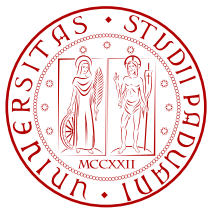


UNIVERSITÀ DI PADOVA FACOLTÀ DI INGEGNERIA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

SCUOLA DI DOTTORATO IN INGEGNERIA DELL'INFORMAZIONE, XXIII CICLO

INDIRIZZO IN SCIENZA E TECNOLOGIA DELL'INFORMAZIONE

Optimization of Cognitive Wireless Networks using Compressive Sensing and Probabilistic Graphical Models

Dottorando

GIORGIO QUER

Supervisore:

Chiar.mo Prof. Michele Zorzi

Direttore della Scuola:

Chiar.mo Prof. Matteo Bertocco

Anno Accademico 2010/2011

Dedicated to my mother and my father.

Contents

List of Figures

List of Tables

List of Acronyms

BIC	Bayesian Information Criterion
BN	Bayesian Network
CA	Cognitive Agent
CEF	Cognitive Execution Function
CogAp	Cognitive Access Point
CS	Compressive Sensing
DAG	Directed Acyclic Graph
DBN	Dynamic Bayesian Network
DCP	Data Collection Point
DCR	Data Collection Round
DCT	Discrete Cosine Transform
DOLS	Deterministic Ordinary Least Square method
IP	Internet Protocol
LAN	Local Area Network
MAP	Maximum A Posteriori
MLE	Maximum Likelihood Estimation
NC	Network Coding
PCA	Principal Component Analysis

POLS Probabilistic Ordinary Least Square method

QoS Quality of Service

RS Random Sampling

SCoRe1 Sensing Compression and Recovery through ONline Estimation

TCP Transmission Control Protocol

WSN Wireless Sensor Network

Abstract

In-network data aggregation to increase the efficiency of data gathering solutions for Wireless Sensor Networks (WSNs) is a challenging task. In the first part of this thesis, we address the problem of accurately reconstructing distributed signals through the collection of a small number of samples at a Data Collection Point (DCP). We exploit Principal Component Analysis (PCA) to learn the relevant statistical characteristics of the signals of interest at the DCP. Then, at the DCP we use this knowledge to design a matrix required by the recovery techniques, that exploit convex optimization (Compressive Sensing, CS) in order to recover the whole signal sensed by the WSN from a small number of samples gathered. In order to integrate this monitoring model in a compression/recovery framework, we apply the logic of the cognition paradigm: we first observe the network, then we learn the relevant statistics of the signals, we apply it to recover the signal and to make decisions, that we effect through the control loop. This compression/recovery framework with a feedback control loop is named "Sensing, Compression and Recovery through ONline Estimation" (SCoRe1). The whole framework is designed for a WSN architecture, called WSN-control, that is accessible from the Internet. We also analyze with a Bayesian approach the whole framework to justify theoretically the choices made in our protocol design.

The second part of the thesis deals with the application of the cognition paradigm to the optimization of a Wireless Local Area Network (WLAN). In this work, we propose an architecture for cognitive networking that can be integrated with the existing layered protocol stack. Specifically, we suggest the use of a probabilistic graphical model for modeling the layered protocol stack. In particular, we use a Bayesian Network (BN), a graphical representation of statistical relationships between random variables, in order to describe the relationships among a set of stack-wide protocol parameters and to exploit this cross-layer approach to optimize the network. In doing so, we use the knowledge learned from the observation of the data to predict the TCP throughput in a single-hop wireless network and to

infer the future occurrence of congestion at the TCP layer in a multi-hop wireless network.

The approach followed in the two main topics of this thesis consists of the following phases: (i) we apply the cognition paradigm to learn the specific probabilistic characteristics of the network, (ii) we exploit this knowledge acquired in the first phase to design novel protocol techniques, (iii) we analyze theoretically and through extensive simulation such techniques, comparing them with other state of the art techniques, and (iv) we evaluate their performance in real networking scenarios.

Sommario

La combinazione delle informazioni nelle reti di sensori wireless è una soluzione promettente per aumentare l'efficienza delle techiche di raccolta dati. Nella prima parte di questa tesi viene affrontato il problema della ricostruzione di segnali distribuiti tramite la raccolta di un piccolo numero di campioni al punto di raccolta dati (DCP). Viene sfruttato il metodo dell'analisi delle componenti principali (PCA) per ricostruire al DCP le caratteristiche statistiche del segnale di interesse. Questa informazione viene utilizzata al DCP per determinare la matrice richiesta dalle tecniche di recupero che sfruttano algoritmi di ottimizzazione convessa (Compressive Sensing, CS) per ricostruire l'intero segnale da una sua versione campionata. Per integrare questo modello di monitoraggio in un framework di compressione e recupero del segnale, viene applicata la logica del paradigma *cognitive*: prima si osserva la rete; poi dall'osservazione si derivano le statistiche di interesse, che vengono applicate per il recupero del segnale; si sfruttano queste informazioni statistiche per prenderere decisioni e infine si rendono effettive queste decisioni con un controllo in retroazione. Il framework di compressione e recupero con controllo in retroazione è chiamato "Sensing, Compression and Recovery through ONline Estimation" (SCoRe1). L'intero framework è stato implementato in una architettura per WSN detta WSN-control, accessibile da Internet. Le scelte nella progettazione del protocollo sono state giustificate da un'analisi teorica con un approccio di tipo Bayesiano.

Nella seconda parte della tesi il paradigma cognitive viene utilizzato per l'ottimizzazione di reti locali wireless (WLAN). L'architetture della rete cognitive viene integrata nello stack protocollare della rete wireless. Nello specifico, vengono utilizzati dei modelli grafici probabilistici per modellare lo stack protocollare: le relazioni probabilistiche tra alcuni parametri di diversi livelli vengono studiate con il modello delle reti Bayesiane (BN). In questo modo, è possibile utilizzare queste informazioni provenienti da diversi livelli per ottimizzare le prestazioni della rete, utilizzando un approccio di tipo cross-layer. Ad esempio, queste in-

formazioni sono utilizzate per predire il throughput a livello di trasporto in una rete wireless di tipo single-hop, o per prevedere il verificarsi di eventi di congestione in una rete wireless di tipo multi-hop.

L'approccio seguito nei due argomenti principali che compongono questa tesi è il seguente: (i) viene applicato il paradigma cognitive per ricostruire specifiche caratteristiche probabilistiche della rete, (ii) queste informazioni vengono utilizzate per progettare nuove tecniche protocollari, (iii) queste tecniche vengono analizzate teoricamente e confrontate con altre tecniche esistenti, e (iv) le prestazioni vengono simulate, confrontate con quelle di altre tecniche e valutate in scenari di rete realistici.

Chapter 1

Introduction

The area of communication and protocol design for wireless networks has been widely researched in the past decades. More recently, the interest of the research community has been focused on the optimization of wireless networks able to adapt to specific scenarios with peculiar constraints.

In this work, we exploit a set of mathematical tools, that span the literature of machine learning [1], convex optimization [2] and probabilistic graphical models [3], in the framework of cognitive networking [4] for the optimization of different kinds of wireless networks, e.g., Wireless Sensor Networks (WSNs), tactical networks, or standard Wireless Local Area Networks (WLANs). Cognitive networking deals with applying cognition to the entire protocol stack for achieving stack-wide as well as network-wide performance goals, unlike cognitive radios that apply cognition only at the physical layer. The idea is to adopt the cognition paradigm, that consists of four phases: the observation of the network (Observe), learning key network parameters from the data observed (Learn), the analysis of the information collected and its exploitation using specific techniques to make decisions (Plan and Decide), and finally effecting such decisions to optimize the network (Act). Designing a cognitive network is challenging since learning the relationships between network protocol parameters in an automated fashion is very complex.

This thesis is divided in two main topics. The former deals with a framework for the transmission of multi-dimensional environmental signals sensed by a WSN and its optimization. Here the aim is to design a simple strategy that requires the minimum computation and the minimum energy consumption among the wireless sensors, while being able to reconstruct the signals at the data collection point with a reconstruction error below a

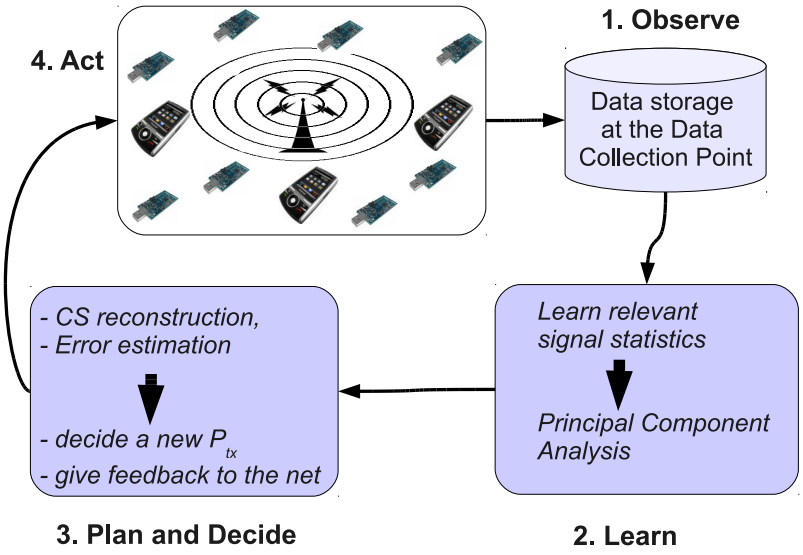


Figure 1.1. *Cognition cycle scheme applied to a monitoring scenario in a WSN.*

given threshold. With this aim, the cognitive paradigm is adopted to learn the statistics of the signals in order to fully exploit the spatial and temporal correlation characteristics of the signal. The latter topic of the thesis deals with the optimization of data transmissions in a WLAN with cognition capabilities. This optimization is performed with the same cognition paradigm, by learning of relevant network statistics and exploiting this information to optimize specific network performance.

In the following, we detail both these topics and we conclude the chapter with a description of the organization of the rest of the thesis.

1.1 Compressive Sensing (CS) in Wireless Sensor Networks (WSNs)

In the first part of this thesis, we investigate the topic of in-network aggregation and data compression to increase the efficiency of data gathering solutions, while being able to measure large amounts of data with high accuracy in a WSN for environmental monitoring. We address the problem of accurately reconstructing distributed signals through the collection of a small number of samples at a data gathering point. With this aim, we exploit the cognition paradigm, whose logical blocks are depicted in Fig. 1.1. The first phase of the cognition cycle is the observation of the signals to be gathered and stored at the Data Collection Point (DCP) for a given period of training (1. Observe phase). At the DCP, the signals gath-

ered are exploited to learn the relevant correlation statistics of the signal through Principal Component Analysis, PCA (2. Learn phase). After the initial training, the WSN starts to exploit the cognition capabilities: the sensors transmit only the minimum amount of data, the sampled version of the signal is efficiently reconstructed at the DCP by means of a convex optimization technique provided by Compressive Sensing (CS) [5–8], and the DCP also estimates the reconstruction quality and gives feedback to the WSN. If the reconstruction quality is above a certain threshold, some sensors might stop their transmissions, thus saving precious energy resources, while if the reconstruction quality is below a given threshold, all nodes should transmit their data to guarantee the required signal reconstruction quality at the DCP (3. Plan and Decide phase). Finally, the DCP should send a broadcast message to all the sensors to communicate its decision and to make it effective (4. Act phase).

In detail, the core of the signal reconstruction system is based on CS, a convex optimization technique. CS shows great promise for fully distributed compression in WSNs. In theory, CS allows the approximation of the readings from a sensor field with excellent accuracy, while collecting only a small fraction of them at a data gathering point. However, the conditions under which CS performs well are not necessarily met in practice. In order to meet these conditions, we need a learning phase to capture the relevant correlation characteristics of the signal to sense, and we use PCA to learn from the past data. With PCA we can design a transformation basis that sparsifies the signal, i.e., concentrates all the energy of the signal in only few components. A transformation with these characteristics is required for CS to retrieve, with good approximation, the original signal from a small number of samples. Our approach dynamically adapts to non-stationary real world signals through the online estimation of their correlation properties in the space and time domains. The approach is tunable and robust, independent of the specific routing protocol in use and able to substantially outperform standard data collection schemes. In particular, we propose a mathematical framework for monitoring, in which we show how it is possible to make joint use of CS and PCA in a WSN scenario, and we depict the probabilistic relations among all the variables involved through a Bayesian Network (BN). Moreover, we use a two level Bayesian analysis to analyze the statistics of the principal components of the signals gathered by existing WSNs, designing a probabilistic model that approximates the distribution of the principal components of these real signals. Then we provide empirical evidence of the effectiveness of CS in an actual WSN monitoring scenario, underlying the conditions under which the approach is optimal in the sense of Bayesian estimation (Maximum A Posteriori,

MAP). Furthermore, we integrate this mathematical framework into an actual technique, "Sensing, Compression and Recovery through ONline Estimation" (SCoRe1), that collects the data gathered by the sensor network for an initial training phase, exploits PCA for learning the structure of the signal, uses CS for recovering the sub-sampled signal through convex optimization and adopts a feedback control to bound the reconstruction error and to adapt to non stationary signals. The effectiveness of our recovery algorithm, in terms of number of transmissions in the network *vs* reconstruction error, is demonstrated for synthetic and for real world signals, gathered from the WSN deployment at the University of Padova and from other WSN testbeds whose data is available in the Internet. We stress that our solution is not limited to WSNs, but can be readily applied to other kinds of network infrastructures that require the online approximation of large and distributed data sets.

The main contributions of this part of the thesis are:

1. a preliminary study to quantify the benefits of CS in a realistic multi-hop WSN scenario when CS in used in conjunction with routing for a static signal (no temporal evolution);
2. the design of a novel mathematical framework to jointly use the statistics learned through PCA, that exploits the temporal correlation of the signals, and the recovery technique with CS;
3. a Bayesian analysis to find a suitable statistical model to approximate the statistical distribution of the principal components of real WSN signals, used to justify theoretically the mathematical framework proposed;
4. the design of a novel, effective and flexible technique for distributed sampling, data gathering and recovery of WSN signals, SCoRe1, based on the proposed mathematical framework;
5. the integration of the sampling/recovery technique into an actual WSN architecture, named WSN-Control;
6. a performance comparison of different data collection techniques that exploit the proposed framework;
7. a performance comparison of data fitting techniques, with real signals in realistic WSN scenarios.

Note: the study on the application of CS and PCA in a WSN scenario was developed in collaboration with Riccardo Masiero, so part of the material may be also found in his thesis [9], although analyzed under a different perspective.

1.2 Bayesian Networks for Cognitive Control

The second main contribution of this thesis deals with the application of cognition to the entire network protocol stack for the optimization of different metrics in a WLAN. Such approach is intentionally left as general as possible, so it can be easily applied to different kinds of network, like, e.g., Tactical Networks, Ad-Hoc networks or standard WLAN. In our concept of cognitive network, all networking elements track the spatial, temporal, and spectral dynamics of their own behavior and the behaviors associated with the environment. The information so gathered is used to learn, plan and act in a way that meets network or application requirements.

In this work, we propose an architecture for cognitive networking that can be integrated with the existing layered protocol stack and that exploits new and hitherto unused tools from artificial intelligence. Specifically, we suggest the use of a probabilistic graphical model for modeling the layered protocol stack. In particular, we use Bayesian Network (BN), a graphical representation of statistical relationships between random variables, widely used for statistical inference and machine learning. A graphical model is represented by a graph structure consisting of vertices connected by directional edges that represent conditional probability distributions. The structure of the model consists of the specification of a set of conditional independence relations for the probability model, represented by a set of missing edges in the graph. Conditional probabilities are used to capture the statistical dependences between variables. The joint probability distribution of a set of random variables can be easily obtained using the chain rule [3].

In order to exploit the favorable properties of this mathematical tool in a wireless network scenario, we adopt again the four phases of the cognition cycle. In detail, we start observing at discrete time intervals a set of network parameters from different layers of the protocol stack (Observe phase). At this point we exploit the BN learning tools to infer a suitable structure of probabilistic relations that connects these parameters (Learn phase). With this structure, we design an inference engine and make inference on present or future values of some parameters, and we can also infer the optimal value of a controllable parameter of

interest (Plan and Decide phase). With the opportunities given by the exploitation of this inference engine, we can act on the network to tune an appropriate set of controllable parameters, in order to optimize the performance or to avoid an undesirable behavior of the network, e.g., the occurrence of congestion at the transport layer (TCP).

The main contributions of this part of the thesis are:

1. the integration of BN into our Cognitive Network framework for wireless networks;
2. the application of BN to study network parameters in realistic wireless LAN scenarios (single-hop and multi-hop);
3. a performance analysis of the BN inference engine's accuracy to infer the value of TCP throughput in a single-hop wireless network scenario;
4. a performance analysis of the BN inference engine accuracy to predict TCP's congestion status in a realistic multi-hop wireless network scenario.

1.3 Organization of the Thesis

The rest of this thesis work is organized as follows:

Chapter 2. introduces the mathematical tools that are used in the thesis, specifically, CS, PCA, BN and the Structure Learning (SL) and Parameter Learning (PL) tools to learn a BN from the observation of the data;

Chapter 3. presents some preliminary results on the possibility to apply CS in a multi-hop WSN scenario in conjunction with routing; this chapter does not deal with the time evolution of signals;

Chapter 4. deals with the mathematical framework for monitoring that is the core of the data compression/recovery technique; it analyzes the learning phase of the cognition cycle to derive the relevant statistical characteristics of WSN signals and derives the conditions for optimality (under a Bayesian perspective) of such an approach;

Chapter 5. proposes the compression/recovery techniques that exploit the mathematical framework for monitoring; different data collection techniques that use this framework are proposed and compared; moreover, the chapter presents a comparison of dif-

ferent data fitting techniques, including CS, and shows how the compression/recovery techniques can be integrated into an existing WSN architecture;

Chapter 6. deals with the use of BNs for learning the statistical relationships among WLAN stack-wide network parameters, and shows how to apply this knowledge for the TCP-throughput inference in a single-hop network and for the congestion prediction in a multi-hop network.

Chapter 7. concludes the thesis and discusses some directions for future extensions of the research material presented.

Chapter 2

Mathematical Preliminaries

In this chapter [1] we briefly overview the mathematical tools that are used in the thesis. Section 2.1 is dedicated to the mathematical techniques that are used for the compression and recovery framework in Wireless Sensor Networks (WSN). These techniques are exploited in Chapters 4 and 5. Section 2.2.1 describes two probabilistic graphical models [3], i.e., Bayesian Networks (BNs) and Dynamic Bayesian Networks (DBNs), that are exploited to describe the conditional independence relations among the parameters of a WLAN. Moreover, these tools are also exploited to make inference on present and future values of some parameters of interest, with the aim of enhancing the network performance, like e.g., avoiding TCP congestions.

2.1 Compressive Sensing (CS)

Compressive Sensing (CS) [5–7] is a novel data compression technique that uses convex optimization methods [2]. CS exploits the inherent correlation in some input data set

[1] The material presented in this chapter has been published in:

[C1] **G. Quer**, H. Meenakshisundaram, B.R. Tamma, B.S. Manoj, R. Rao and M. Zorzi, "Cognitive Network Inference through Bayesian Network Analysis", *IEEE Globecom 2010*, Miami, FL, Dec. 2010.

[C2] **G. Quer**, D. Zordan, R. Masiero, M. Zorzi and M. Rossi, "WSN-Control: Signal Reconstruction through Compressive Sensing in Wireless Sensor Networks", *IEEE LCN (SenseApp Workshop)*, Denver, CO, Oct. 2010.

[C3] **G. Quer**, H. Meenakshisundaram, B.R. Tamma, B.S. Manoj, R. Rao and M. Zorzi, "Using Bayesian Networks for Cognitive Control of Multi-hop Wireless Networks", *MILCOM 2010*, San Jose, CA, Nov. 2010.

$\mathbf{x}$ to compress such data by means of quasi-random matrices. If the compression matrix and the original data $\mathbf{x}$ have certain properties, $\mathbf{x}$ can be reconstructed from its compressed version Y, with high probability, by minimizing a distance metric over a solution space. CS was originally developed for the efficient storage and compression of digital images, which show high spatial correlation. Recently, there has been a growing interest in these techniques by the telecommunications and signal processing communities [10]. In contrast to classical approaches, where the data is first compressed and then transmitted to a given destination, with CS the compression phase can be jointly executed with data transmission. This is important for WSNs as compressing the data before the transmission to the Data Collection Point (DCP) requires to know in advance the correlation properties of the input signal over the entire network [11] (or over a large part of it) and this implies high transmission costs. With CS, the content of packets can be mixed as they are routed towards the DCP. Under certain conditions, CS allows to reconstruct all sensor readings of the network using much fewer transmissions than routing or aggregation schemes. These characteristics make CS very promising for jointly acquiring and aggregating data from distributed devices in a multi-hop wireless sensor network [10].

In Section 2.1.1 we start describing CS, the convex optimization techniques that will be exploited for the recovery of the previously compressed signal sensed by a WSN. In this section we also recall the key concepts of the Nesterov algorithm to efficiently solve the convex optimization problem in CS [12]. In Section 2.1.3 we detail how to obtain the transformation basis required by CS, that we will use to obtain a sparse representation of the signal. We learn this basis exploiting the spatial and temporal correlation of the signal sensed, through Principal Component Analysis (PCA) [13].

2.1.1 CS in WSNs

CS is exploited to perform distributed compression of an N-dimensional signal and centralized recovery of the signal at the server. We represent the signal as a column vector $\mathbf{x}^{(k)} \in \mathbb{R}^N$, where each element of the vector corresponds to the value measured by one of the N sensors, collected according to a fixed sampling rate at discrete times $k = 1, 2, \ldots, K$. The signal should be recovered at the server from an ideally small number of random projections of $\mathbf{x}^{(k)}$, namely $\mathbf{y}^{(k)} \in \mathbb{R}^L$ with $L \leq N$, according to the equation:

$$\mathbf{y}^{(k)} = \mathbf{\Phi}^{(k)}\mathbf{x}^{(k)} \ . \tag{2.1}$$

In our framework $\mathbf{\Phi}^{(k)}$ captures the way in which the sensor data is gathered at the DCP, i.e., $\mathbf{\Phi}^{(k)}$ is an $L \times N$ sampling matrix with a one in each row and at most a single one in each column, so vector $\mathbf{y}^{(k)}$ becomes a subsampled version of $\mathbf{x}^{(k)}$. In order to recover the original sensed signal $\mathbf{x}^{(k)}$ from $\mathbf{y}^{(k)}$, we assume for the moment that there exists an invertible $N \times N$ sparsifying matrix $\mathbf{\Psi}$ such that

$$\mathbf{x}^{(k)} = \mathbf{\Psi}\mathbf{s}^{(k)} \,, \tag{2.2}$$

where $\mathbf{s}^{(k)} \in \mathbb{R}^N$ and $\mathbf{s}^{(k)}$ is M-sparse with $M \leq L$, i.e. it has only M significant components, while the other $N - M$ are negligible with respect to the average energy per component, defined as:

$$E_s^{(k)} = \frac{\sqrt{\left\langle \mathbf{s}^{(k)}, \mathbf{s}^{(k)} \right\rangle}}{N} \,, \tag{2.3}$$

where for any two column vectors $\mathbf{a}$ and $\mathbf{b}$ of the same length, we define $\langle \mathbf{a}, \mathbf{b} \rangle = \mathbf{a}^T\mathbf{b}$. At the receiver, it is equivalent to calculate a good approximation of either of the two vectors $\mathbf{x}^{(k)}$ or $\mathbf{s}^{(k)}$, as due to (2.2) there is a one-to-one mapping between them. Using (2.1) and (2.2) we can write

$$\mathbf{y}^{(k)} = \mathbf{\Phi}^{(k)}\mathbf{x}^{(k)} = \mathbf{\Phi}^{(k)}\mathbf{\Psi}\mathbf{s}^{(k)} = \mathbf{\Theta}^{(k)}\mathbf{s}^{(k)} \,, \tag{2.4}$$

that is in general an ill-posed and ill-conditioned system with $\mathbf{\Theta}^{(k)} = \mathbf{\Phi}^{(k)}\mathbf{\Psi}$ of dimensions $L \times N$, since the number of variables N is larger than the number of equations L and a small variation in the input signal can cause a large variation in the output. However, since we design the matrix $\mathbf{\Psi}$ such that $\mathbf{s}^{(k)}$ is a sparse vector, as explained in Section 2.1.3, we can invert the system and find the optimal solution with high probability solving a convex optimization problem [6], as described in Section 2.1.2.

2.1.2 Convex Optimizer

At time k, in order to reconstruct the original signal $\mathbf{x}^{(k)}$ at the receiver we must invert the ill-posed system defined by Eq. (2.4), where $\mathbf{\Psi}$ is obtained as detailed in the previous section. For simplicity of the notation, we hereby assume that $\overline{\mathbf{x}} = 0$, as this only counts as an additional term. Moreover, under the assumption that $\mathbf{s}^{(k)}$ has a certain degree of sparsity and under specific assumptions on the matrix $\mathbf{\Theta}^{(k)}$ (that are verified in our case, see, e.g., [14]), inverting (2.4) has been proven [7] to be equivalent to solving the convex

0. Initialize $\mathbf{x}_0$.

For $t \geq 0$,

1. Compute $\nabla f(\mathbf{x}_t)$.

2. Compute $\mathbf{r}_{t+1}$:

$$\mathbf{r}_{t+1} = \underset{\mathbf{x}\in\mathcal{Q}_p}{\operatorname{argmin}} \left\{ p(\mathbf{x}, \mathbf{x}_t) + \langle \nabla f(\mathbf{x}_t), \mathbf{x} - \mathbf{x}_t \rangle \right\}.$$

3. Compute $\mathbf{z}_{t+1}$:

$$\mathbf{z}_{t+1} = \underset{\mathbf{x}\in\mathcal{Q}_p}{\operatorname{argmin}} \left\{ p(\mathbf{x}, \mathbf{x}_0) + \sum_{i=0}^{t} \alpha_i \langle \nabla f(\mathbf{x}_i), \mathbf{x} - \mathbf{x}_i \rangle \right\}.$$

4. Update $\mathbf{x}_{t+1}$:

$$\mathbf{x}_{t+1} = \tau_t \mathbf{z}_{t+1} + (1 - \tau_t)\mathbf{r}_{t+1} \ .$$

5. Stop if given criterion is satisfied.

Table 2.1. *The Nesterov minimization algorithm for smooth functions.*

minimization problem

$$\begin{aligned} \hat{\mathbf{s}}^{(k)} &= \underset{\mathbf{s}^{(k)}}{\operatorname{argmin}} \|\mathbf{s}^{(k)}\|_{\ell_1} \\ \text{s.t.} \quad \mathbf{y}^{(k)} &= \boldsymbol{\Theta}^{(k)}\mathbf{s}^{(k)} \ , \end{aligned} \tag{2.5}$$

where $\|\cdot\|_{\ell_1}$ is the ℓ_1-norm of a vector, i.e., for a given vector $\mathbf{a}$ of N elements, $\|\mathbf{a}\|_{\ell_1} = \sum_{i=1}^{N} |a_i|$. In WSN-Control, as suggested in [12], this optimization problem is solved through NESTA, which is an application to CS of the Nesterov minimization algorithm extended to non-smooth functions. As a first step, in the following we review the Nesterov minimization method [15]. Subsequently, we discuss the extension of this method to non-smooth functions and finally we explain how it is applied to CS.

Nesterov minimization: this method solves convex optimization problems of the type

$$\min_{\mathbf{x}\in\mathcal{Q}_p} f(\mathbf{x}) \ , \tag{2.6}$$

where the convex function to minimize, $f(\mathbf{x}) : \mathcal{Q}_p \to \mathbb{R}$, is defined in the convex set $\mathcal{Q}_p \subseteq \mathbb{R}^N$, e.g., of the form

$$\mathcal{Q}_p = \{\mathbf{x} : \mathbf{b} = \mathbf{Q}\mathbf{x}\} \ , \tag{2.7}$$

where $\mathbf{Q}$ is an $M \times N$ matrix, with $M \leq N$, and $\mathbf{b} \in \mathbb{R}^M$ is a given constant vector. Moreover, the function $f(\mathbf{x})$ must be smooth, i.e., it must be differentiable and its gradient must be Lipschitz:

$$\|\nabla f(\mathbf{x}) - \nabla f(\mathbf{y})\|_{\ell_2} \leq C\|\mathbf{x} - \mathbf{y}\|_{\ell_2} \ , \tag{2.8}$$

where $C > 0$ is a constant [15]. The algorithm proposed by Nesterov to solve (2.6) is listed in Table 2.1 and discussed in the following:

0. Initialize $\mathbf{x}_0$: a possible initialization method for $\mathbf{x}_0$ is $\mathbf{x}_0 = \mathbf{Q}^T\mathbf{b}$. Set $t = 0$.

1. Computation of the gradient of $f(\mathbf{x}_t)$.

2. Computation of $\mathbf{r}_{t+1}$: $\mathbf{r}_{t+1}$ is a first sequence of vectors that converges towards the minimum of $f(\mathbf{x})$. The first term $p(\mathbf{x}, \mathbf{x}_t)$ is a proximity function (also referred to as *penalty function*) weighing more those points that are farther away from the current solution $\mathbf{x}_t$. We have

$$p(\mathbf{x}, \mathbf{x}_t) = \frac{C}{2}\|\mathbf{x} - \mathbf{x}_t\|^2_{\ell_2} \ . \tag{2.9}$$

   The second term corresponds to a gradient descent minimization with step $|\mathbf{x} - \mathbf{x}_t|$. Note that the step size is controlled by the first term, which penalizes large deviations from $\mathbf{x}_t$.

3. Computation of $\mathbf{z}_{t+1}$: $\mathbf{z}_{t+1}$ is a second sequence of vectors that also converges to the minimum of $f(\mathbf{x})$. The first term is equal to (2.9) but with $\mathbf{x}_0$ in place of $\mathbf{x}_t$. The second term corresponds to a gradient descent minimization accounting for all previous partial solutions $\mathbf{x}_i$, $i \leq t$.

4. The solution is updated as a weighted average of $\mathbf{r}_t$ and $\mathbf{z}_t$, using a suitable combination coefficient τ_t.

5. A possible stopping criterion, originally proposed in [12], is the following. Let $\overline{f}(\cdot)$ be the average of $f(\cdot)$ during the last ten iterations

$$\overline{f}(\mathbf{x}_t) = \frac{1}{\min\{10, t\}} \sum_{i=1}^{\min\{10,t\}} f(\mathbf{x}_{t-i}) \ . \tag{2.10}$$

The algorithm is terminated when

$$\Delta f = \frac{|f(\mathbf{x}_t) - \overline{f}(\mathbf{x}_t)|}{\overline{f}(\mathbf{x}_t)} < \delta \; . \tag{2.11}$$

The coefficients α_t, τ_t must be chosen to guarantee convergence, see [16]. The impact of the constant δ is studied in the Appendix A.1.

Application of Nesterov minimization to CS: reference [16] extended the Nesterov algorithms to *non-smooth* functions, showing that this extension is possible when these functions can be re-written as a maximization problem. Subsequently, with the NESTA algorithm [12], the theory of [16] has been applied to CS. In detail, (2.5) is re-written as

$$\min_{\mathbf{s}^{(k)} \in \mathcal{Q}'_p} \|\mathbf{s}^{(k)}\|_{\ell_1} \; , \tag{2.12}$$

where $\mathcal{Q}'_p$, at time k, is the convex set defined as

$$\mathcal{Q}'_p = \left\{ \mathbf{s}^{(k)} : \|\mathbf{y}^{(k)} - \mathbf{\Theta}^{(k)}\mathbf{s}^{(k)}\|_{\ell_2} \leq \epsilon \right\} , \tag{2.13}$$

where $\mathbf{s}^{(k)} \in \mathbb{R}^N$ is a sparse vector with only M significant elements with $M \ll N$, $\epsilon \geq 0$ is a small number and $\mathbf{\Theta}^{(k)}$ is an $L \times N$ and real matrix having linearly independent rows ($M \leq L \leq N$). In [12], $\|\mathbf{s}^{(k)}\|_{\ell_1}$ is re-written as a maximization problem, i.e.,

$$\|\mathbf{s}^{(k)}\|_{\ell_1} = \max_{\mathbf{u} \in Q_d} \langle \mathbf{u}, \mathbf{s}^{(k)} \rangle \; , \tag{2.14}$$

where $Q_d \subseteq \mathbb{R}^N$ is the unit sphere defined as

$$Q_d = \{ \mathbf{u} : \|\mathbf{u}\|_\infty \leq 1 \} \; . \tag{2.15}$$

Hence, $\|\mathbf{s}^{(k)}\|_{\ell_1}$ is approximated by the smooth function

$$\|\mathbf{s}^{(k)}\|_{\ell_1} \simeq f_\mu(\mathbf{s}^{(k)}) = \max_{\mathbf{u} \in Q_d} \left\{ \langle \mathbf{u}, \mathbf{s}^{(k)} \rangle - \frac{\mu}{2} \|\mathbf{u}\|^2_{\ell_2} \right\} . \tag{2.16}$$

It can be shown that $\nabla f_\mu(\mathbf{s}^{(k)})$ is Lipschitz with constant $C = 1/\mu$ and thus the Nesterov optimization algorithm can be applied to such function. In conclusion, the NESTA method of [12] amounts to solving

$$\min_{\mathbf{s}^{(k)} \in \mathcal{Q}'_p} \max_{\mathbf{u} \in Q_d} \left\{ \langle \mathbf{u}, \mathbf{s}^{(k)} \rangle - \frac{\mu}{2} \|\mathbf{u}\|^2_{\ell_2} \right\} . \tag{2.17}$$

Note that (2.17) can now be solved using the algorithm in Table 2.1, where the inner maximization problem (2.16) can be solved in linear time through the sequential evaluation of the elements of $\mathbf{u}$. In fact, defining $\hat{\mathbf{u}}$ as

$$\hat{\mathbf{u}} = \operatorname*{argmax}_{\mathbf{u} \in Q_d} \left\{ \langle \mathbf{u}, \mathbf{s}^{(k)} \rangle - \frac{\mu}{2} \|\mathbf{u}\|^2_{\ell_2} \right\} \tag{2.18}$$

we have

$$\hat{u}_i = \begin{cases} s_i^{(k)}/\mu & |s_i^{(k)}| \leq \mu \\ +1 & |s_i^{(k)}| > \mu \text{ and } s_i^{(k)} > 0 \\ -1 & |s_i^{(k)}| > \mu \text{ and } s_i^{(k)} < 0 \end{cases} , \; i = 1, \ldots, N , \tag{2.19}$$

so the maximum $f_\mu(\mathbf{s}^{(k)}) = \mathbf{z}$ of (2.16) becomes

$$z_i = \begin{cases} {s_i^{(k)}}^2/2\mu & |s_i^{(k)}| \leq \mu \\ |s_i^{(k)}| - \frac{\mu}{2} & |s_i^{(k)}| > \mu \end{cases} , \; i = 1, \ldots, N . \tag{2.20}$$

2.1.3 Principal Component Analysis (PCA)

In standard CS [7], the sparsifying basis $\mathbf{\Psi}$ is assumed to be given and fixed with time, but this is not the case for a realistic WSN scenario, where the signal of interest, $\mathbf{x}^{(k)}$, is unknown and its statistical characteristics can vary with time. To face this problem we advocate the use of Principal Component Analysis (PCA) [13], which is based on the Karhunen-Loève expansion, that is a method to represent the best M-terms approximation of a given N-dimensional signal, with $M < N$, exploiting the knowledge of the correlation structure of the signal. Since we do not have perfect knowledge of the correlation structure of the signal in a WSN monitoring application, we can have a good approximation through PCA, which is based on estimating the covariance matrix of the signal of interest $\mathbf{x}^{(k)}$. We assume to collect measurements of the signal at discrete times $k = 1, \ldots, K$, i.e., $\mathcal{T} = \{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(K)}\}$, and from these measurements we can approximate the mean vector $\overline{\mathbf{x}}$ as:

$$\overline{\mathbf{x}} = \frac{1}{K} \sum_{k=1}^{K} \mathbf{x}^{(k)} , \tag{2.21}$$

and the covariance matrix $\widehat{\mathbf{\Sigma}}$ as:

$$\widehat{\mathbf{\Sigma}} = \frac{1}{K} \sum_{k=1}^{K} (\mathbf{x}^{(k)} - \overline{\mathbf{x}})(\mathbf{x}^{(k)} - \overline{\mathbf{x}})^T . \tag{2.22}$$

Given the above equations, we consider the orthonormal matrix $\mathbf{U}$ whose columns are the eigenvectors of the covariance matrix $\widehat{\mathbf{\Sigma}}$, placed in decreasing order with respect to the corresponding eigenvalues. If we define the vector $\mathbf{s}^{(k)}$ as:

$$\mathbf{s}^{(k)} \stackrel{def}{=} \mathbf{U}^T (\mathbf{x}^{(k)} - \overline{\mathbf{x}}) , \tag{2.23}$$

by construction we have that the entries of vector $\mathbf{s}^{(k)}$ are in decreasing order, i.e., $s_1^{(k)} \geq s_2^{(k)} \geq \cdots \geq s_N^{(k)}$. Assuming that the instances $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(K)}$ of the process $\mathbf{x}$ are correlated, as is often the case in WSN monitoring applications, there exists an $M \leq N$ such

that all the component $s_i^{(k)}$ with $i = M+1, \ldots, N$ are negligible with respect to the average energy defined in (2.3), where the actual value of M depends on the spatial correlation of the signal. According to (2.23) we can write

$$\mathbf{x}^{(k)} = \overline{\mathbf{x}} + \mathbf{\Psi}\mathbf{s}^{(k)} \,, \tag{2.24}$$

where we have defined the sparsifying matrix $\mathbf{\Psi} = \mathbf{U}$. The N-dimensional vector $\mathbf{s}^{(k)}$ obtained through PCA turns out to be M-sparse, so it can be efficiently recovered with CS, solving a convex optimization problem as detailed in the previous section.

In Section 4.1, we will use PCA to calculate at each time k an appropriate matrix $\mathbf{\Psi}$ in order to map the signal $\mathbf{x}^{(k)}$ into a sparse vector $\mathbf{s}^{(k)}$. Since the matrix changes over time, and it is based on the signal set $\mathcal{T}_K^{(k)} = \left\{\mathbf{x}^{(k-1)}, \mathbf{x}^{(k-2)}, \cdots, \mathbf{x}^{(k-K)}\right\}$, we specify the time dependences also in the sparsification matrix, that is named $\mathbf{\Psi}^{(k)}$. This notation will be used in Chapter 4 and in Chapter 5.

2.2 Bayesian Networks (BNs)

In this section we briefly describe an interesting tool from probabilistic graphical modeling [3], i.e., Bayesian Network (BN). In the following we give a definition of a BN and present some interesting properties about this graphical model. Then in Sections 2.2.2 and 2.2.3 we detail the techniques to learn such model from the observation of the data, i.e., Structure Learning and Parameter Learning, respectively. In Section 2.2.4 we describe a method to exploit a BN to make inference on the relevant parameters' values. Finally in Section 2.2.5 we propose an extension of the BN model, the DBN model [3], that takes into account also the temporal probabilistic relations among the parameters.

2.2.1 BN for Wireless Network optimization

A BN $\mathcal{B}_0$ is a graphical model for representing the conditional independence relations among a set of random variables through a Directed Acyclic Graph (DAG). This graph will be used to efficiently compute the marginal and conditional probabilities that are required for inference. A node in the DAG represents a random variable, while an arrow that connects two nodes represents a direct probabilistic relation between the two corresponding variables. Node i, representing the random variable x_i, is a parent of node h if there exists a direct arc from i to h and we write $i \in \text{pa}_h$, where pa_h is the set of parents of node h.

From the graph it is always possible to determine the conditional independence between two variables, applying a set of rules known as *d-separation* rules, e.g., see [1], [17] for a detailed description about BN properties.

In this section we assume to have M discrete variables, $x_1, \ldots, x_M$, with unknown dependence relations, and we write the realization of the variable x_i at time k as $x_i^{(k)}$. We assume to deal with a complete dataset, i.e., the collection of M column vectors of length N, where N is the number of the independent realizations of the variables at time samples that for simplicity we indicate as $k = 1, \ldots, N$. In other words we know the values of all the elements $x_i^{(k)} \in \mathcal{D}_{N,M}$, with $i \in \{1, \ldots, M\}$ and $k \in \{1, \ldots, N\}$. The first step for designing the BN is to learn from the dataset the *qualitative* relations between the variables and their conditional independences, in order to represent them in a DAG, as shown in Section 2.2.2. The second step, given the DAG, is to infer the *quantitative* relations in order to obtain a complete probabilistic structure that describes the variables of interest, as shown in Section 2.2.3. With the procedure described in this section we obtain a BN $\mathcal{B}_0$ that describes the probabilistic relationships among the variables, but does not take into account any temporal correlation among them, so we call such model a Static BN (SBN), to differentiate it from the Dynamic BN (DBN) described in Section 2.2.5.

2.2.2 BN Structure Learning

Structure learning is the procedure to define the DAG that represents the quantitative relation between the random variables. In the literature there are two main methods to design such DAG. The first is the constraint based method [17], in which a set of conditional independence statements is established based on some a priori knowledge or on some calculation from the data and this set of statements is used to design the DAG following the rules of d-separation. The second method is the score based method [18], commonly used in the absence of a set of given conditional independence statements. This method is able to infer a suboptimal DAG from a sufficiently large data set $\mathcal{D}_{N,M}$, and consists of two parts:

1. a function to score each DAG based on how accurately it represents the probabilistic relations between the variables based on the dataset $\mathcal{D}_{N,M}$;

2. a search procedure to select the DAGs to be scored within the set of all possible DAGs.

