SPRINGER BRIEFS IN ELECTRICAL AND COMPUTER ENGINEERING · SIGNAL PROCESSING

Giulio Coluccia
Chiara Ravazzi
Enrico Magli

Compressed Sensing for Distributed Systems

Springer

SpringerBriefs in Electrical and Computer Engineering

Signal Processing

More information about this series at http://www.springer.com/series/11560

Giulio Coluccia · Chiara Ravazzi
Enrico Magli

Compressed Sensing for Distributed Systems

Springer

Giulio Coluccia
Department of Electronics
and Telecommunications
Polytechnic University of Turin
Turin
Italy

Enrico Magli
Department of Electronics
and Telecommunications
Polytechnic University of Turin
Turin
Italy

Chiara Ravazzi
Department of Electronics
and Telecommunications
Polytechnic University of Turin
Turin
Italy

ISSN 2191-8112 ISSN 2191-8120 (electronic)
SpringerBriefs in Electrical and Computer Engineering
ISSN 2196-4076 ISSN 2196-4084 (electronic)
SpringerBriefs in Signal Processing
ISBN 978-981-287-389-7 ISBN 978-981-287-390-3 (eBook)
DOI 10.1007/978-981-287-390-3

Library of Congress Control Number: 2015939233

Springer Singapore Heidelberg New York Dordrecht London

Printed on acid-free paper

Springer Science+Business Media Singapore Pte Ltd. is part of Springer Science+Business Media (www.springer.com)

Preface

Compressed sensing is a new technique for nonadaptive compressed acquisition, which takes advantage of signal sparsity and allows signal recovery starting from few linear measurements. Distributed scenarios commonly arise in many applications, where data are inherently scattered across a large geographical area. This applies, for example, to sparse event detection in wireless networks, distributed indoor localization, and distributed tracking in sensor networks. Also distributed sources naturally arise in wireless sensor networks, where sensors may acquire over time several readings of the same natural quantity, e.g., temperature, in different points of the same environment. Such data can be transmitted to a fusion center for joint processing. In this case, a centralized reconstruction system employs joint recovery algorithms exploiting the correlations among the signals coming from the various sensors to enhance the reconstruction fidelity. A drawback of this model is that, particularly in large-scale networks, gathering all data to a central processing unit may be inefficient, as a large number of hops have to be taken, requiring a significant amount of energy for communication over the wireless channel. Moreover, this may also introduce delays, severely reducing the sensor network performance. In this case, distributed protocols are required so that the reconstruction can be performed in-network. In both centralized and distributed scenarios, a common element is the necessity of limiting computations and memory usage, making compressed sensing very appealing as a cost-constrained representation in order to exploit data redundancies. This book presents a survey of the state of the art of *Compressed Sensing for Distributed Systems*. It has to be noted that, while compressed sensing has been studied for some time now, its distributed applications are relatively new. Remarkably, such applications are ideally suited to exploit all the benefits that compressed sensing can provide. The objective of this book is to provide the reader with a comprehensive survey of this topic, from the basic concepts to different classes of centralized and distributed reconstruction algorithms, as well as a comparison of these techniques. This book collects different contributions on these aspects. It presents the underlying theory in a complete and unified way for the first time, presenting various signal models and their use cases. It contains a theoretical part collecting latest results in rate-distortion analysis of

distributed compressed sensing, as well as practical implementations of algorithms obtaining performance close to the theoretical bounds. It presents and discusses various distributed reconstruction algorithms, summarizing the theoretical reconstruction guarantees and providing a comparative analysis of their performance and complexity. In summary, this book will allow the reader to get started in the field of distributed compressed sensing from theory to practice. We believe that it can find a broad audience among researchers, scientists, or engineers with very diverse backgrounds, having interests in mathematical optimization, network systems, graph-theoretical methods, linear systems, stochastic systems, and randomized algorithms. To help the reader become familiar with the theories and algorithms presented, accompanying software is made available on the authors website (www.crisp-erc.eu), implementing several of the algorithms described in the book.

Turin
March 2015

Giulio Coluccia
Chiara Ravazzi
Enrico Magli

Acknowledgments

Part of the material published in this book is based on works coauthored by Sophie Marie Fosson, Aline Roumy and Diego Valsesia, whom the authors express their sincere gratitude to.

This work is supported by the European Research Council under the European Community's Seventh Framework Programme (FP7/2007-2013)/ERC Grant agreement n.279848.

Contents

Chapter 1
Introduction

Distributed applications have gained a lot of interest lately in several areas of engineering, such as signal processing and information theory. Such applications are characterized by a large number of entities that acquire, process, transmit, or manage information, in contrast to the conventional setting involving one information source and one receiver. Central to the concept of distributed systems is the collaboration among these entities, which is known to provide huge potential benefits. Indeed, many application scenarios have shown that distributed collaborative systems can provide performance superior to that of a centralized system; notable examples include peer-to-peer storage and streaming, sensor networks, ad hoc networks, as well as collaborative communication paradigms such as network coding, to mention a few [1].

