Cv}\right)[i]$. The power iteration method can thus be implemented in a fully distributed manner.

Normalization and orthogonalization The norm can be computed using the primitives detailed in Section 2.1.2, while the orthogonalization is obtained by means of these primitives:

$$\begin{aligned} init(v_t[i]) &= \langle (v_t[i]\mathbf{w_l}[i])_{1 \leq l \leq k-1} \rangle \\ f(\langle X \rangle, \langle Y \rangle) &= \langle X + Y \rangle \\ e(\langle X \rangle) &= X. \end{aligned}$$

The resulting $k-1$ scalar products $\{\langle \mathbf{w}_l, \mathbf{v_t} \rangle\}_{1 \leq l \leq k-1}$ are then communicated to the sensors by means of an F operation, so that each sensor i can locally orthogonalize $\mathbf{v}_{t+1}[i] \leftarrow \mathbf{v}_t[i] - \sum_{l=1}^{k-1} \langle \mathbf{v}_t, \mathbf{w}_l \rangle \mathbf{w}_l[i]$.

3.4.4 Synchronization

The nodes have to be synchronized so that they all work on the same copy of $\mathbf{v}$ at the same time. The scheduling policy is presented in Algorithm 3.4.4. The implementation is straightforward since the aggregation service is synchronized.

Network synchronization scheme for the distributed power iteration algorithm $k \leftarrow 0$ $\forall i$ node i is initialized with $C[i,i]$ The nodes get their neighbor's value $(\mathbf{v_t}[j])_{j \in \mathcal{N}_i}$ The computation of $C\mathbf{v_t}$ is performed in parallel $\|\mathbf{v_t}\|$ and $\{\langle \mathbf{w}_l, \mathbf{v_t} \rangle\}_{1 \leq l \leq k-1}$ are computed by the aggregation service $\|\mathbf{v_t}\|$ and $\{\langle \mathbf{w}_l, \mathbf{v_t} \rangle\}_{1 \leq l \leq k-1}$ are fed back in the network For all i, $\mathbf{v_{t+1}}[i] \leftarrow \frac{(C\mathbf{v_t})[i] - \sum_{l=1}^{k-1} \langle \mathbf{v}_t, \mathbf{w}_l \rangle \mathbf{w}_l[i]}{\|\mathbf{v_t}\|}$ (in parallel on all nodes) convergence is obtained $\mathbf{w}_k \leftarrow \mathbf{v}$ $k = q$ or $\lambda_k \leq 0$

3.4.5 Scalability analysis

Highest network load: At each iteration, the computation of the product $\mathbf{Cv_t}$ requires a node i to send one packet and to receive $|\mathcal{N}_i|$ packets. The highest network load for this operation is therefore $O(|\mathcal{N}_{i^*_{\mathcal{N}}}|)$. The normalization step implies one A and one F operations (as defined in 2.1.3), and the orthogonalization step implies $k-1$ operations of type A and F respectively, where k is the index of

the principal component computed. The highest network load for the computation of the q first principal components amounts to $O(q|\mathcal{N}_{i^*_{\mathcal{N}}}| + q^2|\mathcal{C}_{i^*_{\mathcal{C}}}|)$.

The highest network load related to the power iteration method is therefore quadratic in the number of principal components computed. Depending on the constraints of the network, it may be more interesting to retrieve the approximated covariance matrix from the network at the base station for a centralized computation of the eigenvectors. The tradeoff depends here on the communication and computational resources available in the network, and will be discussed more in detail in Section 3.5.

Computational and memory costs For a node i, the cost of the computation of $\mathbf{Cv_t}$ is $O(|\mathcal{N}_i|)$. The cost of the normalization step is $O(|\mathcal{C}_i|)$ for node i, and the orthogonalization step is $O(k|\mathcal{C}_i|)$. The overall highest computational cost therefore amounts to $O(q(|\mathcal{N}_{i^*_{\mathcal{N}}}| + |\mathcal{C}_{i^*_{\mathcal{C}}}|)$. Regarding memory costs, each node i needs to maintain variables for storing its local w_{ik} and its neighbors parameters w_{jk}, $j \in \mathcal{N}_i$. The complexity of the highest memory cost is therefore $O(q + |\mathcal{N}_{i^*_{\mathcal{N}}}|)$.

3.5 Summary

The benefits of the different approaches for computing the covariance matrix and its principal components depend on the communication, computational and memory constraints available in the network nodes and at the base station. A summary of the complexities of these approaches is given in Table 1.

Operation	Communication	Computation	Memory
Covariance			
Centralized	$O(pT)$	$O(p^2T)$	$O(p^2)$
Distributed	$O(\|\mathcal{N}_{i^*_{\mathcal{N}}}\|T)$	$O(\|\mathcal{N}_{i^*_{\mathcal{N}}}\|T)$	$O(\|\mathcal{N}_{i^*_{\mathcal{N}}}\|)$
Eigenvectors			
Centralized	$O(qp)$	$O(p^3)$	$O(p^2)$
Distributed	$O(q^2\|\mathcal{N}_{i^*_{\mathcal{N}}}\|)$	$O(q(\|\mathcal{N}_{i^*_{\mathcal{N}}}\| + \|\mathcal{C}_{i^*_{\mathcal{C}}}\|))$	$O(q + \|\mathcal{N}_{i^*_{\mathcal{N}}}\|)$

Table 1. Scalability of the centralized, hybrid and distributed schemes.