The latter is necessary since it is not computationally tractable to score all the possible DAGs given a set of M random variables and then choose the one with highest score. An exhaus-

tive enumeration of all structures is intractable for a large value of the number of nodes M, as the total number of possible DAGs, $f(M)$, increases super exponentially with M:

$$f(M) = \sum_{i=1}^{M} (-1)^{i+1} \frac{M!}{(M-i)!i!} 2^{i(M-i)} f(M-i) \ . \tag{2.25}$$

For this reason, it is necessary to define a search procedure that selects a small and possibly representative subset of the space of all DAGs. There exist plenty of search strategies based on heuristics consisting in different methods to explore the search space and look for a local maximum of the score function. In general these heuristics do not provide any guarantee of finding a global maximum for the score function on the space of possible DAGs, as highlighted in [3]. In our specific case, we do not deal with a large number of variables, so the problem can be addressed with a simple heuristic that will be described more in detail in Section 6.3.2. The former step to infer the DAG from the dataset, the score function, should be computationally tractable and should balance the accuracy and the complexity of the structure, i.e., the number of arrows in the graph. It is worth to highlight that a complete DAG, with all nodes connected with each other directly, can describe all possible probabilistic relations among the nodes and that it is the absence of an arrow that brings information in the form of conditional independence, discriminating within all the possible probabilistic relations among the variables. In this work we have chosen the Bayesian Information Criterion (BIC) as a score function. BIC is easy to compute and is based on the maximum likelihood criterion, i.e., how well the data suits a given structure, and penalizes DAGs with a higher number of edges. In general it is defined as [19]:

$$\text{BIC}(S|\mathcal{D}_{N,M}) = \log_2 P(\mathcal{D}_{N,M}|S, \hat{\theta}_S) - \frac{\text{size}(S)}{2} \log_2(N) \ , \tag{2.26}$$

where S is the DAG to be scored, $\mathcal{D}_{N,M}$ is the dataset, θ_S is the maximum likelihood estimation of the parameters of S and N is the number of realizations for each variable in the dataset. In the case in which all the variables are multinomial, with a finite set of outcomes r_i for each variable x_i, we define q_i as the number of configurations over the parents of x_i in S, i.e., the number of different combinations of outcomes for the parents of x_i . We define also N_{ijk} as the number of outcomes of type k in the dataset $\mathcal{D}_{N,M}$) for the variable x_i, with parent configuration of type j and N_{ij} as the total number of realizations of variable x_i in $\mathcal{D}_{N,M}$) with parent configuration j. Given these definitions, it is possible to rewrite the BIC

for multinomial variables as [18]:

$$\text{BIC}(S|\mathcal{D}_{N,M}) = \sum_{i=1}^{M}\sum_{j=1}^{q_i}\sum_{k=1}^{r_i}\log_2\left(\frac{N_{ijk}}{N_{ij}}\right) - \frac{\log_2 N}{2}\sum_{i=1}^{M} q_i(r_i-1)\ , \tag{2.27}$$

which is a computationally tractable function that reduces the scoring function to a counting problem. For a detailed comparison of scoring functions for structure learning, please refer to [20].

2.2.3 BN Parameter Learning

Parameter learning is a phase of the learning procedure that consists in estimating the best set of parameters for the probability structure that connects the variables considered in the DAG. According to the definition of BN, each variable is directly conditioned only by its parents, so the estimation of the parameters for each variable x_i should be performed only conditioned on the set of its parents pa_i in the chosen DAG. There are two main methods, both asymptotically equivalent, consistent and suitable to be implemented in an on-line manner, given a sufficient size of the dataset $\mathcal{D}_{N,M}$, namely Maximum Likelihood Estimation (MLE) and Bayesian estimation given Dirichlet priors [21]. In this work we choose MLE for parameter learning, coherently with the choice of BIC as a scoring function for the structure learning algorithm. In the case of multinomial variables, we can write with an abuse of notation[2] the ML estimation as:

$$\hat{\theta}_{x_i=k|\text{pa}_i=j} = \frac{N_{ijk}}{N_{ij}}\ , \tag{2.28}$$

where the estimated value $\hat{\theta}_{x_i=k|\text{pa}_i=j}$ gives an approximation of the posterior distribution of x_i given the evidence $\text{pa}_i = j$, i.e.:

$$\hat{\theta}_{x_i=k|\text{pa}_i=j} \simeq P\left[x_i = k|\text{pa}_i = j\right]\ , \tag{2.29}$$

2.2.4 BN Inference

After performing the structure learning and parameter learning phases on the dataset $\mathcal{D}_{N,M}$, we obtain a complete BN that represents the probabilistic relations between the variables x_i, with $i \in \{1,\dots,M\}$. At this point we can use the BN to compute marginal and conditional probabilities on the variables x_i in order to make inference. Basically, we use the

[2] With the notation $\text{pa}_i = j$ we mean that the parents of node i are in the configuration of type j.

probabilities learned with the MLE method in Section 2.2.3 to design an inference engine, that is able to calculate the most probable value for the variable of interest. We explain more clearly this procedure with an example. Given a singly connected[3] linear BN with $M = 4$, with connections $1 \to 2 \to 3 \to 4$, we can apply *belief propagation* [22] for inference, according to the d-separation rules. We observe at time k the evidence $x_1 = x_1^{(k)}$ and $x_2 = x_2^{(k)}$ and we want to infer the realization of x_4. The belief propagation algorithm updates the marginal probabilities of all the variables in the network based on the given evidence, yielding as output the probabilities:

$$P\left[x_4 = x_4^{(k)} | x_1, x_2\right] = P\left[x_4 = x_4^{(k)} | x_2\right] \ . \tag{2.30}$$

These probabilities are exploited to calculate the expected value for the variable of interest, $E[x_4]$. In Chapter 6 the inference engine is used to make predictions for some network parameters in the Cognitive Network framework.

2.2.5 Dynamic Bayesian Networks (DBNs)

The weak point of the static BN representation $\mathcal{B}_0$, described in Section 2.2.1, is that it considers each realization as independent in time of the previous or the following realizations. An alternative representation is described in the following. Each variable in the dataset can be viewed as a stochastic process $\mathbf{x}_i$, with $i = 1, \ldots, M$, that is a collection of random variables indexed by a discrete time index $k = 1, \ldots, N$. The set of all stochastic processes is

$$\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_M\} \ , \tag{2.31}$$

and for each time instant k and for each stochastic process i we define the random variable $x_i^{(k)}$, that gives the value assumed by the stochastic process $\mathbf{x}_i$ at time k. In Section 2.2.1 we have assumed that the realizations of the processes are independent in time, so for each $i = 1, \ldots, M$ the random variables $x_i^{(k)}$ and $x_i^{(k+1)}$ are independent, and we write $x_i^{(k)} \perp x_i^{(k+1)}$. Given this assumption, the stochastic process $\mathbf{x}_i$ becomes a collection of independent and identically distributed (iid) random variables $x_i^{(k)}$, with $k = 1, \ldots, N$. Given the set of all the stochastic processes in (2.31), we define the set of the random variables at time k as:

$$\mathcal{X}^{(k)} = \{x_1^{(k)}, x_2^{(k)}, \ldots, x_M^{(k)}\} \ . \tag{2.32}$$

[3] A singly connected network is a network in which the undirected graph has no loops.

In the static case we assume that $\mathcal{X}^{(k)} \perp \mathcal{X}^{(j)}$, with $k \neq j$. This is an unrealistic assumption, since we deal with a set of processes that evolve in time. Indeed, the realizations $x_i^{(k)}$ and $x_i^{(k+1)}$ are, in general, highly correlated, and we should exploit this correlation to have a better statistical characterization of the dependence relations among the processes.

A more accurate model should be able to capture also the temporal probabilistic evolution of the processes considered, describing the temporal connections among the variables $x_i^{(k)}$, as well as the temporal connections among the processes. In order to capture these relations, we use a temporal BN, also known as DBN [3]. The total number of random variables involved in this model is in principle $N \times M$, but since the complexity increases super exponentially with the number of variables [17], we should make some assumptions to be able to deal with this number of variables and to derive a compact model to represent the temporal evolution of the processes. In particular, we make:

1. the Markov assumption, that is

$$P\left(\mathcal{X}^{(k+1)}|\mathcal{X}^{(0)}, \mathcal{X}^{(1)}, \ldots, \mathcal{X}^{(k)}\right) = P\left(\mathcal{X}^{(k+1)}|\mathcal{X}^{(k)}\right) \ , \tag{2.33}$$

   and

2. the homogeneity assumption, i.e., the transition probability from a state ξ' to a state ξ'' is the same for any time index k, so we can represent the transition probabilities using a transition model $P\left(\mathcal{X}''|\mathcal{X}'\right)$ for all k:

$$P\left(\mathcal{X}^{(k+1)} = \xi''|\mathcal{X}^{(k)} = \xi'\right) = P\left(\mathcal{X}'' = \xi''|\mathcal{X}' = \xi'\right) \ . \tag{2.34}$$

A BN that connects $\mathcal{X}^{(k+1)}$ and $\mathcal{X}^{(k)}$, for all k, and that satisfies the two assumptions above is called a 2-time-slice BN (2-TBN), and can be uniquely defined using Eq. (2.34). Now we can define a DBN [3], that is a pair $\langle \mathcal{B}_0, \mathcal{B}_\rightarrow \rangle$, where $\mathcal{B}_0$ is a SBN that defines the relation among the random variables at time $k = 0$, while $B_\rightarrow$ is a 2-TBN that connects the variables at time k with the variable at time $k + 1$, for all $k = 1, \ldots, N$.

As in the case of the BN, also for a DBN we can calculate the *qualitative* and the *quantitative* relations among the parameters, as detailed in [3].

Chapter 3

Exploiting CS in WSNs: interplay between routing and signal representation

An important research topic in the area of communication and protocol design for Wireless Sensor Networks (WSNs) which needs further investigation is in-network aggregation and data management to increase the efficiency of data gathering solutions (in terms of energy cost) while being able to measure large amounts of data with high accuracy.

In this chapter [1] we address the problem of exploiting CS in WSNs taking into account network topology and routing, which is used to transport random projections of the sensed data to the Data Collection Point (DCP). Thus, the main contribution of this chapter is the quantification of the benefits of CS in realistic multi-hop WSNs when CS is used in conjunction with routing. In addition, we study the problem of finding good transformations to make real sensed data meet the sparsity requirements of CS and show that widely used transformations are not suitable for a large spectrum of real signals. We also provide a simulation based comparison between the commonly used random sampling (considered here in conjunction with spline interpolation) and CS based data gathering. For this comparison

[1] The material presented in this chapter has been published in:

[C6] **G. Quer**, R. Masiero, D. Munaretto, M. Rossi, J. Widmer and M. Zorzi, "On the Interplay Between Routing and Signal Representation for Compressive Sensing in Wireless Sensor Networks", *Information Theory and Applications Workshop (ITA 2009)*, San Diego, CA, Jan.-Feb. 2009.

we use some data matrices synthetically generated, as well as real field data available in the Internet. In this preliminary study we do not take into account the temporal evolution of the signals, so we do not deal with a learning phase to infer a suitable matrix to sparsify the data. In Chapter 4 instead we study the spatial and temporal correlations of time-varying WSN signals and we exploit the statistical characteristics of such signals in a novel data collection technique, described in Chapter 5. Such technique, differently from the techniques proposed in this chapter, learns the signals' statistical characteristics with PCA, detailed in Section 2.1.3, and exploits this on-line learning of the sparsification basis to recover the signal with CS.

The rest of this chapter is structured as follows. In Section 3.1 we summarize the related work on CS applied to WSNs, then in Section 3.2 we present the mathematical notation adopted. In Sections 3.3, 3.4 and 3.5 we describe the signals, the network model, and the data gathering protocols, respectively, which we used for the investigation of the benefits of CS applied to multi-hop WSNs. The simulation results are presented in Section 3.6 and Section 3.7 concludes the chapter.

3.1 Related Work

The problem of gathering data while jointly performing compression has been receiving increasing attention. One of the first studies addressing this issue is [23], which highlights the interdependence among bandwidth, decoding delay and the routing strategy employed. Under the assumption of dealing with spatial processes satisfying the regularity condition (justifiable from a physical point of view), the authors claim the feasibility of large-scale multi-hop networks from a transport capacity perspective. Classical source coding, suitable routing algorithms and re-encoding of data at relay nodes are key ingredients for joint data gathering and compression. In fact, sensor network applications involve multiple sources which are correlated both temporally and spatially. Thus, subsequent work such as [24–27] and [28] proposed algorithms that involve collaboration among sensors to implement classical source coding (see e.g., [29–31]) in a distributed fashion. Along the same line, [32] shows the relation between routing and location of the aggregation/compression points according to the joint correlation of data among sources. In this way, it is possible to enforce the collaboration among nodes that are well suited to the statistical description of the signal measurements.

In [33], the authors consider a scenario where a number of different compression schemes are available at each node in the network. The selection of which compression scheme to use is based on the expected tradeoff between computation and communication costs; each node contributes to this goal through its local data processing. Following the same objective of minimizing the total energy for compressing and transporting information, [34] investigates a tunable data compression technique to deal with the tradeoff between computation and communication costs. In general, for a given connectivity structure, this technique needs to compute the optimal data gathering tree, which is topology dependent. Moreover, the authors show that when node entropies and the cost for compression are not known, a simple greedy approximation of the Minimal Steiner Tree provides acceptable performance.

Recently, new methods for distributed sensing and compression have been developed based on CS theory (see e.g., [6,7,35]). Early contributions are [8,36], where CS is used in a distributed communication scheme for energy efficient estimation of sensed data in a WSN. Multi-hop communication and in-network data processing are not considered. Instead, data packets are directly transmitted by each node to the DCP. This requires synchronization among nodes.

[37] proposes an early application involving CS for network monitoring. The considered simulation scenario is a network where a small set of nodes fails. The goal is to correctly identify these nodes through the transmission of random projections (i.e., linear combinations) indicating the status of the nodes. However, these random projections are obtained by means of a pre-distribution phase (via simple gossiping algorithms), which is very expensive in terms of number of transmissions. [38] also addresses the problem of gathering data in distributed WSNs through multi-hop routing. In detail, tree topologies are exploited for data gathering and routing, and the Wavelet transformation [39] is used for data compression. Even though CS is presented as one of the possible methods for data compression, the authors do not investigate the impact of the network topology and that of the routing scheme on the compression process. An interesting application for network monitoring exploiting CS is presented in [40], where the aim is to efficiently monitor communication metrics, such as loss or delay, over a set of end-to-end network paths by observing a subset of them. The topology is given a priori and the algorithm works in three steps: 1) compression, 2) non linear estimations and 3) suitable path selection. This last step in particular allows the selection of the best measurements for CS recovery, and therefore highly impacts the overall performance of the algorithm. In [41] and [42] an approach to distributed coding and com-

pression in sensor networks based on CS is presented. The authors advocate the need to exploit the data both temporally and spatially. The projections of the signal measurements are performed at each source node, taking into account only the temporal correlation of the generated information. Thus, it is possible to design the best approximation of the collection of measurements for each node, since the projections can contain all the elements of this set. The spatial correlation is then exploited at the DCP by means of suitable decoders through a joint sparsity model that well characterizes the different types of signals of interest. Finally, the technical report [43] follows an approach similar to the one adopted in this chapter, concluding that CS is not an effective solution when routing costs are considered.

A further related line of research is that of real and complex network coding [44, 45]. These papers highlight the analogies between network coding (NC) and CS from the viewpoint of distributed data processing and routing rules. An important difference between NC and CS is that CS works in real fields whereas NC exploits algebraic operations over Galois fields. This leads to practical issues, such as round-off errors that arise when dealing with real numbers, which are treated in [44].

In our work we address the joint routing and compression problem by exploiting the spatial correlation among sensor readings in a 2D WSN. The sensor nodes do not need to be aware of any correlation structure of the input signal. In particular, we only require that the sensed data has a sparse representation and that the sensor nodes can locally perform random combinations of the incoming information. The goal is to reconstruct the original signal with good accuracy from a small subset of samples using distributed CS. To the best of our knowledge, no papers so far quantified the performance of CS in multi-hop wireless networks by exploiting actual routing topologies to obtain random projections of the signal measurements, except for [43]. However, their conclusions about the effectiveness of CS for synthetic signals are different from ours and they did not address real signal analysis. The objective of our work is to fill this gap investigating the tradeoffs between energy consumption and reconstruction error for realistic scenarios. Furthermore, we analyze under which conditions CS performs well and under which conditions it fails to improve the performance.

3.2 CS notation

For the sake of exposition, we briefly present in this section the CS notation that is adopted in this chapter. We consider signals representable through one dimensional vectors $\mathbf{x}$ in $\mathbb{R}^N$, where N is the vector length. We assume that these vectors contain the sensor readings of a network with N nodes. We further assume that these vectors are such that there exists a transformation under which they are sparse. Specifically, there must exist an invertible transformation matrix $\mathbf{\Psi}$ of size $N \times N$ such that we can write

$$\mathbf{x} = \mathbf{\Psi s} \tag{3.1}$$

and $\mathbf{s}$ is sparse. We say that a vector $\mathbf{s}$ is P-sparse if it has at most P non-zero entries, with $P < N$.

The compression of $\mathbf{x}$ entails a linear combination of its elements through a further *measurement matrix* $\mathbf{\Phi}$ of size $L \times N$, with $L < N$. The compressed version of $\mathbf{x}$ is thus obtained as

$$\mathbf{y} = \mathbf{\Phi x} \ . \tag{3.2}$$

Now, using (3.1) we can write

$$\mathbf{y} = \mathbf{\Phi x} = \mathbf{\Phi \Psi s} \stackrel{def}{=} \tilde{\mathbf{\Phi}} \mathbf{s} \ . \tag{3.3}$$

These systems are ill-posed as the number of equations L is smaller than the number of variables N. Nevertheless, if $\mathbf{s}$ is sparse, it has been shown that the above system can be inverted with high probability through the use of specialized optimization techniques [6, 46]. The problem to be solved at the receiver is thus to invert the above system so as to find vector $\mathbf{s}$. Note that in order to do this, the receiver should know the transformation matrix $\mathbf{\Psi}$ that sparsifies $\mathbf{x}$.We do not deal in this chapter with the on-line learning of this matrix, but we assume that the matrix is given a-priori. Once we know a sparse solution $\mathbf{s}^\star$ that verifies (3.3), the original data $\mathbf{x}$ can be recovered through (3.1), with the convex optimization method detailed in Section 2.1

From a data gathering point of view, the signal $\mathbf{x}$ stores the data readings measured by the N nodes. These are mixed during their transmission towards the DCP as explained in Section 3.5. Thus, each route followed by a given packet specifies the coefficients of a row of $\mathbf{\Phi}$. The data gathering point will receive the compressed vector $\mathbf{y}$ along with the coefficients of matrix $\mathbf{\Phi}$. In the following, $\mathbf{\Phi}$ is referred to as *routing matrix*. Note that the DCP can obtain

the coefficients of $\mathbf{\Phi}$ through different ways, e.g., these coefficients can be sent along with the information packet if the relative overhead is small, or we can use the same pseudo-random number generator at the nodes and the DCP and synchronize the seeds. The problem to be solved at the receiver is thus to invert the system (3.3) so as to find vector $\mathbf{s}$. Note that in order to do this, the receiver should also know the transformation matrix $\mathbf{\Psi}$ that sparsifies $\mathbf{x}$.[2] We emphasize that the transformation $\mathbf{\Psi}$ is only used at the DCP and not during the data gathering and routing process, that is instead captured by $\mathbf{\Phi}$. In particular, $\mathbf{\Psi}$ does not need to be known at any node but the DCP.

A generalization of the CS technique for 2D signals is detailed in the Appendix at the end of this chapter.

3.3 Considered Signals and Transformations

In this section we discuss the signals that we consider for the performance evaluation in this chapter. First, we investigate synthetic signals that are sparse by construction under the DCT transformation. For these signals the degree of sparseness can be precisely controlled. As expected, when they are sufficiently sparse CS achieves substantial gains compared to plain routing schemes. Furthermore, we select a number of signals from real sensor networks measuring different physical phenomena. With such signals, we can much better characterize the performance expected for actual WSN deployments. The problem with real signals, however, is to find a good transformation that sparsifies them in some domain. This issue is discussed at the end of the section.

Synthetic signals. Here, for the input signal we use a matrix $\mathbf{X}$ that we build starting from a sparse and discrete 2D signal $\mathbf{S}$ in the frequency (DCT) domain. $\mathbf{S}$ is obtained through the following steps:

1. Let K be defined as $K = \sqrt{N}$, where N is the number of values of the 2D signal. We build a preliminary signal $\mathbf{S}_1$ of size $K \times K$ having all frequencies (i.e., all entries in the matrix) with amplitude $s_1(p, q)$, where $s_1(p, q)$ is picked uniformly at random in the interval $[0.5, 1.5], \forall\ p, q = 1, 2, \ldots, K$.

[2] This is a reasonable assumption. For example, for image processing it has been verified that the Fourier transformation is a good tool for sparsifying real images [35]. Signals gathered by sensor fields usually show high spatial correlation [47] and can thus be sparsified as we discuss in Section 3.3.

2. We define a frequency mask as a 2D function that is one for entries in position (p, q) where $p+q \leq p_{\text{low}}$ or $p+q > p_{\text{high}}$ and zero otherwise. p_{low} and p_{high} are two thresholds in the value range $\{1, 2, \ldots, K\}$. This function is defined as

$$\texttt{triang}(p, q) \stackrel{def}{=} \begin{cases} 1 & p + q \leq p_{\text{low}} \text{ or} \\ & p + q > p_{\text{high}} \\ 0 & \text{otherwise .} \end{cases} \tag{3.4}$$

3. We obtain a second signal $\mathbf{S}_2$ of size $K \times K$, whose entries $s_2(p, q)$ are calculated as

$$s_2(p, q) = s_1(p, q)\texttt{triang}(p, q) \ . \tag{3.5}$$

4. We finally obtain $\mathbf{S}$ as follows: if $s_2(p, q) = 0$ then $s(p, q) = \xi$ where $\xi \in [0, 0.01]$ is a constant. If instead $s_2(p, q) > 0$, $s(p, q) = \xi$ with probability p_d and $s(p, q) = s_2(p, q)$ otherwise. The parameter p_d represents the fraction of entries that are on average deleted from $\mathbf{S}_2$. The case $\xi > 0$ is accounted for to mimic non ideal signals, where the significant components lie within specific regions according to (3.4) and some *noise floor* is also present outside these regions. In this case, with CS we would like to only retrieve the significant values, while ignoring the noise.

Therefore, the signal $\mathbf{S}$ is obtained by first applying a frequency mask, which helps to assess the reconstruction performance for low-frequency, mid-frequency, and high-frequency signals. In addition, we delete some randomly picked frequencies according to a given probability p_d. This is a simple method to control the characteristics of the signal in the DCT domain (i.e., the sparsity of the signal and its dominant frequency components) and allows to understand the effects of the signal structure on the performance of CS. For the results in Section 3.6 synthetic signals are mapped into matrices $\mathbf{X}$ of size 20×20, which is consistent with the network topology in Section 3.4 with $N = 400$ nodes.

Real Signals. We also used real signals from different environmental phenomena, considering what is likely to be of interest for a realistic wireless sensor network in terms of size of the network (i.e., number of spatial samples) and type of phenomenon to sense. For the sensor network, we considered the topology in Section 3.4 with $N = 400$ sensor nodes.

The following real signals were utilized:

S1. Two signals representing the Wi-Fi strength of the access points in the MIT campus (Cambridge, MA) [48] and in the Stevens Institute of Technology (Hoboken, NJ) [49].

S2. Two sets of measurements from the EPFL SensorScope WSN [50], representing ambient temperature and solar radiation.

S3. Two data readings, one from the Tropical Rainfall Measuring Mission [51] concerning rain fall in Texas, and one on the temperature of the ocean off the coast of California [52].

S4. Two signals on the level of pollution in two European regions, namely, Benelux and Northern Italy [53].

These signals were quantized into five levels and rescaled in grids of 20×20 pixels. The assumption of measuring quantized signals was made as we think this is likely to be the case in actual WSN deployments, where the devices, due to communication, energy constraints or accuracy of the on-board sensor, can only sense or communicate the physical phenomena of interest according to a few discrete levels. In addition, for many signals of interest a quantized representation suffices to fully capture the needed information about the sensed phenomenon. The eight sample signals, quantized and rescaled as discussed above, are shown in Fig. 3.7.

Transformations. By construction, for the above synthetic signals the DCT is the right sparsification method. These signals were in fact created sparse in the DCT domain. An effective utilization of CS for real signals requires a good sparsification approach. It is not clear, however, which approach is best for a given class of signals. Here, we consider four different transformations, which are commonly used in the image processing literature:

T1. *DCT:* this is the standard 2D discrete cosine transformation, see the appendix for further details.

T2. *Haar Wavelet:* the Haar Wavelet is recognized as the first known Wavelet and is a good Wavelet transformation for the sparsification of piece-wise constant signals as the ones in S1–S4, see [54].

T3. *Horz-diff:* this is a transformation that we propose here to exploit the spatial correlation of our signals. First, the 2D signal matrix $\mathbf{X}$ is written in vector form as follows:

$$\begin{aligned}\mathrm{svec}(\mathbf{X}) \quad &= (x(1,1), x(1,2), \ldots, x(1,K), x(2,K), x(2,K-1), \ldots, x(2,1), \\ &\quad x(3,1), x(3,2), \ldots, x(3,K), x(4,K), x(4,K-1), \ldots, x(4,1), \ldots) \end{aligned} \quad (3.6)$$

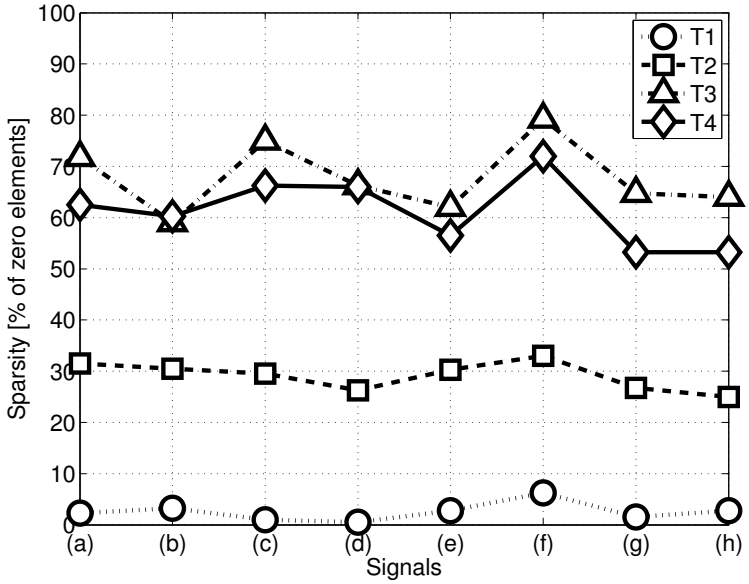


Figure 3.1. *Degree of sparsity for transformations T1–T4. The plot shows the percentage of zero elements of vector* $\mathbf{s}$ *after using transformations T1–T4.*

At this point we obtained the sparse vector $\mathbf{s}$ from $\mathrm{svec}(\mathbf{X})$ by pair-wise subtraction of its elements.

T4. *HorzVer-diff:* according to this transformation the input signal $\mathbf{X}$ is processed by: 1) pair-wise subtraction of the elements along the columns of $\mathbf{X}$ and then 2) pair-wise subtraction of the elements of the resulting matrix, along its rows.

In Fig. 3.1 we show the degree of sparseness achievable using the above transformations T1–T4 with the considered real signals (a)–(h). Notably, DCT (T1) and Haar Wavelet (T2) are not effective, whereas T3 and T4 perform best.

DCT and Wavelet transformations in this case have poor performance as, even though the sampled input signals $\mathbf{X}$ are quite large ($N = 400$ data points) for typical sensor deployments (where each node gathers a single data point), their size is still too small for T1 and T2 to perform satisfactorily. T3 and T4 perform best since they exploit the characteristics of piece-wise constant signals, even if the sparsity obtained is not sufficient for CS to work properly. Since standard techniques as T1–T4 are not satisfactory, a more fundamental approach, i.e., via estimation of the correlation $\mathbf{X}$ and Karhunen-Loève expansion, may be needed. We leave this for future research.

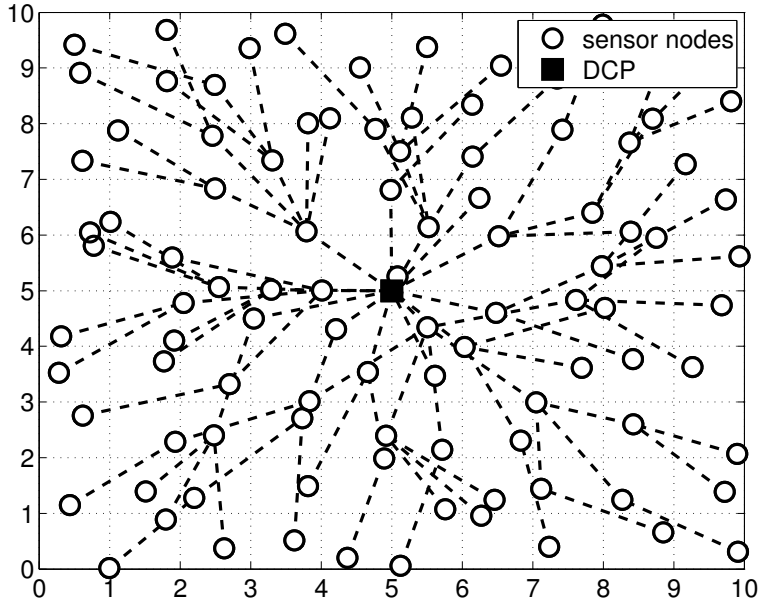


Figure 3.2. *Example of the considered multi-hop topology.*

3.4 Network Model

The concern of this chapter is about data gathering in 2D WSNs. Hence, for the rest of the chapter we consider sensor grids of N nodes as follows. We consider N nodes to be deployed in a square area with side length L. This area is split into a grid with N square cells and we place each of the N nodes uniformly within a given cell so that each cell contains exactly one node. For the transmission range R of the nodes we adopt a unit disk model, i.e., nodes can only communicate with all other nodes placed at a distance less than or equal to R.[3] We use $R = \sqrt{5}L/\sqrt{N}$ as this guarantees that the structure is fully connected under any deployment of the nodes. A further node, the Data Collection Point (DCP), is placed in the center of the deployment area. We consider geographic routing to forward the data towards the DCP, where each node considers as its next hop the node within range that provides the largest geographical advancement towards the DCP. In Fig. 3.2, we show an example topology; as per the above construction process, each cell has a node and the network is always connected. The tree in this figure is obtained through the above geographic routing approach, and is used by the data aggregation protocols to route data towards the DCP.

According to this network scenario, the input signal is a square matrix $\mathbf{X}$ with N ele-

[3]The unit disk graph model is used here for simplicity of explanation and topology representation. However, the presented methodology can be readily applied to more realistic propagation models, e.g., fading channels.

ments, where element (i, j) (referred to as $x(i, j)$) is the value sampled by the sensor placed in cell (i, j) of the sensor grid.

Despite its simplicity and the assumption that each cell contains a sensor node, this scenario captures the characteristic features (multi-hop routing and all to one transmission paradigm) of actual WSN deployments and allows to study the interplay between data gathering and CS.

3.5 Data Gathering Protocols

In this section we present the data gathering protocols that will be considered for the investigation of the benefits of CS when used in multi-hop WSNs. As pointed out in Section 3.1, there is a well studied line of research on the application of CS to data gathering in wireless networks. Previous studies however adapted the routing technique or the data transmission phase so as to take full advantage of CS. What we do here is different as we pick a distributed WSN and consider the usual data gathering paradigm where sensors forward the packet(s) they receive along shortest paths towards the DCP. This occurs in a completely unsynchronized and distributed manner, without knowledge about the correlation structure of the data and without knowing how it is processed at the DCP through CS. Thus, our aim is to assess whether CS provides performance benefits with respect to standard schemes even in such distributed and unsynchronized network scenarios.

In what follows we present two schemes: the first is a standard geographical routing protocol, whereas the second is the same protocol in terms of routing, but it exploits CS for data recovery at the DCP. We then characterize the structure of the $\mathbf{\Phi}$ matrix (see Section 3.2) which is determined by the routing policy.

Data gathering protocols. To simplify the investigation and to pinpoint the fundamental performance tradeoffs, in this first study we neglect channel access considerations (i.e., collisions, transmission times, etc.). Also, we assume a unit cost for each packet transmission and we ignore processing overhead at the nodes, as it is expected to be cheap compared to the cost of packet transmission.

P1. *Random sampling with Spline Interpolation (RS-Spline):* this is the simplest protocol that we consider. In this case, each node becomes a source with probability $P_T = L/N$, which was varied in the simulations to obtain tradeoff curves for an increasing transmission overhead. On average, L nodes transmit a packet containing their own sensor

reading. Each packet is routed to the DCP following the path that minimizes the number of transmissions (as defined by our geographical routing approach). Along this path, the packet is not processed but simply forwarded. The cost of delivering a single packet to the DCP is given by the number of hops that connect the originating node to the data gathering point. The signal is reconstructed by interpolation of the collected values according to the method in [55].

P2. *Random sampling with CS (RS-CS):* this protocol is similar to RS-Spline. As above each node becomes a source with probability $P_T = L/N$. Again, each of these source nodes transmits a packet containing the reading of its own sensor. As this packet travels towards the DCP, we combine the value contained therein with that of any other node that is encountered along the path. Specifically, let v_i^m with $i = 1, 2, ..., \ell_m$ be the readings of the sensors along the path from node m to the DCP, where v_1^m is the reading of the node itself and ℓ_m is the length of the path. Node m sends a packet containing the value $y_1^m = \alpha_1 v_1^m$ as well as the combination coefficient α_1, where α_1 is a value chosen uniformly at random either from $(0, 1]$ or from the set $\{-1, +1\}$.[4] The next node along the path will update the transmitted value and send out $y_2^m = y_1^m + \alpha_2 v_2^m$ where α_2 is again a random value. Also the coefficient α_2 is included in the data packet along with with α_1. We proceed with these random combinations, where in general node $i + 1$ sends out

$$y_{i+1}^m = y_i^m + \alpha_{i+1} v_{i+1}^m \,, \tag{3.7}$$

until the packet finally reaches the DCP. The DCP extracts $y_{\ell_m}^m = \sum_{i=1}^{\ell_m} \alpha_i v_i^m$, together with the vector of α coefficients that were used along the route. These coefficients, according to the CS formalism in Section 3.2, form the mth row of matrix $\mathbf{\Phi}$, referred to as φ_m. Note that some optimizations are possible. First, if we know in advance the network topology, we can assign combination coefficients at setup time to all nodes, rather than including them in the packets. We can further use the same pseudo-random number generator at the nodes and the DCP and synchronize the seeds. However, all of this goes beyond the scope of this chapter and we do not focus on how to optimize the control overhead of CS.

A few observations are in order. When we use CS at the DCP, we receive packets carrying more valuable information than in the plain forwarding case. The received values are *linear*

[4] The implications of the selection of the set to use are discussed in Section 3.6.

random combinations of the readings of several sensor nodes. For example, considering RS-CS, when the DCP receives the mth packet it can build a system of the form

$$\mathbf{y} \stackrel{def}{=} \begin{pmatrix} y^1_{\ell_1} \\ y^2_{\ell_2} \\ \vdots \\ y^m_{\ell_m} \end{pmatrix} = \begin{pmatrix} \varphi_1 \\ \varphi_2 \\ \vdots \\ \varphi_m \end{pmatrix} \mathtt{vec}(\mathbf{X}) = \mathbf{\Phi}\mathtt{vec}(\mathbf{X}) \ , \tag{3.8}$$

where the $y^r_{\ell_r}$ with $r = 1, 2, \ldots, m$ are the combined values that were received by the DCP in the packet that traversed the rth path, $\mathbf{X}$ is the input 2D signal, $\mathbf{\Phi}$ is an $m \times N$ matrix whose generic row r, φ_r, contains the vector of coefficients α included in the packet. Note that, in general, some of these coefficients might be equal to zero. Specifically, the node in cell (i, j) of the 2D grid can only contribute to entry $(i-1)K + j$ of vector φ_r (see also the ordering shown in the appendix in (3.13)). Thus, the combination coefficient in position $(i-1)K + j$ of φ_r, with $i, j = 1, 2, \ldots, K$, is non-zero if and only if node (i, j) was included in the path followed by the rth packet and is set to zero otherwise.[5] Hence, matrix $\mathbf{\Phi}$ highly depends on the *network topology* and on the *selected routing rules* as each of its rows will have non-zero elements only in those positions representing nodes that were included in the path followed by the corresponding packet.

Note that (3.8) is a system of linear equations that is in general ill-posed (as $m \leq L$ and L is expected to be smaller than N). At the DCP, we know vector $\mathbf{y}$ and matrix $\mathbf{\Phi}$ and we need to find the 2D input signal $\mathbf{X}$. We can now use the derivations in the Appendix and rewrite $\mathbf{y} = \tilde{\mathbf{\Phi}}\mathtt{vec}(\mathbf{S})$ which is solved for $\mathtt{vec}(\mathbf{S})$ using standard CS tools for the 1D case [56], thus finding the sparsest $\mathtt{vec}(\mathbf{S})$ that verifies the system, referred to here as $\mathbf{S}^\star$. $\mathbf{S}^\star$ is finally used to reconstruct $\mathbf{X}$, i.e., $\mathbf{X}^\star = \mathbf{\Psi}\mathbf{S}^\star\mathbf{\Psi}^T$ (see also (3.12) in the appendix).

Characterization of the routing matrix $\mathbf{\Phi}$. According to our network model, the nodes that transmit their packet to the DCP are chosen at random. As said above, every row φ_j of $\mathbf{\Phi}$ represents a path from a given sensor to the DCP and each forwarding node in this path contributes with a non zero coefficient. We characterize the sparsity ν_j of φ_j counting the number of elements in this row that differ from zero: $\nu_j = \sum_{i=1}^{N} \mathbb{1}\{\alpha_i^j \neq 0\}$, where α_i^j is the ith entry of vector φ_j and $\mathbb{1}\{E\}$ is the indicator function, which is 1 when event E is true and zero otherwise. ν_j is the cost, in terms of number of transmissions, for sending

[5] Given this, we see that setting an entire column of the matrix to zero, say column $c = (i-1)K + j$ for given i and j, means that we completely ignore the contribution of the node placed in cell (i, j). This happens when none of the m received packets passes through this node while being routed to the DCP.

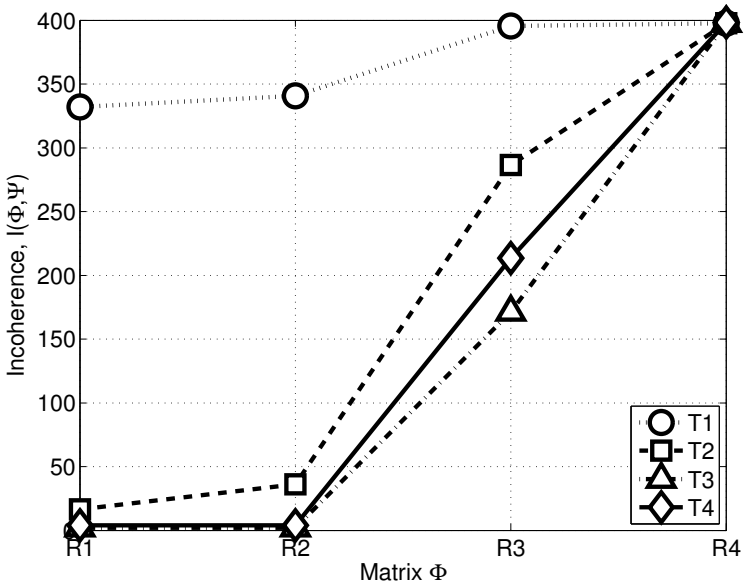


Figure 3.3. *Incoherence $I(\mathbf{\Phi}, \mathbf{\Psi})$ between the routing matrix $\mathbf{\Phi}$, cases R1–R4, and the transformation matrix $\mathbf{\Psi}$, transformations T1–T4. The maximum value for $I(\mathbf{\Phi}, \mathbf{\Psi})$ equals the number of nodes in the network, $N = 400$.*

the jth packet to the DCP. With the network scenario in Section 3.4 it is easy to see that, for any source node in the network, the number of transmissions required for its packet to reach the DCP is $O(\sqrt{N})$. Hence, the total cost for the transmission of L packets is $O(L\sqrt{N})$. As an example, for a network with $N = 400$ nodes the cost of delivering a packet to the DCP is $\simeq 4.5$ transmissions, which is close to $\sqrt{N}/4$. The sparsity of φ_j directly translates into the sparsity of $\mathbf{\Phi}$ that, in turn, affects the *coherence* between the matrices $\mathbf{\Phi}$ and $\mathbf{\Psi}$. In the literature, the concept of coherence (or its dual, called *incoherence*) between these two matrices is directly related to the effectiveness of the CS recovery phase and is well defined when they are orthonormal. Specifically, the routing matrix $\mathbf{\Phi}$ and $\mathbf{\Psi}$ must be incoherent for CS to work properly [7].

In our settings, however, $\mathbf{\Phi}$ is built on the fly according to the routing topology, whereas $\mathbf{\Psi}$ is obtained according to any of the transformations T1–T4 that we discussed in Section 3.3. In the literature the concept of coherence is not defined for non-orthogonal matrices. However, according to the rationale in [7, 41] a quantity that is strictly related to the incoherence can be computed as follows. Roughly speaking, incoherence between two matrices means that none of the elements of one matrix has a sparse representation in terms of the columns of the other matrix (if used as a basis). Put differently, two matrices are highly coherent

when each element of the first can be represented linearly combining a small number of columns of the second. Hence, to characterize the incoherence we first project each row of $\mathbf{\Phi}$ into the space generated by the columns of $\mathbf{\Psi}$. After this, we take the sparsest projections obtained in this space as an indication of the incoherence. Formally, we have:

$$\zeta_j = (\mathbf{\Psi}^T\mathbf{\Psi})^{-1}\mathbf{\Psi}^T\varphi_j^T \ , \tag{3.9}$$

where φ_j is the jth row of $\mathbf{\Phi}$ and ζ_j is the vector of coefficients corresponding to its projection on the space generated by the columns of $\mathbf{\Psi}$. A measure of the incoherence is then obtained as

$$I(\mathbf{\Phi},\mathbf{\Psi}) = \min_{j=1,\ldots,N}\left[\sum_{i=1}^{N}\mathbb{1}\{\beta_i^j \neq 0\}\right] \in \left[1, N\right] \ , \tag{3.10}$$

where β_i^j is the ith entry of vector ζ_j.

In Fig. 3.3 we show the incoherence, obtained from (3.10), for the four transformation methods T1–T4 and for the following matrices $\mathbf{\Phi}$: R1) $\mathbf{\Phi}$ is built according to the CS routing protocol that we explained above, picking random coefficients in $\{-1, +1\}$, R2) $\mathbf{\Phi}$ is built as in case R1, picking random coefficients in $(0, 1]$, R3) $\mathbf{\Phi}$ has all coefficients randomly picked in $\{-1, +1\}$ and R4) $\mathbf{\Phi}$ has coefficients uniformly and randomly picked in $(0, 1]$. As can be deduced from the results of [37], cases R3 and R4 are near optimal in terms of projections of the measurements and can be built through a pre-distribution of the data (that in a multi-hop WSN is in general demanding in terms of number of transmissions).

From this plot we see that the DCT transformation (T1) has a high incoherence with respect to all of the considered routing matrices. The remaining transformations T2–T4 all perform similarly and give satisfactory performance only for cases R3 and R4, whereas for random projections obtained through the actual routing scheme they are highly coherent to $\mathbf{\Phi}$. This has strong negative implications on the CS recovery performance and will be discussed in the following section.