Oftentimes, nodes belonging to the system tend to have a selfish behavior. This is typically a consequence of the fact that collaborating with other nodes implies a price to be paid. The currency is the most important albeit scarce resource owned by the node, i.e., energy. Particularly in the context of wireless devices, battery power is limited and determines the node's lifetime. Therefore, it is clear that a node will not be willing to give up some of its battery life, unless it has a clear advantage to be gained. In order to see that collaboration incurs an immediate cost upon the node, consider that collaboration may require to synchronize the node's activities, to receive and transfer information from and to a node's neighbors, to serve as a relay by forwarding messages received by other nodes, and in general to be active even when the node would otherwise not necessarily stay awake. All these collaboration activities will require the node to process and transmit information, thereby draining their energy resources from processor usage and access to the communication channel.

In order to save energy, therefore, it is imperative to employ compact and efficient representations of the information sensed by the nodes. A straightforward approach would be to apply lossless or lossy compression techniques to reduce the data size. This will reduce the amount of data to be transmitted on the communication channel, thereby reducing energy consumption. However, the computations needed to compress the information will themselves generatean extra energy consumption,

G. Coluccia et al., *Compressed Sensing for Distributed Systems*,
SpringerBriefs in Signal Processing, DOI 10.1007/978-981-287-390-3_1

and possibly a delay. As is shown in [2], depending on the complexity of the compression algorithm and the available computational power, this extra consumption may outweigh the benefits of reduced access to the channel. Moreover, compressed files may not be typically processes in-network, because in the compressed format the information is coded in a variable-length fashion.

Many of these issues can be successfully addressed using compressed sensing (CS) techniques. CS is a mathematical technique that exploits the inherent redundancy of natural signals in order to represent them in a compact way through universal, nonadaptive, and computationally simple algorithms. In CS [3, 4], a signal is represented by a small set of its linear projections onto some suitable sequences. This compressed representation is significantly more compact than conventional sampling as described by Shannon's theorem, since the number of projections is typically much smaller than the number of signal samples. Surprisingly, under suitable signal models it can be shown that a signal can be reconstructed exactly from its linear measurements employing nonlinear reconstruction algorithms. The most typical signal model calls for the signal to be "sparse," i.e., the signal has only few nonzero coefficients in a suitable domain. The availability of strong guarantees regarding signal recovery from linear measurements has spurred a huge amount of research in the area of CS and its applications, including more sophisticated signal models, low-complexity reconstruction algorithms, application to multisensor systems, analog-to-digital conversion, detection and estimation problems, machine learning, imaging, geophysics, computational biology, radar, astronomy, system identification, communications, audio and speech processing, remote sensing, computer graphics, data retrieval, and many more. Interestingly, it has been shown [5] that the CS signal representation is amenable to perform some signal processing tasks directly on the linear measurements, such as signal detection, estimation, and classification. Moreover, it is known that CS can also be used as an encryption layer on the data [6, 7]. Therefore, CS represents a flexible tool that can provide many functionalities needed in a distributed system, such as data reduction, processing, and security, within a single low-complexity framework.

While much of the initial work on CS has been focused on the problem of one sensing entity and one receiver, more recently the application of CS to distributed systems has attracted a lot of interest. Indeed, distributed CS (DCS) has obvious applications in sensor networks, where a set of nodes acquiring information may leverage the energy-efficient CS representation to reduce power consumption. Applications include sensors for smart cities and environments, sensors for Internet of things, visual sensor networks, video surveillance networks, and in general, any wireless sensor network that acquires time series of data that need to be processed and communicated efficiently. Distributed systems raise specific technical challenges for CS. Generally speaking, in a distributed system where energy is the scarcest resource, data reduction and low-complexity algorithms are key aspects of the performance optimization process. The distributed sensing process of a physical phenomenon entails that the information signals sensed by different nodes are not independent among each other due to the spatial smoothness of the phenomenon. Therefore, besides the sparsity model that describes the degrees of freedom of each sensed

signal, the sensing model must be augmented by taking into account the correlations of the information signals sensed by different nodes. These models must be exploited in the design of signal recovery algorithms that take advantage of these correlations, requiring fewer linear measurements to reconstruct the ensemble of signals exactly or within some prescribed tolerance. This is where DCS can really make a difference, because acquiring and transmitting fewer linear measurements can provide large savings in the communication and computation energy.

Recovery, however, can be performed in different ways. The simplest scenario involves that each node transmits its linear measurements to a fusion center, and the fusion center employs a suitable signal model and cost criterion to recover the signals. This is possible for small-scale networks, but becomes increasingly difficult as the network grows in size. Indeed, for a large-scale network, transmission from a node to the fusion center may require a large number of hops, incurring a significant energy cost and a large delay. In large-scale networks, it may be more convenient that nodes communicate locally with their neighbors, attempting to iteratively finding a consensus on the signal to be estimated. This avoids long-range communications and also makes the network more resilient to failures, since the fusion center does not represent any more a single breakdown point of the network.