4 Experimental results

This section illustrates by means of a compression task the ability of the proposed approach to properly identify the principal component subspace, and discusses the tradeoffs involved between the compression accuracy and the communication costs. The experimental study is based on a set of real world temperature measurements involving a network of 52 sensors (Section 4.1), and on a network simulation used to vary the structure of the routing trees to study the impact of the network topology on the network load (Section 4.2). Results on the ability of a few components to retain most of the information are first illustrated in Section 4.3, followed in Section 4.4 by an analysis of the network loads as the number of

retained components increases. The ability of the distributed PCA to properly identify the principal components is then studied in Sections 4.5 and 4.6, which focus on the covariance matrix approximation and eigenvector extraction stages, respectively. Section 4.7 concludes by a discussion on the main results.

4.1 Dataset description

Experiments were carried out using a set of five days of temperature readings obtained from a 54 Mica2Dot sensor deployment at the Intel research laboratory at Berkeley [35]. The sensors 5 and 15 were removed as they did not provide any measurement. The readings were originally sampled every thirty-one seconds. A preprocessing stage where data was discretized in thirty second intervals was applied to the dataset. After preprocessing, the dataset contained a trace of 14400 readings from 52 different sensors. The code associated to the preprocessing and the network simulation was developed in R, an open source statistical language, and is available from the authors' web site [36].

Examples of temperature profiles and dependencies between measurements are reported in Fig. 4 and 5, respectively. The sensors 21 and 49 were the least correlated ones over that time period, with a correlation coefficient of 0.59. They were situated on opposite sides of the laboratory. Temperature over the whole set of data ranged from about 15°C to 35°C.

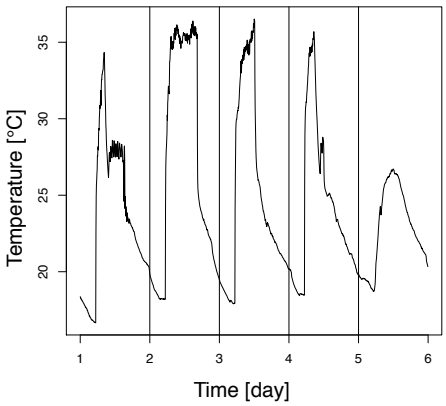

Figure 4. Temperature measurements collected by sensor 21 over a five day period.

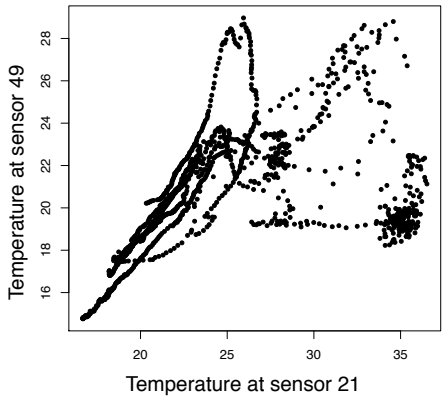

Figure 5. Examples of the dependencies between the measurements of sensor 21 and sensor 49.

4.2 Network simulation

The positions of the sensors are provided in [35], and the distribution of the sensors in the laboratory can be seen in Fig. 6. We analyzed the communication costs in different routing trees which were generated in the following way. The root node was always assumed to be the top right sensor node in Fig. 6 (node 16 in [35]). The routing trees were generated on the basis of the sensor positions and the radio range was varied from 6 meters (minimum threshold such that all sensor could find a parent) to 50 meters (all sensors in radio range of the root node). Starting from the root node, sensors were assigned

to their parent in the routing tree using a shortest path metric, until all sensors were connected. An illustration of the routing tree obtained for a maximum communication range of $10\mathrm{m}$ is reported in Fig. 6.

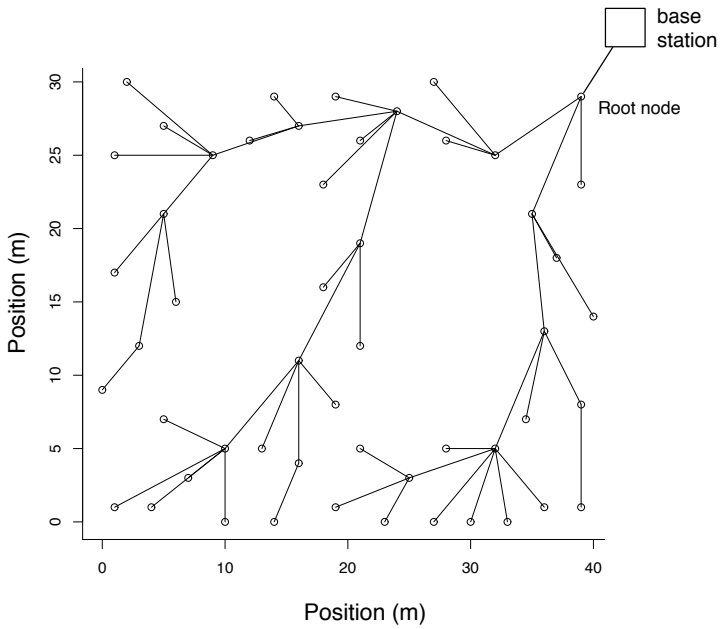

Figure 6. Map of sensors and the routing obtained for a connectivity of 10 meters. The root node connecting the sink is the top right sensor. The routing tree has a depth of seven.

4.3 Principal component aggregation

We study in this section the amount of variance that the PCA can retain in the measurements for different training periods and number of components. We rely on a 10-fold cross validation technique to simulate the fact that the measurements used to compute the principal components are not used for assessing the accuracy of the approach. More precisely, the dataset is split in ten consecutive blocks (1440 observations – i.e., half a day of measurements). Each of the ten blocks is used in turn as a *training* set to compute the covariance matrix and its eigenvectors, while the remaining observations, referred to as *test* set, are used to estimate the percentage of retained variance. This provides ten estimates of the percentage of retained variance on measurements not used for computing the eigenvectors, which we average to obtain an overall estimate of the method.