3.6 Results

In this section we discuss the results we obtained by simulating the RS and RS-CS data gathering schemes for synthetic and real signals. The metric of interest is the reconstruction quality at the DCP, which is defined as follows. Given a 2D input signal $\mathbf{X}$, a matrix $\mathbf{\Phi}$ and a vector $\mathbf{y}$ (containing the received values that are linear combinations of the sensor readings

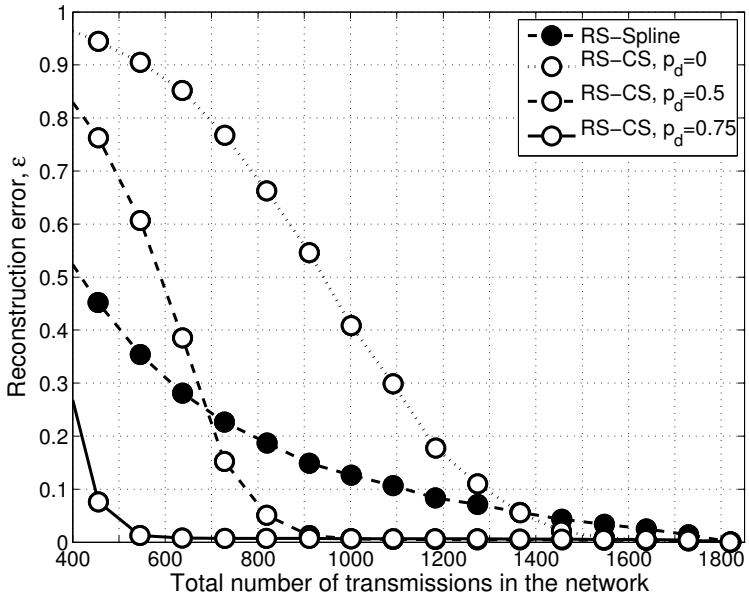


Figure 3.4. *Reconstruction quality ε as a function of the total number of packets transmitted in the network: comparison between RS-Spline and RS-CS for synthetic signals and different values of p_d.*

in the network) we have that $\mathbf{y} = \mathbf{\Phi}\mathtt{vec}(\mathbf{X})$. This system, that in general is ill-posed (as $L \leq N$), is solved for $\mathtt{vec}(\mathbf{X}) = \mathtt{vec}(\mathbf{\Psi S \Psi}^T)$ either through norm one [7] or smoothed zero norm [46] minimization. These methods efficiently find the sparsest $\mathbf{S}$, referred to as $\mathbf{S}^\star$, that verifies the previous system.[6] If $\mathbf{X}^\star = \mathbf{\Psi S}^\star \mathbf{\Psi}^T$ is the solution found for this system and $\mathbf{X}$ is the true input signal, the *reconstruction error* is defined as

$$\varepsilon = \frac{\|\mathtt{vec}(\mathbf{X}) - \mathtt{vec}(\mathbf{X}^\star)\|_2}{\|\mathtt{vec}(\mathbf{X})\|_2} \ . \tag{3.11}$$

3.6.1 Results for Synthetic signals

In Fig. 3.4 we show the reconstruction error ε as a function of the total number of packets sent in the network for RS-Spline and RS-CS. For this plot we considered a low-pass signal with $p_{\text{low}} = \sqrt{N}/2 + 1$ and $p_{\text{high}} = \sqrt{N}$, with $N = 400$. Also, we considered three values of $p_d \in \{0, 0.5, 0.75\}$ so as to vary the sparseness of the signal. As a first observation, random sampling performs nicely for low-pass signals. Nevertheless, a perfect reconstruction of the sensed signal at the DCP requires the transmission of a large number of packets (up to 1800). When the signal is sufficiently sparse ($p_d \geq 0.5$) CS outperforms standard data

[6] We found that these two methods are nearly equivalent in terms of quality of the solution, although the zero norm is simplest and faster. This might be important for practical implementations.

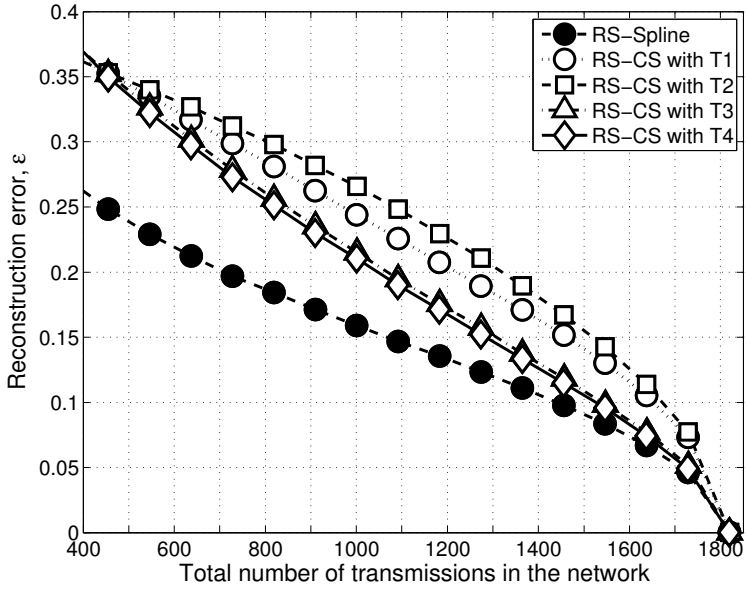


Figure 3.5. *Reconstruction error ε vs total number of packets transmitted in the network: comparison between RS-Spline and RS-CS (for transformations T1–T4) for the real signals in Section 3.3.*

gathering schemes, requiring less than half the packet transmissions (about 900) to achieve the same recovery performance. We noticed that values of ε larger than 0.3 always led to very inaccurate reconstructions of the original signal. Fig. 3.4 was obtained using L_1 minimization and combination coefficients in the set $\{-1, +1\}$. However, we obtained similar performance using smoothed L_0 norm and/or coefficients in the set $(0, 1]$. Note that using the set $\{-1, +1\}$ allows for reduced overhead as, in practical implementations, a single bit suffices to transmit each coefficient.

For high-pass signals the performance of CS is unvaried for the same degree of sparseness. This is expected as CS recovery operates in the frequency domain and is only affected by the number of non-zero frequency components and not by their position. Clearly, RS-Spline with the considered interpolation technique is not appropriate for high-pass signals, in which case it shows poor recovery performance.

As a consequence, RS-CS shows good recovery performance for synthetic signals as, by construction, the DCT transformation effectively sparsifies the signal and this transformation is incoherent with respect to the routing matrix $\mathbf{\Phi}$ (see Fig. 3.3).

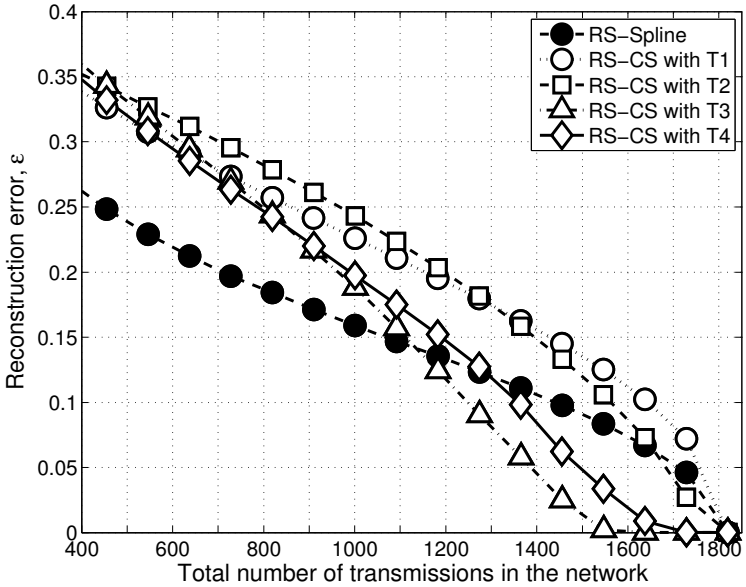


Figure 3.6. *Reconstruction error ε vs total number of packets transmitted in the network: comparison between RS-Spline and RS-CS (for transformations T1–T4) when a pre-distribution of the data is allowed so that the routing matrix $\mathbf{\Phi}$ approaches that of case R4 of Section 3.5.*

3.6.2 Results for Real Signals

In Fig. 3.5 we show the reconstruction error ε as a function of the total number of packets sent in the network for RS-Spline and RS-CS. The sensed signals belong to the data sets presented in Section 3.3. In this case, differently from the case of synthetic signals, RS-CS does not outperform RS-Spline, even though the performance of the two methods is very close. The reason for this is twofold. First, the considered transformations T1–T4 sparsify the real signals only up to 70% (see Section 3.3). This is mainly due to the characteristics of the signals and to the small size of the sample set. Second, the transformations with the best performance in terms of sparsification have a high coherence with respect to the routing matrix of RS-CS. Hence, while the sparsification performance may suffice, matrix $\mathbf{\Phi}$ (routing) does not have the required properties in terms of coherence for CS to perform satisfactorily.

In fact, for good recovery performance CS needs a good transformation in terms of sparsification. Also, transformation and routing matrices must be incoherent. From Figs. 3.1, 3.3 and 3.5 we see that transformations T3 and T4 are the most suitable to sparsify the considered real signals and this allows them to perform better than T1 and T2 (even though

they perform poorly in terms of incoherence, see Section 3.5). In addition, although T2 can sparsify real signals better than T1 (Fig. 3.1), the latter performs better than T2 in terms of transmission cost *vs* error reconstruction (Fig. 3.5), since it has better incoherence properties $I(\mathbf{\Phi}, \mathbf{\Psi})$ (Fig. 3.3).

Finally, in Fig. 3.6 we accounted for a pre-distribution phase of the data so that matrix $\mathbf{\Phi}$ is as close as possible to that of case R4 of Section 3.5 (we verified that case R3 gives similar performance). In this case, RS-CS outperforms RS-Spline as T1 and T2 provide a sparse representation of the signal and the routing matrix is sufficiently incoherent with respect to these transformations. However, this pre-distribution phase (which is similar to that proposed in [37]) has a high transmission cost for static networks, which is ignored in Fig. 3.6. In mobile networks, the pre-distribution could take advantage of the nodes' mobility so as to decrease the cost associated with the construction of $\mathbf{\Phi}$. This is not dealt within this chapter and is left for future research.

3.7 Conclusions

In this chapter we studied the behavior of CS when used jointly with a routing scheme for recovering two types of signals: synthetically generated and obtained from real sensor data. We showed that for the synthetic signals the reconstruction at the DCP is enhanced when applying CS, whereas the application of CS for real sensor data is not straightforward. Thus, as a next step of our ongoing research, we intend to further investigate which signal representation and routing allow CS to outperform random sampling in realistic WSN deployments. This requires to jointly investigate the design of the two matrices $\mathbf{\Phi}$ and $\mathbf{\Psi}$, since the requirements on sparsity and on the incoherence between routing and signal representation have to be met. With this in mind, in Chapter 4 we study the problem of recovering a real WSN signals and we exploit the temporal evolution of the signal, i.e., its temporal correlation. In this way, we can apply directly the cognition paradigm to our problem, learning a suitable sparsifing matrix Ψ and exploiting such matrix to recover the sampled signal.

3.8 Appendix: CS for 2D WSN signals

In this appendix, we review a known method from image processing to generalize the CS theory in Chapter 3 to 2D signals, as those gathered by the WSN of Section 3.4. Accordingly, the input signal is a $K \times K$ square matrix $\mathbf{X}$ with $N = K^2$ elements. Element (i, j) of this matrix, $x(i, j)$, is the value sampled by the sensor placed in cell (i, j) of the sensor grid. We assume that the 2D signal $\mathbf{X}$ is sparse under a given transformation. Thus, $\mathbf{X}$ can be written as

$$\mathbf{X} = \mathbf{BSA} \ , \tag{3.12}$$

where $\mathbf{B}$ and $\mathbf{A}$ are two non singular matrices and $\mathbf{S}$ is a $K \times K$ matrix representing the signal in the transformation domain.

In what follows, we use tools from linear algebra to reformulate the 2D problem as an equivalent 1D problem. It is worth noting that this transformation does not lose any information and preserves the correlation among sensed values in the 2D space.

Now we define a $\texttt{vec}(\cdot)$ function, transforming a $K \times K$ matrix into a vector of length N (through a reordering of the matrix elements)

$$\texttt{vec}(\mathbf{X}) = \big(x(1,1), \ldots, x(k,1), x(1,2), \ldots, x(k,2), \ldots, x(1,k), \ldots, x(k,k)\big)^T \ . \tag{3.13}$$

As explained in Section 3.5, the values that we collect at the DCP can be represented through a vector $\mathbf{y}$ of $M < N$ elements. They are linear combinations of the sensor readings represented by the matrix $\mathbf{X}$ of size $K \times K$, and thus $\mathbf{y} = \mathbf{\Phi}\texttt{vec}(\mathbf{X})$. The $M \times N$ matrix $\mathbf{\Phi}$ contains the combination coefficients that are picked at random according to a given distribution. From linear algebra we know that the vector form of a given product among three matrices $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{S}$ can be rewritten as [57]

$$\texttt{vec}(\mathbf{BSA}) = (\mathbf{A}^T \otimes \mathbf{B})\texttt{vec}(\mathbf{S}) \ , \tag{3.14}$$

where $\otimes$ is the Kronecker product. Hence, using (3.12) and (3.14) we can write $\texttt{vec}(\mathbf{X}) = (\mathbf{A}^T \otimes \mathbf{B})\texttt{vec}(\mathbf{S})$. Using $\mathbf{y} = \mathbf{\Phi}\texttt{vec}(\mathbf{X})$ we obtain $\mathbf{y} = \mathbf{\Phi}(\mathbf{A}^T \otimes \mathbf{B})\texttt{vec}(\mathbf{S})$ that, defining $\tilde{\mathbf{\Phi}} = \mathbf{\Phi}(\mathbf{A}^T \otimes \mathbf{B})$, can be rewritten as

$$\mathbf{y} = \tilde{\mathbf{\Phi}}\texttt{vec}(\mathbf{S}) \ , \tag{3.15}$$

where $\mathbf{y}$ is the vector containing the received (combined) values and $\texttt{vec}(\mathbf{S})$ is a column vector of length N containing the input signal in the transformation domain. Given (3.15)

we can recover the sparse signal $\texttt{vec}(\mathbf{S})$ using the solvers developed for standard CS theory in 1D.

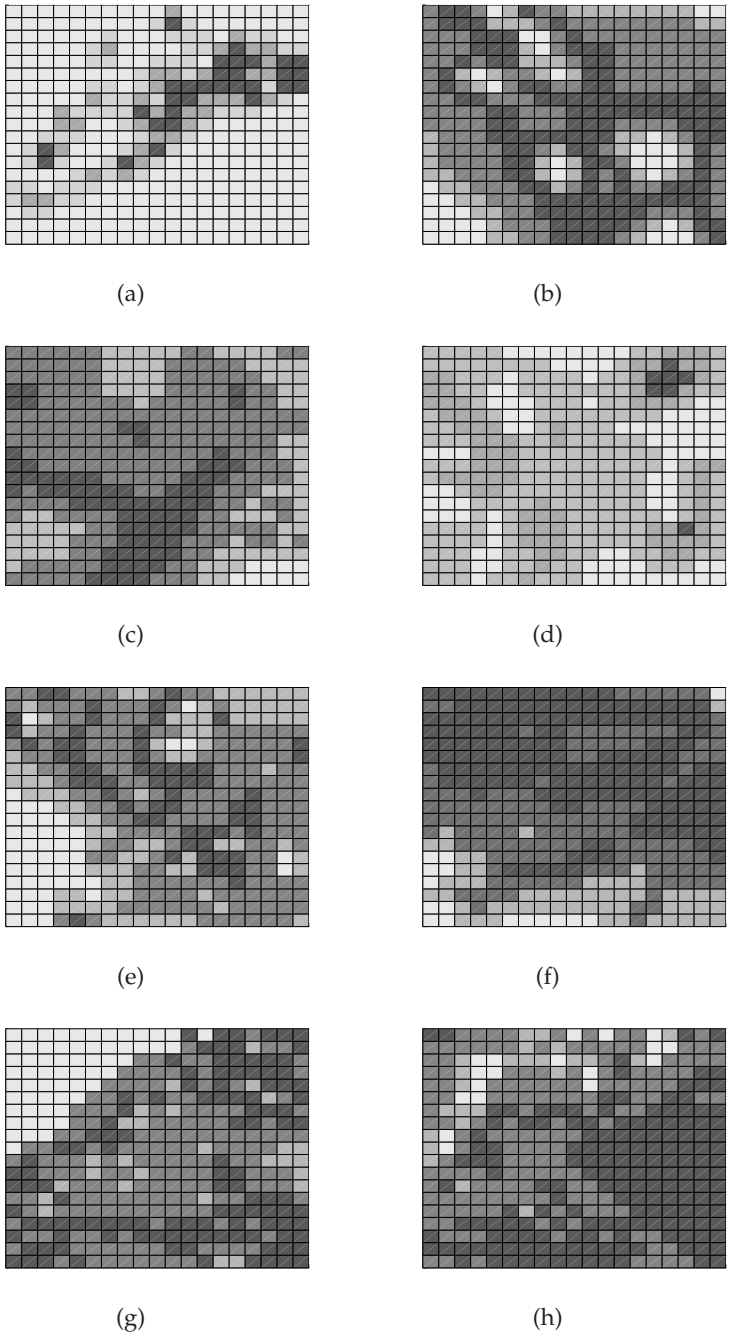


Figure 3.7. *Real signals: (a) Wi-Fi strength from MIT, (b) Wi-Fi strength from Stevens Institute of Technology, (c) Ambient temperature from EPFL SensorScope WSN, (d) Solar radiation from EPFL SensorScope WSN, (e) Rainfall in Texas, (f) Temperature of the ocean in California, (g) Level of pollution in Benelux and (h) in northern Italy.*

Chapter 4

Mathematical framework for monitoring WSN signals over time

In this chapter[1] we look at the problem of designing a mathematical framework for the compression, collection and reconstruction of real signals gathered from an actual Wireless Sensor Network (WSN). The approach is suitable to measure large amounts of data with high accuracy by only requiring the collection of a small fraction of the sensor readings. In the past few years, the research community has been providing interesting contributions on this topic.

In particular, we exploit Compressive Sensing (CS) [5–7], a convex optimization technique that has been detailed in Section 2.1, that takes advantage of the inherent correlation of the input data by means of quasi-random matrices. CS was originally developed for the efficient storage and compression of digital images, which show high spatial correlation. Since the pioneering work of Nowak [8, 10, 36, 58], there has been a growing interest in this

[1] The material presented in this chapter has been published in:

[C5] R. Masiero, **G. Quer**, M. Rossi and M. Zorzi, "A Bayesian Analysis of Compressive Sensing Data Recovery in Wireless Sensor Networks", *The International Workshop on Scalable Ad Hoc and Sensor Networks, SASN'09*, Saint Petersburg, Russia, Oct. 2009.

[J1] **G. Quer**, R. Masiero, M. Rossi and M. Zorzi, "SCoRe1: Sensing Compression and Recovery through Online Estimation for Wireless Sensor Networks", *Under submission to IEEE Trans. Wireless Communication*.

[J2] R. Masiero, **G. Quer**, G. Pillonetto, M. Rossi and M. Zorzi, "Sampling and Recovery with Compressive Sensing in Real Wireless Sensor Networks", *Under submission to IEEE Trans. Wireless Communication*.

technique also by the networking community. In contrast to classical approaches, where the data is first compressed and then transmitted to a given Data Collection Point (DCP), with CS the compression phase can be jointly executed with data transmission.

In Chapter 3 we considered a data field sensed once by the WSN and that should be reconstructed at the DCP. In this chapter and in Chapter 5 instead we deal with a time varying signal that is monitored over time by a WSN. In this case, we exploit the spatial and temporal correlation characteristics of the signal, by learning at the DCP the relevant statistics needed by CS to work properly, through the use of Principal Component Analysis (PCA), that is detailed in Section 2.1.3. Then we exploit the learned statistics to recover the signal through CS from its sampled version that is received at the DCP. Moreover, we propose a mathematical framework capable of exploiting CS and PCA, and we depict the probabilistic relations among all the variables involved through a Bayesian Network (BN), whose characteristics are described in Section 2.2.1.

In order to analyze our framework, we consider different WSN testbeds whose data is available on-line. We analyze the statistics of the principal components of the signals gathered by these WSNs, designing a probabilistic model that approximates the distribution of the principal components. To this aim we exploit Bayesian theory, which provides a general framework for data modeling [59, 60]. The Bayesian approach, in fact, has been addressed in the very recent literature to develop efficient and auto-tunable algorithms for CS, see [61]. However, previous work addressing CS from a Bayesian perspective has mainly been focused on the theoretical derivation of CS and its usefulness in the image processing field. In this chapter, instead, we provide empirical evidence of the effectiveness of CS in an actual WSN monitoring scenario. Moreover, we identify the conditions under which the recovery through convex optimization (CS) is optimal, i.e., it is equivalent to the Maximum A Posteriori (MAP) approach. We conclude the chapter with a brief analysis of the recovery performance with a simple data gathering technique. The integration of the proposed mathematical framework into a Data Collection and Recovery technique for WSN signals is described in detail in Chapter 5, where we compare this technique with similar data collection and with standard data recovery techniques.

The rest of the chapter is structured as follows: in Section 4.1 we show the mathematical details to jointly use CS and PCA, in Section 4.2 we propose the framework to exploit these two techniques, then we analyze a large number of WSN testbeds and signals in Section 4.3. In Section 4.4 we design the probabilistic model, that exploits a two level Bayesian infer-

ence to study the principal components of real signals, and in Section 4.5 we identify the optimality conditions for the recovery with CS. Section 4.6 concludes the chapter.

4.1 Joint Compressive Sensing and Principal Component Analysis

In this section we combine the information learned with Principal Component Analysis (PCA) with the Compressive Sensing (CS) technique. This scheme is exploited to solve the system in (2.4) after L data packets have been collected from the WSN, and PCA is the technique to learn the transformation matrix $\mathbf{\Psi}$ from the past data.

In Chapter 3 we use CS for the recovery of 2D real signals by considering different transformation matrices. However, we notice that none of them is sufficiently good in terms of

1. sparsification and
2. incoherence with respect to $\mathbf{\Phi}$.

In summary, while the results[2] obtained for synthetic signals are very promising, those achieved for real signals are unsatisfactory. In this section we solve this issue showing that the theoretical performance benefits of CS can still be retained if we use PCA to build the transformation matrix needed by CS.

We have seen in Section 2.1.3 that PCA is a method for representing through the best M-term approximation a generic N-dimensional signal, where $N > M$, and in Section 2.1.1 we have introduced CS, a technique to recover an N-dimensional signal through the reception of a small number of samples L, with $L < N$. In this section we propose a technique that jointly exploits PCA and CS to reconstruct a signal $\mathbf{x}^{(k)}$ at each time k, assuming that the signal is correlated both in time and in space, but that in general it is non-stationary. This means that the statistics that we have to use in our solution (i.e., sample mean and covariance matrix, in (2.21) and (2.21), respectively) must be learned at runtime and might not be valid throughout the entire time frame in which we want to reconstruct the signal. We should also make the following assumption, that will be justified in the next sections:

1. at each time k we have perfect knowledge of the previous K process samples, namely we perfectly know the set $\mathcal{T}^{(k)} = \left\{\mathbf{x}^{(k-1)}, \mathbf{x}^{(k-2)}, \cdots, \mathbf{x}^{(k-K)}\right\}$, referred to in what follows as training set;[3]

[2] In terms of reconstruction error *vs* number of transmissions.

[3] In Chapter 5 we present a practical scheme that does not need this assumption in order to work.

2. there is a strong temporal correlation between $\mathbf{x}^{(k)}$ and the set $\mathcal{T}^{(k)}$ that will be explicated in the next session via a Bayesian network. The size K of the training set is chosen according to the temporal correlation of the observed phenomena to validate this assumption.

Using PCA, from Eq. (2.23) at each time k we can map our signal $\mathbf{x}^{(k)}$ into a sparse vector $\mathbf{s}^{(k)}$. The matrix $\mathbf{U}$ and the average $\overline{\mathbf{x}}$ can be thought of as computed iteratively from the set $\mathcal{T}^{(k)}$, at each time sample k. Accordingly, at time k we indicate matrix $\mathbf{U}$ as $\mathbf{U}^{(k)}$ and we refer to the temporal mean and covariance of $\mathcal{T}^{(k)}$ as $\overline{\mathbf{x}}^{(k)}$ and $\widehat{\mathbf{\Sigma}}^{(k)}$, respectively. Hence, we can write:

$$\mathbf{x}^{(k)} - \overline{\mathbf{x}}^{(k)} = \mathbf{U}^{(k)}\mathbf{s}^{(k)} \ . \tag{4.1}$$

Now, using Eqs. (2.1) and (4.1), we can write:

$$\mathbf{y}^{(k)} - \mathbf{\Phi}^{(k)}\overline{\mathbf{x}}^{(k)} = \mathbf{\Phi}^{(k)}(\mathbf{x}^{(k)} - \overline{\mathbf{x}}^{(k)}) = \mathbf{\Phi}^{(k)}\mathbf{U}^{(k)}\mathbf{s}^{(k)} \ , \tag{4.2}$$

where with the symbol $\mathbf{\Phi}^{(k)}$ we make explicit that also the routing matrix $\mathbf{\Phi}$ can change over time. The form of Eq. (4.2) is similar to that of (3.3) with $\tilde{\mathbf{\Phi}} = \mathbf{\Phi}^{(k)}\mathbf{U}^{(k)}$. The original signal $\mathbf{x}^{(k)}$ is approximated as follows:

1. finding a good estimate[4] of $\mathbf{s}^{(k)}$, namely $\widehat{\mathbf{s}}^{(k)}$, using the techniques in [6] or [46] and

2. applying the following calculation:

$$\widehat{\mathbf{x}}^{(k)} = \overline{\mathbf{x}}^{(k)} + \mathbf{U}^{(k)}\widehat{\mathbf{s}}^{(k)} \ , \tag{4.3}$$

   where $\widehat{\mathbf{x}}^{(k)}$ is the approximation of the original signal $\mathbf{x}^{(k)}$.

4.2 Mathematical Framework for monitoring and Sparse Signal Model

In this section we describe a model to represent a broad range of environmental signals that can be gathered from a Wireless Sensor Network (WSN). The aim is to analyze the stochastic properties of these signals, in order to select the most appropriate sampling, compression and recovery techniques to minimize the number of transmitting nodes while keeping a certain level of reconstruction accuracy, as detailed in Section 3.6.

[4] In this chapter we refer to a good estimate of $\mathbf{s}^{(k)}$ as $\widehat{\mathbf{s}}^{(k)}$ such that $\|\mathbf{s}^{(k)} - \widehat{\mathbf{s}}^{(k)}\|_2 \leq \epsilon$. Note that by keeping ϵ arbitrarily small, assumption 1) above is very accurate.

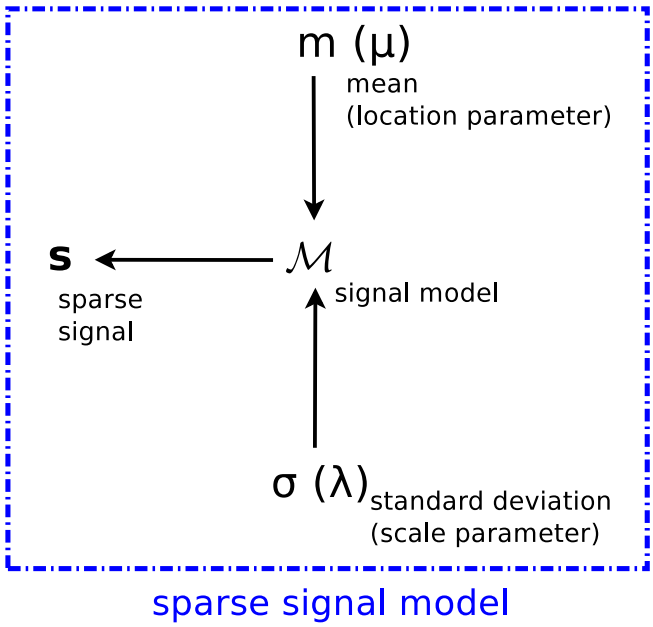


Figure 4.1. *Bayesian network used to model the probability distribution of the innovation signal* $\mathbf{s}$.

We have chosen to represent the variables involved with a Bayesian Network (BN), i.e., a Directed Acyclic Graph (DAG) where nodes represent random variables and arrows represent conditional dependencies among them, as detailed in Section 2.2. From the DAG it is always possible to determine the conditional independence between two variables, applying a set of rules known as *d-separation* rules, e.g., see [1] for a detailed description about BNs properties. In this section, we propose two graphical models which illustrate the perspective we adopted in Sec. 4.4 and Sec. 3.6, respectively:

1. Fig. 4.1 represents a stochastic model for the signal $\mathbf{s}$;

2. Fig. 4.2 is a BN which links together all the variables involved in our analysis, highlighting those required to define our monitoring framework.

In detail, with Fig. 4.1 we introduce a Bayesian model to describe the statistical properties of the elements of $\mathbf{s}^{(k)}$. Given the realizations of the signal $\mathbf{s}^{(k)}$ at time $k = 1, \ldots, K$, we use a Bayesian estimation method, described in Sec. 4.4, to infer a suitable model $\mathcal{M}$ along with the best-fitting values of its parameters. In particular, for a Gaussian model the parameters to infer are the mean value m of each component and the standard deviation σ, whereas for a Laplacian model are the location parameter μ and the scale parameter λ, respectively.

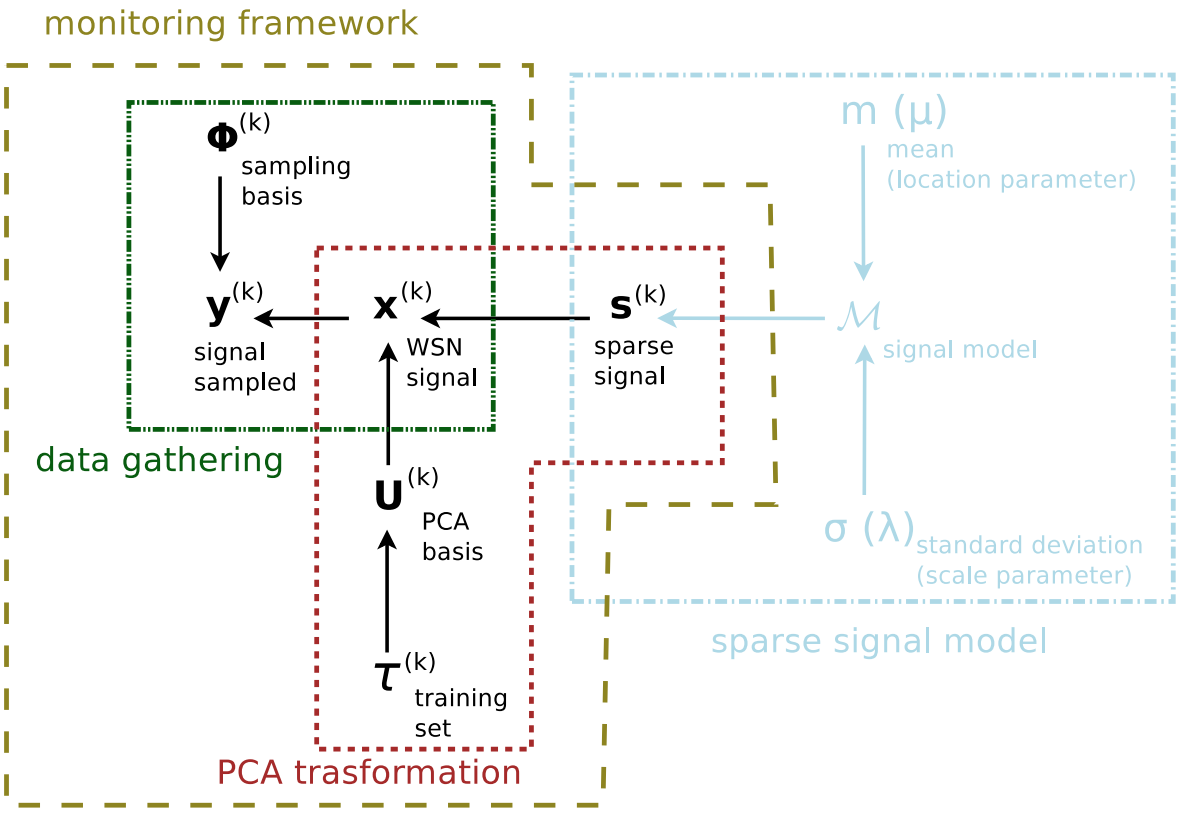


Figure 4.2. *Bayesian network used to model the considered real signals. In the scheme we highlight the monitoring framework at each time sample k.*

This modeling approach is exploited in Sec. 4.4 to determine which stochastic model, chosen among a set of plausible ones, better describes the signal $\mathbf{s}^{(k)}$.

Fig. 4.2, instead, depicts the whole considered framework that involves the following variables for each time sample k: the training set $\mathcal{T}^{(k)}$, the WSN signal $\mathbf{x}^{(k)}$, its compressed version $\mathbf{y}^{(k)}$, obtained sampling $\mathbf{x}^{(k)}$ according to matrix $\mathbf{\Phi}^{(k)}$, the invertible matrix $\mathbf{\Psi}^{(k)}$, obtained through PCA, and the sparse representation $\mathbf{s}^{(k)}$, as it has been detailed in Section 2.1. From the results presented in Sec. 4.4, it turns out that $\mathbf{s}^{(k)}$ is well approximated by a Laplacian distribution. Analyzing the DAG in Fig. 4.2, based on the *d-separation* rules, we can make the following observations:

- **data gathering**: the WSN signal $\mathbf{x}^{(k)}$ is independent of the stochastic sampling matrix $\mathbf{\Phi}^{(k)}$, whose nature is described in Chapter 5, but the observation of $\mathbf{y}^{(k)}$ reveals a link between these two variables;

- **PCA transformation**: this is the core of our model, that describes how the system

learns the statistics of the signal of interest $\mathbf{x}^{(k)}$. According to the dynamic system framework, $\mathbf{\Psi}^{(k)}$ can be seen as the state of a system, since it summarizes at each instant k all the past history of the system, represented by the matrix $\mathcal{T}^{(k)}$. The system input is the signal $\mathbf{s}^{(k)}$, that can be seen as a Laplacian or Gaussian innovation process. This type of priors on the signal induces estimators that use, respectively, the L_1 and L_2 norm of the signal as regularization terms. Hence, such priors are often used in the literature in view of the connection with powerful shrinkage methods such as ridge regression and LASSO, as well as for the many important features characterizing them, e.g., see Section 3.4 in [62] for a thorough discussion. Note also that the observation of the WSN signal $\mathbf{x}^{(k)}$ has a twofold effect: the former is the creation of a deterministic dependence between the PCA basis $\mathbf{\Psi}^{(k)}$ and the sparse signal $\mathbf{s}^{(k)}$, that otherwise are independent; the latter is the separation of $\mathbf{\Psi}^{(k)}$ and $\mathbf{s}^{(k)}$ from the data gathering variables, i.e., they become independent of $\mathbf{y}^{(k)}$ and $\mathbf{\Phi}^{(k)}$;

- **sparse signal model**: we observe that the priors assigned to the variable $\mathcal{M}$ and to the corresponding parameters μ (resp. m) and λ (resp. σ) are non informative, except for the non-negativity of the variance. Here the observation of the sparse signal $\mathbf{s}^{(k)}$ separates the sparse signal model from the monitoring framework, i.e., after observing the signal $\mathbf{s}^{(k)}$, the variable $\mathcal{M}$ and the corresponding parameters μ (resp. m) and λ (resp. σ) will no longer be dependent on the variables of the monitoring framework, so they can be analyzed separately as we do in Sec. 4.4.

In the next section we will describe the real world signals that will be used to develop a statistical analysis on the principal component distribution and from which we obtain a set of realizations for the signal $\mathbf{s}^{(k)}$. In Sec. 4.4 we will show that the Laplacian is a good model to represent the principal components of typical WSN data. In turn, this provides a justification for using CS in WSNs, as detailed in Section 4.5.

4.3 WSN online testbeds

The ultimate aim of WSN deployments is to monitor the evolution of a certain physical phenomenon over time. Examples of applications that require such infrastructure include monitoring for security, health-care or scientific purposes. Many different types of signals can be sensed, processed and stored, e.g., the motion of objects and beings, the heart beats,

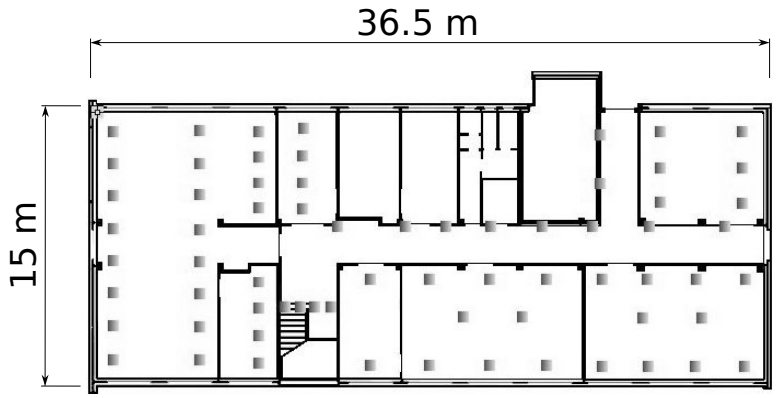


Figure 4.3. *Layout of the WSN testbed at the University of Padova.*

or environmental signals like the values of temperature and humidity, indoor or outdoor. Very often the density of sensor network deployments is very high and therefore sensor observations are strongly correlated in the space domain. Furthermore, the physics itself of the observed signals makes consecutive observations of a sensor node to be also temporally correlated.

The spatial and temporal correlation represents a huge potential that can be exploited designing collaborative protocols for the nodes constituting a WSN. In this perspective, we can think of reducing the energy consumption of the network by tuning the overall number of transmissions required to monitor the evolution of a given phenomenon over time. The appeal of the techniques presented in Chapter 2 follows from the fact that CS enables us to significantly reduce the number of samples needed to estimate a signal of interest with a certain level of quality. Clearly, the effectiveness of CS is subject to the knowledge of a transformation basis for which the observed signals result sparse.

In this section we illustrate the WSNs and the gathered signals that will be used in Section 4.4 to test, using the Bayesian framework presented in Section 4.2, whether CS and PCA are effective for real signals, i.e., whether the real signal transformed by the PCA matrix is actually sparse.

Networks. In addition to our own experimental network deployed on the ground floor of the Department of Information Engineering at the University of Padova, we consider other three WSNs whose sensor reading databases are available on-line, and a further deployment called Sense&Sensitivity, whose data has been kindly provided to the authors by Dr. Thomas

Watteyne of the Dust Networks, Incorporation. A brief technical overview of each of these five experimental network scenarios follows.

T1 WSN testbed of the Department of Information Engineering (DEI) at the University of Padova, shown in Figure 4.3. The WSN is collecting data from 68 TmoteSky wireless sensor nodes [63], [64]. The node hardware features an IEEE 802.15.4 Chipcon wireless transceiver working at 2.4 GHz and allowing a maximum data rate of 250 Kbps. These sensors have a TI MSP430 micro-controller with 10 Kbytes of RAM and 48 Kbytes of internal FLASH;

T2 LUCE (Lausanne Urban Canopy Experiment) WSN testbed at the Ecole Polytechnique Fédérale de Lausanne (EPFL), [50]. This measurement system exploits 100 SensorScope weather sensors which have been deployed across the EPFL campus. The node hardware is based on a TinyNode module equipped with a Xemics XE1205 radio transceiver operating in the 433, 868 and 915 MHz license-free ISM (Industry Scientific and Medical) frequency bands. Also these sensors have a TI MSP430 micro-controller;

T3 St-Bernard WSN testbed at EPFL, [65]. This experimental WSN deployment is made of 23 SensorScope stations deployed at the Grand St. Bernard pass at 2400 m, between Switzerland and Italy. See point T2 for a brief description of the related hardware;

T4 CitySense WSN testbed, developed by Harvard University and BBN Technologies, [66]. CitySense is an urban scale deployment that will consist of 100 wireless sensor nodes equipped with an ALIX 2d2 single-board computer. The transmitting interface is reconfigurable by the user and by default it operates in 802.11b/g ad hoc mode at 2.4 GHz. Nowadays this WSN deployment counts about twenty nodes;

T5 The Sense&Sensitivity [67] testbed is a WSN of 86 WSN430 nodes, which embed Texas technology: a MSP430 micro-controller and a CC1100 radio chip operating in the ISM band (from 315 to 915 MHz).

Signals. From the above WSNs, we gathered seven different types of signals: **S1)** temperature; **S2)** humidity; **S3-S4)** luminosity in two different ranges ($320 - 730$ and $320 - 1100$ nm, respectively); **S5)** wind direction; **S6)** voltage and **S7)** current. Concerning the signals gathered from our testbed T1, we collected measurements from all nodes every 5 minutes for 3 days. We repeated the data collection for three different measurement campaigns, choosing

WSN Testbed T1 (DEI)						
	# of nodes	**frame length**	**# of frames**	**starting time (G.M.T)**	**stopping time (G.M.T)**	**signals**
Camp. A	37	5 min	783	13/03/2009, 09:05:22	16/03/2009, 18:20:28	S1 S2 S3 S4 S6
Camp. B	45	5 min	756	19/03/2009, 10:00:34	22/03/2009, 17:02:54	S1 S2 S3 S4 S6
Camp. C	31	5 min	571	24/03/2009, 11:05:10	26/03/2009, 10:15:42	S1 S2 S3 S4 S6
WSN Testbed T2 (EPFL LUCE)						
	# of nodes	**frame length**	**# of frames**	**starting time (G.M.T)**	**stopping time (G.M.T)**	**signals**
Camp. A	85	5 min	865	12/01/2007, 15:09:26	15/01/2007, 15:13:26	S1 S2 S5
Camp. B	72	5 min	841	06/05/2007, 16:09:26	09/05/2007, 14:13:26	S1 S2 S5
Camp. C	83	30 min	772	02/02/2007, 17:09:26	18/02/2009, 19:09:26	S6 S7
WSN Testbed T3 (EPFL St Bernard)						
	# of nodes	**frame length**	**# of frames**	**starting time (G.M.T)**	**stopping time (G.M.T)**	**signals**
Camp. A	23	5 min	742	03/10/2007, 12:35:37	06/10/2007, 02:35:37	S1 S2 S5
Camp. B	22	5 min	756	19/10/2007, 12:35:37	22/10/2007, 03:35:37	S1 S2 S5
Camp. C	22	30 min	778	02/10/2007, 07:06:05	19/10/2007, 12:06:50	S6 S7
WSN Testbed T4 (CitySense)						
	# of nodes	**frame length**	**# of frames**	**starting time (G.M.T)**	**stopping time (G.M.T)**	**signals**
Camp. A	8	60 min	887	14/10/2009, 14:01:57	21/11/2009, 00:01:57	S1
Camp. B	8	60 min	888	14/10/2009, 13:00:01	21/11/2009, 00:00:01	S5
WSN Testbed T5 (Sense&Sensitivity)						
	# of nodes	**frame length**	**# of frames**	**starting time (G.M.T)**	**stopping time (G.M.T)**	**signals**
Camp. A	77	15 min	65	26/08/2008, 14:46:46	27/08/2008, 07:31:07	S1 S3 S4 S6

Table 4.1. *Details of the considered WSN and gathered signals.*

different days of the week. Regarding the data collection from WSNs T2–T5, we studied the raw data available on-line with the aim of identifying a portion of data that could be used as a suitable benchmark for our research purposes. This task has turned out to be really challenging due to packet losses, device failures and battery consumption that are very common and frequent in currently available technology. For the acquisition of the signals we divided the time axis in frames (or time slots) such that each of the working nodes was able to produce a new sensed data per frame. Details of the signals extracted from the records of T1–T5, and organized in different campaigns, are reported schematically in Table 4.1.