The objective of this book is to provide a comprehensive treatment of recent developments in the area of CS for distributed systems, covering all aspects related to the sensing and recovery process of an ensemble of signals. As has been said, this is the most important, but arguably not the only aspect of interest in this area. CS techniques may provide the ability to process the data in a domain of reduced dimensionality, as well as providing some form of encryption. For these aspects, we refer the reader to the available literature, whereas this book is only concerned with the sensing/recovery problem.

In particular, the book is organized as follows. Chapter 2 introduces the basic notation for CS, and reviews different types of recovery algorithms. The purpose of this chapter is not to provide a complete review of the existing techniques, which are countless. Rather, we focus on the techniques that are more amenable to be generalized to distributed systems, and particularly Lasso and iterative thresholding techniques. We review sparsity models for ensembles of signals, and outline design criteria for centralized and distributed recovery algorithms. In Chap. 3 we consider correlated and distributed sources without cooperation at the encoder, and perform their rate-distortion analysis. In particular, for these sources, we derive the best achievable performance in the rate-distortion sense of any distributed compressed sensing scheme, under the constraint of high-rate quantization. Moreover, we derive a closed-form expression of the rate gain achieved by taking into account the correlation of the sources at the receiver and a closed-form expression of the average performance of the oracle receiver for independent and joint reconstruction. We report experimental results validating the theoretical performance analysis. In Chap. 4 we focus on the problem of centralized recovery when a fusion center is available. We consider different joint sparsity models, and review and compare the performance of existing algorithms, also considering a recent class of algorithms where one signal is used as reference to perform differential encoding and reconstruction. In Chap. 5

we consider distributed reconstruction without a fusion center. We review different types of consensus-based estimators and several algorithms for distributed estimation, including the distributed sub-gradient method, the alternating direction method of multipliers, and distributed iterative thresholding. Then we focus on algorithms optimized for energy saving, providing performance, and energy comparisons among these algorithms. Finally, in Chap. 6 we draw some conclusions.

References

1. Magli, E., Wang, M., Frossard, P., Markopoulou, A.: Network coding meets multimedia: a review. IEEE Trans. Multimed. **15**(5), 1195–1212 (2013)
2. Chiasserini, C.F., Magli, E.: Energy-efficient coding and error control for wireless video-surveillance networks. Telecommun. Syst. **26**(2–4), 369–387 (2004)
3. Candès, E.J., Tao, T.: Near-optimal signal recovery from random projections: universal encoding strategies? IEEE Trans. Inf. Theory **52**(12), 5406–5425 (2006)
4. Donoho, D.L.: Compressed sensing. IEEE Trans. Inf. Theory **52**(4), 1289–1306 (2006)
5. Davenport, M.A., Boufounos, P., Wakin, M., Baraniuk, R.: Signal processing with compressive measurements. IEEE J. Sel. Top. Signal Process. **4**(2), 445–460 (2010)
6. Bianchi, T., Bioglio, V., Magli, E.: On the security of random linear measurements. In: 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 3992–3996 (2014)
7. Cambareri, V., Haboba, J., Pareschi, F., Rovatti, H., Setti, G., Wong, K.W.: A two-class information concealing system based on compressed sensing. In: 2013 IEEE International Symposium on Circuits and Systems (ISCAS), pp. 1356–1359 (2013)

Chapter 2
Distributed Compressed Sensing

This chapter first introduces CS in the conventional setting where one device acquires one signal and sends it to a receiver, and then extends it to the distributed framework in which multiple devices acquire multiple signals. In particular, we focus on two key problems related to the distributed setting. The former is the definition of sparsity models for an ensemble of signals, as opposed to just one signal. The second is the structure of the corresponding recovery algorithm, which can be centralized or distributed; each solution entails specific advantages and drawbacks that are preliminarily discussed in this chapter, whereas a detailed description of the corresponding recovery algorithms is given in Chaps. 4 and 5.

2.1 Compressed Sensing for Single Sources

Before starting, we define some notations. We denote column vectors with small letters, and matrices with capital letters. Given a matrix A, $A^\top$ denotes its transpose. We consider $\mathbb{R}^n$ as an Euclidean space endowed with the following norms:

$$\|x\|_p = \left(\sum_{i=1}^{n} |x_i|^p \right)^{1/p}$$

with $p = 1, 2$. Given $x \in \mathbb{R}^n$, we denote the ℓ_0 pseudo-norm as

$$\|x\|_0 = \sum_{i=1}^{n} |x_i|^0,$$

where we use the convention $0^0 = 0$. For a rectangular matrix $M \in \mathbb{R}^{m \times n}$, we consider the Frobenius norm, which is defined as follows:

G. Coluccia et al., *Compressed Sensing for Distributed Systems*,
SpringerBriefs in Signal Processing, DOI 10.1007/978-981-287-390-3_2

$$\|M\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} M_{ij}^2}$$

and the operator norm

$$\|M\|_2 = \sup_{z \neq 0} \frac{\|Mz\|_2}{\|z\|_2}.$$

We denote the sign function as

$$\operatorname{sgn}(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{if } x = 0 \\ -1 & \text{otherwise.} \end{cases}$$

If x is a vector in $\mathbb{R}^n$, $\operatorname{sgn}(x)$ is intended as a function to be applied elementwise.