Fig. 7 provides the average retained variance on the 10 test sets for the first 25 principal components. The upper line gives the average amount of variance retained when the principal components are computed with the test sets, and provides an upper bound on the compression efficiency that the PCA can achieve on this dataset. The lower curve gives the average amount of variance retained on the test set when the components are computed with the training set. This figure shows that the first principal com-

ponent accounts on average for almost 80% of the variance, while 90% and 95% of variance are retained with 4 and 10 components, respectively. The confidence level of these estimates was about $\pm 5\%$. Additional experiments were run using K-fold cross validation with K ranging from 2 to 30. The percentages of retained variance on the test data blocks tended to decrease with K. Losses of a few percents were observed for K higher than 15 (less than nine hours of data).

The amount of retained variance increases very fast with the first principal component, and becomes almost linear after about ten components. A linear increase of retained variance with the number of principal components reflects the fact that the components obtained by the PCA are actually no better than random components [28]. From Fig. 7, it therefore seems that from 10 or 15 components onwards, the remaining variations can be considered as white noise.

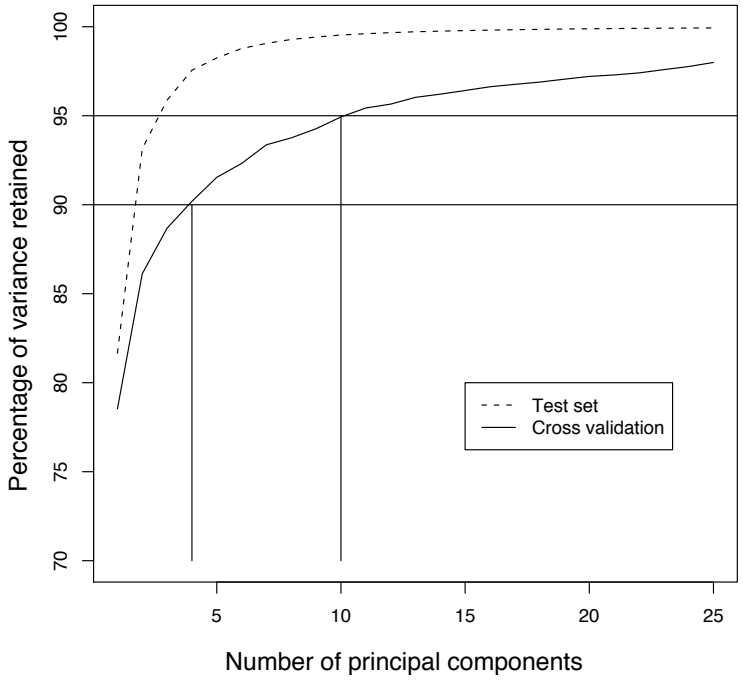

Figure 7. Capacity of principal components to retain the measurement variance.

Fig. 8 illustrates the approximations obtained during the first round of the cross validation (i.e., principal components are computed from the first 12 hours of measurements) for the sensor 49, using one, five and ten principal components. A single principal component provides rough approximations, which cannot account for fine-grained details of some of the sensor measurements. For example, the stabilization of the temperature around $20^{\circ}\mathrm{C}$ around noon during the second, third and fourth day (probably due to the activation of an air conditioning system at a location close to sensor 49) are not captured by the approximations. Increasing the number of principal components allows to better approximate the local variations, and passing to five components provides for example a much better approximation of the sensor measurements.

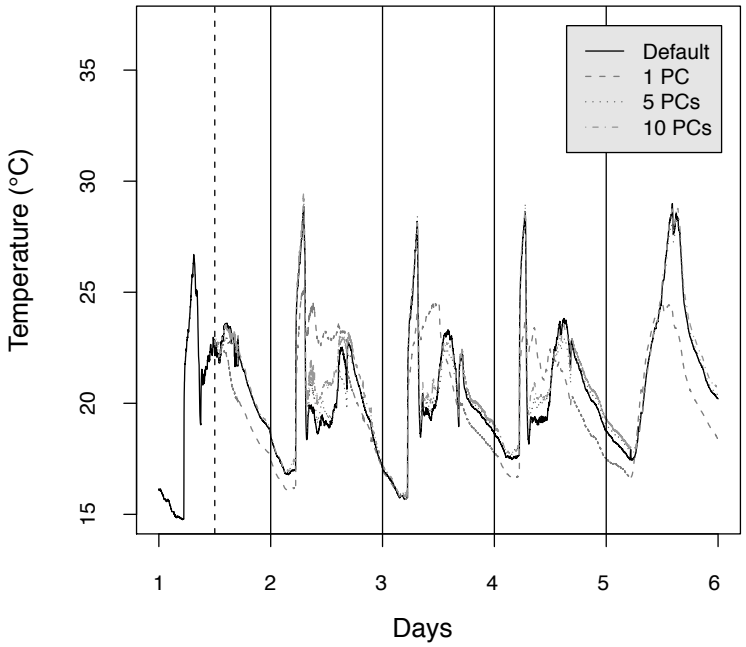

Figure 8. Approximations obtained on the test set for the sensor 21, using one, five and fifteen principal components. The covariance matrix was computed from the first twelve hours of measurements (before the vertical dashed line).

4.4 Communication costs

We compare in the following the communication costs entailed by the default and PCAg schemes for different types of routing trees. Fig. 9 reports the total number of packets processed by nodes (received and transmitted) during an epoch, for the default and PCAg scheme, as the radio communication range increases. The subfigure on the left reports the overall sensor network load, while subfigures in the middle and on the right detail with boxplots the distributions of the network load per node in the network, for the default and PCAg schemes, respectively.