4.4 Sparsity Analysis of Real Signal Principal Components

In this section we aim to infer the statistical distribution of the vector random process $\mathbf{s}$ from the samples $\{\mathbf{s}^{(1)}, \mathbf{s}^{(2)}, \ldots, \mathbf{s}^{(T)}\}$ which are obtained from the above WSN signals. The parameter T is the duration (number of time samples) of each monitoring campaign in Table 4.1.

From the theory [13] we know that signals in the PCA domain (in our case $\mathbf{s}$) have in general uncorrelated components. Also, in our particular case we experimentally verified that this assumption is good since $E[s_i s_j] \simeq E[s_i]E[s_j]$ for $i, j \in \{1, \ldots, N\}$ and $i \neq j$. In our analysis, we make a stronger assumption, i.e., we build our model of $\mathbf{s}$ considering statistical independence among its components, i.e., $p(s_1, \ldots, s_N) = \prod_{i=1}^{N} p(s_i)$. A further assumption that we make is to consider the components of $\mathbf{s}$ as stationary over the entire monitoring period[5]. The model developed following this approach can be integrated into a data collection technique for WSNs and it leads to good results, as shown in Chapter 5, which allow us to validate these assumptions.

Owing to these assumptions, the problem of statistically characterizing $\mathbf{s}$ reduces to that of characterizing the random variables

$$s_i = \sum_{j=1}^{N} u_{ji}(x_j - \overline{x}_j) \ , \ i = 1, \ldots, N \ , \tag{4.4}$$

where the r.v. u_{ji} is an element of matrix $\mathbf{U}$ in eq. (4.1) and the r.v. x_j is an element of vector $\mathbf{x}$.

[5]Note that this model is able to follow also signals whose frequency content varies over time since the signal basis adapts to the data.

A statistical model for each s_i can be determined through the Bayesian estimation procedure detailed below. Similarly to the approach adopted in [68], we rely upon two levels of inference.

First level of inference. Given a set of competitive models $\{\mathcal{M}_1, \cdots, \mathcal{M}_N\}$ for the observed phenomenon, each of them depending on the parameter vector $\boldsymbol{\theta}$, we fit each model $\mathcal{M}_i$ to the collected data denoted by $\mathcal{D}$, i.e., we find the $\boldsymbol{\theta}_{\text{MAP}}$ that maximizes the a posteriori probability density function (pdf)

$$p(\boldsymbol{\theta}|\mathcal{D}, \mathcal{M}_i) = \frac{p(\mathcal{D}|\boldsymbol{\theta}, \mathcal{M}_i)p(\boldsymbol{\theta}|\mathcal{M}_i)}{p(\mathcal{D}|\mathcal{M}_i)} \ , \tag{4.5}$$

i.e.,

$$\boldsymbol{\theta}_{\text{MAP}} = \arg\max_{\theta} p(\boldsymbol{\theta}|\mathcal{D}, \mathcal{M}_i) \ , \tag{4.6}$$

where $p(\mathcal{D}|\boldsymbol{\theta}, \mathcal{M}_i)$ and $p(\boldsymbol{\theta}|\mathcal{M}_i)$ are known as the *likelihood* and the *prior* respectively, whilst the so called *evidence* $p(\mathcal{D}|\mathcal{M}_i)$ is just a normalization factor which plays a key role in the second level of inference.

Second level of inference. According to Bayesian theory, the most probable model is the one maximizing the posterior $p(\mathcal{M}_i|\mathcal{D}) \propto p(\mathcal{D}|\mathcal{M}_i)p(\mathcal{M}_i)$. Hence, when the models $\mathcal{M}_i$ are equiprobable, they are ranked according to their evidence. In general, evaluating the evidence involves the computation of analytically intractable integrals. For this reason, we rank the different models according to a widely used approximation, the Bayesian Information Criterion (BIC) [19], that we define as:

$$\text{BIC}(\mathcal{M}_i) \stackrel{def}{=} \ln\left[p(\mathcal{D}|\boldsymbol{\theta}_{MAP}, \mathcal{M}_i)p(\boldsymbol{\theta}_{MAP}|\mathcal{M}_i)\right] - \frac{\ell_i}{2}\ln(T) \ , \tag{4.7}$$

where $\boldsymbol{\theta}_{MAP}$ is defined in (4.6), ℓ_i is the number of free parameters of model $\mathcal{M}_i$ and T is the cardinality of the observed data set $\mathcal{D}$. Roughly speaking, the Bayesian Information Criterion (BIC) provides insight in the selection of the best fitting model penalizing those models requiring more parameters.

According to the introduced formalism we consider $\{\mathbf{s}^{(1)}, \mathbf{s}^{(2)}, \ldots, \mathbf{s}^{(T)}\}$ as the set of collected data $\mathcal{D}$; further, the observation of the experimental data gives empirical evidence for the selection of four statistical models $\mathcal{M}_i$ and corresponding parameter vectors $\boldsymbol{\theta}$:

$\mathcal{M}_1$ a Laplacian distribution with $\boldsymbol{\theta} = [\mu, \lambda]$, that we call $\mathcal{L}$;

$\mathcal{M}_2$ a Gaussian distribution with $\boldsymbol{\theta} = [m, \sigma^2]$, that we call $\mathcal{G}$;

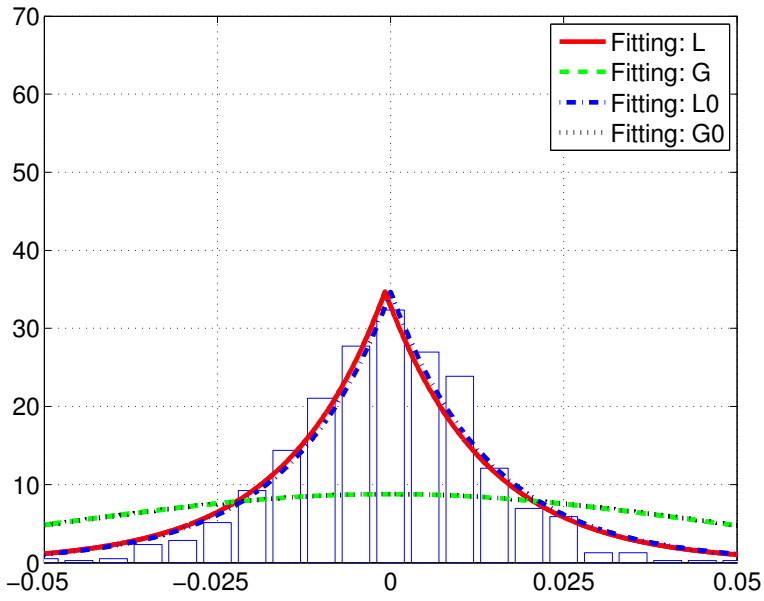


Figure 4.4. *Empirical distribution and model fitting for a principal component (the second) of signal S1, temperature.*

$\mathcal{M}_3$ a Laplacian distribution with $\mu = 0$ and $\theta = \lambda$, that we call $\mathcal{L}_0$;

$\mathcal{M}_4$ a Gaussian distribution with $m = 0$ and $\theta = \sigma^2$, that we call $\mathcal{G}_0$.

The space of models for each $\mathbf{s}_i$ is therefore described by the set $\{\mathcal{L}, \mathcal{G}, \mathcal{L}_0, \mathcal{G}_0\}$. In detail, for each signal $S1 - S7$ in the corresponding WSNs and campaigns of Table 4.1, we collected the $T + K$ signal samples $\left\{\mathbf{x}^{(1-K)}, \ldots, \mathbf{x}^{(-1)}, \mathbf{x}^{(0)}, \mathbf{x}^{(1)}, \ldots \mathbf{x}^{(T)}\right\}$ from which we computed $\left\{\mathbf{s}^{(1)}, \mathbf{s}^{(2)}, \ldots, \mathbf{s}^{(T)}\right\}$ according to what explained in Sec. 4.1. Then, for each component $s_i, i = 1, \ldots, N$, and for each model $\mathcal{M}_i, i = 1, \ldots, 4$, we have estimated the parameters (i.e., the most probable *a posteriori*, MAP) that best fit the data according to (4.5). These estimations are related to the BN in Fig. 4.1 and since we deal with Gaussian and Laplacian distributions, they have well known and closed form solutions [60]. In detail, for each component s_i:

$\mathcal{M}_1$ $\hat{\mu} = \mu_{1/2}(s_i)$ and $\hat{\lambda} = \frac{\sum_{k=1}^{T}\left|s_i^{(k)} - \hat{\mu}\right|}{T}$, where $\mu_{1/2}(s_i)$ is the median of the data set $\left\{s_i^{(1)}, \ldots, s_i^{(T)}\right\}$;

$\mathcal{M}_2$ $\hat{m} = \frac{\sum_{j=1}^{T} s_i^{(k)}}{T}$ and $\hat{\sigma}^2 = \frac{\sum_{k=1}^{T}\left(s_i^{(k)} - \hat{m}\right)^2}{T-1}$;

$\mathcal{M}_3$ $\hat{\lambda} = \frac{\sum_{k=1}^{T}\left|s_i^{(k)}\right|}{T}$;

$\mathcal{M}_4$ $\hat{\sigma}^2 = \frac{\sum_{k=1}^{T}\left(s_i^{(k)}\right)^2}{T}$.

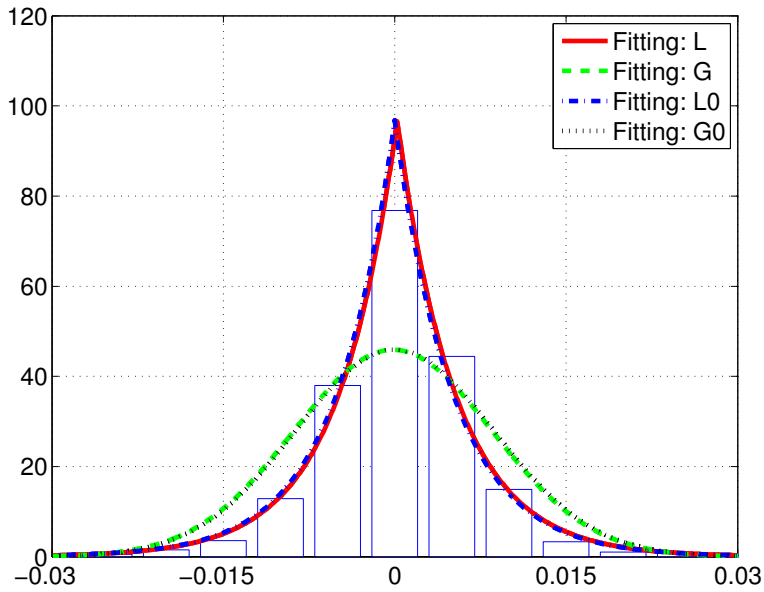


Figure 4.5. *Empirical distribution and model fitting for a principal component (the second) of signal S3, luminosity in the range* $320 - 730$ *nm.*

Figs. 4.4–4.5 show two examples of data fitting according to the aforementioned models; in these figures we plot the empirical distribution and the corresponding inferred statistical model for a generic principal component (but not the first one, as explained in the following) of the temperature (S1) and the luminosity (S3), respectively. Both these signals have been observed during the data collection of the campaign A, in the WSN testbed T1 (DEI). From the graphs in Figs. 4.4–4.5 we see that the distribution of the principal components of our signals is well described by a Laplacian distribution. Formally, the best among the four considered models can be determined ranking them according to the Bayesian Information Criterion (BIC) introduced in Eq. (4.7). Since we assigned non informative priors to the model parameters, $p(\boldsymbol{\theta}_{\text{MAP}}|\mathcal{M}_i)$ is a constant for each $\mathcal{M}_i$ and therefore the BIC can be redefined as:

$$\text{BIC}(\mathcal{M}_i) \stackrel{def}{=} \ln p(\mathcal{D}|\boldsymbol{\theta}_{MAP}, \mathcal{M}_i) - \frac{\ell_i}{2} \ln(T) \; . \tag{4.8}$$

Fig. 4.6 shows the BIC for the aforementioned humidity signal, for all its principal components and for all the considered models. From this figure we see that the Laplacian models better fit the data for all principal components s_i, $i = 1, 2, \ldots, N$. The average BIC for each model, for the different signals, campaigns and WSN tesbeds, is shown in Table 4.2. The values of this table are computed averaging over the N principal components. From these

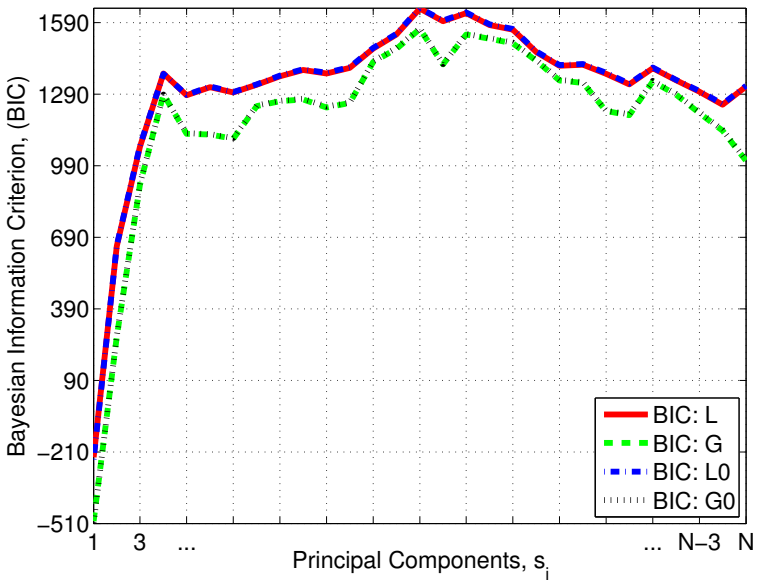


Figure 4.6. *Bayesian Information Criterion (BIC) per Principal Component, for each model $\mathcal{M}_1$–$\mathcal{M}_4$, WSN T1 (DEI), campaign A and signal S2, humidity.*

results we see that model $\mathcal{L}_0$ provides the best statistical description of the experimental data. In fact, the BIC metric is higher for Laplacian models in all cases; furthermore, $\mathcal{L}_0$ has a higher evidence with respect to $\mathcal{L}$, since it implies the utilization of a single parameter. As previously mentioned, the over-parameterization of the model is penalized according to the factor $T^{\frac{-\ell}{2}}$ (see Eq. (4.8)). Based on the above results, we can conclude that the Laplacian model describes slightly better than the Gaussian one the real signals' principal components obtained according to our proposed framework, for all the considered signals. Furthermore, it is worth noting that the first principal components (to be more precise, the first $K-1$ principal components[6] of the signal, where K is the training set length) have different statistics from the remaining ones, in terms of both signal range dynamics and amplitude of the components. This is due to the fact that the first $K-1$ components actually map the observed signal into the training set vector space, instead the remaining ones are random projections of the signals. The former capture the "core" of the signal $\mathbf{x}$, the latter allow to recover its details which can lie outside the linear span of the training data. In our simulations we set $K = 2$, in accordance to the rationale presented in the Appendix A.2, so that only the first

[6]Note that, according to Eq. (4.3), the matrix $\mathbf{U}^{(k)}$ is obtained from the elements of the training set $\mathcal{T}^{(k)}$ minus their mean, i.e., from the set $\left\{\mathbf{x}^{(k-1)} - \overline{\mathbf{x}}^{(k)}, \mathbf{x}^{(k-2)} - \overline{\mathbf{x}}^{(k)}, \cdots, \mathbf{x}^{(k-K)} - \overline{\mathbf{x}}^{(k)}\right\}$ which spans a vector space of dimension at most $K-1$.

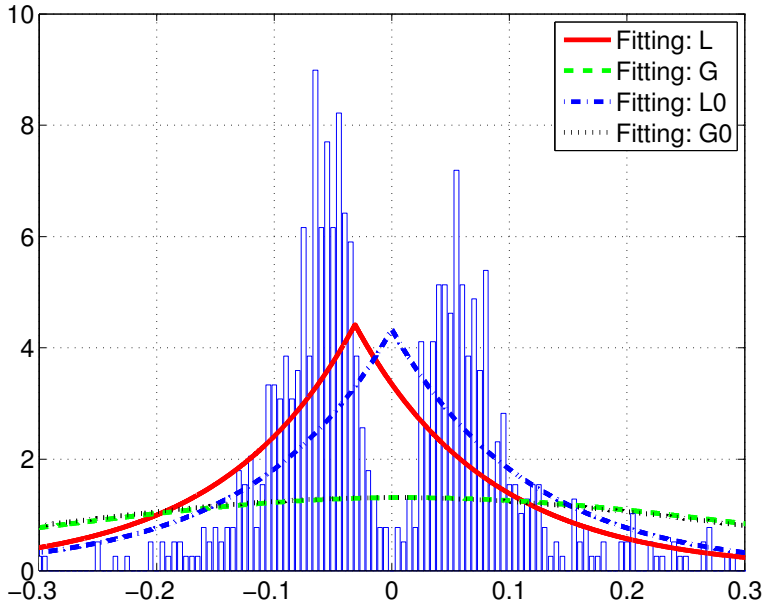


Figure 4.7. *Empirical distribution and model fitting for the first principal component of signal S1, temperature.*

principal component shows a behavior different from the one illustrated in Figs. 4.4–4.5 as reported in Figs. 4.7–4.8. In any case, the Laplacian model still fits better the observed data compared to the Gaussian one.

4.5 Bayesian MAP Condition and CS Recovery for Real Signals

In the previous section we have seen that the Laplacian model is a good representation for the principal components of typical WSN signals. This legitimates the use of CS in WSNs when it is exploited according to the framework presented in Section 4.2; to support this claim we review in this section a Bayesian perspective that highlights the equivalence between the output of the CS reconstruction algorithm and the solution that maximizes the posterior probability in Eq. (4.5).

Assume a Data Collection Point (DCP) is placed in the center of a WSN with N sensor nodes and let our goal be to determine at each time k all the N sensor readings by just collecting at the DCP a small fraction of them. To this end we exploit the joint CS and PCA scheme presented in Section 4.1. Eqs. (4.1)–(4.3) show that the considered framework does not depend on the particular topology considered; the only requirement is that the

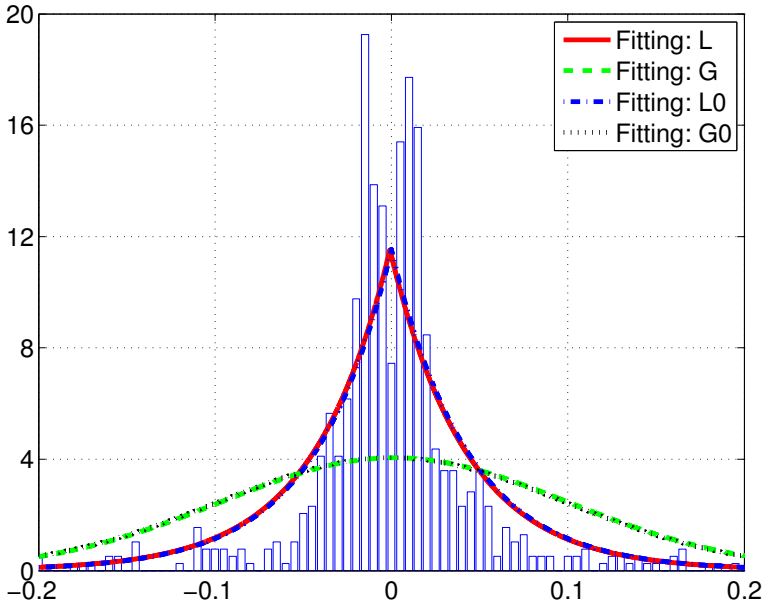


Figure 4.8. *Empirical distribution and model fitting for the first principal component of signal S3, luminosity in the range* $320 - 730$ *nm.*

sensor nodes be ordered (e.g., based on the natural order of their IDs). Our monitoring application can be seen, at each time k, as an interpolation problem: from a sampled M-dimensional vector $\mathbf{y}^{(k)} = \mathbf{\Phi}\mathbf{x}^{(k)} \in \mathbb{R}^M$, we are interested in recovering, via interpolation, the signal $\mathbf{x}^{(k)} \in \mathbb{R}^N$. Typically (e.g., see [68]) this problem can be solved through a linear interpolation on a set $\mathcal{F}$ of h basis functions $\mathbf{f}_i \in \mathbb{R}^N$, i.e., $\mathcal{F} = \{\mathbf{f}_1, \cdots, \mathbf{f}_h\}$. We can assume that the interpolated function has the form:

$$\mathbf{x}^{(k)} = \overline{\mathbf{x}}^{(k)} + \sum_{i=1}^{h} \theta_i \mathbf{f}_i \ . \tag{4.9}$$

In accordance to what explained in Section 2.1.3, at each time k we can do the following associations: the columns of the PCA matrix $\mathbf{U}^{(k)}$ as the set of $h = N$ basis functions, i.e., $\mathcal{F} = \{\mathbf{f}_1, \cdots, \mathbf{f}_N\} = \{\mathbf{u}_1^{(k)}, \cdots, \mathbf{u}_N^{(k)}\} = \mathcal{U}^{(k)}$; the sparse vector $\mathbf{s}^{(k)} = (s_1^{(k)}, \cdots, s_N^{(k)})^T$ as the parameter vector $\boldsymbol{\theta} = (\theta_1, \cdots, \theta_N)^T$. In this perspective the interpolated function has the form (see Eq. (4.1))

$$\mathbf{x}^{(k)} - \overline{\mathbf{x}}^{(k)} = \sum_{i=1}^{N} s_i^{(k)} \mathbf{u}_i^{(k)} \ . \tag{4.10}$$

A Bayesian approach would estimate the most probable value of $\mathbf{s}^{(k)} = (s_1^{(k)}, \cdots, s_N^{(k)})^T$ by maximizing a posterior pdf of the form $p(\mathbf{s}^{(k)} | \mathbf{y}^{(k)}, \mathcal{U}^{(k)}, \mathcal{M})$, where $\mathcal{M}$ is a plausible

model for the vector $\mathbf{s}^{(k)}$. To avoid confusion, it is important to note that in this section the interpretation of all the variables involved is slightly different from the one adopted in Sec. 4.4. In detail, now the vector $\mathbf{s}^{(k)}$ is seen as the parameter vector $\boldsymbol{\theta}$ in Eq. (4.5), whilst the vector $\mathbf{y}^{(k)}$ represents the set $\mathcal{D}$ of collected data. Moreover, the observed phenomenon $\mathbf{x}^{(k)}$ is modeled through both a set $\mathcal{U}^{(k)}$ of basis functions (i.e., the columns of the matrix $\mathbf{U}^{(k)}$) and a model $\mathcal{M}$ for the parameter vector $\mathbf{s}^{(k)}$, according to the BN in Fig. 4.2. In Eq. (4.5) we indicated with the symbol $\mathcal{M}_i$ a possible model for the observed phenomenon: here that symbol is replaced with the couple $(\mathcal{U}^{(k)}, \mathcal{M})$, where $\mathcal{M}$ refers directly to $\mathbf{s}^{(k)}$. Using the symbol $\mathcal{M}$ to indicate a model for $\mathbf{s}^{(k)}$ (even if $\mathbf{s}^{(k)}$ is now interpreted as the parameter vector θ) allows us to highlight the correspondence between the adoption of a particular model for $\mathbf{s}^{(k)}$ and the results of the study carried out in Sec. 4.4. This correspondence will become clear in the following.

As in [68], we assume also that $\mathcal{M}$ can be specified by a further parameter set α (called hyper-prior) related to $\mathbf{s}^{(k)}$, so that the posterior can be written as

$$p(\mathbf{s}^{(k)}|\mathbf{y}^{(k)}, \mathcal{U}^{(k)}, \mathcal{M}) = \int p(\mathbf{s}^{(k)}|\mathbf{y}^{(k)}, \alpha, \mathcal{U}^{(k)}, \mathcal{M}) p(\alpha|\mathbf{y}^{(k)}, \mathcal{U}^{(k)}, \mathcal{M}) \, \mathrm{d}\alpha \, .$$

If the hyper-prior can be inferred from the data and has non zero values $\hat{\alpha}$, maximizing the posterior corresponds to maximizing $p(\mathbf{s}^{(k)}|\mathbf{y}^{(k)}, \hat{\alpha}, \mathcal{U}^{(k)}, \mathcal{M})$, that as shown in [68] corresponds to maximizing the following expression

$$\begin{aligned} p(\mathbf{s}^{(k)}|\mathbf{y}^{(k)}, \mathcal{U}^{(k)}, \mathcal{M}) &\propto p(\mathbf{s}^{(k)}|\mathbf{y}^{(k)}, \hat{\alpha}, \mathcal{U}^{(k)}, \mathcal{M}) \\ &= \frac{p(\mathbf{y}^{(k)}|\mathbf{s}^{(k)}, \mathcal{U}^{(k)}) p(\mathbf{s}^{(k)}|\hat{\alpha}, \mathcal{M})}{p(\mathbf{y}^{(k)}|\hat{\alpha}, \mathcal{U}^{(k)}, \mathcal{M})} \, , \end{aligned} \tag{4.11}$$

where $p(\mathbf{y}^{(k)}|\mathbf{s}^{(k)}, \mathcal{U}^{(k)})$ represents the likelihood function, $p(\mathbf{s}^{(k)}|\hat{\alpha}, \mathcal{M})$ is the prior and $p(\mathbf{y}^{(k)}|\hat{\alpha}, \mathcal{U}^{(k)}, \mathcal{M})$ is a normalization factor. The parameters $\hat{\alpha}$ are estimated maximizing the evidence $p(\mathbf{y}^{(k)}|\alpha, \mathcal{U}^{(k)}, \mathcal{M})$, which is a function of α. Note that here the hyper-prior plays, in regard to $\mathbf{s}^{(k)}$, exactly the same role as the parameter vector $\boldsymbol{\theta}$ in the previous section, where $\mathbf{s}^{(k)}$ was interpreted as the collected data set $\mathcal{D}$ of the observed phenomenon; for example, if we choose $\mathcal{M} = \mathcal{L}_0$ for $\mathbf{s}^{(k)}$ then $\alpha = \lambda$, i.e., the hyper-prior is the scale parameter of the Laplacian prior assigned to $\mathbf{s}^{(k)}$.

In Eq. (4.10), without loss of generality we can assume that $\overline{\mathbf{x}}^{(k)} = 0$, thus the constraints on the relationship between $\mathbf{y}^{(k)}$ and $\mathbf{s}^{(k)}$ can be translated into a likelihood of the form (see Eq. (4.2)):

$$p(\mathbf{y}^{(k)}|\mathbf{s}^{(k)}, \mathcal{U}^{(k)}) = \delta(\mathbf{y}^{(k)}, \boldsymbol{\Phi}^{(k)} \mathbf{U}^{(k)} \mathbf{s}^{(k)}) \, , \tag{4.12}$$

where $\delta(x, y)$ is 1 if $x = y$ and zero otherwise. In Sec. 4.4, we have seen that the statistics of vector $\mathbf{s}^{(k)}$ is well described by a Laplacian density function with location parameter μ equal to 0 ($\mathcal{L}_0$). This pdf is widely used in the literature [46, 61] to statistically model sparse random vectors and, owing to the assumption of statistical independence of the components of $\mathbf{s}^{(k)}$, we can write it in the form:

$$p(\mathbf{s}^{(k)}|\hat{\alpha}, \mathcal{M} = \mathcal{L}_0) = \frac{e^{-\hat{\alpha}\sum_{i=1}^{N}|s_i^{(k)}|}}{(2/\hat{\alpha})^N} \ . \tag{4.13}$$

In this equation, all the components of $\mathbf{s}^{(k)}$ are assumed to be equally distributed. If (4.11) holds, we can obtain the following posterior:

$$\begin{aligned} p(\mathbf{s}^{(k)}|\mathbf{y}^{(k)}, \mathcal{U}^{(k)}, \mathcal{L}_0) &\propto p(\mathbf{s}^{(k)}|\mathbf{y}^{(k)}, \hat{\alpha}, \mathcal{U}^{(k)}, \mathcal{L}_0) \\ &\propto p(\mathbf{y}^{(k)}|\mathbf{s}^{(k)}, \mathcal{U}^{(k)}) p(\mathbf{s}^{(k)}|\hat{\alpha}, \mathcal{L}_0). \end{aligned} \tag{4.14}$$

Using (4.12)–(4.14), maximizing the posterior corresponds to solving the problem

$$\begin{aligned} & \arg\max_{\mathbf{s}^{(k)}} p(\mathbf{s}^{(k)}|\mathbf{y}^{(k)}, \mathcal{U}^{(k)}, \mathcal{L}_0) \\ = \ & \arg\max_{\mathbf{s}^{(k)}} p(\mathbf{y}^{(k)}|\mathbf{s}^{(k)}, \mathcal{U}^{(k)}) p(\mathbf{s}^{(k)}|\hat{\alpha}, \mathcal{L}_0) \\ = \ & \arg\max_{\mathbf{s}^{(k)}} \delta(\mathbf{y}^{(k)}, \mathbf{\Phi}^{(k)}\mathbf{U}^{(k)}\mathbf{s}^{(k)}) \frac{e^{-\hat{\alpha}\sum_{i=1}^{N}|s_i^{(k)}|}}{(2/\hat{\alpha})^N} \\ = \ & \arg\min_{\mathbf{s}^{(k)}} \sum_{i=1}^{N} |s_i^{(k)}|, \text{ given that } \mathbf{y}^{(k)} = \mathbf{\Phi}^{(k)}\mathbf{U}^{(k)}\mathbf{s}^{(k)} \\ = \ & \arg\min_{\mathbf{s}^{(k)}} \|\mathbf{s}^{(k)}\|_1, \text{ given that } \mathbf{y}^{(k)} = \mathbf{\Phi}^{(k)}\mathbf{U}^{(k)}\mathbf{s}^{(k)} \ , \end{aligned} \tag{4.15}$$

which is the convex optimization problem solved by the CS reconstruction algorithms (see [6] and [12]). In our approach, unlike in the classical CS problems, the sparsification matrix $\mathbf{U}^{(k)}$ is not fixed but varies over time adapting itself to the current data.

4.6 Conclusions

In this chapter we investigated the effectiveness of data recovery through joint Compressive Sensing (CS) and Principal Component Analysis (PCA) in Wireless Sensor Networks (WSNs). At first, we have shown that PCA is suitable for the Learn phase of the cognition paradigm, reaching the goal of capturing the relevant signal statistics from the past collected data. Then we have proposed a mathematical framework able to exploit both CS

and PCA for monitoring WSN signals, and we framed our recovery scheme into the context of Bayesian theory proving that, under certain assumptions on the signal statistics, the use of CS is legitimate, and we identified the conditions under which it is optimal, in terms of recovery performance. Hence, as the main contribution we have shown that these assumptions hold for real world data, which we gathered from several WSN deployments and processed according to our framework. This allows us to conclude that the learning phase using PCA is a good solution and the use of CS not only is legitimate in our recovery scheme but also makes it possible to obtain very good performance for the considered data sets. In the next chapter, we will define the compression/recovery technique that integrates this mathematical framework. Logically, this technique will play the role of the Plan and Decide phase of the cognition paradigm, as it will be able to make decisions on the future behavior of the WSN. Moreover, with the introduction of the feedback into the network, we will also define the Act phase, that will effect the decision made.

WSN Testbed T1 (DEI)							
	S1	S2	S3	S4	S5	S6	S7
$\mathcal{L}$	1382.8	1059.8	2191.7	1760.9	-	4656.9	-
$\mathcal{G}$	1042.1	804.9	1690.0	1154.5	-	3814.1	-
$\mathcal{L}_0$	**1385.5**	**1062.4**	**2194.9**	**1764.1**	-	**4660.1**	-
$\mathcal{G}_0$	1044.9	807.60	5078.3	1157.4	-	3816.9	-
WSN Testbed T2 (EPFL LUCE)							
	S1	S2	S3	S4	S5	S6	S7
$\mathcal{L}$	-36.1	-992.3	-	-	-3694.9	1854.1	-972.8
$\mathcal{G}$	-195.3	-1163.7	-	-	-4026.5	1191.4	-1520.3
$\mathcal{L}_0$	**-33.3**	**-989.5**	-	-	**-3691.5**	**1856.3**	**-969.6**
$\mathcal{G}_0$	-192.5	-1160.9	-	-	-4023.6	1194.2	-1517.3
WSN Testbed T3 (EPFL St Bernard)							
	S1	S2	S3	S4	S5	S6	S7
$\mathcal{L}$	-82.3	-1473	-	-	-3700.2	1617.8	-1557.5
$\mathcal{G}$	-487.4	-1700.7	-	-	-3850.3	1087.9	-1877.2
$\mathcal{L}_0$	**-79.3**	**-1469.9**	-	-	**-3697.3**	**1619.0**	**-1554.2**
$\mathcal{G}_0$	-484.7	-1697.8	-	-	-3847.5	1090.7	-1874.2
WSN Testbed T4 (CitySense)							
	S1	S2	S3	S4	S5	S6	S7
$\mathcal{L}$	-858.1	-	-	-	-4309.5	-	-
$\mathcal{G}$	-1094.6	-	-	-	-4384.2	-	-
$\mathcal{L}_0$	**-856.8**	-	-	-	**-4306.4**	-	-
$\mathcal{G}_0$	-1091.9	-	-	-	-4381.2	-	-
WSN Testbed T5 (Sense&Sensitivity)							
	S1	S2	S3	S4	S5	S6	S7
$\mathcal{L}$	-127.7	-	-196.2	-184.4	-	110.0	-
$\mathcal{G}$	-176.1	-	-232.1	-227.5	-	70.2	-
$\mathcal{L}_0$	**-125.7**	-	**-194.2**	**-182.3**	-	**111.9**	-
$\mathcal{G}_0$	-174.7	-	-230.6	-225.8	-	71.8	-

Table 4.2. *Bayesian Information Criterion (BIC) averaged over all Principal Components and relative campaigns, for each model $\mathcal{M}_1$–$\mathcal{M}_4$, for each testbed T1–T5 and each corresponding provided signal among S1 (Temperature), S2 (Humidity), S3 (Light), S4 (IR), S5 (Wind), S6 (Voltage) and S7 (Current).*

Chapter 5

SCoRe1: Sensing, Compression and Recovery through OnNline Estimation for WSNs

5.1 Introduction and Related Work

In this chapter [1] we present a lightweight and self-adapting framework called "Sensing, Compression and Recovery through ONline Estimation", *SCoRe1*, for the estimation of large data sets with high accuracy through the collection of a small number of sensor readings. SCoRe1 integrates the mathematical framework proposed in Chapter 4 into a data compression, collection and recovery technique designed to work in a WSN. The mathematical framework at the core of SCoRe1, presented in Section 4.2, is based on the joint

[1] The material presented in this chapter has been published in:

[C2] **G. Quer**, D. Zordan, R. Masiero, M. Zorzi and M. Rossi, "WSN-Control: Signal Reconstruction through Compressive Sensing in Wireless Sensor Networks", *IEEE LCN (SenseApp Workshop)*, Denver, CO, Oct. 2010.

[C4] R. Masiero, **G. Quer**, D. Munaretto, M. Rossi, J. Widmer and M. Zorzi, "Data Acquisition through joint Compressive Sensing and Principal Component Analysis", *IEEE Globecom 2009*, Honolulu, HW, Nov.-Dec. 2009.

[J1] **G. Quer**, R. Masiero, M. Rossi and M. Zorzi, "SCoRe1: Sensing Compression and Recovery through Online Estimation for Wireless Sensor Networks", *Under submission to IEEE Trans. Wireless Communication*.

use of Compressive Sensing (CS) and Principal Component Analysis (PCA) [13], to devise a scheme where the processing of the signal is only required at the Data Collection Point (DCP), whereas data gathering and routing are independent of it. The main objective of such framework is to be very general, i.e., suitable to be implemented for a monitoring application independently of the observed signal. This requirement is very appealing when we think about a network of nodes equipped with different sensors, and therefore capable of sensing different signals. We do not want protocols specifically designed for signals with given statistical characteristics, so that a node should select the right protocol according to the currently sensed signal. Conversely, we would like to have a transmission protocol that requires no prior knowledge of the observed signal characteristics, but nevertheless is able to adapt to them. We stress that SCoRe1 is proposed for WSNs, but can be readily applied to other types of network infrastructures that require the approximation of large and distributed datasets with spatial or temporal correlation. Furthermore, this technique fulfills the tasks of the Plan and Decide phase, and, with the presence of the feedback to the WSN, also of the Act phase of the cognitive paradigm.

We present also two other simple data collection techniques that exploit the mathematical framework of Chapter 4, in order to show the reason for some specific design choices for SCoRe1, like the adoption of a controller to bound the signal error and to iteratively adapt to the specific statistical characteristics of the signal monitored. Moreover, we exploit a wide set of signal processing techniques for the approximation of the signal in space and time and we devise an original data gathering scheme that autonomously adapts the number of collected measurements according to a runtime estimate of the signal reconstruction error. In detail, the signal of interest can be approximated according to (i) a deterministic approach, i.e., through a proper fitting of the collected measurements, as in the case of the spline method, and to (ii) a probabilistic approach, i.e., the signal is estimated from the collected measurements and some a priori statistical knowledge of the signal, as in the case of CS or the Least Square Error (LSE) method. Chapter 4 showed that CS recovery can be exploited for data interpolation since the signal model which is at the basis of CS describes well real signal characteristics. Here we apply CS within an actual framework for WSN signal monitoring and show that it performs as well as or better than the other state of the art techniques analyzed. Finally, we propose WSN-control, an architecture that integrates data compression, collection and recovery into an actual WSN. The recovery technique is implemented in an application server that can be accessed by the Internet ant that is connected to

the WSN through a WSN gateway.

Our present work is related to the literature on signal recovery, e.g., see [69], and in particular to Bayesian theory, e.g., see [59, 60], and the main contributions of this chapter with respect to the state of the art are:

- the design of a novel, effective and flexible technique for distributed sampling, data gathering and recovery of signals from actual WSN deployments, named SCoRe1, that integrates the mathematical framework described in Chapter 4;
- the proposal of the WSN-Control architecture to access and control all the operations in a WSN from an application server external to the WSN and connected to the Internet, and the integration of the proposed compression/recovery technique into such architecture;
- a performance comparison of different data collection techniques that exploit the proposed framework;
- a performance comparison of different data-fitting techniques with real signals in realistic WSN scenarios.

The rest of this chapter is structured as follows. In Section 5.2 we describe our data collection framework, the distributed sampling method and the controller module to bound the signal reconstruction error. Here we present also two alternative data collection techniques to illustrate our choices of protocol design. In Section 5.3 we give a mathematical overview of the main approaches to recover data from an incomplete measurement set, followed by the description of the recovery algorithms to apply these approaches in our monitoring framework. Then, in Section 5.4 we present the WSN-control software architecture in which we integrate the data collection techniques. In Section 5.5 we analyze the performance of the proposed schemes and recovery techniques for different kinds of real signals gathered from different WSNs. Finally, Section 5.6 concludes the chapter.

5.2 Iterative Monitoring Framework

In this section we present our data collection framework called SCoRe1 (Sensing, Compression an Recovery through ON-line Estimation) for distributed compression and centralized recovery of a multi dimensional signal in a WSN with N sensor nodes. A diagram

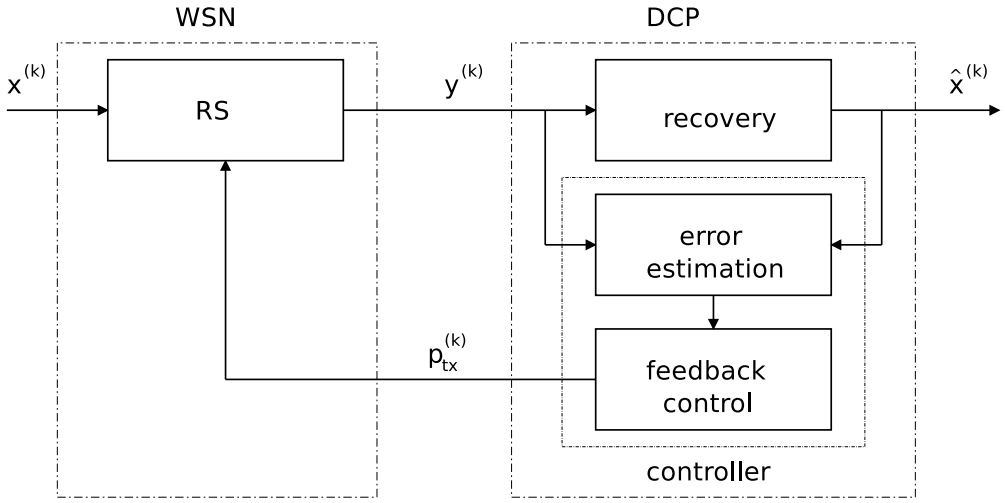


Figure 5.1. *Diagram of the proposed sensing, compression and recovery scheme. Note that the* Controller*, which includes the* Error estimator *and the* Feedback Control *blocks, is a characteristic of* SCoRe1 *and is not present in the other DC techniques.*

showing the logic blocks of this framework is presented in Fig. 5.1 and will be detailed in Section 5.2.1. Let $\mathbf{x}^{(k)} \in \mathbb{R}^N$ be the N-dimensional signal (one reading per sensor node) sampled at discrete times $k = 1, 2, \ldots$. At each time k the Data Collection Point (DCP) collects a compressed version $\mathbf{y}^{(k)} = \mathbf{\Phi}^{(k)}\mathbf{x}^{(k)}$, $\mathbf{y}^{(k)} \in \mathbb{R}^L$, of the original signal $\mathbf{x}^{(k)} \in \mathbb{R}^N$. The *sampling* matrix $\mathbf{\Phi}^{(k)} \in \mathbb{R}^{L\times N}$, with $L \leq N$, has one element equal to 1 per row and at most one element equal to 1 per column, indicating which nodes transmit their data sample to the DCP at time k, while all the other elements are equal to zero. Thus, the elements in $\mathbf{y}^{(k)} \in \mathbb{R}^L$ are a subset of those in $\mathbf{x}^{(k)}$ (spatial sampling). Note that reducing the number of nodes that transmit to the DCP is a key aspect as each sensor is supposed to be a tiny battery powered unit with a finite amount of energy that determines its lifetime. Furthermore, the nodes of the WSN have limited computational resources, so it is necessary to design a compression technique which does not require much computational power at the nodes and which only needs a minimum amount of control messages to be exchanged among them. At each time k the transmitting nodes are chosen in a distributed way according to a simple Random Sampling (RS) technique to be executed in each node of the WSN, as we detail shortly. The DCP is the sink of the WSN and is connected to a remote server that is not battery powered so it does not have stringent energy requirements and has enough computational resources to execute signal recovery algorithms. The DCP is responsible for collecting the compressed data $\mathbf{y}^{(k)}$, sending a feedback to the WSN and recovering the original signal from $\mathbf{y}^{(k)}$. Ac-

cording to the IDs of the nodes that have transmitted, which are included in the received packets, the DCP obtains a further matrix $\mathbf{\Phi}^{(k)}$ that will be used in the recovery phase. Next, we detail the blocks which compose our framework and that are illustrated in Fig. 5.1.