2.1.1 Sensing Model

The basic CS problem can be stated as follows. An unknown signal represented by a column vector $x \in \mathbb{R}^n$ is sensed by taking a number m of its linear projections, i.e.,

$$y = \Phi x, \tag{2.1}$$

where $\Phi \in \mathbb{R}^{m \times n}$ is a given sensing matrix, and $y \in \mathbb{R}^m$, with $m < n$, is the measurements vector. According to this process, x is repeatedly sensed by taking its scalar product with every row of Φ, yielding a measurements vector y. Since $m < n$, the signal representation provided by y is more "compact" than the original representation x, hence the term *compressed* sensing.

The CS reconstruction problem can be stated as follows: given y and Φ, one wishes to reconstruct the original signal x. With $m < n$, this is clearly an underdetermined problem that may have infinitely many valid solutions satisfying $y = \Phi x$. As is the case of most regularization problems, in order to recover x, it is therefore necessary to add some prior knowledge about the signal in order to constrain the solution set of (2.1). In CS, this is done using the concept of sparsity. In plain terms, a signal is said to be sparse if it has a low number of nonzero entries with respect to its length. In particular, x is said to be k-sparse if it has at most k nonzero entries, or equivalently $\|x\|_0 \leq k$. The set of all k-sparse signals in $\mathbb{R}^n$ is denoted as Σ_k, i.e., $\Sigma_k = \{x \in \mathbb{R}^n : \|x\|_0 \leq k\}$.

2.1.2 Sparse Recovery

Armed with the notion of sparsity, one can attempt to recover x solving the following problem, which aims at finding the sparsest possible signal that satisfies the sensing equation (2.1):

$$\hat{x} = \arg\min_{x} \|x\|_0 \quad \text{such that} \quad y = \Phi x. \tag{2.2}$$

Problem (2.2) has very strong guarantees of success. In [1, Chap. 1] it is shown that, under some mild conditions on the independence of the columns of Φ, if $m \geq 2k$, i.e., the number of linear measurements is at least twice as the sparsity of x, then there exists at most one signal $x \in \Sigma_k$ such that $y = \Phi x$, and (2.2) will yield the correct solution for any $x \in \Sigma_k$. However, despite its attractiveness, solving (2.2) is not a viable way to recover x, because this problem has combinatorial complexity. The so-called "oracle" receiver simply assumes to know in advance the set $\mathscr{S} = \text{supp}(x) = \{i \in \{1, \ldots, n\} | x_i \neq 0\}$ identifying the indexes of the nonzero entries of x. Given $\mathscr{S}$, one can construct the reduced matrix $\Phi_{\mathscr{S}}$ which is obtained removing from Φ the columns ϕ_i whose index does not belong to $\mathscr{S}$. Then, the nonzero components of x are readily obtained as $x_{\mathscr{S}} = \Phi_{\mathscr{S}}^{\dagger} y$, where $A^{\dagger}$ denotes the pseudoinverse of A, i.e., $A^{\dagger} = (A^{\top} A)^{-1} A^{\top}$. The oracle receiver is very useful to derive theoretical properties of CS systems. In practice, however, $\mathscr{S}$ is not known, so that in order to solve (2.2) one should consider all possible sets of k out of n index positions, i.e., the sparsity supports of x, and test each of them for correctness. This is an NP-hard problem that is computationally infeasible even for small values of n.

To address this issue, it is possible to solve a slightly different problem where the function to be minimized is convex, namely

$$\hat{x} = \arg\min_{x} \|x\|_1 \quad \text{such that} \quad y = \Phi x. \tag{2.3}$$

Replacing the ℓ_0 with the ℓ_1 norm makes the problem convex, and essentially reduces it to a linear program that can be solved using convex optimization tools, e.g., quadratic programming [2], such as interior-point methods.
The complexity is polynomial and depends on the specific solver employed for (2.3), e.g., $O(n^3)$. The algorithm in (2.3), also known as basis pursuit (BP), has very good performance in terms of success of reconstruction, since the ℓ_1 norm tends to promote a sparse solution. Moreover, it also has some interesting performance guarantees, which are easily described via the Restricted Isometry Property (RIP, see e.g., [3]). In particular, a matrix Φ satisfies the RIP of order k if there exists $\delta \in [0, 1)$ such that the following relation holds for all $x \in \Sigma_k$:

$$(1 - \delta)\|x\|_2^2 \leq \|\Phi x\|_2^2 \leq (1 + \delta)\|x\|_2^2. \tag{2.4}$$

We define the RIP-constant $\delta_k := \inf\{\delta \in [0, 1)$: Φ satisfies the RIP of order $k\}$. Basically, the RIP ensures that the columns of Φ are nearly orthonormal, at least when operating on sparse vectors. Moreover, Φ is an approximately norm-preserving (and hence distance-preserving) mapping for k-sparse signals, all the more so as δ_k approaches zero. It can be shown (see [4]) that, if x is k-sparse and Φ satisfies the RIP of order $2k$ with RIP-constant $\delta_{2k} < \sqrt{2} - 1$, then the solution to (2.3) is the same as the solution to (2.2).