Given that sensors choose as their parent the sensor within radio range that is the closest to the base station, increasing the radio communication range typically leads the routing tree to have a smaller depth, and its nodes to have a higher number of children. As was pointed out in Section 2.5, this is detrimental to the PCAg scheme, as this may lead the numbers of packets received by a node to be higher than in the default scheme.

For the default scheme, increasing the radio range reduces the overall sensor network load (Fig. 9, left), and eventually leads all the nodes but the root to only transmit one packet (Fig. 9, middle, radio range of 50m). In this extreme case, the routing tree has depth one, and all nodes but the root are leaf nodes. Note that the highest network load does not depend on the tree topology. This highest load is sustained by the root node which, whatever the depth of the tree, is required to forward other node's packets (i.e., 51 measurements to receive and send), and to send its own measurement to send. Its network load is therefore of 103 packets per epoch.

For the PCAg scheme, we first report results for the extraction of one component. A nice feature of aggregation is that the overall network load (Fig. 9, left) does not depend on the topology, thanks to the fact that forwarding does not increase the number of transmissions. Looking at the details of the distribution of the network load per node (Fig. 9, right), it is interesting to see that increasing the radio range has a reverse effect for the PCAg scheme, as it tends to increase the network load. This is a direct consequence of the increased number of children induced by routing trees with smaller depths. Eventually, for a fully interconnected network, we observe the same effect than for the default scheme, where all the sensors send only one packet, while the root node sustains the higher network load due to the forwarding task. Note however than thanks to the aggregation process, it only sends one packet, which reduces to 52 packets per epoch its network load (one packet transmitted and 51 packets received). The extraction of one component therefore decreases the network load supported by the root node. The network load incurred by k components is obtained by multiplying the values found in Fig. 9 by k. The PCAg scheme may therefore be less efficient than the the default scheme if many components are extracted.

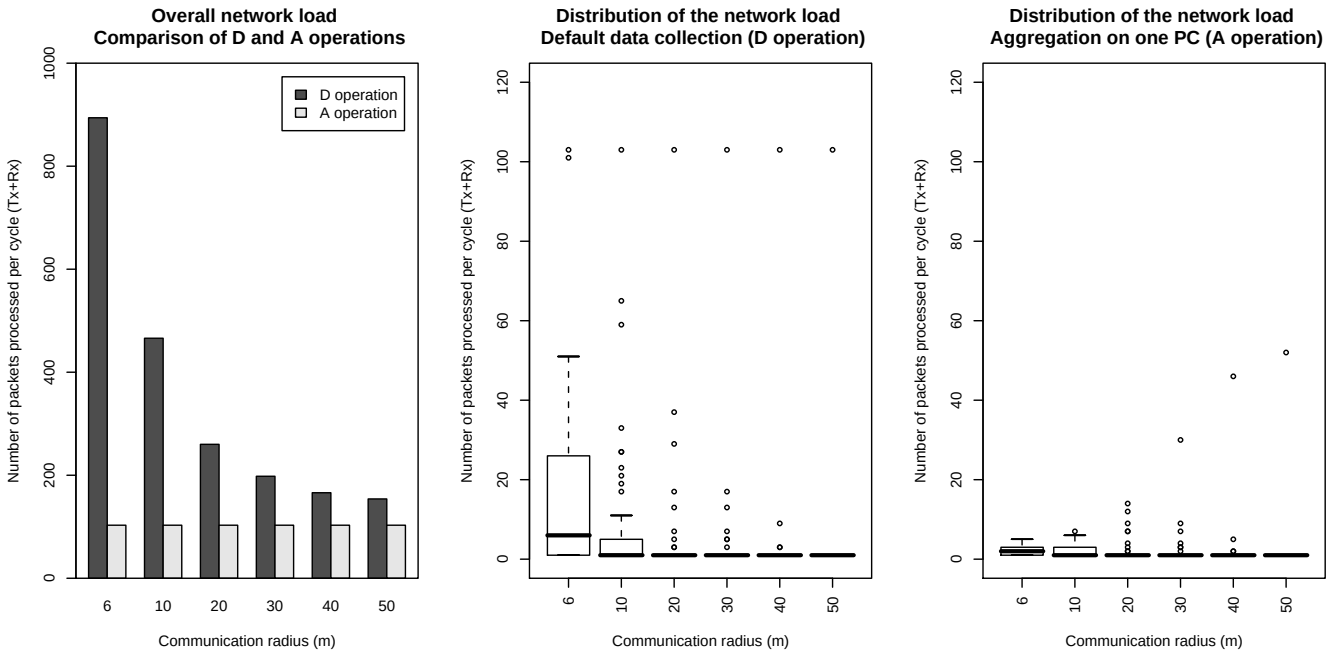

Figure 9. Communication costs entailed by D and A operations. The overall network load (left) is in all cases higher for a D operation than for an A operation. Details of the distribution of the network load per node reported as boxplots (middle and right) reveal that the distributions of the load are more balanced using aggregation. Aggregation also leads to reduce the highest network load.

This is illustrated in Fig. 10 where the number of packets processed (received and sent) is reported as a function of the number of principal components extracted for a radio range of 10 (Illustrated in Fig. 6). In this routing tree, the highest number of children is 6. For the extraction of one PC, the highest

network load is therefore of 7, i.e., 6 receptions and one transmission, to be compared with the highest network load of 103 for the root node in the default scheme. This results in a reduction of about 85% of the network load. Extracting more than 15 components leads however the highest network load to be higher than in the default scheme, as the sensor node aggregating the packets from its 6 children will sustain a network load of 105 packets per epoch.