5.2.1 SCoRe1 Framework: Description of Blocks

Wireless Sensor Network (WSN): in this chapter we considered data collected from our own experimental network deployed on the ground floor of the Department of Information Engineering at the University of Padova and other four WSNs whose sensor reading have been published on-line. A brief technical overview of each of these five experimental network was presented in Section 4.3. The geometry of the considered deployment is not important, i.e., the nodes can be placed arbitrarily in a given area. Our framework, in fact, is flexible and does not depend on a specific topology; the only requirement is that the sensor nodes can be ordered, e.g., based on their IDs. Multi-hop paths are taken into account for transmission energy computation by assigning to each node a weight proportional to its distance from the DCP. Thus, the information coming to the DCP from a node which is three hops distant will "cost" three times as much as the information gathered from a node with one-hop distance.

Random Sampling (RS): the RS scheme is used to decide in a fully distributed way which sensors transmit their data to the DCP and which remain silent, at any given time v. This scheme has been chosen because it allows us to have a simple and general solution that can easily adapt to different signal characteristics and changes. In detail, at each time v each sensor node decides, with probability $p_{tx}^{(v)}$, whether to transmit its measurement to the DCP. This decision is made independently of the past and of the behavior of the other nodes, so there is no need for a large memory in each sensor, nor for further control packets within the network. The probability of transmission $p_{tx}^{(v)}$ can be fixed beforehand and kept constant, or can be varied as a function of the reconstruction error. The random behavior of the network is driven by this probability, so it is interesting to study the performance bounds of the random sampling scheme for a given $p_{tx}^{(v)} = p_{tx}$. In order to give an idea on how the value of the probability $p_{tx}^{(v)}$ translates into the behavior of the sensor network, we want to calculate the average time after which, given a fixed p_{tx}, all the sensors have transmitted their value at least once to the DCP, with probability higher than a given p. To this aim we define the random variable V_N, that gives the minimum number of time samples after

which all nodes have transmitted at least once, and we want to find the expected value of this variable and the minimum number of time samples after which with probability p all nodes have transmitted at least once. We define $a = 1-p_{tx}$ and $b = p_{tx}$ as the probability that a single node is silent and the probability that it transmits in a given instant v, respectively. Interpreting b as the probability of success in a Bernoulli process, the random variable V_N can be seen as the minimum number of time samples after which N independent Bernoulli processes have at least one success each. The probability of success for a single Bernoulli process at the v^{th} slot after $v-1$ failures is:

$$P[V_1 = v] = a^{v-1}b \quad , \tag{5.1}$$

that defines the probability mass function (pmf) of the variable V_1, the minimum number of time samples needed for the first success of a Bernoulli process. The probability of at least one success after v slots is:

$$P[V_1 \leq v] = \sum_{j=1}^{v} a^{j-1}b = 1 - a^v \quad , \tag{5.2}$$

and given that the N Bernoulli processes are independent we find the probability that all of them have had at least a success after v slots:

$$P[V_N \leq v] = (1 - a^v)^N \quad . \tag{5.3}$$

Inverting this formula, it is possible to calculate the minimum number of time samples after which with probability p all nodes have transmitted at least once, that is:

$$v = \left\lceil \frac{\log\left(1 - p^{1/N}\right)}{\log a} \right\rceil \ . \tag{5.4}$$

From the probability in (5.3), we can calculate also the pmf of V_N, that is:

$$p_{V_N}(v) = P[V_N \leq v] - P[V_N \leq v-1] = (1 - a^v)^N - (1 - a^{v-1})^N \quad , \tag{5.5}$$

and from this pmf we can define the expectation of V_N:

$$E[V_N] = \sum_{v=1}^{+\infty} v p_{V_N}(v) = \sum_{j=1}^{N} \binom{N}{j} (-1)^{j+1} \frac{1}{1 - a^j} \ . \tag{5.6}$$

Data Collection Point (DCP): the role of DCP is threefold:

1. it receives as input $\mathbf{y}^{(k)}$ and returns the reconstructed signal $\widehat{\mathbf{x}}^{(k)}$,

2. it adapts $p_{tx}^{(k)}$ and sends its new value to the sensor nodes; this is done to reduce the number of transmissions in the network while bounding the reconstruction error, and

3. it provides the recovery block with a training set $\widehat{\mathcal{T}}_{K_1}$ for $\mathbf{x}^{(k)}$; this training set is used to infer the structure of the signal, which is then exploited by the signal recovery algorithm. $\widehat{\mathcal{T}}_{K_1}$ is formed by the K_1 previously reconstructed signals $\widehat{\mathbf{x}}^{(j)}$ for $j < k$, so it can be written as $\widehat{\mathcal{T}}_{K_1} = \{\widehat{\mathbf{x}}^{(k-K_1)}, \ldots, \widehat{\mathbf{x}}^{(k-1)}\}$.

Recovery: the recovery method adopted in our framework is based on the joint use of CS and PCA. All the details about CS and PCA are presented in Section 2.1, while the mathematical framework that exploits both of them and that is integrated in SCoRe1 is presented in Chapter 4. In Section 5.3 we present a set of different recovery methods that may be adopted in our framework and we compare them in Section 5.5.

Controller: the controller block is responsible for the estimation of the quality of the signal reconstruction at the data collection point and for giving a feedback to the WSN based on this quality. In particular, the quality of reconstruction is quantified with the relative reconstruction error of the signal, and is estimated by the Error Estimation block described in the following.

Error Estimation: the reconstruction error that we want to estimate is defined as

$$\xi_R^{(k)} = \frac{\|\mathbf{x}^{(k)} - \widehat{\mathbf{x}}^{(k)}\|_2}{\|\mathbf{x}^{(k)}\|_2} \,, \tag{5.7}$$

where $\widehat{\mathbf{x}}^{(k)}$ is the signal reconstructed by the recovery block at time k and $\|\cdot\|_2$ is the 2-norm function for a vector. Note that at the DCP we do not have $\mathbf{x}^{(k)}$, but only $\mathbf{y}^{(j)} = \mathbf{\Phi}^{(j)}\mathbf{x}^{(j)}$ and $\widehat{\mathbf{x}}^{(j)}$, for $j \leq k$. Since the quantity $\xi_0^{(k)} = \|\mathbf{y}^{(k)} - \mathbf{\Phi}^{(k)}\widehat{\mathbf{x}}^{(k)}\|_2 / \|\mathbf{y}^{(k)}\|_2$ is always zero, due to the fact that the received samples are reconstructed perfectly, i.e., $\mathbf{\Phi}^{(k)}\widehat{\mathbf{x}}^{(k)} = \mathbf{\Phi}^{(k)}\mathbf{x}^{(k)}$, one might use some heuristics to calculate the error from the past samples. In this chapter we use the following formula:[2]

$$\xi^{(k)} = \left\| \begin{bmatrix} \mathbf{y}^{(k)} \\ \mathbf{y}^{(k-1)} \end{bmatrix} - \begin{bmatrix} \mathbf{\Phi}^{(k)}\widehat{\mathbf{x}}^{(k-1)} \\ \mathbf{\Phi}^{(k-1)}\widehat{\mathbf{x}}^{(k)} \end{bmatrix} \right\|_2 \cdot \left(\left\| \begin{bmatrix} \mathbf{y}^{(k)} \\ \mathbf{y}^{(k-1)} \end{bmatrix} \right\|_2 \right)^{-1} \,, \tag{5.8}$$

With this heuristic we compare the spatial samples collected at time k, i.e., $\mathbf{y}^{(k)}$, with the reconstructed values at time $k-1$, i.e., $\widehat{\mathbf{x}}^{(k-1)}$, sampled in the same points of the compressed

[2] We tried other heuristics and verified through extensive simulation that they perform similarly and worse than the one in (5.8). These are not listed here as they do not provide any additional insight.

values at time k, i.e., $\mathbf{\Phi}^{(k)}\widehat{\mathbf{x}}^{(k-1)}$. Then we compare the same signals inverting k and $k-1$. Note that $\xi^{(k)}$ does not only account for the reconstruction error but also for the signal variability. This introduces a further approximation to the error estimate, but on the other hand it allows the protocol to react faster if the signal changes abruptly and this is a desirable feature. In fact, if the signal significantly differs from time $k-1$ to time k, $\xi^{(k)}$ will be large and this will translate into a higher $p_{tx}^{(k+1)}$ (see below).

Feedback Control: this block calculates the new $p_{tx}^{(k+1)}$ for the next time $k+1$ and sends a broadcast message with this new value to the network nodes. The calculation of the new p_{tx} is made according to a technique similar to TCP's congestion window adaptation, where p_{tx} is exponentially increased in case the error is above a defined error threshold τ (to quickly bound the error) and is linearly decreased otherwise. In detail, we define the constants $C_1 \in [1, +\infty[$, $C_2 \in \{1, 2, \ldots, N\}$ and $p_{tx}^{\min}$, the minimum value allowed for the probability of transmission, and we calculate the new probability of transmission as:

$$p_{tx}^{(k+1)} = \begin{cases} \min\left\{p_{tx}^{(k)} C_1, 1\right\} & \text{if } \xi^{(k)} \geq \tau \\ \max\left\{p_{tx}^{(k)} - C_2/N, p_{tx}^{\min}\right\} & \text{if } \xi^{(k)} < \tau \ . \end{cases} \tag{5.9}$$

5.2.2 Role of the Controller

In order to illustrate the choices done for the design of SCoRe1 and in particular in order to highlight the advantaged of using the Controller block, here we consider two simple strategies for iteratively sensing and recovering a given signal that do not require the use of a Controller, and we compare them with SCoRe1. In detail, we want to show how the use of the Controller, with the Error Estimation and the Feedback Control blocks, can improve the performance of the data collection technique and automatically adapt the recovery to the specific signal analyzed. Moreover, we show the influence of the choice of an iterative or an approximate training set $\widehat{\mathcal{T}}_{K_1}$. The two simple data collection strategies analyzed are referred to as *2 Phases* and *Fixed* p_{tx}, respectively.

2 Phases: according to this data collection technique, the network monitors the entire signal $\mathbf{x}^{(k)}$ for a certain period of time (referred to as *training phase*) and sends it to the DCP, which is responsible for inferring the signal statistics during this period. For the subsequent period (*monitoring phase*), the DCP only requires a small fraction of the nodes to transmit, being able to accurately reconstruct the signal from its under-sampled version. Due to the fact that the monitored signal is non-stationary, its statistics may vary with time and should therefore be

periodically updated at the DCP. In detail, this protocol alternates the following two phases of fixed length in number of Data Collection Rounds (DCRs)[3]:

1. a *training phase* of K_1 DCRs, during which the DCP collects the readings from all N sensors and uses them to compute the statistics needed by the recovery algorithm. During this phase, the probability of transmission at each sensor is set to $p_{tx}^{(k)} = 1$, so the DCP collects a training set $\mathcal{T}_{K_1} = \{\mathbf{x}^{(k-K_1)}, \ldots, \mathbf{x}^{(k-1)}\}$ that will be used to infer the relevant statistics;

2. a subsequent *monitoring phase* of K_2 DCRs, with $K_2 \geq K_1$, during which (on average) only $L \leq N$ nodes transmit, according to the adopted random sampling scheme with $p_{tx}^{(k)} = L/N$. The signal of interest is thus reconstructed from this data set by the recovery algorithm, using the statistics computed in the training phase.

In other words, training and monitoring phases are interleaved as follows:

$$\ldots, \underbrace{\mathbf{y}^{(k)}, \ldots, \mathbf{y}^{(k+K_1-1)}}, \underbrace{\mathbf{y}^{(k+K_1)}, \ldots, \mathbf{y}^{(k+K_1+K_2-1)}}, \ldots$$

training phase	monitoring phase
$p_{\text{tx}} = 1$	$p_{\text{tx}} = L/N$
$\dim(\mathbf{y}^{(j)}) = N$	$E\left[\dim(\mathbf{y}^{(j)})\right] = L$

where $\dim(\mathbf{y}^{(j)})$ is the number of the components of $\mathbf{y}^{(j)}$ and $E[\dim(\mathbf{y}^{(j)})]$ is the expected value of $\dim(\mathbf{y}^{(j)})$.

Note that for the *2 Phases* technique the training set $\mathcal{T}_{K_1}$ does not contain approximations (reconstructions) of the past signals, but those actually collected during the training phase. The major drawback of this technique is that it is very sensitive to the choice of the parameters that govern the compression and the recovery phases. These parameters are: (1) the average number of sensors L that transmit during the monitoring phase, which determines $p_{tx}^{(k)} = L/N$. (2) The length of the training phase (K_1) and of the monitoring phase (K_2), that should be chosen according to the temporal correlation of the observed phenomenon. In this chapter we analyze the performance as a function of the value of $p_{tx}^{(k)}$, while the analysis of the influence of the parameters K_1 and K_2 can be found in Appendix A.2. All these parameters, $p_{tx}^{(k)}$, K_1 and K_2, must be chosen at the beginning of the transmission and they

[3] A Data Collection Round (DCR) is the time period between two consecutive readings of each sensor. In a DCR, we should reconstruct at the DCP the readings from all sensors.

can only be tuned manually. Hence, even if the initial choice is optimal, this technique is not able to adapt to sudden changes in the signal statistics. Moreover, the training phase accounts for the biggest part of the total cost in terms of number of transmissions, as shown by the study reported in Appendix A.2.

Fixed $\mathbf{p}_{tx}$: a solution to the latter problem is to eliminate the training phase, so the nodes at each time k transmit with a fixed probability $p_{tx}^{(k)} = p_{tx}$: this is the *Fixed* p_{tx} technique. Here, the training set needed by the recovery block to infer the statistics that allows the reconstruction of $\mathbf{x}^{(k)}$ from $\mathbf{y}^{(k)}$ is formed by the previously reconstructed signals $\widehat{\mathbf{x}}^{(j)}$ for $j < k$, so it can be written as $\widehat{\mathcal{T}}_{K_1} = \{\widehat{\mathbf{x}}^{(k-K_1)}, \ldots, \widehat{\mathbf{x}}^{(k-1)}\}$. The main drawbacks of this scheme is that without any control loop for p_{tx}, the energy consumption cannot be easily adapted to the observed signal and the reconstruction error can grow unbounded as will be shown shortly.

5.3 Data recovery from an incomplete measurement set

The recovery algorithm (see Fig. 5.1) is executed at the DCP and, at any time k, tries to recover the original signal $\mathbf{x}^{(k)} \in \mathbb{R}^N$ from its compressed version $\mathbf{y}^{(k)} \in \mathbb{R}^L$, with $L \leq N$. To this end, in the Section 4.1 we presented a mechanism that jointly exploits CS and PCA for signal interpolation in WSN and that from here on we will call **CS-PCA**. Obviously, many alternatives exist in the literature, each based on a particular signal model. Given a signal model, theoretical analysis can tell us which is the best, or the optimal, recovery mechanism to adopt. However, when we apply the chosen mechanism to real signals, we can obtain unsatisfactory results if the chosen model does not capture well the signal characteristics. In Chapter 4, we have analyzed the statistical distribution of the principal components for a set of real signals, computed exploiting K_1 past samples of the signals themselves. We have seen that the principal components are well modeled by a Laplacian distribution. Adopting this statistical model for the signal transformed by the PCA basis, the CS recovery algorithm is found to be optimal.

In what follows, we first review well-known state-of-the-art interpolation techniques that formally solve the following problem:

Problem 5.3.1 (Interpolation Problem)**.** *Estimate* $\widehat{\mathbf{x}}^{(k)}$ *(such that* $\|\widehat{\mathbf{x}}^{(k)} - \mathbf{x}^{(k)}\|_2 / \|\mathbf{x}^{(k)}\|_2 \simeq 0$*) knowing that* $\mathbf{y}^{(k)} = \mathbf{\Phi}^{(k)}\mathbf{x}^{(k)}$*, where* $\mathbf{y}^{(k)} \in \mathbb{R}^L$*,* $L \leq N$ *and* $\mathbf{\Phi}^{(k)}$ *is an* $[L \times N]$ *sampling matrix,*

i.e., all rows of $\mathbf{\Phi}^{(k)}$ *contain exactly one element equal to* 1 *and all columns of* $\mathbf{\Phi}^{(k)}$ *contain at most one element equal to* 1*, whilst all the remaining elements are zero.*

Each technique is based on a particular signal model, that we explicitly describe as well. At the end of this section, we detail how the presented recovery techniques can be implemented within the iterative monitoring framework of SCoRe1. In this way they can be compared against the proposed CS-PCA method. The performance results presented in Section 5.5, even if limited to the signals therein explained, will give insights on which of the analyzed techniques is more suitable to be used with real signals (and therefore, which is the signal model among those considered that best describes the reality).

Signal Models and Interpolation Techniques

The *a priori* knowledge that we can have about the signal of interest $\mathbf{x}^{(k)}$ helps us build a model for such signal. This knowledge can be deterministic, e.g., a description of the physical characteristics of the observed process, or probabilistic, e.g., the formulation of a probability distribution, called prior, to describe the possible realizations of $\mathbf{x}^{(k)}$. In both cases, the acquired knowledge on the signal to recover can be obtained by observing or processing a set of representative realizations of the signals of interest (i.e., the training set $\mathcal{T}_{K_1}^{(k)}$ or the approximate training set $\widehat{\mathcal{T}}_{K_1}^{(k)}$). In summary, to compute $\widehat{\mathbf{x}}^{(k)}$ from $\mathbf{y}^{(k)}$, we need a model of $\mathbf{x}^{(k)}$ that can be built according to two different approaches: a *deterministic approach* or a *probabilistic approach*, detailed in Sections 5.3.1 and 5.3.2, respectively. The Deterministic approach allows us to define two recovery methods, the Biharmonic Spline (Spline) and the Deterministic Ordinary Least Square (DOLS). The probabilistic approach instead is adopted in the Probabilistic Ordinary Least Square (POLS), that assumes a Gaussian distribution of the principal components, and in the CS-PCA method, that assumes a Laplacian distribution of the principal components. The implementation details of the four recovery methods are detailed in Section 5.3.3.

5.3.1 Recovery Methods based on Deterministic Signal Models

A possible way to think of $\mathbf{x}^{(k)} \in \mathbb{R}^N$ is as a signal whose elements depend on $d-$dimensional coordinates. To be more concrete, we can think of an environmental monitored signal collected from a WSN of N nodes that we order according to their IDs. Each node i, with $i = \{1, \ldots, N\}$, at time k senses a value which is represented by element $x_i^{(k)}$ of

vector $\mathbf{x}^{(k)}$. Since the considered node i is deployed in a specific location of the network, it is also linked to a set of geographical coordinates (e.g., latitude and longitude, which can be represented with a $d = 2$ dimensional coordinate vector $\mathbf{c}^{(i)}$). $x_i^{(k)}$ represents the reading of the $i-$th network node, which in turn is associated with a vector of d coordinates $\mathbf{c}^{(i)}$, and therefore we can express $x_i^{(k)}$ as a function of $\mathbf{c}^{(i)}$, i.e., $x_i^{(k)}(\mathbf{c}^{(i)})$. A straightforward way to model $\mathbf{x}^{(k)}$ is by defining a proper function of the $d-$dimensional coordinate $\mathbf{c}$, $\phi(\mathbf{c})$ (e.g., the Green function) that satisfies regularity conditions (e.g., smoothness) inferred by "typical" realizations of the signal of interest $\mathbf{x}^{(k)}$ [55]. Thus, we can write each element i of $\mathbf{x}^{(k)}$ as

$$x_i^{(k)}(\mathbf{c}^{(i)}) \simeq \sum_{j=1}^{L} \alpha_j \phi(\mathbf{c}^{(i)} - \mathbf{c}^{(j)}) \,, \tag{5.10}$$

where the function $\phi(\cdot)$ is used as a building block for $\mathbf{x}^{(k)}$ and α_j is the weight associated to $\phi(\cdot)$ centered in $\mathbf{c}^{(j)}$, with $j = 1, \ldots, L$, that is the $d-$dimensional coordinate corresponding to the physical placement of the node from which we received the $j-$th measurement.

The Biharmonic Spline Interpolation (**Spline**) [55] method solves Problem 5.3.1 exploiting the deterministic model in (5.10); the objective is to find a biharmonic function that passes through L data points stored in the $L-$dimensional vector $\mathbf{y}^{(k)}$. In this context, the elements of both $\mathbf{y}^{(k)}$ and $\mathbf{x}^{(k)}$ are seen as a function of d coordinates. Namely, to each element j of the $L-$dimensional vector $\mathbf{y}^{(k)}$ is associated a $d-$dimensional index $\mathbf{c}^{(j)} = [c_1^{(j)}, \ldots, c_d^{(j)}]^T$. Similarly, to each element i of the $N-$dimensional vector $\mathbf{x}^{(k)}$ is associated the $d-$dimensional index $\mathbf{c}^{(i)}$. In order to interpolate the L points in $\mathbf{y}^{(k)}$ we require to satisfy for each element of $\mathbf{x}^{(k)}$ the smoothness condition[4] $\nabla^4 \widehat{x}^{(k)}(\mathbf{c}) = \sum_{j=1}^{L} \alpha_j \delta(\mathbf{c} - \mathbf{c}^{(j)})$, given that, if $\mathbf{c} = \mathbf{c}^{(j)}$ then $\widehat{x}^{(k)}(\mathbf{c}^{(j)}) = y_j^{(k)}$, where $\mathbf{c}^{(j)}$ is the coordinate vector $\mathbf{c}^{(j)} = [c_1^{(j)}, \ldots, c_d^{(j)}]^T \in \mathbb{R}^d$ related to the reading $y_j^{(k)}$ (e.g., the geographical location of the reading $y_j^{(k)}$). The solution is proved to be:

$$\widehat{x}^{(k)}(\mathbf{c}) = \sum_{j=1}^{L} \alpha_j \phi_d(\mathbf{c} - \mathbf{c}^{(j)}) \,, \tag{5.11}$$

where $\phi_d(\cdot)$ is the Green function for the $d-$dimensional problem[5]. The constants $\alpha_1, \ldots, \alpha_L$ are found by solving the linear system $y_i^{(k)} = \sum_{l=1}^{L} \alpha_l \phi_d(\mathbf{c}^{(j)} - \mathbf{c}^{(l)})$, $\forall\ j \in \{1, \ldots, L\}$. To conclude, the solution $\widehat{\mathbf{x}}^{(k)} \in \mathbb{R}^N$ is the vector whose element i is equal to $\widehat{x}^{(k)}(\mathbf{c}^{(i)})$, namely,

[4]Here, ∇^4 is the biharmonic operator which allows to formalize regularity conditions on the fourth-order derivatives; $\delta(\cdot)$ is defined as $\delta(x) = 1$ if $x = 0$, $\delta(x) = 0$ otherwise.

[5]E.g., $\phi_1(\mathbf{c}) = |\mathbf{c}|^3$, $\phi_2(\mathbf{c}) = |\mathbf{c}|^2(\ln|\mathbf{c}| - 1)$ and $\phi_3(\mathbf{c}) = |\mathbf{c}|$.

the recovered value associated with the $d-$dimensional index $\mathbf{c}^{(i)}$.

An alternative way to determine a model for $\mathbf{x}^{(k)}$ allows us to abstract from the knowledge of where the signal sources are placed. Further, this second method is adaptable to the spatio-temporal correlation and structure of the signal. Observing that generally a physical phenomenon is correlated in time and that its spatial correlation can be considered as stationary over a given time period (e.g., from $k - K_1$ until k), a natural way to proceed is by assuming that $\mathbf{x}^{(k)}$ lies in the vector space spanned by the K_1 previous samples contained in the training set $\mathcal{T}_{K_1}^{(k)}$, or in the approximate training set $\widehat{\mathcal{T}}_{K_1}^{(k)}$, i.e., in span $\left\langle \mathcal{T}_{K_1}^{(k)} \right\rangle$, or in span $\left\langle \widehat{\mathcal{T}_{K_1}^{(k)}} \right\rangle$, respectively. According to the formalism introduced in Section 2.1.3, let us refer to the temporal mean and the covariance matrix of the elements in $\mathcal{T}_{K_1}^{(k)}$ as $\overline{\mathbf{x}}^{(k)}$ and $\widehat{\mathbf{\Sigma}}^{(k)}$, respectively. Let us consider also the ordered set $\mathcal{U}^{(k)} = \{\mathbf{u}_1^{(k)}, \ldots, \mathbf{u}_N^{(k)}\}$ of unitary eigenvectors of $\widehat{\mathbf{\Sigma}}^{(k)}$, placed according to the decreasing order of the corresponding eigenvalues. Let $\mathbf{U}_M^{(k)}$ be the $[N \times M]$ matrix whose columns are the first M elements of $\mathcal{U}^{(k)}$. To build a model of $\mathbf{x}^{(k)}$ based on the assumption that this one lies in span $\left\langle \mathcal{T}_{K_1}^{(k)} \right\rangle$, we can write:

$$\mathbf{x}^{(k)} \simeq \overline{\mathbf{x}}^{(k)} + \mathbf{V}^{(k)}\mathbf{s}^{(k)} = \overline{\mathbf{x}}^{(k)} + \mathbf{U}_M^{(k)}\mathbf{s}^{(k)} \,, \tag{5.12}$$

where, in general, $\mathbf{V}^{(k)}$ can be any $[N \times M]$ matrix of orthonormal columns (obtained at time k from the set $\{\mathbf{x}^{(k-K_1)} - \overline{\mathbf{x}}^{(k)}, \ldots, \mathbf{x}^{(k-1)} - \overline{\mathbf{x}}^{(k)}\}$, e.g., through the Gram-Schmidt process), with $M \leq N$; here we set $\mathbf{V}^{(k)} = \mathbf{U}_M^{(k)}$ because given $M \leq N$, the best way to represent with M components each element out of a set of $N-$dimensional elements is through PCA. In order to show that PCA is the solution to this representation problem, we should look at it from a geometric point of view. We can consider each sample $\mathbf{x}^{(k)}$, for all k, as a point in $\mathbb{R}^N$ and look for the M-dimensional plane (with $M \leq N$) which provides the best fit to all the elements in $\mathcal{T}_{K_1}^{(k)}$, and therefore for all the vectors that lie in span $\left\langle \mathcal{T}_{K_1}^{(k)} \right\rangle$, in terms of minimum Euclidean distance. The key point of PCA is the Ky Fan theorem [70], that is reported here to show an important aspect of the Deterministic approach followed in this Section.

Theorem 5.3.1 (Ky Fan Theorem)**.** *Let $\mathbf{\Sigma} \in \mathbb{R}^{N \times N}$ be a symmetric matrix, let $\lambda_1 \geq \cdots \geq \lambda_N$ be its eigenvalues and $\mathbf{u}_1, \ldots, \mathbf{u}_N$ the corresponding eigenvectors (which are assumed to be orthonormal, without loss of generality). Given M orthonormal vectors $\mathbf{b}_1, \ldots, \mathbf{b}_M$ in $\mathbb{R}^N$, with $M \leq N$, it*

holds that

$$\max_{\mathbf{b}_1,\ldots,\mathbf{b}_M} \sum_{j=1}^{M} \mathbf{b}_j^T \boldsymbol{\Sigma} \mathbf{b}_j = \sum_{j=1}^{M} \lambda_i \ , \tag{5.13}$$

and the maximum is attained for $\mathbf{b}_i = \mathbf{u}_i$, $\forall i$.

According to the Ky Fan Theorem, maximizing $\sum_{j=1}^{M} \mathbf{b}_j^T \widehat{\boldsymbol{\Sigma}}^{(k)} \mathbf{b}_j$ corresponds to finding the linear transformation $\mathcal{F}: \mathbb{R}^N \rightarrow \mathbb{R}^M$ that maximally preserves the information contained in the training set $\mathcal{T}_{K_1}^{(k)}$. In other words, this corresponds to maximizing the variance of the M-dimensional (linear) approximation of each element in span $\left\langle \mathcal{T}_{K_1}^{(k)} \right\rangle$ that, in turn, is strictly related to the information content of each signal in $\mathcal{T}_{K_1}^{(k)}$. Because of Theorem 5.3.1, the best M-dimensional approximation of any signal $\mathbf{x} \in$ span $\left\langle \mathcal{T}_{K_1}^{(k)} \right\rangle$ is given by [13]

$$\widehat{\mathbf{x}} = \overline{\mathbf{x}}^{(k)} + \mathbf{U}_M^{(k)} \left(\mathbf{U}_M^{(k)} \right)^T (\mathbf{x} - \overline{\mathbf{x}}^{(k)}) \ ,$$

where $\left(\mathbf{U}_M^{(k)} \right)^T (\mathbf{x} - \overline{\mathbf{x}}^{(k)})$ is the projection of $\mathbf{x} - \overline{\mathbf{x}}^{(k)}$ onto its best fitting M-dimensional plane. In summary, if the original point of interest $\mathbf{x}^{(k)} \in$ span $\left\langle \mathcal{T}_{K_1}^{(k)} \right\rangle$, we can transform it into a point $\mathbf{s}^{(k)} \in \mathbb{R}^M$ as follows:

$$\mathbf{s}^{(k)} \stackrel{def}{=} \left(\mathbf{U}_M^{(k)} \right)^T (\mathbf{x}^{(k)} - \overline{\mathbf{x}}^{(k)}) \ . \tag{5.14}$$

Multiplication of (5.14) by $\mathbf{U}_M^{(k)}$ and summation with the sample mean return the best approximation of the original vector, in accordance to (5.12).

To solve Problem 5.3.1 exploiting the model in Eq. (5.12) we can simply use the Ordinary Least Square (OLS) method [69], thus we refer to this recovery solution as Deterministic Ordinary Least Square (**DOLS**). From $\mathbf{y}^{(k)} = \boldsymbol{\Phi}^{(k)} \mathbf{x}^{(k)}$ and the assumption that Eq. (5.12) holds, we can write

$$\mathbf{y}^{(k)} = \boldsymbol{\Phi}^{(k)} (\overline{\mathbf{x}}^{(k)} + \mathbf{U}_M^{(k)} \mathbf{s}^{(k)}) \ . \tag{5.15}$$

The ordinary least square solution of Eq. (5.15) is given by

$$\widehat{\mathbf{s}}^{(k)} = (\boldsymbol{\Phi}^{(k)} \mathbf{U}_M^{(k)})^{\dagger} (\mathbf{y}^{(k)} - \boldsymbol{\Phi}^{(k)} \overline{\mathbf{x}}^{(k)}) \tag{5.16}$$

and it allows us to estimate the signal $\mathbf{x}^{(k)}$ as $\widehat{\mathbf{x}}^{(k)} = \overline{\mathbf{x}}^{(k)} + \mathbf{U}_M^{(k)} \widehat{\mathbf{s}}^{(k)}$. In the above expression the symbol † indicates the Moore-Penrose pseudo-inverse matrix.

Recalling that in Eq. (5.15) $\mathbf{y}^{(k)}$ is an $[L \times 1]$ vector whilst $\mathbf{s}^{(k)}$ is an $[M \times 1]$ with $M \leq L$, the system in Eq. (5.15) is in general overdetermined and may have no solutions (e.g., when all the L measurements are linearly independent). In this case (5.16) minimizes $\|(\mathbf{y}^{(k)} -$

$\mathbf{\Phi}^{(k)}\overline{\mathbf{x}}^{(k)}) - \mathbf{\Phi}^{(k)}\mathbf{U}_M^{(k)}\mathbf{s}^{(k)}\|_{\ell_2}$, obtaining $\widehat{\mathbf{s}}^{(k)}$ as the nearest (according to the Euclidean norm) possible vector to all the L collected measurements. If $L = M$, instead, the Moore-Penrose pseudo-inverse coincides with the inverse matrix and $\widehat{\mathbf{s}}^{(k)}$ is uniquely determined.

5.3.2 Recovery Methods based on Probabilistic Signal Models

This alternative approach allows us to introduce an uncertainty in the model of $\mathbf{x}^{(k)}$, in order to improve the effectiveness and robustness of the model when exploited for the recovery of the signal at the DCP. The introduction of the uncertainty in the model translates into a significant improvement in the recovery performance, as will be shown in Section 5.5.

Considering Eq. (5.12), this can be reformulated as:

$$\mathbf{x}^{(k)} \simeq \overline{\mathbf{x}}^{(k)} + \mathbf{V}^{(k)}\mathbf{s}^{(k)} = \overline{\mathbf{x}}^{(k)} + \mathbf{U}^{(k)}\mathbf{s}^{(k)} \,, \tag{5.17}$$

where $\mathbf{V}^{(k)}$ is now an $[N \times N]$ matrix of orthonormal columns set equal to the PCA matrix $\mathbf{U}^{(k)}$ following the same rationale as above. Here, the cardinality of the model's parameters is N (i.e., the dimension of vector $\mathbf{s}^{(k)}$), which is surely larger than or equal to the dimension of span $\left\langle \mathcal{T}_{K_1}^{(k)} \right\rangle$. The model in (5.17), therefore, allows us to account for the fact that $\mathbf{x}^{(k)}$ could not perfectly lie in span $\left\langle \mathcal{T}_{K_1}^{(k)} \right\rangle$. This kind of approach has been implicitly adopted also in Chapter 4 and, recalling Fig. 4.2, we can see that we need further assumptions on the system input $\mathbf{s}^{(k)}$ to fully characterize the model in Eq. (5.17), i.e., we have to assign a prior to $\mathbf{s}^{(k)}$. In practice, as it was shown in Chapter 4, $\mathbf{s}^{(k)}$ is a vector random process that we can assume to be, e.g., a Gaussian multivariate process[6] or a Laplacian vector process with i.i.d. components.

When we assign a Laplacian prior to $\mathbf{s}^{(k)}$, we can solve Problem 5.3.1 through our proposed recovery CS-PCA that corresponds to minimizing $\|\mathbf{s}^{(k)}\|_{\ell_1}$, given that $\mathbf{y}^{(k)} = \mathbf{\Phi}^{(k)}\mathbf{\Psi}\mathbf{s}^{(k)}$, as shown in Chapter 4.

Differently, when we assign a Gaussian prior to $\mathbf{s}^{(k)}$, we can solve Problem 5.3.1 again via the Ordinary Least Square Method (OLS); we refer to this recovery method as Probabilistic Ordinary Least Square Method (**POLS**). In this case, we just have to rewrite Eq. (5.16) as

$$\widehat{\mathbf{s}}^{(k)} = (\mathbf{\Phi}^{(k)}\mathbf{U}^{(k)})^{\dagger}(\mathbf{y}^{(k)} - \mathbf{\Phi}^{(k)}\overline{\mathbf{x}}^{(k)}) \,. \tag{5.18}$$

[6]This is the standard way of dealing with such problems, which appeals to the central limit theorem of probability theory [71].

In this equation, the dimension of $\mathbf{y}^{(k)}$ is less than the dimension of $\mathbf{s}^{(k)}$, i.e., $L < N$. Therefore, Eq. (5.18) is the solution of an ill-posed system, which theoretically allows an infinite number of solutions. Nevertheless, a multivariate Gaussian prior on $\mathbf{s}^{(k)}$ with zero mean and independent components[7], i.e., $p(\mathbf{s}^{(k)}) \sim \mathcal{N}(0, \mathbf{\Sigma_s})$ where $\mathbf{\Sigma_s}$ is a diagonal matrix, helps us to choose, among all the possible solutions, the one estimated as[8]

$$\begin{aligned}
\widehat{\mathbf{s}}^{(k)} &= \arg\max_{\mathbf{s}^{(k)}} p(\mathbf{s}^{(k)}|\mathbf{y}^{(k)}) = \arg\max_{\mathbf{s}^{(k)}} p(\mathbf{y}^{(k)}|\mathbf{s}^{(k)})p(\mathbf{s}^{(k)}) \\
&= \arg\max_{\mathbf{s}^{(k)}} \delta(\mathbf{y}^{(k)}, \mathbf{\Phi}^{(k)}\mathbf{U}^{(k)}\mathbf{s}^{(k)}) \frac{1}{(2\pi)^{\frac{L}{2}} \det(\mathbf{\Sigma_s})^{\frac{L}{2}}} \exp\left\{-\frac{\|\mathbf{\Sigma_s}\mathbf{s}^{(k)}\|_2^2}{2}\right\} \\
&= \arg\min_{\mathbf{s}^{(k)}} \|\mathbf{\Sigma_s}\mathbf{s}^{(k)}\|_2^2, \quad \text{given that } \mathbf{y}^{(k)} = \mathbf{\Phi}^{(k)}\mathbf{U}^{(k)}\mathbf{s}^{(k)}
\end{aligned} \tag{5.19}$$

that corresponds to the solution in Eq. (5.18), namely the minimum of $\|\mathbf{s}^{(k)}\|_{\ell_2}$ given that $\mathbf{y}^{(k)} = \mathbf{\Phi}^{(k)}\mathbf{U}^{(k)}\mathbf{s}^{(k)}$.

5.3.3 Implementation of Signal Recovery Methods

Each of the interpolation techniques explained above can be implemented at the DCP of our monitoring framework, specifically in the recovery block shown in Fig. 5.1. As previously remarked, at each time sample k, we can think of $\mathbf{x}^{(k)}$ as an $N-$dimensional signal whose elements depend on coordinates in d dimensions. If we measure $\mathbf{x}^{(k)}$ in L different coordinate points, collecting the measurement set $\{y_1^{(k)}, \ldots, y_L^{(k)}\}$, for the recovery stage we can proceed as follows, using the Deterministic approach:

1) Biharmonic Spline (**Spline**)

a) compute $\alpha_1, \ldots, \alpha_M$ solving $y_j^{(k)}(\mathbf{c}^{(j)}) = \sum_{l=1}^{M} \alpha_l \phi_d(\mathbf{c}^{(j)} - \mathbf{c}^{(l)}) \quad \forall\, j \in 1, \ldots, M$;

b) estimate $x_i^{(k)}$ as $\widehat{x}_i^{(k)}(\mathbf{c}^{(i)}) = \sum_{j=1}^{M} \alpha_j \phi_d(\mathbf{c}^{(i)} - \mathbf{c}^{(j)}) \quad \forall\, i \in 1, \ldots, N$.

Alternatively, if we assume to know the K_1 previous samples $\mathcal{T}_{K_1}^{(k)} = \{\mathbf{x}^{(k-K_1)}, \ldots, \mathbf{x}^{(k-1)}\}$ or the approximate training set $\widehat{\mathcal{T}}_{K_1}^{(k)} = \{\widehat{\mathbf{x}}^{(k-K_1)}, \ldots, \widehat{\mathbf{x}}^{(k-1)}\}$, with $K_1 \leq N$, we can abstract from the knowledge of the physical coordinates associated to $\mathbf{x}^{(k)}$. In this case we need

[7] Note that $\mathbf{s}^{(k)}$ can be assumed to have independent components if obtained through (5.14). If $\mathbf{s}^{(k)}$ is the vector of principal components of $\mathcal{T}_{K_1}^{(k)}$, these are known to be uncorrelated and therefore, under the assumption of gaussianity and zero mean, they are also independent.

[8] We recall here that, in the formulas (5.19), $\delta(\cdot)$ is a function defined as: $\delta(\mathbf{x}, \mathbf{y}) = 1$ if $\mathbf{x} = \mathbf{y}$, $\delta(\mathbf{x}, \mathbf{y}) = 0$ otherwise.

to compute the PCA matrix $\mathbf{U}^{(k)}$ from $\mathcal{T}_{K_1}^{(k)}$ (or $\widehat{\mathcal{T}}_{K_1}^{(k)}$). Then, knowing the matrix $\mathbf{U}^{(k)}$ and knowing also the sampled signal $\mathbf{y}^{(k)} = \mathbf{\Phi}^{(k)}\mathbf{x}^{(k)}$, we can use a Deterministic approach and set $M = K_1 - 1$. At each time k we can estimate $\mathbf{x}^{(k)}$ according to:

2) Deterministic Ordinary Least Square (**DOLS**)
a) estimate $\mathbf{s}^{(k)}$ as $\widehat{\mathbf{s}}^{(k)} = (\mathbf{\Phi}^{(k)}\mathbf{U}_{K_1-1}^{(k)})^{\dagger}(\mathbf{y}^{(k)} - \mathbf{\Phi}^{(k)}\overline{\mathbf{x}}^{(k)})$;
b) estimate $\mathbf{x}^{(k)}$ as $\widehat{\mathbf{x}}^{(k)} = \overline{\mathbf{x}}^{(k)} + \mathbf{U}_{K_1-1}^{(k)}\widehat{\mathbf{s}}^{(k)}$.

If we adopt a probabilistic approach, we can implement one of the two methods described, depending on the statistical distribution that we assume for the principal components of signal $\mathbf{x}^{(k)}$:

3) Probabilistic Ordinary Least Square (**POLS**)
a) estimate $\mathbf{s}^{(k)}$ as $\widehat{\mathbf{s}}^{(k)} = (\mathbf{\Phi}^{(k)}\mathbf{U}^{(k)})^{\dagger}(\mathbf{y}^{(k)} - \mathbf{\Phi}^{(k)}\overline{\mathbf{x}}^{(k)})$;
b) estimate $\mathbf{x}^{(k)}$ as $\widehat{\mathbf{x}}^{(k)} = \overline{\mathbf{x}}^{(k)} + \mathbf{U}^{(k)}\widehat{\mathbf{s}}^{(k)}$.

4) Joint CS and PCA (**CS-PCA**)
a) estimate $\mathbf{s}^{(k)}$ as $\widehat{\mathbf{s}}^{(k)} = \arg\min_{\mathbf{s}^{(k)}} \|\mathbf{s}^{(k)}\|_{\ell_1}$, given that $\mathbf{y}^{(k)} = \mathbf{\Phi}^{(k)}\mathbf{U}^{(k)}\mathbf{s}^{(k)}$;
b) estimate $\mathbf{x}^{(k)}$ as $\widehat{\mathbf{x}}^{(k)} = \overline{\mathbf{x}}^{(k)} + \mathbf{U}^{(k)}\widehat{\mathbf{s}}^{(k)}$.

The performance of the four different reconstruction techniques is compared in Section 5.5, in the case of a perfect knowledge of the training set $\mathcal{T}_{K_1}^{(k)}$, and in the case of an approximate training set $\widehat{\mathcal{T}}_{K_1}^{(k)}$.