Verifying whether a given matrix Φ satisfies the RIP of order k is also an NP-hard problem. However, it has been shown [5] that some classes of random matrices with $m = O(k \log \frac{n}{k})$, and particularly those with independent and identically distributed (i.i.d.) entries drawn from a sub-Gaussian distribution, satisfy the RIP with very high probability. This is the main reason behind the popularity of random sensing matrices with Gaussian, Bernoulli or Rademacher distributions, so that it is also common to refer to linear measurements as "random projections."

In practice, in most cases the sparsity model describes well the signal in a transformed domain, rather than in its natural domain. The formulation above can be easily modified to accommodate this. In particular, we let $x = \Psi\theta$, with $\Psi \in \mathbb{R}^{n\times n}$. Matrix Ψ represents the linear inverse transform of a representation θ that is indeed sparse. Putting this definition into (2.1) yields $y = \Phi x = \Phi\Psi\theta = A\theta$, where $A = \Phi\Psi$ and θ is a sparse vector. This new problem can be solved in exactly the same way as the original one, solving for θ and considering A as the new "sensing" matrix, and eventually recovering $x = \Psi\theta$.

Another limitation of the sensing model (2.1) lies in the fact that the acquired linear measurements are typically affected by noise, leading to the following more accurate model:

$$y = \Phi x + e, \tag{2.5}$$

where e is some unknown perturbation bounded by $\|e\|_2 \leq \varepsilon$.

Under certain assumptions on the sensing matrix and for a sufficiently low level of the signal sparsity [6], *robust signal recovery* is achieved by solving

$$\hat{x} = \arg\min_x \|x\|_0 \quad \text{s.t.} \quad \|y - \Phi x\|_2 \leq \varepsilon. \tag{2.6}$$

This means that the solution $\hat{x}$ of (2.6) obeys $\|\widehat{x} - x\| \leq \kappa\varepsilon$ where κ is a positive constant. Alternatively, an estimation can be provided by the following estimator

$$\hat{x} = \arg\min_x \|y - \Phi x\|_2^2 \quad \text{s.t.} \quad x \in \Sigma_k\ . \tag{2.7}$$

As in the noise-free scenario, (2.6) and (2.7) are known to be NP-hard problems. However, an attractive alternative is given by considering (2.7) and taking the convex relaxation of the ℓ_0 pseudonorm. This problem is also known as basis pursuit with denoise (BPDN) and consists in selecting the element x with residual norm below the tolerance ε which has minimal ℓ_1-norm:

$$\hat{x} = \arg\min_{x} \|x\|_1 \quad \text{s.t.} \quad \|y - \Phi x\|_2 \leq \varepsilon, \tag{2.8}$$

It can be shown (see [4]) that, if x is k-sparse and Φ satisfies the RIP of order $2k$ with RIP-constant $\delta_{2k} < \sqrt{2} - 1$, then the solution to (2.8) is such that $\|\widehat{x} - x\| \leq c\varepsilon$ where c is a positive constant.

Another take on the same problem is the least-absolute shrinkage and selection operator (Lasso) [7], which recasts the problem (2.8) using an unconstrained formulation:

$$\hat{x} = \arg\min_{x} \frac{1}{2} \left[\|y - \Phi x\|_2^2 + 2\lambda \|x\|_1 \right], \tag{2.9}$$

where the parameter $\lambda > 0$ weights the sparsity term of the cost function. It is well known that for problems in which the number of variables n exceeds the number of observations m the cost function in (2.9) is not strictly convex, and hence it may not have a unique minimum. Sufficient conditions guaranteeing the uniqueness of the solution of (2.9) are derived in [8]. The solution x_{Lasso} provides an approximation of (2.6) with a bounded error, which is controlled by λ (see [9, 10]).

2.1.3 Iterative Thresholding Algorithms

Despite their provably good performance, however, solvers of BPDN and Lasso problems have a rather high computational complexity, which may be excessive for certain applications, especially when the signal length n is large. Therefore, alongside these solvers, the literature describes a large number of algorithms for sparse recovery. The main approaches can be classified as optimization-based methods [11], pursuit strategies [12–15], coding-theoretic tools [16, 17], and Bayesian methods (see [18] and reference therein).