The overall network load generated by the aggregation of k components is in the same manner k times the load generated by of the aggregation of one component. For a communication radius of 10 meters, this overall load was of 103 packets as reported in Fig. 9, left, to be compared with an overall load of 466 packets for the default scheme. The overall load generated by the aggregation process is therefore higher than the load of the default scheme from 5 components. This is due to the fact that aggregation, while decreasing the load of the most solicited node, increases evenly the load of all other nodes.

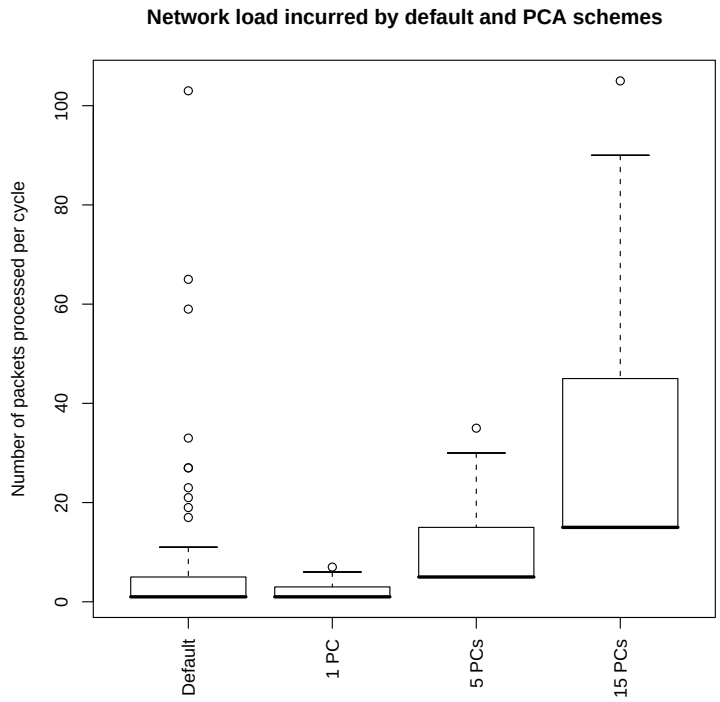

Figure 10. Comparison of the communication costs entailed by a D operation, and A operations with 1, 5 and 15 principal components. Radio range is 10 meters.

4.5 Distributed covariance matrix

We study in this section the ability of the local covariance hypothesis to properly identify the principal component subspace, and discuss how the accuracy loss is counterbalanced by gains in energy consumption and network load. In Fig. 11, the upper curve gives the amount of variance retained if all covariances are computed, and is the same as in Fig. 7. Lower curves correspond to the percent-

age of variance retained as the radio range of sensors is decreased, and illustrate the fact that the local covariance hypothesis may have as a negative consequence a loss of accuracy for identifying the principal components. If the local covariance hypothesis does not hold, the loss can be high, as is illustrated by a radio range of 6 meters. This loss is however attenuated when the number of principal components increases. In any case, the subspace obtained by computing the principal components from the approximate covariance matrix is significantly better in retaining information than a random subspace.

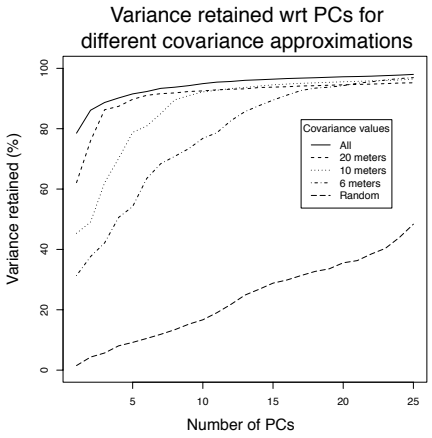

Figure 11. Capacity of the principal components to retain the measurement variance.

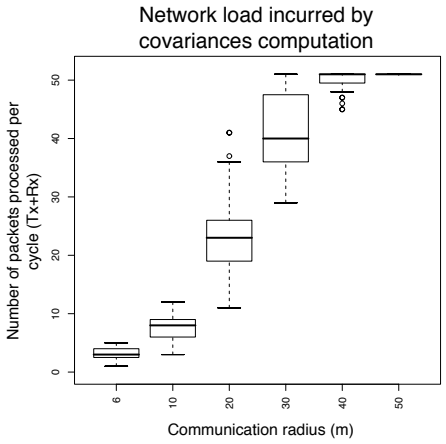

Figure 12. Network load incurred by local covariances updates.

We note that reducing the radio range also decreases the energy spent in communication, as a lower transmission power leads to a lower energy consumption for the radio. We illustrate in Fig. 12 the potential savings in terms of network load obtained by decreasing the radio range. This figure reports the distribution of the network loads related to the computation of the covariance matrix for varying radio ranges. The average network load increases with the radio range, as the neighborhood of sensors gets larger. Fig. 11 and 12 provide together an illustration of the tradeoff between the accuracy and the communication costs entailed by the local covariance hypothesis. A radio range range larger than 20 meters does not bring much gains in terms of accuracy, while it strongly increases the network load.

Also, it is interesting to compare the results reported in Fig. 12 and Fig. 9. The highest network load caused by an update is in all cases lower using the distributed covariance matrix scheme (52 packets processed against 101 at the root for the default data gathering operation). The average network load can however be higher as the radio range increases, due to the fact that nodes close to the leaves process more packets in the distributed scheme than in the centralized scheme.