5.4 WSN-Control Architecture

In this section, we briefly show how to integrate the SCoRe1 technique into an existing WSN architecture for monitoring environmental signals. Up to now, we have described the simple protocol executed at the sensor nodes, i.e., RS, and the recovery algorithms with the feedback control, executed at the Data Collection Point (DCP), that represents the sink to which all the nodes transmit their data and the application server connected to it, in which all the recovery calculations are made. In this section, we see that the DCP entity is physically divided into two separate components, i.e., the WSN gateway that collects the

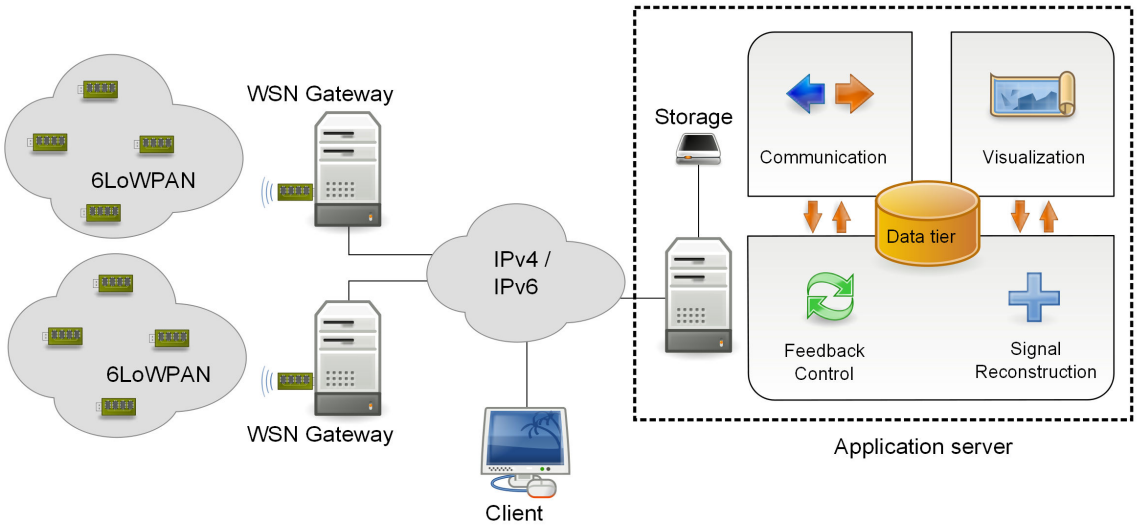


Figure 5.2. *WSN-Control architecture.*

data sent by the sensors and the application server. These two components are connected through the Internet. A diagram of the WSN-Control architecture is given in Fig. 5.2. The WSN (possibly composed of separate sensor islands) is accessed through a number of WSN gateways. Sensor nodes adopt a protocol stack based on 6LoWPAN and run a suitable routing protocol to send the gathered data to the gateways. For a more detailed description of the protocols running in the WSN the reader is referred to [72].

The core of the WSN-Control system is the Application Server (see Fig. 5.2). This server is a Web application composed of the following blocks: 1) Visualization, 2) Communication and 3) Signal Reconstruction and Feedback Control.

1) *Visualization:* this block creates a 3D representation of the gathered data, and is also responsible for the user interface and for the related Applet and Java Server Page (JSP) technology [73].

2) *Communication:* this block is responsible for the reception of data from the WSN and for the transmission of data gathering requests to the sensor nodes. In addition, along with these requests, it also broadcasts *feedback* messages that set the transmission behavior of all the sensor nodes for the next data collection round.

3) *Signal Reconstruction and Feedback Control:* in this block the learning mechanism, the reconstruction technique and the feedback control are implemented. At each data collection round, the entire WSN signal is reconstructed from the received measurements.

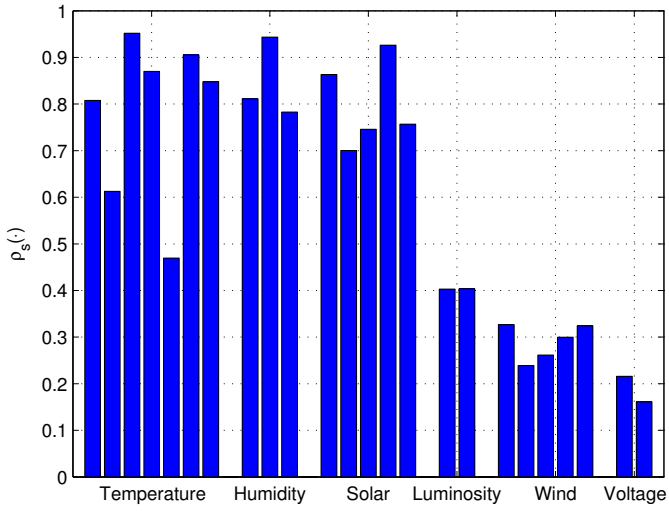


Figure 5.3. *Inter-node correlation for different signals gathered from the 5 different WSNs considered.*

Feedback messages are generated and sent to the sensor nodes based on the time sample according to the SCoRe1 technique so as to adapt the transmission behavior for the following data collection rounds. In particular, our aim is to minimize the number of nodes that send their measurements at each data collection round, while keeping the reconstruction error below a certain threshold.

The Web application is connected to a database that maintains the WSN measurements collected at previous data collection rounds. These are instrumental to the estimation of the signal statistics that, in turn, are used to obtain the incoherent transformation basis needed by CS, as detailed in Chapter 4.

5.5 Performance Analysis

In this section we analyze the performance of the proposed monitoring framework comparing the data collection techniques of Section 5.2 and analyzing the performance of our framework when used in conjunction with the signal recovery methods of Section 5.3. We analyze the statistics of all the signals gathered from the WSN deployments described in Section 4.3 and we choose a relevant subset of them for our performance analysis[9].

[9]The proposed framework is flexible and does not depend on a specific network topology. Its only requirement is that the sensor nodes must be ordered according to some criterion, e.g., using their IDs.

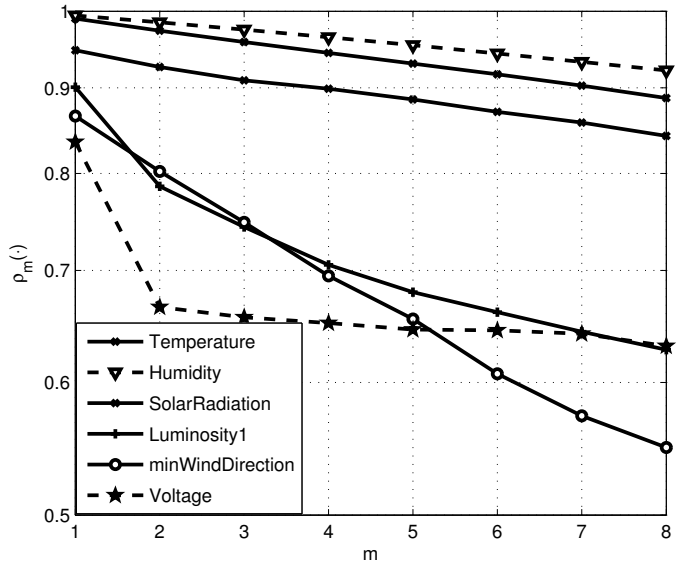


Figure 5.4. *Intra-node correlation for the signals chosen among all the signals considered in Fig. 5.3.*

Signals: We considered five different types of WSNs, each one of them sensing different types of signals for a total of 24 signals. For each signal $\mathbf{x}^{(k)} \in \mathbb{R}^N$, with $k = 1, \ldots, K$, we calculate the average inter-node correlation $\rho_s(\mathbf{x}^{(k)})$, defined as the average correlation between the one dimensional signals sensed by node i, $x_i(k)$, and the one sensed by node j, $x_j(k)$, for all the couples of nodes i, j:

$$\rho_s(\mathbf{x}^{(k)}) = \frac{1}{K} \sum_{k=1}^{K} \sum_{i=1}^{N} \sum_{j>i} \frac{\left(x_i^{(k)} - E[x_i]\right)\left(x_j^{(k)} - E[x_j]\right)}{((N^2 - N)/2)\sigma_{x_i}\sigma_{x_j}} \,. \tag{5.20}$$

$\rho_s(\mathbf{x}^{(k)})$ gives us a measure of the expected sparsity of the principal components $\mathbf{s}^{(k)} \in \mathbb{R}^N$. If we calculate the principal components of a signal with maximum inter-node correlation, i.e., $\rho_s(\mathbf{x}^{(k)}) = 1$, we will obtain a signal $\mathbf{s}^{(k)}$ with only the first component different from zero. On the contrary, if we calculate the principal components of a signal with minimum inter-node correlation $\rho_s = 0$, we will obtain a signal $\mathbf{s}^{(k)}$ with all components that are significant. In Fig. 5.3 we depict the inter-node correlation for all the signals considered and we divide them according to the signal type, i.e., Temperature, Humidity, Solar Radiation, indoor Luminosity, Wind and Voltage. We notice that the signals Temperature, Humidity and Solar Radiation have on average a high inter-node correlation ($\rho_s(\mathbf{x}^{(k)}) \simeq 0.7$), while indoor Luminosity, Wind and Voltage have a lower inter node correlation ($\rho_s(\mathbf{x}^{(k)}) \simeq 0.25$). To further analyze these signals, we consider the intra-node correlation $\rho_m(\mathbf{x}^{(k)})$, that is the

correlation of the one dimensional signal $x_i^{(k)}$ sensed by a single node with the same signal shifted by m time samples, i.e., $x_i^{(k+m)}$, averaged for all the N entries of $\mathbf{x}^{(k)} \in \mathbb{R}^N$. It is defined as:

$$\rho_m(\mathbf{x}^{(k)}) = \frac{1}{N}\sum_{i=1}^{N} \frac{\sum_{k=1}^{K}\left(x_i^{(k)} - E[x_i]\right)\left(x_i^{(k+m)} - E[x_i]\right)}{K\sigma_{x_i}^2} \,. \tag{5.21}$$

For representation purposes, we choose one signal for each type, within the 24 signals depicted in Fig. 5.3, and we represent for each chosen signal the temporal correlation $\rho_m(\mathbf{x}^{(k)})$, for $m = 1, \ldots, 8$ in Fig. 5.4. We notice that Temperature, Humidity and Solar Radiation signals keep a high intra-node correlation even for $m = 8$ ($\rho_8(\mathbf{x}^{(k)}) \geq 0.85$), while for Luminosity and Wind signals the temporal correlation quickly decreases ($\rho_8(\mathbf{x}^{(k)}) \leq 0.65$). The voltage signal has different characteristics, since even if its inter-node and intra-node correlations are similar to the ones of Luminosity and Wind, it is a nearly constant signal, so it should be treated accordingly.

Given this signal characterization, we choose a subset of the signals that is representative for the different statistical characteristics of each one of the signals analyzed. We use the signals gathered from the WSN testbed deployed on the ground floor of the Department of Information Engineering at the University of Padova using $N = 68$ TmoteSky wireless nodes equipped with IEEE 802.15.4 compliant radio transceivers. We have chosen these signals since they are representative of the whole signal set we considered, and since we can set for how long we should gather a given signal, in order to have meaningful performance of the compression/recovery scheme. In particular, we consider 5 signals divided accordingly to their statistical characteristics:

S1) two signals with high temporal and spatial correlation, i.e., the ambient temperature [°C] and the ambient humidity [%];

S2) two signals with lower correlation, i.e., the photo sensitivity [A/W] in the two ranges $320 - 730$ nm and $320 - 1100$ nm;

S3) the battery level [V] of the sensor nodes during the signal collection campaign.

The signals are gathered from the WSN testbed at the University of Padova. The results have been calculated through off-line simulation of the techniques, in particular they have been obtained from 100 independent simulation runs and by averaging the performance over all signals in each class.

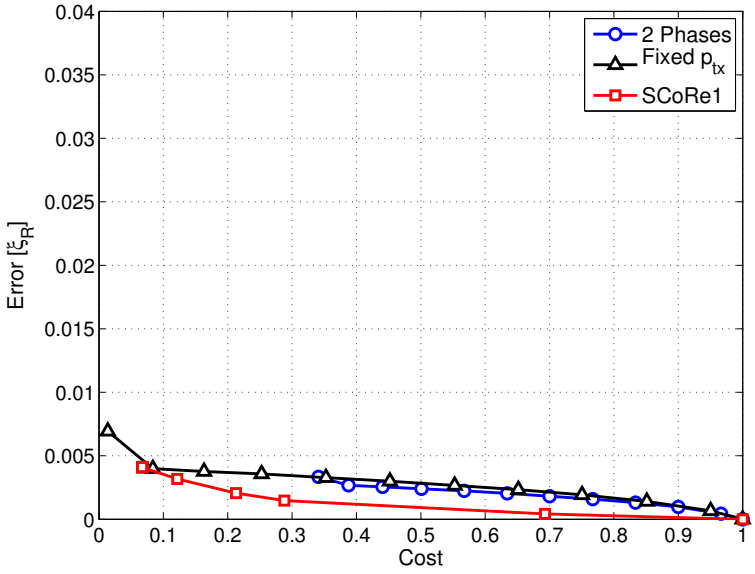


Figure 5.5. *Performance comparison of three iterative monitoring schemes when perfect knowledge of the past is exploited, for signals S1, temperature and humidity.*

The rest of the section is structured as follows: in Section 5.5.1 we analyze the performance of the data collection and control techniques, whereas in Section 5.5.2 we investigate the performance of the signal recovery methods.

5.5.1 Performance of Data Collection techniques

In the following we test the performance of the framework using the three DC techniques described in Section 5.2: *2 Phases*, *Fixed* p_{tx} and *SCoRe1*. These are used in all experiments in conjunction with the joint CS and PCA (CS-PCA) recovery method of Section 5.3, so as to check the impact on the performance of the chosen data collection scheme. Similar results can be obtained for any of the other recovery methods presented in this chapter.

The x-axis of Figs. 5.5–5.15 represents the normalized cost expressed as the average fraction of packet transmissions in the network per time sample, formally:

$$\text{Cost} = \frac{1}{K}\sum_{k=1}^{K}\frac{\sum_{n=1}^{N} D_n I_n(k)}{\sum_{n=1}^{N} D_n}\,, \tag{5.22}$$

where K is the number of considered time instants (i.e., the overall duration of the data collection), N is the total number of nodes in the WSN, D_n is the distance in terms of number of hops from node n to the DCP and $I_n(k)$ is an indicator function, with $I_n(k) = 1$ if node n

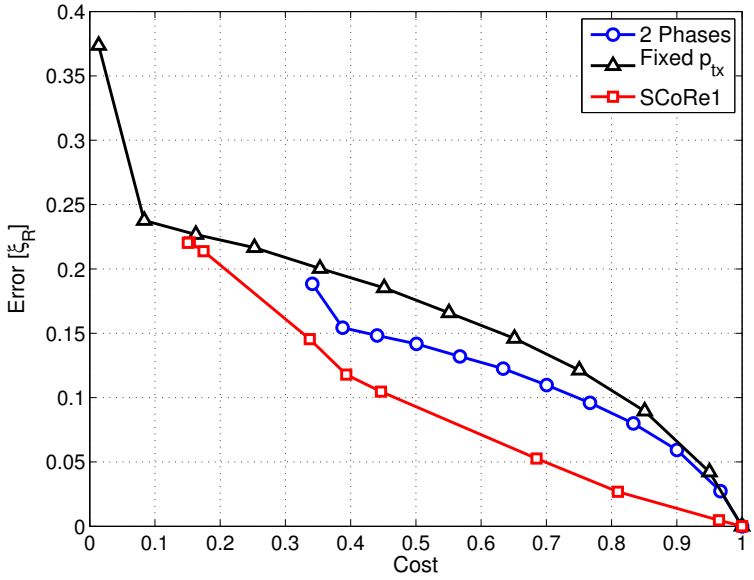


Figure 5.6. *Performance comparison of three iterative monitoring schemes when perfect knowledge of the past is exploited, for signals S2, light.*

transmits and $I_n(k) = 0$ if node n remains silent at time k. Note that a normalized cost equal to 1 corresponds to the case where all nodes transmit during all time instants $1, 2, \ldots, K$, which accounts for the maximum energy consumption for the network. Conversely, the normalized cost is zero when all nodes remain silent during all time instants. The y-axis shows the signal reconstruction error at the end of the recovery process, calculated accordingly to (5.7). In order to vary the cost (x-axis) for the three techniques we modify the following parameters:

1) for *2 Phases* and *Fixed* p_{tx} we vary the probability of transmission p_{tx} that is set at the beginning of the data gathering in the range $]0, 1[$;

2) for *SCoRe1*, we vary the error threshold τ used in (5.9) in the range $]0, 1[$ setting the feedback control parameters as $C_1 = 1.3$, $C_2 = 3$ and $p_{\min} = 0.05$.

In the figures, the training set length is $K_1 = 2$ and the the monitoring phase length is set to $K_2 = 4$, as suggested by the performance analysis in the Appendix A.2. Moreover, the three DC techniques are compared for the following cases:

A) perfect knowledge of the past, i.e., the training set is $\mathcal{T}_K = \{\mathbf{x}^{(k-K)}, \ldots, \mathbf{x}^{(k-1)}\}$;

B) adaptive knowledge of the past, i.e., the training set is $\widehat{\mathcal{T}}_K = \{\hat{\mathbf{x}}^{(k-K)}, \ldots, \hat{\mathbf{x}}^{(k-1)}\}$ for

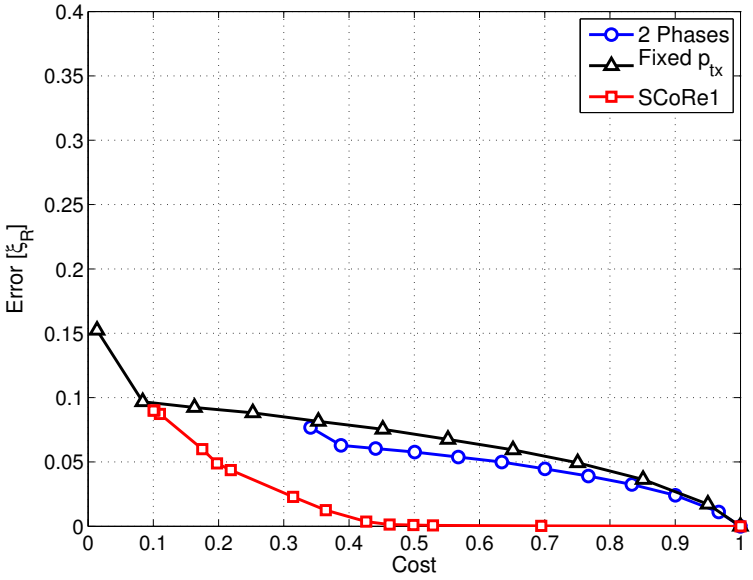


Figure 5.7. *Performance comparison of three iterative monitoring schemes when perfect knowledge of the past is exploited, for signals S1–S3.*

Fixed p_{tx} and *SCoRe1*, while for *2 Phases* the training set consists of the data collected during the last training phase.

Performance for case A: the performance for this case is shown in Figs. 5.5, 5.6 and 5.7. From Fig. 5.5 we note that the three techniques all perform very well in case of a slowly varying signal (S1). However, when the signal varies in an unpredictable way (S2), see Fig. 5.6, *SCoRe1* outperforms the other two techniques. We found that for signal S3 the error is close to zero for all techniques since in this case the signal is nearly constant; thus, the performance for this case is not shown.

In Fig. 5.7 we test the performance of the three data collection techniques in the case of a signal whose statistics change over time, e.g., from high correlation to low correlation due to some unpredictable events. In particular, we assume that this signal assume, over time, the characteristics of all the three signals (S1–S3) in series. In this case, *SCoRe1* significantly outperforms the other two techniques because it is able to iteratively adapt the transmission rate to the different statistical characteristics of the signals, e.g., slow varying or changing in an unpredictable way. This is a very important aspect of *SCoRe1* since, even if the signal characteristics remain unchanged during the monitoring, often we do not know these statistical characteristics of the signal before the compression/recovery mechanism starts and we should fix the parameters of the technique a priori. For this reason, it is very useful to

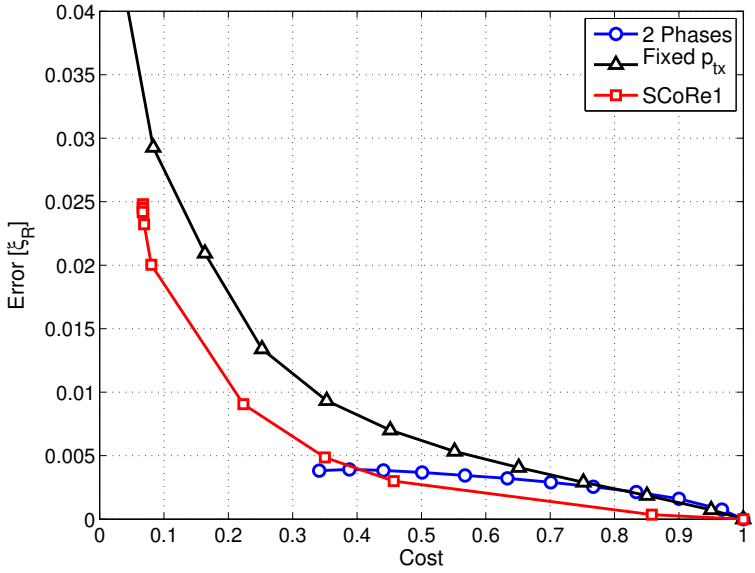


Figure 5.8. *Performance comparison of three iterative monitoring schemes, with online estimation of the past, for signals S1, temperature and humidity.*

have a technique able to perform well even if its parameters are fixed without knowing the exact statistical characteristics of the signal.

Performance for case B: the performance for case B is shown in Figs. 5.8, 5.9 and 5.10. All techniques perform slightly worse with respect to case A. In particular, *Fixed* p_{tx} does not adapt its transmission probability as a function of the reconstruction error that, in turn, gets very large for small values of p_{tx}. As in the previous case, if we look at the average performance over all signals, see Fig. 5.10, *SCoRe1* significantly outperforms the other two techniques.

Finally, comparing Fig. 5.7 with Fig. 5.10 we see that the reconstruction performance of *Fixed* p_{tx} is significantly impacted by inaccuracies in the estimation of the signal statistics, whereas in *2 Phases* and *SCoRe1* these inaccuracies are counter balanced by the training phase in *2 Phases* (with a significant increase in the cost) and by the proposed feedback control loop in *SCoRe1* (with a smaller increase in the cost).

5.5.2 Performance of the Recovery Techniques

In the following, we show performance curves for the different recovery techniques illustrated in Section 5.3 and used in conjunction with the DC technique *Fixed* p_{tx}, chosen because it highlights more accurately the performance differences between the recovery techniques

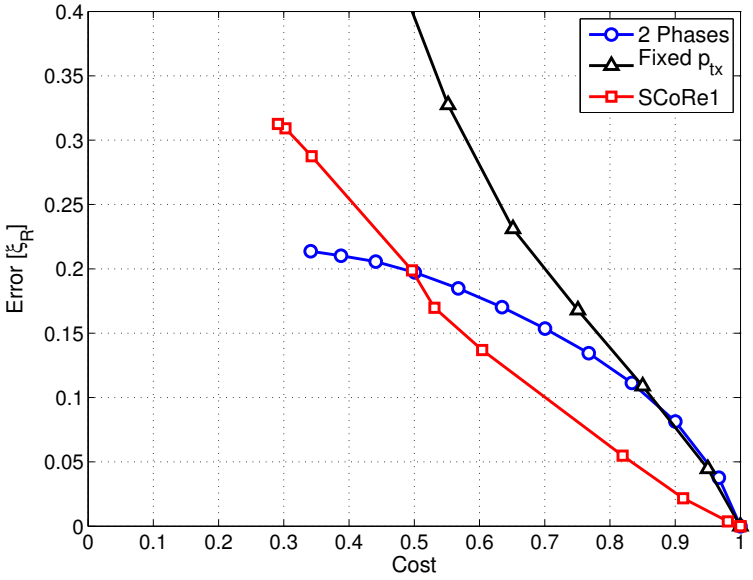


Figure 5.9. *Performance comparison of three iterative monitoring schemes, with online estimation of the past, for signals S2, light.*

analyzed. The considered recovery techniques are: Biharmonic Spline (Spline), Deterministic Ordinary Least Square (DOLS), Probabilistic Ordinary Least Square (POLS) and Joint CS and PCA (CS-PCA). Note that DOLS cannot be considered as an effective solution since it is affected by a numerical stability problem. Nevertheless, we considered it in view of its simplicity and low complexity. We will further come back to this issue in the following.

We compare the recovery techniques considering a fixed transmission probability in the range $[0.1, 1]$ with steps of 0.05 for the *Fixed* p_{tx} data collection scheme. Moreover, we consider the two cases A) and B) (with or without a perfect knowledge of the past) as in Section 5.5.1.

Performance for case A: Figs. 5.11 and 5.12 show the comparison of the recovery techniques performance for signals S1 and S2, respectively. From these figures, we see that CS-PCA and POLS are the best recovery techniques to use in conjunction with our monitoring framework. Both Gaussian and Laplacian seem to be a good choice as a prior for $\mathbf{s}$. Also DOLS, despite its simplicity and low complexity, works well under the assumption that the training set $\mathcal{T}_K$ is perfectly known. As previously mentioned, however, this approach is generally ill-conditioned and in some cases results in out-of-scale outcomes (disregarded in Figs. 5.11 and 5.12) and is also expected not to be robust against noise (i.e., in our case the error due to imperfect knowledge of the past when $\widehat{\mathcal{T}}_K$ is considered as the training set, see Figs. 5.13

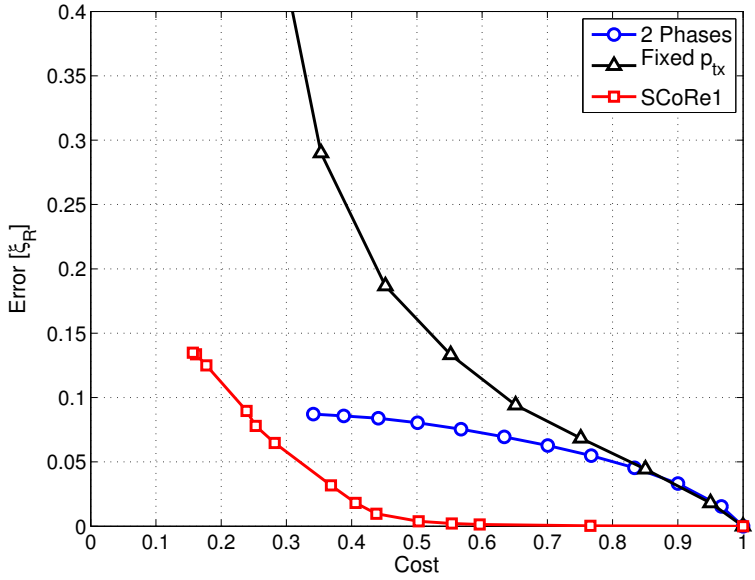


Figure 5.10. *Performance comparison of three iterative monitoring schemes, with online estimation of the past, for signals S1–S3.*

and 5.14). Note also that CS-PCA and POLS can be effective solutions in conjunction with our iterative monitoring framework in the presence of highly variable signals, see Fig. 5.12, signal S2.

Performance for case B: In Figs. 5.13 and 5.14, instead, we can see that an imperfect knowledge of the training set severely impacts the recovery performance of Spline and DOLS. This is however not as dramatic for CS-PCA and POLS. It is also interesting to note that in this second case (i.e., imperfect knowledge of $\mathcal{T}_K$, substituted by $\widehat{\mathcal{T}}_K$), POLS outperforms CS-PCA. In fact, the introduction of a further error in the model, i.e., an uncertainty on the training set, makes the Gaussian prior for $\mathbf{s}$ more effective than the Laplacian one, in accordance to the central limit theorem (e.g., see [71]). Nevertheless, both POLS and CS-PCA remain valid solutions for a monitoring application framework, since the performance loss from the ideal case A to the realistic one B is sufficiently small. Spline allows to reach good performance only above a transmission probability of 0.8. Furthermore, its use in conjunction with our iterative method leads to huge errors due to: (i) the tendency of our protocol to systematically avoid transmissions when possible; (ii) the approximation of the error estimate; (iii) the variability of the signal and (iv) the fact that Spline does not exploit any previous knowledge on the statistics of the signal to recover.

Finally, in Fig. 5.15 we report similar performance curves using the signals gathered from

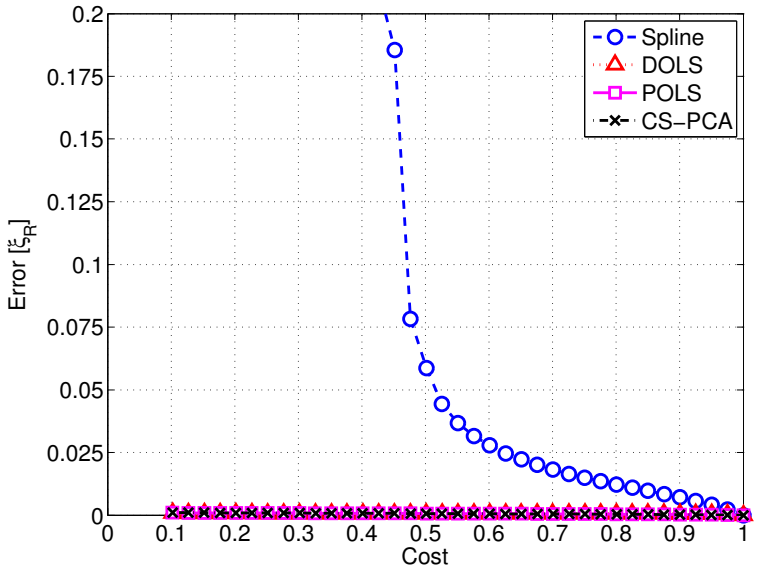


Figure 5.11. *Performance comparison of our iterative monitoring scheme used in conjunction with the recovery techniques, when a perfect knowledge of the past is exploited and for signals S1, temperature and humidity.*

the EPFL WSN deployment LUCE, see [50]. The results shown in these figures are obtained averaging the recovery performance achieved over 100 runs for signals S1, i.e., temperature and humidity, and assuming imperfect knowledge of the signal statistics. The WSN deployment LUCE consists of more than 80 nodes and the performance in Fig. 5.15 shows that all the above observations remain valid in this case as well. This provides evidence that our proposed monitoring framework, used with POLS and CS-PCA, is an effective solution for monitoring applications for WSNs in different scenarios.

5.6 Conclusion

In this chapter we studied joint sampling, recovery and protocol adaptation for distributed signals monitored by a WSN. We proposed a novel technique, called SCoRe1, to perform these tasks based on PCA to learn the data statistics, CS to recover the signal through convex optimization and a feedback controller that tries to bound the error. Using data measured in two different testbeds, we have shown that our technique achieves good performance in terms of reconstruction accuracy *vs* network cost (i.e., number of transmissions required). Thanks to our approach, we showed that CS recovery can be adopted

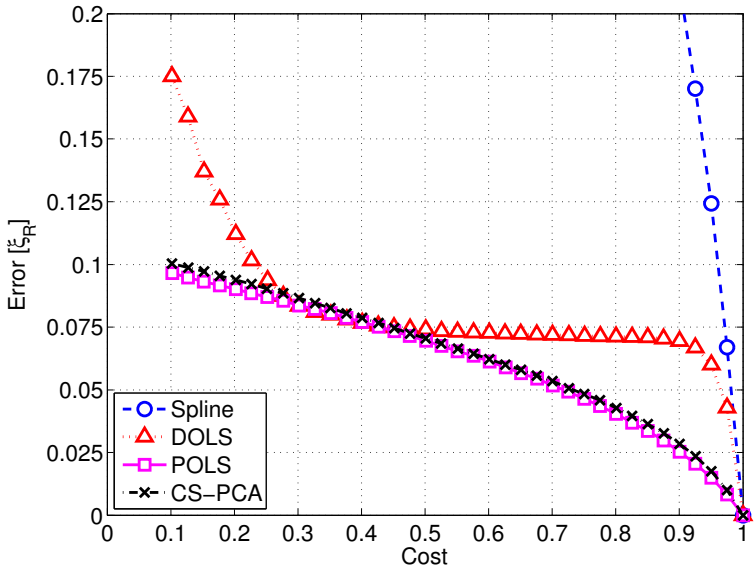


Figure 5.12. *Performance comparison of our iterative monitoring scheme used in conjunction with the recovery techniques, when a perfect knowledge of the past is exploited and for signals S2, light.*

for networking when exploited as an interpolation technique, differently from the literature where CS is mainly used as a method to jointly perform data acquisition and compression. Our approach is also robust to unpredictable changes in the signal statistics, and this makes it very appealing for a wide range of applications that require the approximation of a large and distributed dataset, with a certain spatial or temporal correlation.

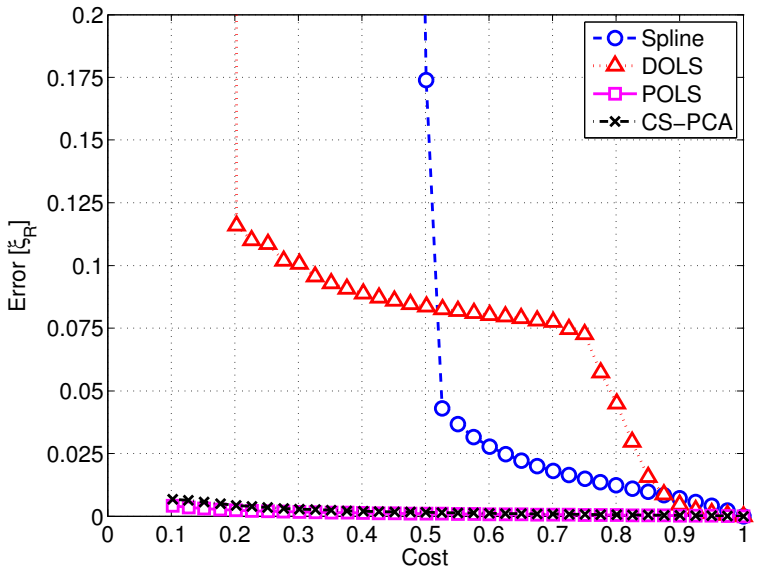


Figure 5.13. *Performance comparison of our iterative monitoring scheme used in conjunction with the recovery techniques, with online estimation of the past, for signals S1, temperature and humidity.*

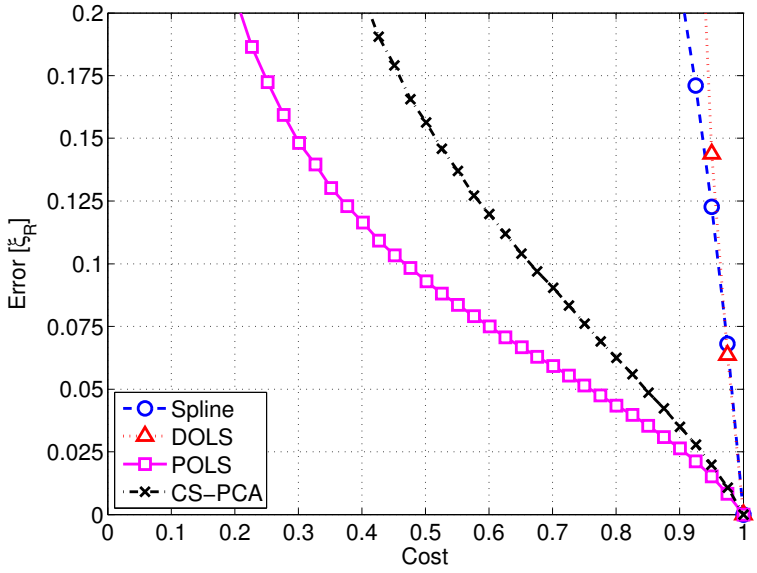


Figure 5.14. *Performance comparison of our iterative monitoring scheme used in conjunction with the recovery techniques, with online estimation of the past, for signals S2, light.*

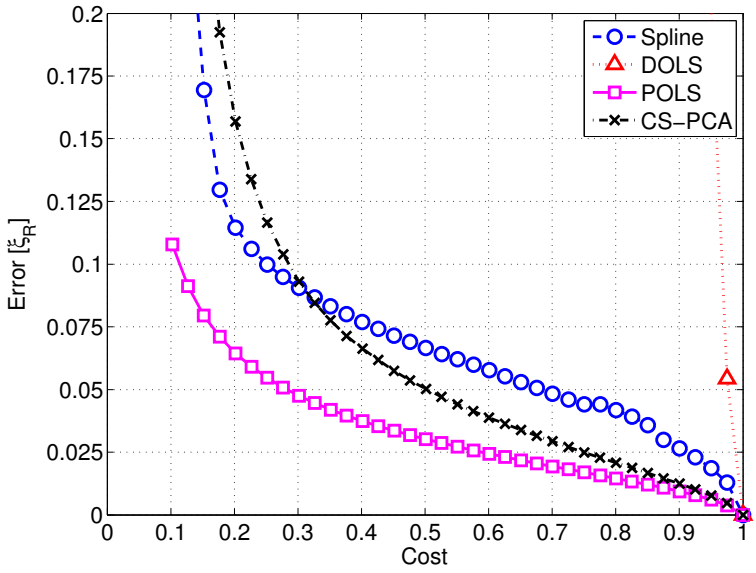


Figure 5.15. *Performance comparison of our iterative monitoring scheme used in conjunction with the recovery techniques,with online estimation of the past, for signals S1, temperature and humidity. These performance curves are obtained with signals gathered from the EPFL WSN deployment LUCE, see [50].*

Chapter 6

Cognitive Network Control using Bayesian Networks

6.1 Introduction

Cognitive networking [74, 75] is an emerging paradigm that deals with how wireless systems learn relationships among network parameters, network events, and observed network performance, plan and make decisions in order to achieve local, end-to-end, and network-wide performance as well as resource management goals. In cognitive networks, all nodes track the spatial, temporal, and spectral dynamics of their own behavior, as well as of the environment. The information gathered is used to learn, plan and act in a way that meets network or application Quality of Service (QoS) requirements.

One of the key requirements of a cognitive network is to learn the relationships among network protocol parameters spanning the entire stack in relation with the operating network environment. In this chapter [1] we use a probabilistic graphical modeling approach, Bayesian Networks (BNs), in order to create a representation of the dependence relationships between significant network parameters in multi-hop wireless network environments.

[1] The material presented in this chapter has been published in:

[C1] **G. Quer**, H. Meenakshisundaram, B.R. Tamma, B.S. Manoj, R. Rao and M. Zorzi, "Cognitive Network Inference through Bayesian Network Analysis", *IEEE Globecom 2010*, Miami, FL, Dec. 2010.

[C3] **G. Quer**, H. Meenakshisundaram, B.R. Tamma, B.S. Manoj, R. Rao and M. Zorzi, "Using Bayesian Networks for Cognitive Control of Multi-hop Wireless Networks", *MILCOM 2010*, San Jose, CA, Nov. 2010.

As an example, tactical communication environments are dynamic and very challenging, and there is no predefined protocol configuration that optimally operates in these conditions. In these scenarios, learning the optimal parameter set for the network protocol stack, by using the BN structure constructed from historical network behavior, can help efficiently meet QoS requirements.

Cognitive networking is different from cognitive radios or cognitive radio networking in that the latter two typically apply cognition only at the PHY layer to dynamically detect and use spectrum holes, and focus strictly on dynamic spectrum access. We notice that there are still some open problems to solve before cognition can be applied to the entire protocol stack. First, the probabilistic relationships among the various parameters that span across the entire protocol stack are not clearly understood. Second, the tools that can be used to determine such complex relationships are not well known. The traditional layered protocol stack has helped establish an order and structure in the role of various protocols and hence has greatly contributed to the faster progress of networking systems. The popular alternative to layered protocol stack operation, cross layer networking, has been cautioned by recent results [76,77] which show that cross-layer research can be less useful if not designed and implemented within the larger scope of abstraction of network systems in a way similar to the time proven relation between architecture and protocols. Unfortunately, there exist no comprehensive approaches, as observed in [77], that can be used to study the crucial cross-layer behavior across all the layers. Therefore, our cognitive networking architecture does not adopt a fully cross-layer approach, but keeps the layered structure of the network intact. At the same time, however, it uses some cross-layer principles, as the core of the cognitive architecture is the BN that represents the relationships among network parameters belonging to different layers. Moreover, the approach is designed to optimize specific performance at the transport layer using the information coming from other layers, in a cross-layer fashion.

In this chapter, we propose a cognitive network node architecture that can be integrated with the existing layered protocol stack. Our work partially addresses the requirement of modeling the layered protocol stack using new and hitherto unused tools from artificial intelligence. Specifically, we consider the use of BNs, a graphical representation of statistical relationships among random variables, widely used in machine learning [1]. An introductory description about BNs and the details about the learning techniques used to infer the BN from the data observation are presented in Section 2.2.

The use of BN for modeling the protocol stack provides us with a unique tool, not only to

learn the influence of certain parameters on others, but also to apply the inferred knowledge and achieve a certain desired level of performance at the higher layers. For example, our modeling enables a node to determine which combinations of lower layer parameters are useful for achieving a certain higher layer throughput performance. The main contributions of this work are:

1. the integration of BN into our Cognitive Network framework;
2. the application of BN to study network parameters in realistic Wireless LAN (WLAN) scenarios (single-hop and multi-hop);
3. a performance analysis of the BN inference engine's accuracy in the single-hop scenario to infer the value of TCP throughput;
4. a performance analysis in a realistic multi-hop wireless network scenario of the BN inference engine predicting the TCP's congestion status;

The rest of this chapter is organized as follows: Section 6.2 discusses our architecture for cognitive networking in detail, Section 6.3 presents the application of BN learning to a WLAN single-hop scenario and shows the performance of the inference engine to predict TCP throughput. Section 6.4 presents the application of BN learning to multi-hop networking scenarios and shows the performance of the inference engine to predict TCP congestion status. We conclude the chapter in Section 6.5.

6.2 Cognitive Network Architecture

Here we present the network architecture in which we want to integrate the BN tools. The network parameters within a stack or within a particular protocol can be classified into two basic categories: observable parameters and controllable parameters. The observable parameters provide important information about the characteristics or behavior of the protocol and the status of the network system. For example, in the MAC layer, the average packet retransmission count provides information about the packet losses and retransmissions; however, it cannot be directly controlled, although we can set the maximum retransmission limit. Examples of controllable parameters include the TCP Congestion Window (T.CW), with its minimum and maximum levels allowed. Note that the controllable parameters in a given layer may affect the observable parameters in some other layer.

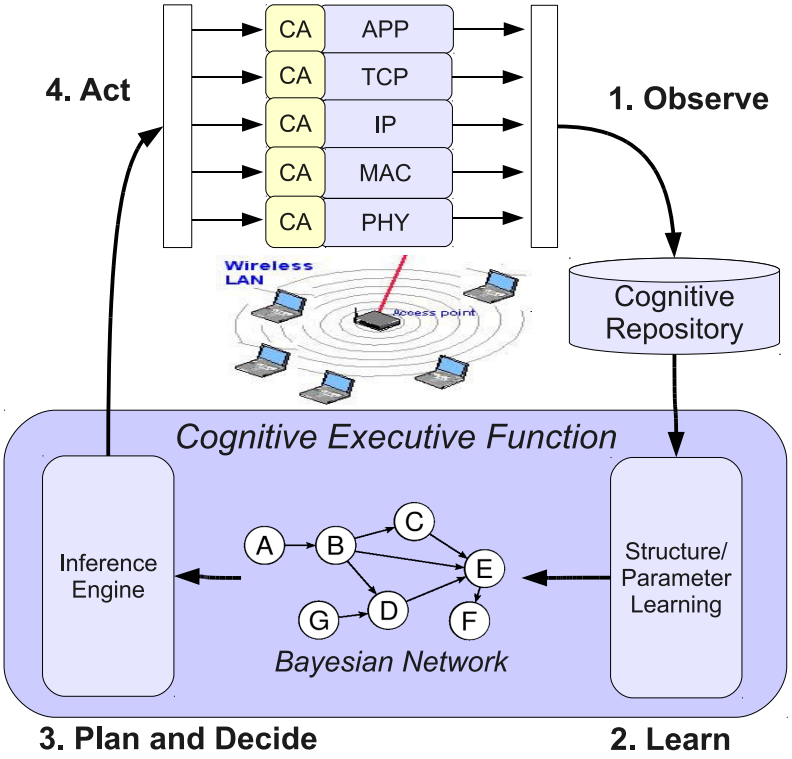


Figure 6.1. *The Cognitive Network Architecture.*

Our approach uses a fully distributed solution where software modules called Cognitive Agents (CA) are plugged into each layer as well as protocols of importance, so that we have access to the protocol parameters in each layer. Communication among the CAs is coordinated through a backplane called CogPlane [75]. Each CA is responsible for periodically sampling a set of protocol parameters and repositorizing the sampled data to a *Cognitive Repository*. The CogPlane contains a Cognitive Execution Function (CEF), that serves as the brain of the cognitive network protocol stack and executes the optimization of protocols within the stack. We decompose the cognitive action into the four phases of the "Cognition Cycle" [4]: 1) Observe, 2) Learn, 3) Plan and Decide, and 4) Act.