For example, the orthogonal matching pursuit algorithm [13] is a very popular solution; this algorithm attempts to estimate the k nonzero components one by one, starting from the strongest one. At the same time, iterative hard thresholding (IHT) [19, 20], and iterative soft thresholding (IST) algorithms [21] have been proposed. These algorithms have lower computational complexity per iteration and lower storage requirements than interior-point methods, and are amenable to a theoretical performance analysis. We briefly review IHT and IST, which will also be employed in later sections of this book. They approximate the solution to (2.6) and (2.9).

The solution is obtained through an iterative technique that alternatively applies a gradient descent step to minimize $\|y - \Phi x\|_2^2$, followed by a scalar thresholding step that enforces sparse estimations. The procedure is iterated until a stopping criterion is met. Generally, if the algorithm is analytically proved to converge, one can stop it when numerical convergence is achieved, that is, when the relative distance between the estimates of two successive iterates is below a fixed threshold. Alternatively, one can fix a maximum number of iterations.

The thresholding operators can be hard or soft and are defined as follows:

$$\sigma_k[x] = \underset{z \in \Sigma_k}{\operatorname{argmin}} \|x - z\|_2 \tag{2.10}$$

and

$$\eta_\lambda[x] = \begin{cases} \operatorname{sgn}(x)(|x| - \lambda) & \text{if } |x| > \lambda \\ 0 & \text{otherwise.} \end{cases} \tag{2.11}$$

It should be noticed that the hard thresholding operator takes a best-k term approximation for some value of k. This is equivalent to selecting a certain number of components that have the largest magnitude and sets the remaining ones to zero at each iteration. The soft thresholding operator is also called the "proximity operator" of the ℓ_1-norm, and it acts component-wise by taking the components with magnitude above a certain threshold and shrinks remaining ones. IHT and IST are described in Algorithm 1 and in Algorithm 2, respectively.

Algorithm 1 IHT

Input: Sensing matrix Φ, measurement y, sparsity level k
  Set $x^{(0)} = 0$, iterate
  for $t = 1$ to StopIter **do**
    $x^{(t)} \leftarrow \sigma_k[x^{(t-1)} + \Phi^T(y - \Phi x^{(t-1)})]$
  end for

Algorithm 2 IST

Input: Sensing matrix Φ, measurement y, sparsity parameter λ
  Set $x^{(0)} = 0$, iterate
  for $t = 1$ to StopIter **do**
    $x^{(t)} \leftarrow \eta_\lambda[x^{(t-1)} + \Phi^T(y - \Phi x^{(t-1)})]$
  end for

The convergence of these algorithms was proved under the assumption that $\|\Phi\|_2^2 < 1$ in [21] (for ISTA) and [20] (for IHTA). A dissertation about the convergence results can be found in [11].

2.2 Compressed Sensing for Distributed Systems

The problem addressed in the previous section refers to the sensing and reconstruction process of a single signal. In many cases, however, it is of interest to consider a set of signals sensed by independent nodes. The sensing setup can be extended accordingly, and suitable recovery algorithms can be derived.

2.2.1 Distributed Setup

In the following, we refer to a scenario where CS is applied to a distributed system in which several nodes independently sense a signal according to (2.5), as in Fig. 2.1, where each black dot represents a node and dashed lines represent available communication links between pairs of nodes. Let $\mathscr{V}$ be the set of sensor nodes of size $J = |\mathscr{V}|$. The vth node acquires a signal $x_v \in \mathbb{R}^n$ via linear measurements of the form $y_v = \Phi_v x_v + e_v$. According to this model, each node senses a different signal x_v, using an individual sensing matrix Φ_v, the sensing process being affected by an individual noise signal e_v. Nodes are represented as vertexes of a graph $\mathscr{G}$ where the set $\mathscr{E}$ of edges between pairs of nodes identify the communications links, so that $(\mathscr{G}, \mathscr{V}, \mathscr{E})$ defines the network structure.

The distributed setup poses particular challenges for the sensing and reconstruction process.

- If the signals x_v are independent, then the reconstruction problem is essentially equivalent to J individual problems, one at each node. In other terms, each node will have to acquire a number of linear measurements that is sufficiently large to enable reconstruction of y_v from x_v and Φ_v. No collaboration among nodes is needed or useful at all, because a node does not have any information that may help in reconstructing the signal measured by another node.
- The more interesting case is when the signals x_v are correlated among each other. This typically occurs when the nodes sense a physical phenomenon that exhibits a spatially smooth behavior, e.g., a temperature or pressure field. In this scenario, there are two types of sparsity to be exploited. Namely, intranode sparsity describes the degree of dependency among samples of the same signal at different time instants, while internode sparsity describes the dependency among samples acquired by different nodes at the same time instant. Internode sparsity is a specific

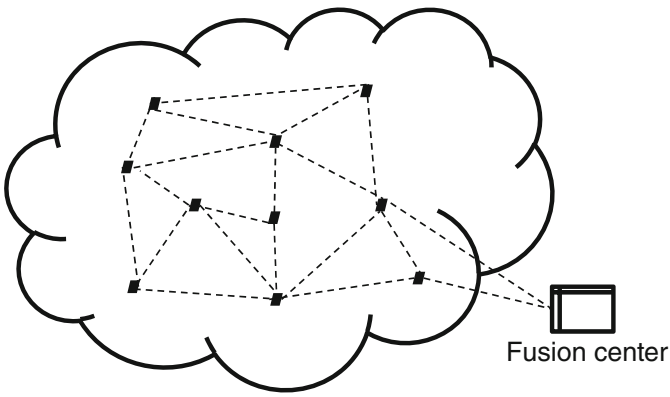

Fig. 2.1 Distributed system setting

aspect of distributed systems, which will be addressed in the remaining part of this chapter and in the next chapters.