4.6 Distributed principal component computation

We finally discuss the ability of the power iteration method to properly identify the principal components. The main parameter for the estimation of the eigenvectors is the convergence criteria used. We illustrate in Fig. 13 the difference in accuracy obtained on the test set, for different convergence criteria,

between the set of exact eigenvectors (Computed in a centralized manner with the QR method), and the set of approximated eigenvectors obtained by means of the power iteration method. The convergence threshold δ was set to 10^{-3}, and we tested the accuracy obtained for 5, 10, 20, 30, 40 and 50 iterations. Results reported are averages of ten-fold cross validations, and the confidence level of the results was about 2%.

This Figure allows to experimentally illustrate a set of possible behaviors, which are understood by keeping in mind that the rate of convergence of the power method depends on the ratio of the two dominant eigenvalues. Typically, in correlated data, the ratio of subsequent eigenvalues decreases exponentially (which can be seen qualitatively in Fig. 7), making the convergence speed lower as the number of principal components computed increases. As a result, few iterations are usually enough to converge to the first eigenvectors. This is seen in Fig. 13 for the first PC, for which as little as five iterations are enough to nicely converge. In the computation of the subsequent PCs, we observe that the number of iterations required to properly converge is higher (about 20 iterations led to accuracy similar to the centralized approach). If the number of iterations is not high enough, the power iteration leads to a PC that may not represent the data as well as the centralized approach, as is observed from the second PC for a maximum number of five iterations. Note that, given the fact that PCs are estimated on the basis of past measurements whose distribution is not exactly the same as upcoming measurements, it may also happen that approximations provide better accuracies than the centralized approach. This explains the gains in accuracy obtained around the 8-th component where the difference between subsequent eigenvalues gets very low.

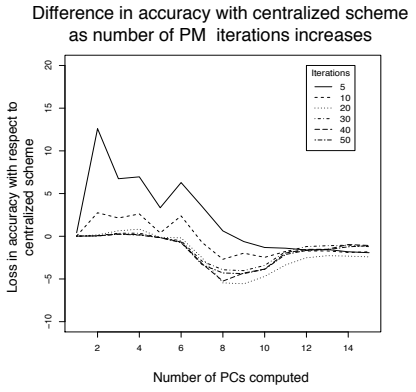

Figure 13. Comparison of the accuracy obtained using the exact and the approximated eigenvectors, for different number of iterations and principal components.

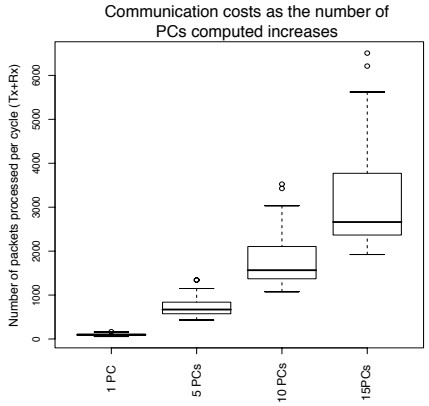

Figure 14. The network load is quadratic in the number of principal components. Radio range is 10 meters.

Additional experiments were also run to observe the consequence of the local covariance hypothesis on the non positive semi-definiteness of the matrix. We observed that negative eigenvalues could lead the algorithm to stop at stages as early as between 5 to 10 components (this was observed for radio ranges

of 30 and 40 meters). It is however important to note that despite the early stopping of the algorithm, the set of principal components identified was enough to retain more than 90% of the variance (cf. Fig. 11).

Finally, we report in Fig. 14 estimates of the communication costs entailed by the computation of the principal components. The radio range was of 10 meters, and the costs were computed following the analysis of Sections 2.1.3 and 3.5. While about two hundred packets per node suffice for the computation of the first eigenvector, this quantity can reach a significant network load of 6000 packets on average for the computation of 15 components. As discussed in Section 3.5, the main advantage of the distributed computation of the principal components lies in the distribution of the computational and memory costs, and the approach does not scale well to the computation of a large number of principal components.

4.7 Summary

The main parameter of the principal component aggregation, which determines the tradeoff between the compression accuracy and the communication costs, is the number of principal components to extract. As illustrated in Fig. 10 or 14, the network load incurred by the computation of the principal components quickly increases with their number. The PCAg scheme therefore proves useful only if a few components are required. In such a case, we emphasize that the gains can be dramatic, as was reported in Fig. 10 or 14.

The amount of information retained by a set of principal components depends on the correlation existing between sensor measurements. We illustrated the method for a compression task using a real-world temperature dataset where measurements had an degree of correlation that can be assumed to be representative of a typical sensor network scenario. An interesting result to recall is that for 5 principal components, 90% of the variance could be retained. This was shown to return a nice approximation of the original signals (Fig. 8), and to reduce the highest network load from about 100 packets per epoch to about 40 packets per epoch (Fig. 10).

Regarding the scalability, an interesting property of the PCAg is that the network load is better distributed in the network. This property first prevents congestion issues observed at the root node using the default scheme. Second, as the radio communication is a primary source of energy depletion for a wireless node, it better distributes the lifetime of the nodes in the network. The PCAg therefore provides an interesting framework for dealing with data in multi-hop networks.

5 Related work

The distribution of the PCA has been addressed in the signal processing and information theory communities [4, 37–39]. These approaches, based on distributed source coding, involve compression techniques related to Slepian-Wolf and Wyner-Ziv coding together with Karhunen-Loeve transform for compressing data in distributed systems. Each component of the system is however assumed to observe several dimensions of the problem, and they require the base station to support the task of defining the coding scheme. The network architectures and communication costs analyses are also left as open research areas. Although these approaches may lead to attractive applications for WSN, they still remain

in that respect at an early research stage.