Fig. 6.1 shows the architecture of the cognitive network protocol stack that (in the Observe phase) collects parametric information from each layer and (in the Learn phase) learns the effect of controllable parameters on observable parameters and (in the Plan and Decide phase) determines the values to be assigned to various controllable parameters in order to meet the performance requirements imposed by the application layer, and finally (in the Act phase) reconfigures the network elements.

The Observe phase includes in-stack and out-of-stack parameter observation. The observe action, within the protocol stack, is done by periodically sampling the observable and controllable protocol parameters across all layers. For in-stack parameter observation, the CAs will be designed to periodically sample the state of the network parameters. The main out-of-stack parameter is the wireless traffic information which is collected by the CA in the physical layer.

The Cognitive Execution Function (CEF) is the brain of the cognitive network node where the optimization decisions for protocol within the stack are made. CEF realizes the Learn and the Plan and Decide phases of the cognition cycle. The Learn phase consists of the process of building the BN structural relationship between the observable and controllable network parameters within the network protocols, network layers, and environment parameters such as spectrum, location, and time. Our approach is to first identify a set of important parameters from each layer of the protocol stack or from representative protocols in each layer. Once the temporal behavior data of such parameters is collected in a given spatial domain, the challenge is to derive critical causality relationships between parameters. We use the above discussed BN model for deriving the critical causality relationship between parameters. The BN learning model has been presented in Section 2.2, and in this chapter it is applied to learn the relationships among some in-stack network parameters for single-hop networks in Section 6.3 and for multi-hop networks in Section 6.4.

The Plan and Decide phase of the system focuses on decomposing the network or user performance objectives into real actionable information within the controllable parameter set of the entire protocol stack. The spatio-temporal-spectral characteristics derived from historical information, in the form of the probabilistic structure in the graphical models, will provide significant knowledge for this phase. The Plan and Decide phase would translate abstract objectives, e.g., minimum throughput required or maximum end-to-end delay that can be tolerated by the application layer protocol, into the real controllable network parameters, e.g., combinations of multi-layer protocol parameter values. A key role in this phase is the prediction of the values of the network parameters of interest given the observations, in order to choose the appropriate actions to perform in the network.

Finally, in the Act phase the decision is effected. Compared to the other phases, this phase is far less complex. For example, in this phase, the CEF sends instructions to be executed at all or selected layers or protocols in the form of identified controllable parameters and their suggested values. As an example, the initial congestion window and slow

start threshold of TCP can be instructed to a new node based on the network environment's spatio-temporal characteristics [75]. The process of Act may be carried out at different scales on different devices. For example, a client node in a Wireless LAN uses this framework to observe local node parameters and to optimize certain flows of that node, whereas a wireless AP can use LAN wide parameters to help optimize the network for certain high priority client nodes.

In conclusion, a cognitive network's protocol stack has a Cognitive Agent (CA) at each layer. During the Observation phase, these agents collect traces of a variety of network parameters. This information is then propagated to the Cognitive Repository. The Learning phase begins once sufficient data has been collected in the repository. In this phase, the learning module of the Cognitive Controller uses the repository data to learn both the structure and parameters of a BN representing the relationships between the various network parameters. The learned BN and its parameters are passed to the Inference Module in the Decision & Inference phase. This module infers the values of certain controllable parameters required to achieve target values of some other parameters. The Decision Module communicates the inferred values of these controllable parameters to the CA of the appropriate network layer.

6.3 Single-Hop networks: Learning a Bayesian model for Cognitive Networks

In this section, we look at how the BN Structure Learning (SL) and Parameter Learning (PL) algorithms, described in Section 2.2.2 and in Section 2.2.3, respectively, are used in the learning phase to infer the probabilistic relations between the MAC and TCP parameters of the single-hop network considered. While we focus on the relationship between TCP and MAC as an example in this section, our strategy can be generalized for the entire protocol stack.

6.3.1 Network Scenario

We analyze a scenario with up to 40 active users that produce 20 independent TCP flows within the WLAN and communicate through one Cognitive Access Point (CogAP). The CogAP is responsible for learning a suitable Bayesian Network (BN) based on the data col-

lected at each active node and for taking consequent decisions to optimize the traffic in the network. The 20 TCP flows are simulated using the ns-3 discrete event network simulator [78], augmented with hooks that collect the values of selected TCP and MAC parameters at regular sampling intervals for each flow. In particular, the TCP parameters collected are the following: x_1 is the congestion window value (T.CW), i.e., the total amount of data [bytes] that can remain unacknowledged at any time; x_2 is the congestion window status (T.CWS), which can take one of three values based on the TCP algorithm: linear increasing, exponentially increasing or exponentially decreasing; x_3 is the Round Trip Time (T.RTT), i.e., the last value registered in the sampling interval of the time between when a packet is sent and when the corresponding acknowledgement is received; x_4 is the TCP throughput (T.THP), i.e., the total amount of unique data [bytes] acknowledged in a sampling interval. The selected MAC parameters are the following: x_5 is the Contention Window (M.CTW), i.e., the maximum number of slots the node will wait before transmitting a packet at the MAC level, according to a random back off interval between zero and CW; x_6 is the number of MAC transmissions (M.TX), i.e., the total number of original MAC packets transmitted in the sampling interval; x_7 is the number of MAC retransmissions (M.RTX), i.e., the total number of MAC retransmissions in the sampling interval. Among the parameters described above, T.CWS, T.THP, M.CTW, M.TX, and M.RTX have a finite number of outcomes, less than or equal to $n_q = 30$. Also the remaining two parameters, T.CW and T.RTT, have been quantized to n_q levels, in order to apply the structure and parameter learning algorithms for multinomial variables, since they are computationally more efficient and require less training data. These parameters have been quantized in n_q levels chosen according to their estimated n_q-quantiles[2]. We perform the learning methods in two datasets, namely $\mathcal{D}^1_{N,M}$ and $\mathcal{D}^2_{N,M}$, that differ by the sampling intervals in which we register the network parameters values, $\Delta T_1 = 100\ ms$ and $\Delta T_2 = 1000\ ms$, respectively. The number of samples is of the order of $N \simeq 5 \times 10^4$ samples and the number of parameters is $M = 7$ in both datasets.

6.3.2 Designing the Bayesian Network from the data

The SL algorithm, described in Section 2.2.2, is performed by a score based method implemented in Matlab and available in [80]. This phase is the most computationally demand-

[2]Since it is unrealistic to introduce a complex non-uniform quantizer in each node, we have chosen this quantizer that performs better than the uniform one in the case of non uniform realistic variables, as suggested by intuition looking at the cdf of the variables, and as formally explained in [79].

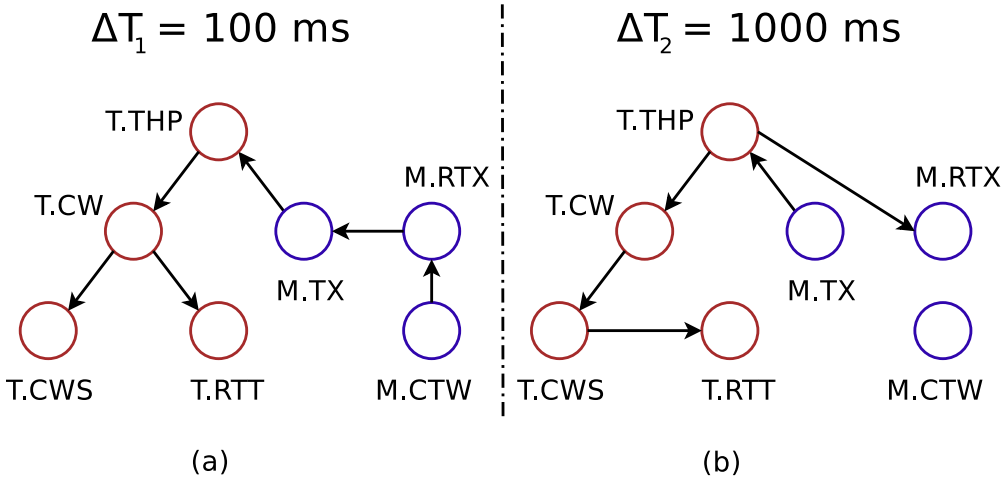


Figure 6.2. *BNs learned (a) from the dataset $\mathcal{D}^1_{N,M}$ with sampling time $\Delta T_1 = 100\ ms$ and (b) from the dataset $\mathcal{D}^2_{N,M}$ with $\Delta T_2 = 1000\ ms$.*

ing, as for $M = 7$ parameters it requires to discriminate from a set of about 1.1×10^9 DAGs. We have used three different search procedures, namely 1) Hill Climbing (HC), 2) a Markov Chain Monte Carlo method (MCMC) and 3) a simple heuristic. The first two are classical methods described in [81] and implemented in [80]. The simple heuristic we propose in this section consists in dividing the net into two subnets, one for the MAC parameters ($M_{MAC} = 3$) and one for the TCP parameters ($M_{TCP} = 4$), and finding the best structure for each one of them, separately scoring all the possible DAGs, that for each subnet are less than 10^3. Then these two separate subnets form the initial structure given as input to the HC algorithm. Each search procedure gives a DAG as a result and we choose among these three DAGs the one with the highest BIC score in Eq. (2.27). From dataset $\mathcal{D}^1_{N,M}$ we obtained the DAG depicted in Fig. 6.2-(a) that represents the static connection between the parameters sampled at intervals of $\Delta T_1 = 100\ ms$. From dataset $\mathcal{D}^2_{N,M}$ instead we obtain the DAG depicted in Fig. 6.2-(b), that is slightly different from the previous one. The reasons for this difference are that when a longer sampling period is used, the data shows that there is a direct probabilistic relation between the TCP throughput (T.THP) and the number of MAC retransmissions (M.RTX), while the MAC contention window value (M.CTW) becomes independent of the other parameters. In the graph the former translates into the appearance of a direct edge between M.RTX and T.THP and the latter translates into the disappearance of the edge connecting M.CTW with M.RTX. All the other differences in the graph can be explained similarly.

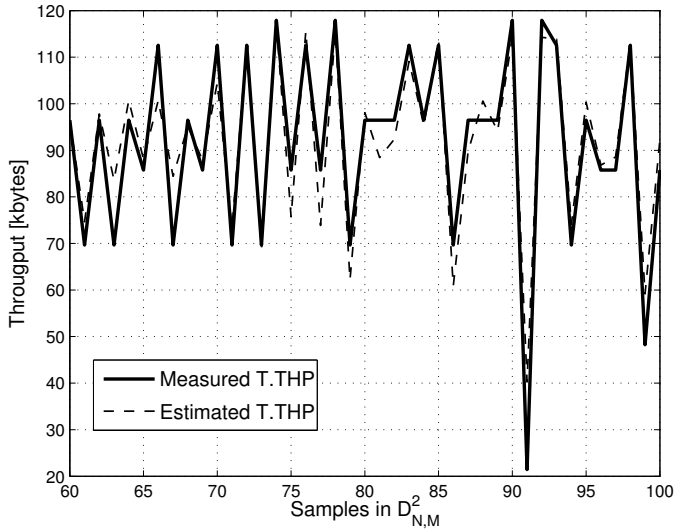


Figure 6.3. *Measured value and estimated value of T.THP (with evidence $X_e = \{all\}$) for 40 consecutive samples in $\mathcal{D}^2_{N,M}$.*

6.3.3 Learning the Inference Engine

The Plan and Decide phase of our cognitive network is implemented using a Bayesian inference engine, that is able to predict the value of certain network parameters based on the observation of some other parameters. The inference engine is designed using the PL algorithm described in Section 2.2.3, that defines quantitatively the probability relations among the parameters based on the given training set. The estimate of an unobserved parameter x_i is the expectation of such parameter based on the probability mass function (pmf) learned using Eq. (2.28), where the probability is conditioned on the set of observed parameters, i.e., the evidence X_e. The estimated value for x_i at time k becomes:

$$\hat{x}_i^{(k)} = E\left[x_i^{(k)} | X_e\right] \ . \tag{6.1}$$

6.3.4 Performance Evaluation of the Inference Engine

The Bayesian inference engine exploits the BN structure to infer the expected values of certain important parameters based on the evidence. We apply the inference engine to the two different DAGs that connect the network parameters in the cases of sampling interval $\Delta T_1 = 100\, ms$ and $\Delta T_2 = 1000\, ms$, depicted in Fig. 6.2. Now we use such structures to infer the value of the TCP throughput (T.THP) based on some evidence, i.e., the measured values

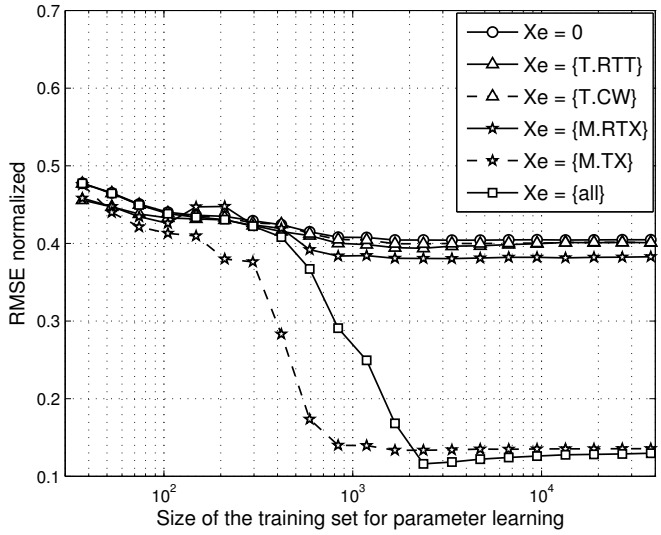


Figure 6.4. *Performance of the T.THP inference engine for* $\Delta T_1 = 100\ ms$.

of a subset, possibly empty, of the network parameters, X_e. In Fig. 6.3 we show the measured and the estimated T.THP for 40 consecutive samples, with sampling time ΔT_2, in the case where we have as evidence all the other network parameters. In order to measure the accuracy of the T.THP estimate, we define the normalized Root Mean Square Error (RMSE), i.e.:

$$\xi = \frac{\sqrt{\sum_{k=1}^{N}\left(\hat{x}_4^{(k)} - x_4^{(k)}\right)^2}}{\sqrt{\sum_{k=1}^{N}\left(x_4^{(k)}\right)^2}}\ , \tag{6.2}$$

where $x_4^{(k)}$ is the actual value of T.THP at time k and $\hat{x}_4^{(k)}$ is the estimated value based on the evidence in the set X_e. In the simulations we used the data collected from the ns-3 simulator mentioned in Section 6.3.1, while the inference engine is written in Matlab. In Figs. 6.4 and 6.5 we represent the performance of the inference engine for the TCP throughput estimate for $\Delta T_1 = 100\ ms$ and $\Delta T_2 = 1000\ ms$, respectively. In the x-axis we vary the length of the training set for parameter learning, in logarithmic scale, and in the y-axis we represent the normalized RMSE calculated as in Eq. (6.2). The total number of sample intervals used in this simulation is $N = 4 \cdot 10^4$. The curves in the graphs correspond to different evidence subsets for the estimate. In particular, we analyze the performance of the inference engine in these cases: 1) $X_e = \emptyset$, there is no evidence and the T.THP is estimated based on its marginal distribution; 2) $X_e = \{\text{T.RTT}\}$, the evidence is the TCP RTT; 3) $X_e = \{\text{T.CW}\}$,

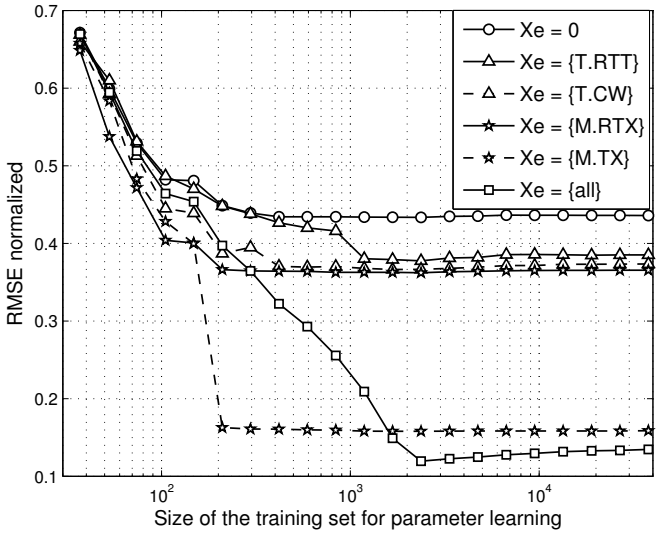


Figure 6.5. *Performance of the T.THP inference engine for* $\Delta T_2 = 1000\ ms$.

the TCP Congestion Window; 4) $X_e = \{\text{M.RTX}\}$, the number of MAC retransmissions; 5) $X_e = \{\text{M.TX}\}$, the number of MAC transmissions; 6) $X_e = \{$T.RTT, T.CW, T.CWS, M.TX, M.RTX, M.CTW$\}$, where all the parameters but the TPC throughput are observed.

In particular in Fig. 6.4, with $\Delta T_1 = 100\ ms$, we have a small improvement compared to the case of no evidence when we measure the number of MAC retransmissions, while we have a bigger improvement when measuring the number of MAC transmissions, which is the parameter most related to TCP throughput, as expected in our scenario. It is interesting to notice that for a training set smaller than $2 \cdot 10^3$ samples the inference engine performs better with the knowledge of only M.TX than with the knowledge of all the parameters. In order to explain this behavior, we should notice from Fig. 6.2-(a) that M.TX and T.CW separate T.THP from the rest of the network, according to the rules of *d-separation*. Moreover, when the training set is too short the estimates of the values of the variables are not precise, so that including additional variables (T.CW in this case) adds potentially inaccurate information which negatively affects the overall throughput estimate performance. Instead, with $\Delta T_2 = 1000\ ms$ (see Fig. 6.5), the (now more accurate) knowledge of T.CW and T.RTT does provide some advantages for throughput inference, even though M.TX still has a dominant role in the considered single-hop scenario.

Observing the performance results we conclude that our inference engine helps the Plan and Decide phase not only in predicting the behavior of certain network parameters but

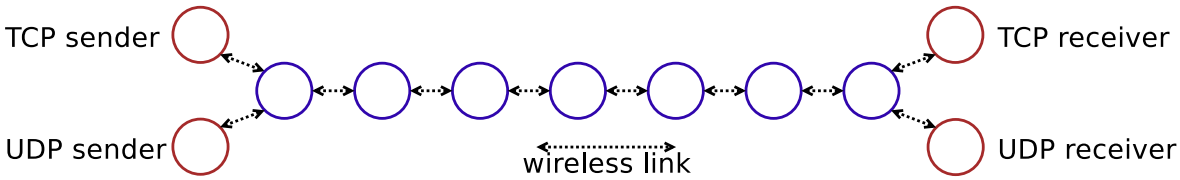


Figure 6.6. *Network scenario: dumbbell topology.*

also in guaranteeing quality of T.THP estimate with proper choice of the evidence. A key application to exploit the T.THP inference engine is to adapt the controllable parameters such as T.CW, T.CWS and M.CTW to achieve the desired T.THP.

6.4 Multi-Hop networks: Inferring Network Congestion through a BN Model

In this section, we derive Bayesian Network (BN) structures that probabilistically connect some of the significant protocol stack parameters in different multi-hop wireless network scenarios. Then with the BN structure we design an engine that can be implemented in each cognitive network node to predict in advance the congestion status of the network. Knowing when congestion will arise is very useful for the efficiency of transport layer protocols; however, TCP and its popular variants do not have any mechanism to predict congestion in advance. Such a reactive nature of TCP leads to packet losses and wastage of precious network resources like bandwidth and energy, which are essential for efficient communication in mobile network environments such as tactical networks. In this section, with the help of the BN structure derived observing the network environment and the current network state, we infer the congestion status of the network that will help TCP to proactively make decisions on how to adapt the value of the congestion window.

6.4.1 Network scenarios

We considered two multi-hop network topologies, a dumbbell topology depicted in Fig. 6.6 and a random topology, and we conducted experiments using the ns-3 discrete event network simulator [78]. The protocol stack in the simulator is augmented with hooks that collect the values of selected TCP and MAC parameters at regular sampling intervals for each flow, as in the case of single-hop scenario in Section 6.3. These hooks simulate the

behavior of the Cognitive Agents (CAs), that are responsible for reading and collecting the parameters' values. In the dumbbell topology, we have two nodes on either side connected by a string of 7 intermediate nodes. Adjacent nodes are separated by 250 m, have a transmission range of 300 m, and have fixed PHY data rate of 2 Mbps. The Optimized Link State Routing (OLSR) is employed as the routing protocol. We have one TCP flow (*ftp* file transfer session from TCP sender to TCP receiver in Fig. 6.6) and one Constant Bit Rate (CBR) flow (UDP sender to UDP receiver in Fig. 6.6), hence each flow has eight hops. The CBR sending rate is equal to either 20 Kbps or 40 Kbps, to provide two levels of congestion to the TCP flow, Low Congestion (LC) and High Congestion (HC), respectively. The cognitive TCP source node samples the parameters of interest at 100 ms and 200 ms interval, respectively, in two sets of experiments. This scenario is very simple but sufficient to understand the basic relations between parameters.

In the second scenario, we have a mesh network with 40 nodes, initially arranged in an 8×5 pattern with 250 m between horizontally and vertically adjacent nodes and moving randomly at a speed of 1 m/s within a square area of 2500×2500 m^2. Each node has a transmission range of 300 m and is set up to be either the sender or the receiver in an *ftp* session over TCP. We have 20 such TCP flows that go on throughout the duration of the experiment. We have 20 of the nodes that were also involved as either transmitter or receiver of CBR traffic with rate 20 Kbps. The routing protocol is again OLSR and all the nodes were set to use the Minstrel physical layer rate control algorithm [82]. One of the *ftp* sender nodes was cognitive and capable of observing its network stack parameters at 100 ms intervals.

6.4.2 Network parameters

In this section, we deal with MAC and TCP parameters, similarly to what has been done in Section 6.3, and briefly reported here. Specifically, we take three parameters from the MAC layer and four parameters from the transport layer. IEEE 802.11 and TCP are chosen as MAC and transport protocols, respectively. The MAC parameters are the number of packets transmitted at the MAC layer (M.TX), the value of the 802.11 contention window (M.CTW), and the number of retransmissions at MAC layer (M.RTX). M.CTW is a controllable parameter and the other two are observable parameters. The TCP parameters are the congestion window value (T.CW), the Round Trip Time (T.RTT), the instantaneous TCP throughput

(T.THP) and the network congestion status (T.CS), that is a binary parameter, with outcome 1 when a congestion is detected and 0 otherwise.

In each sampling interval k, the network parameters are obtained as follows. T.RTT(k) is the last value registered in the sampling interval k of the time between when a packet is sent and when the corresponding acknowledgement is received; T.THP(k) is the total amount of unique data [bytes] acknowledged in a sampling interval k, divided by the length of the sampling interval [s]; T.CW(k) is the value of the Congestion Windows at time k [bytes]; M.CTW(k) is the maximum number of slots the node will wait before transmitting a packet at the MAC level, according to a random back off interval between zero and M.CTW(k); M.TX(k) is the total number of original MAC packets transmitted in the sampling interval; M.RTX(k) is the total number of MAC retransmissions in the sampling interval. T.CS(k) at each time sample k is defined as:

$$\text{T.CS}(k) = \begin{cases} 1\ , & \text{if T.CW}(k)/\text{T.CW}(k-1) \leq 0.6\ , \\ 0\ , & \text{if T.CW}(k)/\text{T.CW}(k-1) > 0.6\ . \end{cases} \tag{6.3}$$

The threshold is set to 0.6 because we are mainly interested in detecting significant drops of the congestion window. We aim to infer at time k the value of T.CS($k+n$), with $n = 1, \ldots, 5$, using current values of all the observable parameters, so we want to predict the occurrence of congestion to be able to act before it strongly affects the network.

All the parameters collected, except T.RTT, are multinomial, with a finite but possibly large number of outcomes. In order to make the calculation simpler and more efficient we quantize all the parameters to a maximum of $n_q = 30$ levels, so it is possible to apply the SL and PL algorithms for multinomial variables explained in Sections 2.2.2 and 2.2.2, respectively, without the need for a very long training set to learn the probabilistic structure. Indeed, a finer quantization would lead to a more accurate estimation, but at the price of requiring a longer training set for proper learning of the probabilistic structure. Moreover, it was not realistic to introduce a complex non-uniform quantizer, given the limited computation capacity of the CA, so we chose to quantize the parameters' values according to their estimated n_q-quantiles, that translates in our case into better estimation performance than uniform quantization.

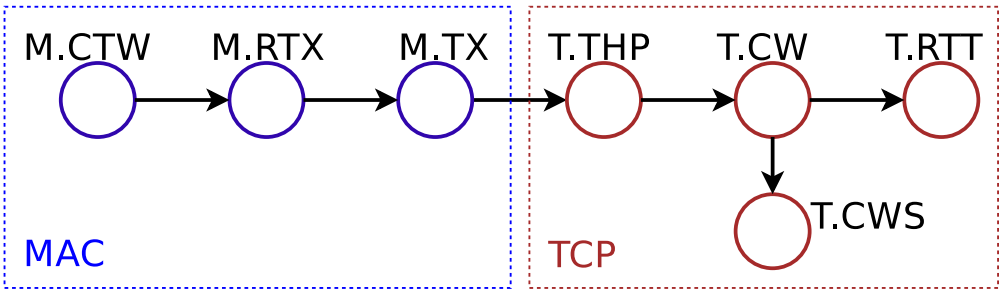


Figure 6.7. *BN structure learned from the historical network dataset and chosen as representative for all the multi-hop scenarios considered.*

6.4.3 BN Structure Learning

As a first step for the Learn phase, we infer the structure of the BN using a score based method available in the MATLAB BNT toolbox [80], with a similar procedure to the one described in Section 6.3, exploiting the three search heuristics described in Section 6.3.2. For each one of the network scenarios described in Section 6.4.1 we obtain a slightly different DAG, and we choose the DAG in Fig. 6.7 as a structure for the inference in the rest of this section, since it has a good BIC score (see Eq. (2.27)) in all the scenarios considered, even if not optimal.

6.4.4 Congestion Inference

The prediction of T.CS, the congestion status, presents some fundamental issues that are addressed in this section. First of all, the BN gives us the qualitative (the DAG) and quantitative (Eq. (2.28)) probabilistic relations between the network parameters at a given time instant. In order to predict at time k the value of T.CS$(k+n)$, with $n \geq 1$, we need to introduce the time dimension in our model. In order to do so, we put the variable T.CS$(k+n)$ in our BN structure in place of T.CS(k) and we use the ML estimation in Eq. (2.28) to calculate the quantitative relations among the parameters of the new BN structure.

A second issue that arises is due to the fact that the prior for the variable T.CS$(k+n)$ is not uniform, but instead $P[\text{T.CW}(k+n)=0] \gg P[\text{T.CW}(k+n)=1]$, since congestion rarely occurs. If we simply aim at minimizing the misclassification rate, the predictor would always infer a value of $\widehat{\text{T.CS}}(k+n)=0$, because this is by far the most likely outcome. This predictor brings no information, so it is not useful. In order to solve this problem we introduce a loss function [1], i.e., we penalize with different weights the misclassification

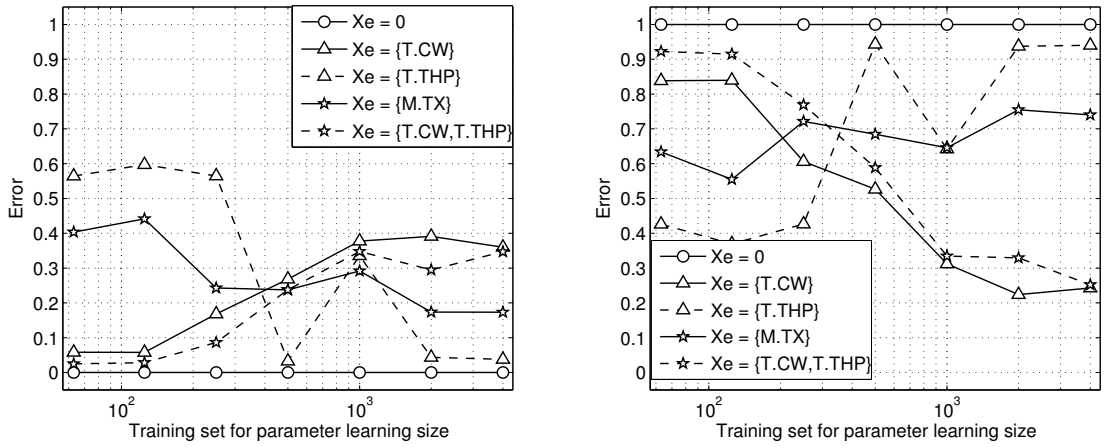


Figure 6.8. *Average prediction error for T.CS$(k+2)$ as a function of the training set length for different evidence sets, in case (a) the value we want to infer is T.CS$(k+2)=0$ and (b) T.CS$(k+2)=1$.*

when congestion occurs, $L_{1,0}$, and when congestion does not occur, $L_{0,1}$. In other words, with the introduction of the loss function we aim at maximizing the probability of a certain occurrence multiplied by the corresponding loss function weight, i.e.:

$$P\left[\text{T.CS}(k+n)=b|\text{evidence}\right]\cdot L_{b,1-b}\ , \tag{6.4}$$

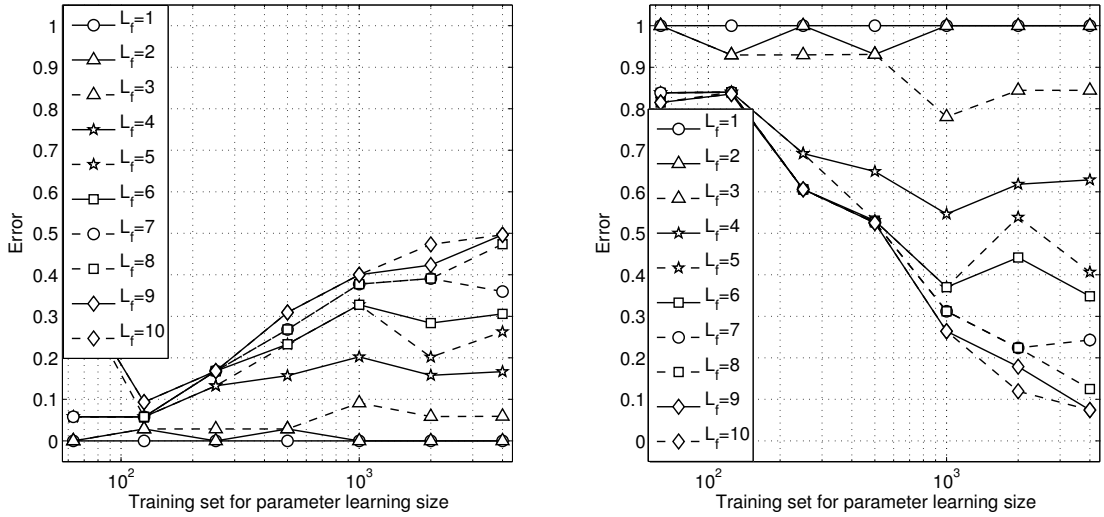
for $b\in\{0,1\}$. As we are interested in the relative values of these weights, we fix $L_{0,1}=1$ and we vary $L_{1,0}$. $L_{1,0}=1$ corresponds to the case in which we just aim at minimizing the overall misclassification rate.

The inference engine we propose uses the probabilities in Eq. (2.28) to maximize Eq. (6.4) and as a result the inferred parameter given the evidence X_e is:

$$\widehat{\text{T.CS}}(k+n)=\max_{b\in\{0,1\}} P\left[\text{T.CS}(k+n)=b|X_e\right]\cdot L_{b,1-b}\ . \tag{6.5}$$

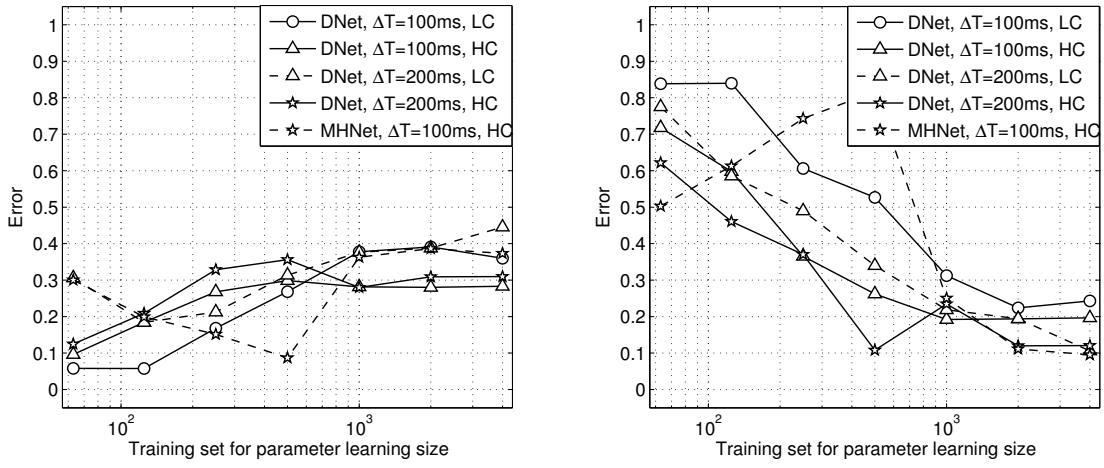
6.4.5 Performance analysis

In this section, we analyze the accuracy of the inference engine in predicting at time k the value of T.CS$(k+n)$, i.e., the presence or absence of congestion at time $k+n$, with $n\geq 1$. The performance of the engine is analyzed as a function of the length (in number of samples) of the training set used to learn the relations between the parameters and to define the inference engine with Eqs. (2.28) and (6.5). During the training set the parameters are



(a) (b)

Figure 6.9. *Average prediction error for T.CS$(k+2)$ as a function of the training set length for different values of the loss function weight $L_{1,0}$, in case (a) the value we want to infer is T.CS$(k+2)=0$ and (b) T.CS$(k+2)=1$.*



(a) (b)

Figure 6.10. *Average prediction error for T.CS$(k+2)$ as a function of the training set length for different network topologies (dumbbell network (DNet) and a random mobile network (RNet)), for different values of the time sampling ΔT and congestion levels (LC: low congestion and HC: high congestion), in case (a) the value we want to infer is T.CS$(k+2)=0$ and (b) T.CS$(k+2)=1$.*

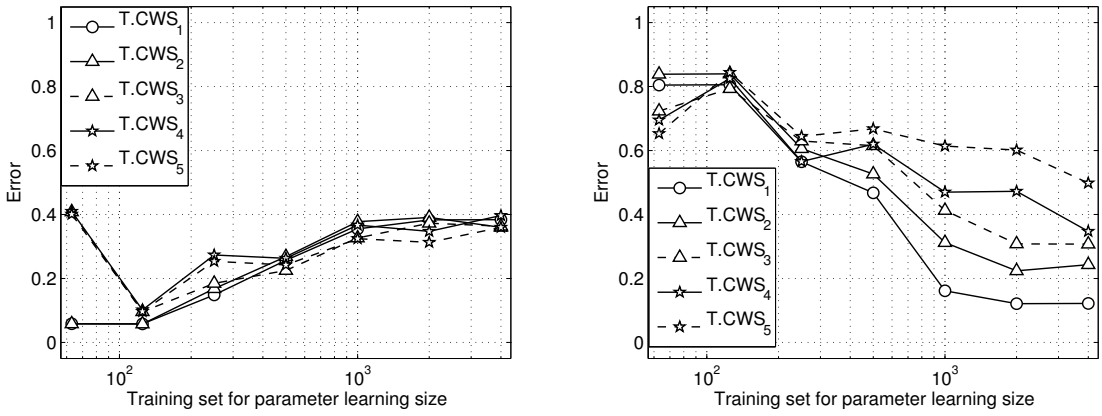


Figure 6.11. *Average prediction error for T.CS$(k+n)$ as a function of the training set length for different values of n, in case (a) the value we want to infer is T.CS$(k+n)=0$ and (b) T.CS$(k+n)=1$.*

recorded and then they become the input for the inference engine. In the y-axis of the figures we represent the average error for the inference, that is the expected value of $|\text{T.CS}(k+n) - \widehat{\text{T.CS}}(k+n)|$, where T.CS$(k+n)$ is the actual value of T.CS at time $k+n$ and $\widehat{\text{T.CS}}(k+n)$ is the inferred value. Since T.CS is a binary variable, this value can be viewed also as the frequency of an error in the estimation. We need to analyze separately the two cases in which the value we aim to infer is T.CS$(k+n)=0$ and T.CS$(k+n)=1$, since otherwise the average error would be dominated by the error in the former case (no congestion), that is a much more frequent event, and the predictor that minimizes the total error would simply give $\widehat{\text{T.CS}}(k+n)=0$, as discussed earlier.

In Figs. 6.8, 6.10 and 6.11, we vary the evidence set, the network scenario and the value of n, respectively, fixing the value of the loss function to $L_{1,0}=7$, and we analyze the performance of the inference in case the value we aim to infer is T.CS$(k+n)=0$ (no congestion), in Fig. 6.8-(a), Fig. 6.10-(a) and Fig. 6.11-(a), and T.CS$(k+n)=1$ (congestion), in Fig. 6.8-(b), Fig. 6.10-(b) and Fig. 6.11-(b). In Fig. 6.8 we represent the average error for the inference of T.CS$(k+2)$ in the dumbbell topology scenario of Fig. 6.6, in the case of low congestion (LC) and we vary the evidence set. In the case of no evidence, the prediction is equal to the prior for T.CS, so we always predict T.CS$(k+2)=0$, indeed in this case the error in Fig. 6.8-(a) is equal to zero, while in Fig. 6.8-(b) the error is equal to one, the maximum. The only predictors that give us useful information are the ones with T.CW in the evidence set. For a

sufficient length of the training set, they give an error of almost 0.25 and 0.35, for congestion and no congestion (false positive), respectively.

In Fig. 6.10 we consider as evidence T.CW(k) and compare the performance for different multi-hop network scenarios, i.e., the dumbbell topology in Fig. 6.6 with low congestion (LC) and high congestion (HC) and with the parameters sampled every $\Delta T = 100$ ms and $\Delta T = 200$ ms, and the random multi-hop network with mobility described in Section 6.4.1. The results depicted in the figure show that the inference engine performs similarly in these cases, with a frequency of false positives between 0.3 and 0.45 and a misclassification rate in case of congestion between 0.1 and 0.25 in all cases. Accordingly, we expect the inference engine to work well also in different kinds of topologies with different environmental conditions.

In Figs. 6.11 we compare the performance of the inference engine for T.CS$(k+n)$, varying the value of $n = 1, \ldots, 5$, where k is the time sample at which we make the prediction and $k+n$ is the time sample of the predicted value $\widehat{\text{T.CS}}(k+n)$. We observe that the performance is almost constant when the actual value to be inferred is T.CS$(k+n) = 0$, as shown in Fig. 6.11-(a), while it varies significantly when T.CS$(k+n) = 1$. In this case, as expected, the performance decreases with n, and for $n = 5$ we have a misclassification rate in case of congestion and in the absence of congestion of about 0.5 and 0.4, respectively, a performance close to the extreme case of the random predictor, and this gives an approximate limit in time to the possibility of predicting congestion with these models.

In Fig. 6.9 we vary the value of the loss function weight to determine a suitable value, considering the evidence X_e =T.CW(k). In case $L_{1,0} = 1$ the inference engine is just predicting the most probable value, $\widehat{\text{T.CS}}(k+n) = 0$. A good choice for $L_{1,0}$ is the one that guarantees a false positive error significantly smaller than 0.5 and minimizes the error in case of congestion, i.e., $L_{1,0} = 7$. Furthermore, Fig. 6.9, in case T.CS$(k+n) = 0$, may seem misleading since the average error grows with longer training set lengths, but this can be explained observing that, for a short training set, $N < 2 \cdot 10^2$, we have a large misclassification error (> 0.7) in case of congestion, so the predicted value is almost always $\widehat{\text{T.CS}}(k+n) = 0$ in this case.

6.5 Conclusions

In this work we proposed the integration of the Bayesian Network (BN) model into an architecture for Cognitive Networking for the optimization of a wireless network scenario. Cognitive networks learn their operating environment and the protocol parameter relationships in order to achieve network-wide performance objectives. In this context, we have shown that the use of BN is very promising for learning the probabilistic structure and designing an inference engine for the chosen parameters. We have seen that for a single-hop network the inference engine can be exploited to achieve performance goals like a minimum guaranteed level for the TCP throughput. Moreover, as an interesting application of BNs for multi-hop cognitive wireless networks, we studied the problem of predicting in advance the congestion status of the network.

A typical cognitive network has the following four major phases of operation: Observe, Learn, Plan and Decide, and Act. The Observe phase focuses on sampling the network protocol parameters as a function of time. In our Cognitive Network approach, the Learn phase deals with learning the structure and parameters of the BN from the observed data. Our use of BN included the construction of a BN model where nodes represent the network protocol parameters and the edges represent the conditional dependencies between them. The Plan and Decide phase deals with inferring the values of necessary protocol parameters from the BN such that the network protocol performance objectives can be achieved. The results support the following observations: (i) BN is a useful tool for cognitive networking to determine, represent and exploit the probabilistic dependencies and conditional independences between protocol parameters; (ii) exploiting BN, we can design an inference engine to accurately predict the behavior of protocol parameters; (iii) we can obtain useful insights on the influence of the data size used for training on the accuracy of network parameter behavior prediction; and (iv) we found that the current value of TCP's congestion window has a strong influence on accurately predicting the congestion status of a multi-hop network at future time instants.

Our future plans include the following: (a) implementing the Act phase of the cognitive network operation, (b) studying more complex BN models to also describe the time dependencies between network parameters, (c) investigating other methods for efficient inference over BN, (d) improving the prediction accuracy, (e) devising algorithms that exploit the predicted congestion status of the network to adapt the value of the congestion window

before severe congestion occurs, and (f) studying the effect of misclassification rates on TCP performance in terms of throughput and packet losses.

Chapter 7

Conclusions

In this thesis we have discussed the application of the cognition paradigm to the performance optimization of different kinds of wireless networks. To perform the four phases of the cognition paradigm, we used techniques from machine learning, convex optimization and probabilistic graphical modeling adapted to our specific needs, and exploited them in a number of realistic scenarios.