- While the nodes individually acquire linear measurements of the signals, two approaches are available for the reconstruction, namely *centralized* and *distributed*. In the centralized approach, the nodes offload the reconstruction process to a fusion center, which receives the linear measurements $\{y_v\}_{v=1}^{J}$ acquired by all nodes and recovers the corresponding set of signals $\{x_v\}_{v=1}^{J}$. Whether centralized reconstruction is possible or even useful depends on many aspects, including the energy cost incurred by transmitting the measurements to the fusion center; this latter also depends on several factors, including distance and the existence and willingness of neighboring nodes to serve as relays. Moreover, the nodes do not get to know the reconstructed signal x_v, which is known only at the fusion center, unless it is transmitted back to each node.
- In the distributed approach, the network does away with a fusion center. This has many advantages, as it avoids the need to transfer all the data to the fusion center, thereby saving a lot of energy. Moreover, the network can function even in the case that the fusion center suddenly becomes unavailable, e.g., because of a hardware failure. In addition, the information does not travel long distances, avoiding the danger of eavesdropping or other security threats. Conversely, distributed reconstruction is based on *local* short-range communications of each node with its neighbors. Short-range communications are very convenient in terms of privacy and energy consumption, and the failure of few nodes will generally not break down the operation of the whole network. On the other hand, local communications decrease the speed at which the information spreads through the network, calling for iterative reconstruction techniques to allow time for each node to contribute to the reconstruction of all other nodes in the network. Since each iteration has an energy cost, the design of a distributed reconstruction algorithm must be done carefully, in order to avoid that too many iterations outweigh the energy benefits of local communications.

2.3 Joint Sparsity Models

This section extends the signal model for one source to the case of distributed signals. Several joint sparsity models are considered, in which all or parts of the signal model are assumed to be sparse. Joint sparsity models entail signals having a twofold source of correlation:

- Intracorrelation, denoting how samples of the same signal are correlated to each other (typically, correlation in time).
- Intercorrelation, denoting how samples of different signals are correlated to each other at the same time instant (correlation in space).

These models represent a good fit for physical signals acquired during time by a sensor network in different points of space. In particular, the original signal sensed by the vth node can be written as

$$x_v = x_{\mathsf{C}} + x_{\mathsf{I},v} \,, \tag{2.12}$$

and node v acquires linear projections of this signal as

$$y_v = \Phi_v x_v = \Phi_v \left(x_{\mathsf{C}} + x_{\mathsf{I},v} \right) \,. \tag{2.13}$$

According to this model, each signal is composed of a common part x_{C}, which is the same for all sensors and is referred to as the *common* component, and an individual part $x_{\mathsf{I},v}$, called *innovation* component, which is specific to each individual sensor. We also assume that x_{C} has k_{C} nonzero entries, and each $x_{\mathsf{I},v}$ has $k_{\mathsf{I},v}$ nonzero entries, so that x_v has at most $k_v = k_{\mathsf{C}} + k_{\mathsf{I},v}$ nonzero entries. If each node had to recover x_v from y_v without receiving help from other nodes, it should acquire a number of linear measurements m_v proportionally larger than k_v, i.e., $m_v \simeq C k_v$, with C sufficiently large depending on the recovery algorithm employed. If each node acted like this, the total number of measurements acquired by the network would be equal to

$$\sum_v C k_v = C \sum_v \left(k_{\mathsf{C}} + k_{\mathsf{I},v} \right) = C \left(J k_{\mathsf{C}} + \sum_v k_{\mathsf{I},v} \right) .$$

What is clear from this analysis is that, while intrasensor sparsity is exploited at each node, the common component is measured J times individually. This is a clear waste of resource that is caused by the lack of exploitation of intersensor sparsity.

The model described in (2.12) can be further detailed according to the structure of x_{C} and $x_{\mathsf{I},v}$. In particular, in [22, 23] the following cases are identified:

- Both the common and innovation components are sparse, namely $k_{\mathsf{C}} \ll n$ and $k_{\mathsf{I},v} \ll n$ for all v. This model is also referred to as JSM-1 in [23]. The common and innovation components need not necessarily be sparse in the same basis. This is a very general model that encompasses many cases of practical interest. For example, a physical phenomenon that is spatially smooth over the coverage area of the sensor network will typically yield a sparse common component that represents the large-scale behavior of the phenomenon, and an innovation component that accounts for the local behavior.
- The common and innovation components have the same sparsity support for all signals (model JSM-2 in [23]). This model is relevant e.g., when a signal described by few frequency components undergoes a frequency-selective attenuation such as the multipath effect, whereby each component is multiplied by a different coefficient, but no new component is created.