In the fields of machine learning and data mining, distributed PCA schemes have also been addressed in [40, 41] and [19]. The former approaches aimed at combining the PCA results obtained from different sources that share the same variables (*vertically* distributed databases). The latter work provided an architecture aimed at network anomaly detection, where the PCA is first computed in a centralized manner, and a subsequent distributed and adaptive feedback mechanism is proposed. Despite the similarity in the choice for the method name, the corresponding work is however clearly different from the approach proposed in this article.

The distributed computation of eigenvectors of large and sparse matrices has been tackled from several angles in the literature, and good reviews of existing state of the art techniques are for example detailed in [21, 22]. The approach proposed in this article is however assumed to be innovative to the best of authors' knowledge, as it leverages two specific architectural properties of sensor networks. First, a wireless sensor network is an *intrinsically* distributed system where each component only captures one dimension of the system variations. Second, the communication constraints can be coupled with the local covariance hypothesis.

Recent work in the domain of link analysis and recommendation systems has led authors in [42] to propose a distributed implementation of the power iteration method, which closely matches the approach proposed in this article. Their algorithm aims at computing the principal eigenvector of the adjacency matrix of a peer-to-peer network, which leads to the ranking of the network components, in a way similar to the *page rank* algorithm. The underlying hypotheses of the distributed system are interestingly close to ours (each component has only access to one dimension the problem, and can communicate with components whose data is related). The network structure is however different, as no coordinating entity is assumed. This leads authors to rely on the harmonic mean to achieve the normalization step of the power iteration method. The computation of subsequent eigenvectors is also not addressed.

Related work on the use of basis change for sensor networks has been addressed in [43], where authors proposed the radical position of using random bases to project sensor measurements. The work is analyzed in the context of compressed sensing, an emerging field in signal processing (see also [44]). Their work has however mainly focused on the theoretical ability of random bases to retain the sensor measurements variations.

Conclusion

This article extended previous work related to the distribution of the principal component analysis by presenting an implementation suitable for wireless sensor networks. The approach relies on the hypothesis that sensor measurements collected by distant sensors are uncorrelated, and was shown to provide significant gains in terms of radio communication. Additionally, we showed that the distributed principal component analysis led to balance the network load among the sensors, making the method particularly suitable for multi-hop sensor networks. We plan to extend this work by showing that spatiotemporal aggregation and independent component analysis can also be formulated in the same framework.

Acknowledgments

This work was supported by the COMP2SYS project, sponsored by the Human Resources and Mobility program of the European Community (MEST-CT-2004-505079), and by the PIMAN project, supported by the Institute for the Encouragement of Scientific Research and Innovation of Brussels, Belgium.

References

1. Kumar, S.; Zhao, F.; Shepherd, D. Collaborative signal and information processing in microsensor networks. *Signal Processing Magazine, IEEE* **2002**, *19*(2), 13–14.
2. Intanagonwiwat, C.; Govindan, R.; Estrin, D. Directed diffusion: A scalable and robust communication paradigm for sensor networks. In *Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking*, pages 56–67, **2000**.
3. Pattem, S.; Krishnamachari, B.; Govindan, R. The impact of spatial correlation on routing with compression in wireless sensor networks. In *Proceedings of the third international symposium on Information processing in sensor networks*, pages 28–35. ACM Press, **2004**.
4. Xiong, Z.; Liveris, A.; Cheng, S. Distributed source coding for sensor networks. *Signal Processing Magazine* **2004**, *21*(5), 80–94.
5. Akyildiz, I.; Su, W.; Sankarasubramaniam, Y.; Cayirci, E. Wireless sensor networks: a survey. *Computer Networks* **2002**, *38*(4), 393–422.
6. Ilyas, M.; Mahgoub, I.; Kelly, L. *Handbook of Sensor Networks: Compact Wireless and Wired Sensing Systems*. CRC Press, **2004**.
7. Krishnamachari, L.; Estrin, D.; Wicker, S. The impact of data aggregation in wireless sensor networks. In *Proceedings of the Distributed Computing Systems Workshops*, pages 575–578, **2002**.
8. Heidemann, J.; Silva, F.; Intanagonwiwat, C.; Govindan, R.; Estrin, D.; Ganesan, D. Building efficient wireless sensor networks with low-level naming. In *Proceedings of the eighteenth ACM symposium on Operating systems principles*, pages 146–159. ACM Press, **2001**.
9. Intanagonwiwat, C.; Estrin, D.; Govindan, R.; Heidemann, J. Impact of network density on data aggregation in wireless sensor networks. In *Proceedings of the 22nd International Conference on Distributed Computing Systems.*, pages 457–458, **2002**.
10. Madden, S.; Franklin, M.; Hellerstein, J.; Hong, W. TAG: a Tiny AGgregation Service for Ad-Hoc Sensor Networks. In *Proceedings of the 5th ACM Symposium on Operating System Design and Implementation (OSDI)*, volume 36, pages 131 – 146. ACM Press, **2002**.
11. Madden, S.; Franklin, M.; Hellerstein, J.; Hong, W. TinyDB: an acquisitional query processing system for sensor networks. *ACM Transactions on Database Systems* **2005**, *30*(1), 122–173.
12. Yao, Y.; Gehrke, J. The cougar approach to in-network query processing in sensor networks. *ACM SIGMOD Record* **2002**, *31*(3), 9–18.
13. Burri, N.; Wattenhofer, R. Dozer: ultra-low power data gathering in sensor networks. In *Pro-*

ceedings of the 6th international conference on Information processing in sensor networks, pages 450–459. ACM Press, **2007**.