In detail, the task of the first approach studied in this thesis is to design an efficient in-network aggregation and data compression technique for Wireless Sensor Networks (WSNs), in order to increase the efficiency of data gathering solutions, while being able to measure large amounts of data with high accuracy. After a preliminary study to quantify the benefit of the use of Compressive Sensing (CS) in a realistic multi-hop scenario, we designed a novel mathematical framework that follows the cognition paradigm. This framework is able to learn the signal statistics required by CS through the use of Principal Component Analysis (PCA) from the past data that is observed and stored. We performed a Bayesian analysis to statistically model the principal components of real WSN signals in order to justify the approach and to show the optimality conditions from a Bayesian perspective. Moreover, we designed a novel and effective data compression/recovery technique, able to estimate the reconstruction error and to make decisions on the future behavior of the network. We proposed also a feedback mechanism to iteratively change the number of nodes transmitting at each round, making the decisions effective. Such techniques were integrated in an existing WSN architecture. Finally, the performance of the data collection techniques that exploit the proposed framework was evaluated and we proposed also a performance comparison of data fitting techniques using real signals from actual WSN testbeds.

In the second part of this thesis we applied the cognitive framework to a wireless network, e.g., a WLAN or a tactical network, in order to optimize specific performance metrics using information from different network layers in a cross-layer fashion. We proposed an architecture for cognitive networking that exploits a probabilistic graphical model, i.e., Bayesian Network (BN), for learning the statistical relationships among a set of parameters of the protocol stack. In particular, we learned such relationships from the observation of the past behavior of the network, using Structure Learning (SL) and Parameter Learning (PL) techniques for BNs. We used the knowledge learned with the BN to design an inference engine and to make inference on present or future values of some parameters of the protocol stack. We applied these tools to study and optimize realistic wireless scenarios (single-hop and multi-hop). Finally, we evaluated the performance of the BN inference engine's accuracy to infer the value of TCP throughput in a single-hop wireless network scenario and we made a performance analysis of the BN inference engine's accuracy to predict TCP's congestion status in a realistic multi-hop wireless network scenario.

7.1 Future Directions

The application of cognition for performance optimization in a wireless network is a challenging topic, since it is necessary to choose the appropriate optimization technique from the literature and to appropriately tune its parameters to deal with a realistic scenario, where many ideal assumptions may not hold. Nonetheless, using cognition may lead to significant performance increasing in many scenarios that still need optimization. In this thesis, we applied these tools to real data, and we solved some specific problems, but there are still a lot of optimization opportunities. Regarding the first part, the approach with the learning phase (PCA) and the recovery with convex optimization (CS) is very promising for WSNs, but there is still a need for a general model that is able to capture all the relevant statistical characteristics of the signals that may be sensed by a WSN. Ideally, the technique should be able to adapt also to the extreme cases of a signal fully correlated or made of independent components. This general model should be used to classify the signals according to their characteristics. The technique for compression/recovery should be able to recognize the class to which the sensed signal belongs and to act on the WSN accordingly, e.g., choosing the sampling rate or the data fitting technique depending on the signal characteristics. Moreover, the extended technique should be implemented and tested in an existing WSN.

As regards the second approach proposed in this thesis, the exploitation of BNs and all the learning techniques to infer a suitable probabilistic structure from the observation of the data is a very promising way for the optimization of wireless networks. In this thesis, we used these methods to face very specific tasks, such as the prediction of congestion in multi-hop networks. On the one side, it is very interesting to complete this study designing an ad hoc transport layer technique able to fully exploit the knowledge of the probability of congestion, i.e., able to iteratively adapt the congestion windows to an optimal value in order to maximize the throughput and at the same time minimize the occurrence of congestion. On the other side, the BN approach in this thesis has been presented in a very general way and can be easily adapted to face different optimization problems in a wireless network, e.g., it may be used to predict the QoS for call admission control in a wireless network with VoIP traffic. Furthermore, this approach has a natural extension, that is the study of the temporal evolution of the probabilistic relations among the set of chosen parameters. In this thesis we presented the notation for the Dynamic Bayesian Network (DBN) model, but in order to learn the relations directly from the data there is still a need to design specific Structure Learning (SL) and Parameter Learning (PL) techniques for learning the DBN model from the temporal evolution of the data. The DBN model is more general than the BN model and it may be exploited for learning and optimization of a wider set of networks.

Appendix A

Further performance analysis

In this Appendix we analyze the performance of the data collection framework varying some parameters of the techniques, in order to infer an acceptable value for each parameter, i.e., a value for which the technique perform well in the simulation considered. This value is used in the data collection performance analysis of Chapter 5. In Section A.1 we set up the parameters δ and ϵ for the NESTA algorithm, using the notation introduced in Section 2.1.2. In Section A.2 instead we set up the parameters K_1 and K_2 of the *2 Phases* data collection technique, that is used for the data collection techniques comparison in Section 5.5.

A.1 NESTA: Parameter setting

For the sake of this performance analysis, we use signals gathered from the WSN testbed named T1 in Section 4.3, which has been deployed on the ground floor of the Department of Information Engineering, University of Padova, Italy, including $N = 68$ TmoteSky wireless sensor nodes that communicate with IEEE 802.15.4 radio transceivers. These sensors can measure five different signals: temperature, humidity, luminosity in two different ranges (320–730 and 320–1100 nm), and the voltage of their battery. For the performance analysis we have considered temperature and humidity, which have high spatial and temporal correlation, and luminosity, that instead is an unpredictable signal with high variability in space and time. In the following we test the performance of the NESTA algorithm integrated in the Signal Reconstruction and Feedback Control module.

The NESTA algorithm has been rewritten in C++ and integrated into the signal reconstruction module of WSN-Control. The implementation of the algorithm is available

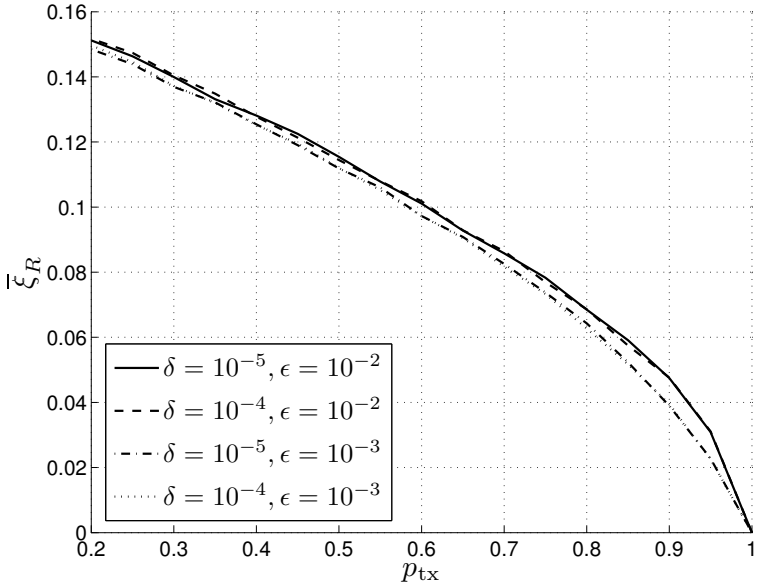


Figure A.1. *Performance of the NESTA algorithm for a luminosity signal, varying δ and ϵ.*

from [83], while we refer the reader to [12] for the original version of NESTA. As a first step, we selected the parameters of the optimization δ, which determines when the algorithm stops, see (2.11), ϵ, which is a constant used to relax the equality in (2.7), see (2.13) and μ, which is related to the accuracy of the approximation of the ℓ_1-norm, see (2.16). In our results we have fixed the value of μ, keeping into account that smaller values imply a better approximation and bigger values imply a faster convergence. We have chosen $\mu = 10^{-2}$, since by simulations we have noticed that this value only marginally impacts the result of the minimization while guaranteeing a fast convergence. In Figs. A.1 and A.2 we represent the performance of the algorithm varying δ and ϵ for the luminosity signal. The x-axis represents p_{tx} during the monitoring phase, while the y-axis represents the average relative reconstruction error in the whole simulation ($k = 1, \ldots, K$), defined in (5.7), and reported here:

$$\overline{\xi}_R = \frac{1}{K} \sum_{k=1}^{K} \xi_R^{(k)} \ , \tag{A.1}$$

where $\xi_R^{(k)}$ is the relative reconstruction error at time k, i.e.,

$$\xi_R^{(k)} = \frac{\|\mathbf{x}^{(k)} - \hat{\mathbf{x}}^{(k)}\|_2}{\|\mathbf{x}^{(k)}\|_2} \ . \tag{A.2}$$

In Fig. A.1 we notice that varying the value of δ in the range $[10^{-4}, 10^{-5}]$ does not affect the recovery performance; for our results we picked $\delta = 10^{-5}$. In Fig. A.2 instead we vary

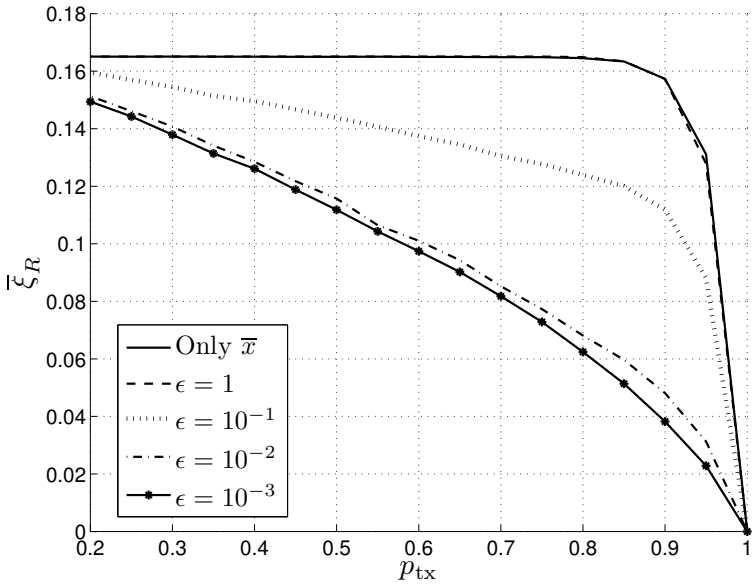


Figure A.2. *Performance of the NESTA algorithm for a luminosity signal, with $\delta = 10^{-5}$ and varying ϵ.*

$\epsilon \in [10^{-3}, 1]$. For $\epsilon \leq 10^{-2}$, the reconstruction error is quite insensitive to ϵ, while it sharply increases for larger values of ϵ. We decided to adopt $\epsilon = 10^{-2}$ as it provided reasonable performance across all our experiments. It is worth noting that for $\epsilon = 1$ we reach the upper bound of the reconstruction error, represented by the error in the case where we have no optimization, but we consider as reconstructed signal $\hat{\mathbf{x}}^{(k)} = \overline{\mathbf{x}}$, i.e., the average calculated over the previous training set, see (2.21). This is due to the fact that for a large value of ϵ the constraints are relaxed, so the set of possible solutions $\mathcal{Q}'_p$ includes also the null solution $\hat{\mathbf{s}}^{(k)} = 0$, that is the sparsest one in Eq. (2.5). In this case the algorithms (2.24) gives as outcome $\hat{\mathbf{x}}^{(k)} = \overline{\mathbf{x}}$.

A.2 2 Phases Data Collection technique

In this section we apply the joint CS and PCA recovery technique integrated in the *2 Phases* Data Collection Technique (DCT) described in Section 5.2.2. The aim of this performance analysis is to compare the performance of *2 Phases* with a standard data compression and recovery technique, names *RS-Spline*, described in detail in Section 3.5, that consists in a distributed sampling of the signal, following the random sampling scheme detailed in Section 5.2.1, and a standard reconstruction by spline interpolation [55]. Moreover, we study the impact of the two parameters K_1 and K_2 in the performance of *2 Phases*. K_1 and K_2 are

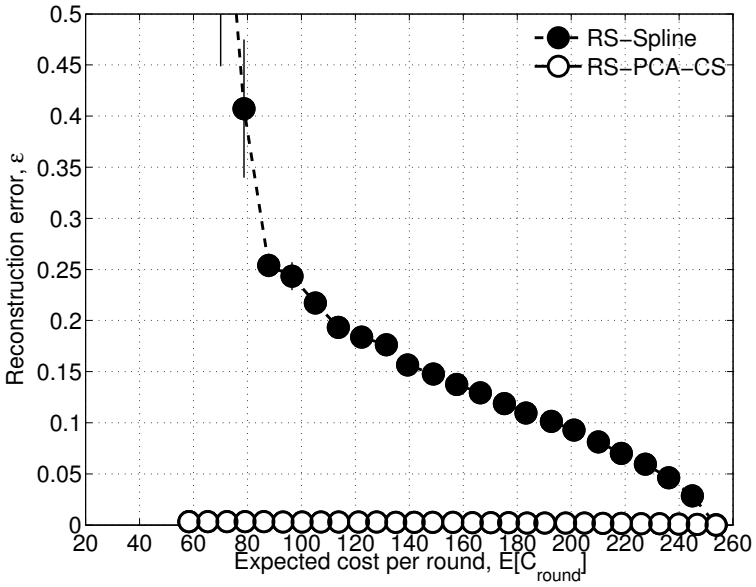


Figure A.3. *Reconstruction Error $\overline{\xi}_R$ vs $E[C_{\mathrm{round}}]$: humidity.*

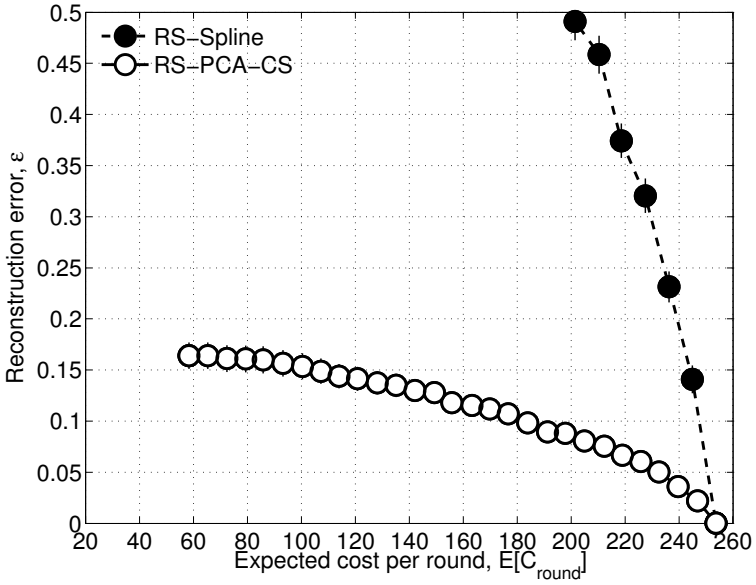


Figure A.4. *Reconstruction Error $\overline{\xi}_R$ vs $E[C_{\mathrm{round}}]$: luminosity.*

the length in data collection rounds of the training phase and the length of the monitoring phase, respectively. In this section we consider signals gathered from the WSN deployment T1, described in Section 4.3.

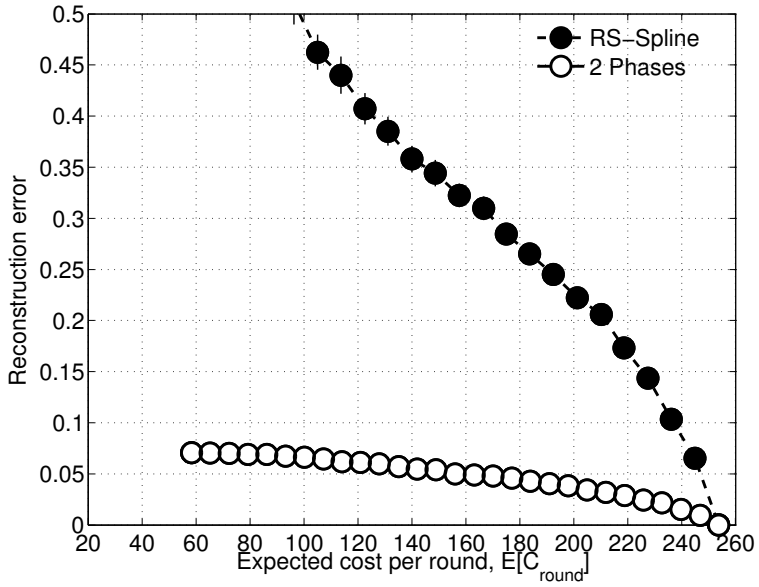


Figure A.5. *Average Reconstruction Error $\overline{\xi}_R$ (signals 1–5) vs $E[C_{\text{round}}]$.*

We define the ratio ζ between the duration of monitoring and training phases:

$$\zeta = \frac{K_2}{K_1} \,, \tag{A.3}$$

that should be chosen according to the temporal correlation of the observed phenomena.

In Figs. A.3–A.5 we show the performance in terms of reconstruction error $\overline{\xi}_R$, defined in (A.1), as a function of the average cost per round, which in this section is given by the number of transmissions for the collection of a single instance of the signal $\mathbf{x}^k$ in a multi-hop network[1]. In these plots each training phase lasts $K_1 = 2$ rounds and $\zeta = 4$ (the impact of these parameters is addressed at the end of this section). A training phase entails a cost $K_1 C_N$, where C_N is the total number of transmissions needed to gather the readings from all nodes. The average number of packets sent during the following $2\zeta = 8$ monitoring phases depends on p_{tx}, which is varied from $1/N$ to 1, and $\overline{\xi}_R$ is computed for each case. For a given $p_{tx} = L/N$ each monitoring phase has an average total cost of $\zeta K_1 E[C_L]$, where $E[C_L]$ is the total number of transmissions needed to collect the readings from the source nodes during a data collection round. Thus, the average cost per round is calculated as:

$$E[C_{\text{round}}] = \frac{C_N + \zeta E[C_L]}{1 + \zeta} \,. \tag{A.4}$$

[1]The reason for which we represent in the x-axis the cost in terms of total number of transmissions instead of the probability of transmission P_{tx} is due to the fact that in this section we want to do a fair comparison with a standard technique like RS-Spline, so we need to define a cost function that is suitable for both the techniques.

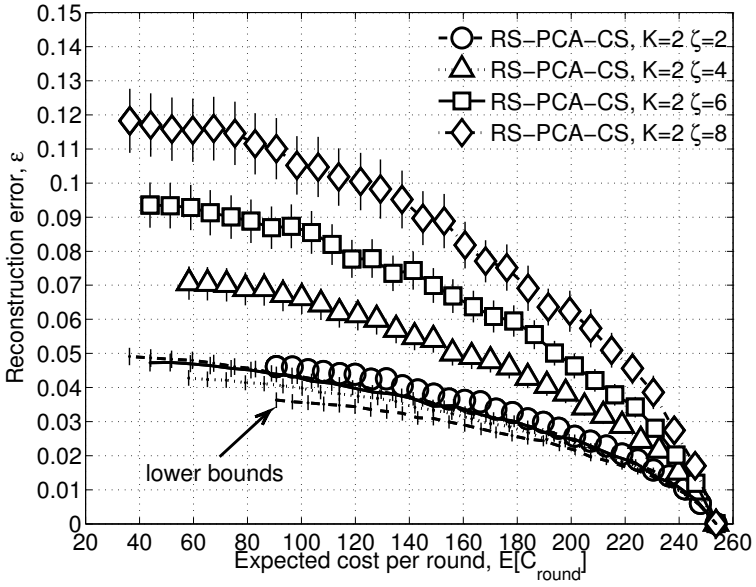


Figure A.6. *Average $\overline{\xi}_R$ (signals temperature, humidity, luminosity and voltage) vs $E[C_{\text{round}}]$, $K_1 = 2$, $\zeta \in \{2, 4, 6, 8\}$.*

For comparison, in the plots we also show the recovery performance of RS-Spline. The cost per round for RS-Spline is $E[C_L]$.

In Figs. A.3–A.5 we demonstrate the effectiveness of our recovery technique, i.e., *2 Phases*. These results show that the mathematical framework of Chapter 4 provides substantial benefits with respect to standard data gathering schemes. In Fig. A.3 $\overline{\xi}_R$ is close to zero as this specific signal varies slowly in time, i.e., its correlation structure is quasi-stationary during a monitoring phase. Also, we note that for those signals showing higher variations over space and time, such as luminosity, *RS-Spline* has unsatisfactory performance.

In the last two graphs, Figs. A.6 and A.7, we show the impact of K_1 and ζ on the performance. From Fig. A.6 (fixed K) we see that decreasing ζ leads to: 1) a higher minimum admissible cost to bear per round due to an increase of the overhead 2) despite the increase of overhead, a decreased cost per round for a given quality goal since the signal's reconstuction algorithm uses fresher information and 3) a smaller variance for $\overline{\xi}_R$. From Fig. A.7 (fixed ζ) we see that decreasing K is beneficial. This means that, for the considered signals, a smaller reconstruction error is achievable through more frequent updates of $\overline{\mathbf{x}}$ and $\widehat{\mathbf{\Sigma}}$. In Figs. A.6 and A.7 solid and dotted lines without marks represent lower bounds on the error recovery performance, which are obtained as follows. For each (K, ζ) pair, the actual recovery performance evaluates the reconstruction accuracy of the signal when training and

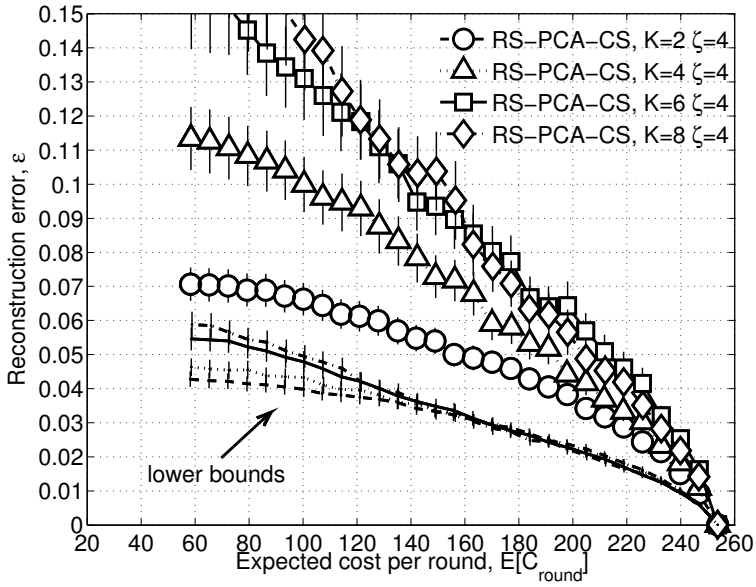


Figure A.7. *Average $\overline{\xi}_R$ (signals temperature, humidity, luminosity and voltage) vs $E[C_{\text{round}}]$, $K_1 \in \{2, 4, 6, 8\}$, $\zeta = 4$.*

monitoring phases alternate. In this case, during a monitoring period each input signal $\mathbf{x}^k$ is reconstructed using *2 Phases* with $\overline{\mathbf{x}}$ and $\widehat{\mathbf{\Sigma}}$ calculated exploiting the signals gathered during the last training phase. Differently, the lower bound on the reconstruction error of *2 Phases* for each (K_1, ζ) pair and for each time k is obtained using *2 Phases* with $\overline{\mathbf{x}}$ and $\widehat{\mathbf{\Sigma}}$ calculated assuming perfect knowledge of the previous K instances of the signal $\mathbf{x}^{k-1}, \ldots, \mathbf{x}^{k-K_1}$. The cost associated with the new $\overline{\xi}_R$ is set equal to that of the real *2 Phases* scheme for the given (K_1, ζ) pair. These curves reveal the impact of the obsolescence of $\overline{\mathbf{x}}$ and $\widehat{\mathbf{\Sigma}}$ during the monitoring phase for the considered signals. In particular, the recovery performance degrades for either increasing ζ (Fig. A.6) or K_1 (Fig. A.7).

List of Publications

The work presented in this thesis has appeared in the articles reported below.

Conference papers

[C1] **G. Quer**, H. Meenakshisundaram, B.R. Tamma, B.S. Manoj, R. Rao and M. Zorzi, "Cognitive Network Inference through Bayesian Network Analysis", *IEEE Globecom 2010*, Miami, FL, Dec. 2010.

[C2] **G. Quer**, D. Zordan, R. Masiero, M. Zorzi and M. Rossi, "WSN-Control: Signal Reconstruction through Compressive Sensing in Wireless Sensor Networks", *IEEE LCN (SenseApp Workshop)*, Denver, CO, Oct. 2010.

[C3] **G. Quer**, H. Meenakshisundaram, B.R. Tamma, B.S. Manoj, R. Rao and M. Zorzi, "Using Bayesian Networks for Cognitive Control of Multi-hop Wireless Networks", *MILCOM 2010*, San Jose, CA, Nov. 2010.

[C4] R. Masiero, **G. Quer**, D. Munaretto, M. Rossi, J. Widmer and M. Zorzi, "Data Acquisition through joint Compressive Sensing and Principal Component Analysis", *IEEE Globecom 2009*, Honolulu, HW, Nov.-Dec. 2009.

[C5] R. Masiero, **G. Quer**, M. Rossi and M. Zorzi, "A Bayesian Analysis of Compressive Sensing Data Recovery in Wireless Sensor Networks", *The International Workshop on Scalable Ad Hoc and Sensor Networks, SASN'09*, Saint Petersburg, Russia, Oct. 2009.

[C6] **G. Quer**, R. Masiero, D. Munaretto, M. Rossi, J. Widmer and M. Zorzi, "On the Interplay Between Routing and Signal Representation for Compressive Sensing in Wireless Sensor Networks", *Information Theory and Applications Workshop (ITA 2009)*, San Diego, CA, Jan.-Feb. 2009.

Journal papers under submission

[J1] **G. Quer**, R. Masiero, M. Rossi and M. Zorzi, "SCoRe1: Sensing Compression and Recovery through Online Estimation for Wireless Sensor Networks", *Under submission to IEEE Trans. Wireless Communication*.

[J2] R. Masiero, **G. Quer**, G. Pillonetto, M. Rossi and M. Zorzi, "Sampling and Recovery with Compressive Sensing in Real Wireless Sensor Networks", *Under submission to IEEE Trans. Wireless Communication*.

Bibliography

[1] C. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[2] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[3] D. Koller and N. Friedman, *Probabilistic Graphical Models: Principles and Techniques*. The MIT Press, 2009.

[4] J. Mitola III, "Cognitive Radio: An Integrated Agent Architecture for Software Defined Radio," Ph.D. dissertation, Royal Institute of Technology (KTH), Sweden, May 2000.

[5] D. Donoho, "Compressed sensing," *IEEE Trans. on Information Theory*, vol. 52, no. 4, pp. 4036–4048, Apr. 2006.

[6] E. Candès, J. Romberg, and T. Tao, "Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information," *IEEE Trans. on Information Theory*, vol. 52, no. 2, pp. 489–509, Feb. 2006.

[7] E. Candès and T. Tao, "Near optimal signal recovery from random projections: Universal encoding strategies?" *IEEE Trans. on Information Theory*, vol. 52, no. 12, pp. 5406–5425, Dec. 2006.

[8] W. Bajwa, J. Haupt, A. Sayeed, and R. Nowak, "Joint source-channel communication for distributed estimation in sensor networks," *IEEE Trans. on Information Theory*, vol. 53, no. 10, pp. 3629–3653, Oct. 2007.

[9] R. Masiero, "Distributed Optimization and Data Recovery for Wireless Networking," Ph.D. dissertation, Department of Information Engineering, University of Padova, Italy, Jan. 2011.

[10] J. Haupt, W. Bajwa, M. Rabbat, and R. Nowak, "Compressive Sensing for Networked Data: a Different Approach to Decentralized Compression," *IEEE Signal Processing Magazine*, vol. 25, no. 2, pp. 92–101, Mar. 2008.

[11] J. Liu, M. Adler, D. Towsley, and C. Zhang, "On optimal communication cost for gathering correlated data through wireless sensor networks," in *ACM MOBICOM*, Los Angeles, CA, USA, Sep. 2006.

[12] S. Bercker, J. Bobin, and E. J. Candés, "NESTA: a fast and accurate first order method for sparse recovery." *Submitted for publication*. [Online]. Available: http://www-stat.stanford.edu/~candes/papers/NESTA.pdf

[13] C. R. Rao, "The Use and Interpretation of Principal Component Analysis in Applied Research," *Sankhya: The Indian Journal of Statistics*, vol. 26, pp. 329–358, 1964.

[14] E. Candès and M. Wakin, "An Introduction to Compressive Sampling," *IEEE Signal Processing Magazine*, vol. 2, no. 25, pp. 21–30, Mar. 2008.

[15] Y. E. Nesterov, "A method for unconstrained convex minimization problem with rate of convergence $\mathcal{O}(1/k^2)$," *Doklady AN USSR (Translated as Soviet Math. Docl.)*, vol. 269, no. 3, pp. 543–547, 1983.

[16] ——, "Smooth minimization of non–smooth functions," *Mathematical Programming*, vol. 103, no. 1, pp. 127–152, may 2005.

[17] D. Margaritis, "Learning Bayesian Network Model Structure from Data," Ph.D. dissertation, School of Computer Science - Carnegie Mellon University, Pittsburgh, PA, May 2003.

[18] F. V. Jensen and T. D. Nielsen, *Bayesian Networks and Decision Graphs*. Springer, 2007.

[19] G. Schwarz, "Estimating the Dimension of a Model," *The Annals of Statistics*, vol. 6, no. 2, pp. 461–464, 1978.

[20] S. Yang and K.-C. Chang, "Comparison of score metrics for Bayesian network learning," *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans* , vol. 32, pp. 419–428, May 2002.

[21] N. Friedman and D. Koller, "Being Bayesian About Network Structure. A Bayesian Approach to Structure Discovery in Bayesian Networks," *Machine Learning Journal*, vol. 50, pp. 95–125, 2003.

[22] T. Richardson and R. Urbanke, *Modern Coding Theory*. Cambridge University Press, 2007.

[23] A. Scaglione and S. D. Servetto, "On the Interdependence of Routing and Data Compression in Multi-Hop Sensor Networks," in *ACM MOBICOM*, Atlanta, GA, USA, Sep. 2002.

[24] S. Servetto, "Distributed Signal Processing Algorithms for the Sensor Broadcast Problem," in *Conference on Information Sciences and Systems (CISS)*, Baltimore, MD, USA, Mar. 2003.

[25] ——, "Sensing Lena - Massively Distributed Compression of Sensor Images," in *IEEE Conference on Image Processing (ICIP)*, Ithaca, NY, USA, Sep. 2003.

[26] S. Pradhan and K. Ramchandran, "Distributed Source Coding Using Syndromes (DISCUS): Design and Construction," *IEEE Transactions on Information Theory*, vol. 49, no. 3, pp. 626–643, Mar. 2003.

[27] H. Luo and G. Pottie, "Routing Explicit Side Information for Data Compression in Wireless Sensor Networks," in *Distributed Computing in Sensor Systems (DCOSS)*, Marina del Rey, CA, USA, Jun. 2005.

[28] M. Gastpar, P. Dragotti, and M. Vetterli, "The Distributed Karhunen-Loeve Transform," *IEEE Transactions on Information Theory*, vol. 52, no. 2, pp. 5177–5196, Dec. 2006.

[29] D. Slepian and J. Wolf, "Noiseless Coding of Correlated Information Sources," *IEEE Trans. on Information Theory*, vol. 19, no. 4, pp. 471–480, Jul. 1973.

[30] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. John Wiley, 1991.

[31] Z. Xiong, A. Liveris, and S. Cheng, "Distributed Source Coding for Sensor Networks," *IEEE Signal Processing Magazine*, vol. 21, no. 5, pp. 80–92, Sep. 2004.

[32] S. Pattem, B. Krishnamachari, and R. Govindan, "The Impact of Spatial Correlation on Routing with Compression in Wireless Sensor Networks," in *Int. Conf. on Information Processing in Sensor Networks (IPSN)*, Berkeley, CA, USA, Apr. 2004.

[33] A. Ciancio, S. Pattem, A. Ortega, and B. Krishnamachari, "Energy-Efficient Data Representation and Routing for Wireless Sensor Networks Based on a Distributed Wavelet Compression Algorithm," in *Information Processing in Sensor Networks (IPSN)*, Nashville, Tennessee, USA, Apr. 2006.

[34] Y. Yu, B. Krishnamachari, and V. K. Prasanna, "Data Gathering with Tunable Compression in Sensor Networks," *IEEE Trans. on Parallel and Distributed Systems*, vol. 19, no. 2, pp. 276–287, 2008.

[35] J. Romberg, "Imaging via Compressive Sampling," *IEEE Signal Processing Magazine*, vol. 25, no. 2, pp. 14–20, Mar. 2008.

[36] J. Haupt and R. Nowak, "Signal reconstruction from noisy random projections," *IEEE Trans. on Information Theory*, vol. 52, no. 9, pp. 4036–4048, Sep. 2006.

[37] M. Rabbat, J. Haupt, A. Singh, and R. Novak, "Decentralized Compression and Predistribution via Randomized Gossiping," in *Information Processing in Sensor Networks (IPSN)*, Nashville, Tennessee, USA, Apr. 2006.

[38] G. Shen, S. Y. Lee, S. Lee, S. Pattem, A. Tu, B. Krishnamachari, A. Ortega, M. Cheng, S. Dolinar, A. Kiely, M. Klimesh, and H. Xie, "Novel distributed wavelet transforms and routing algorithms for efficient data gathering in sensor webs," in *NASA Earth Science Technology Conference (ESTC2008)*, University of Maryland, MD, USA, Jun. 2008.

[39] M. Duarte, M. Wakin, and R. Baraniuk, "Wavelet-domain compressive signal reconstruction using a Hidden Markov Tree model," in *Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on*, Apr. 2008, pp. 5137–5140.

[40] M. Coates, Y. Pointurier, and M. Rabbat, "Compressed Network Monitoring for IP and All-optical Networks," in *Internet Measurement Conference (IMC)*, San Diego, CA, USA, Oct. 2007.

[41] M. Duarte, M. Wakin, D. Baron, and R. Baraniuk, "Universal Distributed Sensing via Random Projections," in *Information Processing in Sensor Networks (IPSN)*, Nashville, TN, USA, Apr. 2006.

[42] M. Duarte, S. Sarvotham, D. Baron, M. Wakin, and R. Baraniuk, "Distributed Compressed Sensing of Jointly Sparse Signals," in *Signals, Systems and Computers, 2005. Conference Record of the Thirty-Ninth Asilomar Conference on*, Oct. 2005.

[43] S. Lee, S. Pattem, M. Sathiamoorthy, B. Krishnamachari, and A. Ortega, "Compressed Sensing and Routing in Multi-Hop Networks," Tech. Rep., Feb. 2009.

[44] S. Shintre, B. Dey, S. Katti, S. Jaggi, D. Katabi, and M. Medard, ""Real" and "Complex" Network Codes: Promises and Challenges," in *Network Coding, Theory and Applications, 2008. NetCod 2008. Fourth Workshop on*, Jan. 2008, pp. 1–6.

[45] S. Katti, S. Shintre, S. Jaggi, D. Katabi, and M. Medard, "Real Network Codes," in *Forty-Fifth Annual Allerton Conference*, Allerton House, UIUC, IL, USA, Sep. 2007.

[46] H. Mohimani, M. Babaie-Zadeh, and C. Jutten, "A fast approach for overcomplete sparse decomposition based on smoothed L0 norm," *IEEE Trans. on Signal Processing*, 2009, Accepted for publication.

[47] A. Jindal and K. Psounis, "Modeling Spatially Correlated Data in Sensor Networks," *ACM Transactions on Sensor Networks*, vol. 2, no. 4, pp. 466–499, Nov. 2006.

[48] "MIT Wireless Network Coverage," Last time accessed: January 2009. [Online]. Available: http://nie.chicagotribune.com/activities_120505.htm

[49] T. Kamakaris and J. V. Nickerson, "Connectivity maps: Measurements and applications," in *Proceedings of the 38th Hawaii International Conference on System Sciences*, Hilton Waikoloa Village Island of Hawaii, HI, USA , Jan. 2005.

[50] "EPFL LUCE SensorScope WSN," Last time accessed: January 2009. [Online]. Available: http://sensorscope.epfl.ch/

[51] "Tropical Rainfall Measuring Mission," Last time accessed: January 2009. [Online]. Available: http://trmm.gsfc.nasa.gov

[52] "Partnership for Interdisciplinary Studies of Coastal Oceans," Last time accessed: January 2009. [Online]. Available: www.piscoweb.org

[53] "ENEA: Ente Nuove Tecnologie e l'Ambiente," Last time accessed: January 2009. [Online]. Available: www.enea.it

[54] Napler Addison, *The Illustrated Wavelet Transform Handbook*. Taylor & Francis, 2002.

[55] D. T. Sandwell, "Biharmonic Spline Interpolation of GEOS-3 and SEASAT Altimeter Data," *Geophysical Research Letters*, vol. 14, no. 2, pp. 139–142, 1987.

[56] E. Candès, M. B. Wakin, and S. Boyd, "Enhancing Sparsity by Reweighted ℓ_1 Minimization," *Journal of Fourier Analysis and Applications*, vol. 14, pp. 877–905, Dec. 2008.

[57] G. R. Belitskii and Yu. I. Lyubich, *Matrix norms and their applications*. Birkhäuser, 1988.

[58] A. Figueiredo, R. Nowak, and S. Wright, "Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse problems," *IEEE Journal of Selected Topics in Signal Processing*, no. 1, pp. 586–597, 2007.

[59] S. Gull, "The Use and Interpretation of Principal Component Analysis in Applied Research," *Maximum Entropy and Bayesian Methods in Science and Engineering*, vol. 1, pp. 53–74, 1988.

[60] J. Skilling, *Maximum Entropy and Bayesian Methods*. Kluwer academic, 1989.

[61] S. Ji, Y. Xue, and L. Carin, "Bayesian Compressive Sensing," *IEEE Trans. on Signal Processing*, vol. 56, no. 6, pp. 2346–2356, Jun. 2008.

[62] T. Hastie, R. Tibshirani, and J. Friedman, *The elements of statistical learning*. Springer, 2008.

[63] P. Casari, A. P. Castellani, A. Cenedese, C. Lora, M. Rossi, L. Schenato, and M. Zorzi, "The "Wireless Sensor Networks for City-Wide Ambient Intelligence (WISE-WAI)" Project," *Sensors*, vol. 9, no. 6, pp. 4056–4082, May 2009.

[64] R. Crepaldi, S. Friso, A. F. Harris III, M. Mastrogiovanni, C. Petrioli, M. Rossi, A. Zanella, and M. Zorzi, "The Design, Deployment, and Analysis of SignetLab: A Sensor Network Testbed and Interactive Management Tool," in *IEEE Tridentcom*, Orlando, FL, US, May 2007.

[65] "EPFL Grand-St-Bernard SensorScope WSN," Last time accessed: September 2009. [Online]. Available: http://sensorscope.epfl.ch/index.php/Grand-St-Bernard_Deployment

[66] "CitySense," Last time accessed: October 2009. [Online]. Available: http://www.citysense.net

[67] T. Watteyne, D. Barthel, M. Dohler, and I. Auge-Blum, "Sense&Sensitivity: A Large-Scale Experimental Study of Reactive Gradient Routing," *Measurement Science and Technology, Special Issue on Wireless Sensor Networks: Designing for Real-world Deployment and Deployment Experiences*, vol. 21, no. 12, Oct. 2010.

[68] D. J. MacKay, "Bayesian Interpolation," *Neural Computation Journal*, vol. 4, no. 3, pp. 415–447, May 1992.

[69] K.M. Hanson, *Bayesian and Related Methods in Image Reconstruction from Incomplete Data*. H. Stark, ed., 1987.

[70] K. Fan, "On a theorem of Weil concerning eigenvalues of linear transformation I," *Proc. of the National Academy of Sciences*, vol. 35, pp. 652–655, 1949.

[71] A. Barron, "Entropy and the Central Limit Theorem," *The Annals of Probability*, vol. 14, no. 1, pp. 336–342, Jan. 1986.

[72] A. P. Castellani, N. Bui, P. Casari, M. Rossi, Z. Shelby, and M. Zorzi, "Architecture and Protocols for the Internet of Things: A Case Study," in *International Workshop on the Web of Things (WoT)*, Mannheim, Germany, Mar. 2010.

[73] D. Heffelfinger, *Java EE 5 Development with NetBeans 6*. Packt Publishing, 2008.

[74] R. W. Thomas, D. H. Friend, L. A. DaSilva, and A. B. MacKenzie, "Cognitive networks: Adaptation and learning to achieve end-to-end performance objectives," *IEEE Communications Magazine*, vol. 44, no. 12, pp. 51–57, December 2006.

[75] B. Manoj, R. Rao, and M. Zorzi, "Cognet: a cognitive complete knowledge network system," *IEEE Wireless Communications*, vol. 15, no. 6, pp. 81–88, December 2008.

[76] V. Srivastava and M. Motani, "Cross-layer design: A survey and the road ahead," *IEEE Communications Magazine*, vol. 43, no. 12, pp. 112–119, December 2005.

[77] V. Kawadia and P. R. Kumar, "A cautionary perspective on cross layer design," *IEEE Wireless Communications Magazine*, vol. 12, no. 1, pp. 3–11, February 2005.

[78] http://www.nsnam.org/.

[79] D. Lampasi, "An Alternative Approach to Measurement based on Quantile functions," *Measurement*, vol. 41, no. 9, pp. 994–1013, 2008.

[80] K. Murphy, "Bayes Net toolbox for Matlab," Last time accessed: March 2010. [Online]. Available: code.google.com/p/bnt/

[81] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2003.

[82] "Minstrel physical layer rate adaptation algorithm," Last time accessed: April 2010. [Online]. Available: madwifi-project.org/browser/madwifi/trunk/ath_rate/minstrel/minstrel.txt

[83] D. Zordan, G. Quer, R. Masiero, and M. Rossi, "C++ Implementation of the NESTA Algorithm for Signal Recovery through Compressive Sensing," May 2010. [Online]. Available: http://www.dei.unipd.it/~rossi/software.html

Acknowledgments

To my family. You have supported me in all my choices, you let me know you are with me, even when I am far from home, you let me feel home, whenever I come back.

To my sister, Laura. We have spent a lot of time together, sharing our past and our view of the future, thank you for asking and for giving me the joy to answer.

To my shining star, Daniela. You are an important presence in my life, even when we are so far. You wait for me when I need to go, and you hug me when I come back. Our way has just started.

To my friends at DEI. To Riccardo, you have been a valid colleague in these years, and you are a great friend. To Davide, for all the laughs at work and for all the times we meet outside. To Alfred, for the funny situations you have created in these years. To all the people at DEI, the PhD students, the Post-Docs, the researchers, the professors and all the stuff, I have spent a wonderful time here. Thank you.

To all my friends, that I have met in Fonte, Bassano, Padova, Oulu. Thank you for being around me in all these years. To my friends from San Diego, Emanuele, Lorenzo, Riccardo, Santi, Carol and George, and all the people I have met there, it has been an unforgivable experience.

To Prof. Michele Zorzi. You gave me the opportunity to do research, you supported my motivation and gave me new ideas when I was lost, thanks. To Prof. Ramesh Rao, working with you at UC San Diego was a great chance for professional and human growth.

Giorgio