14. Le Borgne, Y.; Bontempi, G. Unsupervised and supervised compression with principal component analysis in wireless sensor networks. In *Proceedings of the Workshop on Knowledge Discovery from Data, 13th ACM International Conference on Knowledge Discovery and Data Mining*, pages 94–103. ACM Press, **2007**.
15. Hyvarinen, A.; Karhunen, J.; Oja, E. *Independent Component Analysis*. J. Wiley New York, **2001**.
16. Li, J.; Zhang, Y. Interactive sensor network data retrieval and management using principal components analysis transform. *Smart Materials and Structures* **December 2006**, *15*, 1747–1757(11).
17. Bontempi, G.; Le Borgne., Y. An adaptive modular approach to the mining of sensor network data. In *Proceedings of the Workshop on Data Mining in Sensor Networks, SIAM SDM*, pages 3–9. SIAM Press, **2005**.
18. Duarte, M.; Hen Hu, Y. Vehicle classification in distributed sensor networks. *Journal of Parallel and Distributed Computing* **2004**, *64*(7), 826–838.
19. Huang, L.; Nguyen, X.; Garofalakis, M.; Jordan, M.; Joseph, A.; Taft, N. In-network PCA and anomaly detection. In B. Scholkopf, J. P.; Hoffman, T., editors, *Proceedings of the 19th conference on Advances in Neural Information Processing Systems*. MIT Press, **2006**.
20. Lakhina, A.; Crovella, M.; Diot, C. Diagnosing network-wide traffic anomalies. In *Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 219–230. ACM Press, **2004**.
21. Bai, Z.; others. *Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide*. Society for Industrial and Applied Mathematics, **2000**.
22. Golub, G.; Van Loan, C. *Matrix computations*. Johns Hopkins University Press, **1996**.
23. Akkaya, K.; Younis, M. A survey on routing protocols for wireless sensor networks. *Ad Hoc Networks* **2005**, *3*(3), 325–349.
24. TinyOS. Project Website: `http://www.tinyos.net`.
25. Polastre, J.; Szewczyk, R.; Culler, D. Telos: enabling ultra-low power wireless research. In *Proceedings of the 4th International Symposium on Information Processing in Sensor Networks*, pages 364–369, **2005**.
26. Zhao, F.; Guibas, L. *Wireless Sensor Networks: An Information Processing Approach*. Morgan Kaufmann, **2004**.
27. Raghunathan, V.; Srivastava, C. Energy-Aware Wireless Microsensor Networks. *Signal Processing Magazine* **2002**, *19*(2), 40–50.
28. Jolliffe, I. *Principal Component Analysis*. Springer, **2002**.
29. Miranda, A. A.; Le Borgne, Y.-A.; Bontempi, G. New routes from minimal approximation error to principal components. *Accepted for publication in the Neural Processing Letters* **2008**.
30. Duda, R.; Hart, P.; Stork, D. *Pattern Classification*. Wiley-Interscience, **2000**.
31. Webb, A. *Statistical Pattern Recognition*. Hodder Arnold Publication, **1999**.

32. Barrenetxea, G. SensorScope: on-line urban environmental monitoring network. *Geophysical Research Abstracts* **2007**, *9*, 07501.
33. Jindal, A.; Psounis, K. Modeling spatially-correlated sensor network data. *Sensor and Ad Hoc Communications and Networks, 2004. IEEE SECON 2004. 2004 First Annual IEEE Communications Society Conference on* **2004**, pages 162–171.
34. Rousseeuw, P.; Molenberghs, G. Transformation of non positive semidefinite correlation matrices. *Communications in Statistics-Theory and Methods* **1993**, *22*(4), 965–984.
35. Intel Lab Data webpage. `http://db.csail.mit.edu/labdata/labdata.html`.
36. Wireless Sensor Lab. Machine Learning Group. University of Brussels. `http://www.ulb.ac.be/di/labo/index.html`.
37. Vosoughi, A.; Scaglione, A. Precoding and Decoding Paradigms for Distributed Data Compression. *IEEE Transactions on Signal Processing* **2007**.
38. Pradhan, S.; Ramchandran, K. Distributed source coding using syndromes (DISCUS): design and construction. *IEEE Transactions on Information Theory* **2003**, *49*(3), 626–643.
39. Gastpar, M.; Dragotti, P. L.; Vetterli, M. The Distributed Karhunen-Loève Transform. *IEEE Transactions on Information Theory* **2006**, *52*(12), 5177–5196.
40. Bai, Z.-J.; Chan, R.; Luk, F. Principal component analysis for distributed data sets with updating. In *Advanced Parallel Processing Technologies*, volume 3756, pages 471–483. Springer Berlin, Heidelberg, **2005**.
41. Kargupta, H.; Huang, W.; Sivakumar, K.; Park, B.; Wang, S. Collective Principal Component Analysis from Distributed, Heterogeneous Data. In *Proceedings of the 4th European Conference on Principles of Data Mining and Knowledge Discovery*, pages 452–457. Springer-Verlag, **2000**.
42. Jelasity, M.; Canright, G.; Engø-Monsen, K. Asynchronous Distributed Power Iteration with Gossip-Based Normalization. In *Proceedings of Euro-Par*, volume 4641 of *Lecture Notes in Computer Science*, pages 514–525. Springer, **2007**.
43. Duarte, M. F.; Sarvotham, S.; Baron, D.; Wakin, M. B.; Baraniuk, R. G. Distributed compressed sensing of jointly sparse signals. In *Proceedings of the 39th Asilomar Conference on Signals, Systems and Computation*, pages 1537–1541, **2005**.
44. Donoho, D. Compressed Sensing. *IEEE Transactions on Information Theory* **2006**, *52*(4), 1289–1306.