- The common component is not sparse, but the innovation components $x_{\mathrm{I},v}$ are sparse (model JSM-3). This is a more general version of JSM-1.

Similar sparsity models were proposed in [24], along with respective reconstruction algorithms.

2.4 Reconstruction for Distributed Systems

It should be noted that, given the number of possible setups, there is no one-size-fits-all reconstruction algorithm that can be used in all cases. In the next chapters we will overview a few algorithms which have been developed for specific scenarios, i.e., centralized (Chap. 4) versus distributed (Chap. 5) reconstruction for a specific joint sparsity model.

The centralized problem entails that all linear measurements $\{y_v\}_{v=1}^{J}$ are available at the fusion center, which attempts to take advantage of the correlation among the signals x_v. Early work on this reconstruction problem goes back to the multiple-measurement vectors problem [25, 26]. Indeed, [25] showed that the availability of multiple measurements improves recovery performance. Later, [26] extended the equivalence between ℓ_0 and ℓ_1 reconstruction to the multiple-measurement vectors case. A few practical algorithms have also been proposed, e.g., M-FOCUSS and M-OMP. Typically, these algorithms are based on a sensing model $\tilde{Y} = \tilde{\Phi}\tilde{X}$, with $\tilde{Y} = [y_1, \ldots, y_v] \in \mathbb{R}^{m \times |\mathcal{V}|}$, $\tilde{X} = [x_1, \ldots, x_v] \in \mathbb{R}^{n \times |\mathcal{V}|}$, and $\tilde{\Phi} \in \mathbb{R}^{m \times n}$. Such algorithms are convenient extensions of the corresponding algorithm in the single-sensor case. However, they have significant limitations. First, the dictionary $\tilde{\Phi}$ must be the same for all vectors. Second, they work well when all signals x_v have the *same sparsity support*, e.g., in the case of model JSM-2. This is because a common sparsity support leads to an unknown vector $\tilde{X}$ that is row-sparse, facilitating the recovery task as well as the derivation of theoretical recovery guarantees. In Chap. 4 we will show more general application scenarios.

The distributed reconstruction problem is more challenging, because at any stage no node has a complete knowledge of the measurements sensed by all other nodes. This information spreads in the network over time, and nodes make greedy decisions based on limited knowledge of the information circulating in the network. Distributed reconstruction algorithms raise the following questions.

- Given the design of a specific distributed recovery algorithm, does the algorithm converge at all?
- If it does converge, does it converge to the global or local minimum of some given cost function?
- Is this functional a sensible one, e.g., the same functional solved by a corresponding centralized reconstruction algorithm?
- Do all nodes individually converge to a sensible solution?

As will be seen in Chap. 5, and as is the case of many single-sensor recovery algorithms, these questions can be answered for some classes of algorithms, but

sometimes only partial responses can be obtained. In particular, Chap. 5 will describe a distributed generalization of thresholding algorithms, for which strong guarantees can be obtained, and some extensions aimed at minimizing communication cost, which are very interesting from the practical standpoint, but less amenable to a complete analytical characterization.

References

1. Eldar, Y.C., Kutyniok, G. (eds.): Compressed sensing: theory and applications. Cambridge University Press, Cambridge (2012)
2. Boyd, S., Vandenberghe, L.: Convex optimization. Cambridge University Press, New York (2004)
3. Candès, E.J., Tao, T.: Decoding by linear programming. IEEE Trans. Inf. Theor. **51**(12), 4203–4215 (2005)
4. Candès, E.J., Romberg, J.K., Tao, T.: Stable signal recovery from incomplete and inaccurate measurements. Commun. Pure Appl. Math. **59**(8), 1207–1223 (2006)
5. Devore, R., Petrova, G., Wojtaszczyk, P.: Instance-optimality in probability with an l1-minimization decoder. Appl. Comput. Harmon. Anal **27**(3), 275–288 (2009)
6. Donoho, D.L., Elad, M., Temlyakov, V.N.: Stable recovery of sparse overcomplete representations in the presence of noise. IEEE Trans. Inf. Theor. **52**(1), 6–18 (2006)
7. Hastie, T., Tibshirani, R., Friedman, J., Hastie, T., Friedman, J., Tibshirani, R.: The elements of statistical learning, vol. 2. Springer, Heidelberg (2009)
8. Tibshirani, R.J.: The Lasso problem and uniqueness. Electron. J. Stat. **7**, 1456–1490 (2013)
9. Candès, E.J.: The restricted isometry property and its implications for compressed sensing. Compte Rendus de l'Academie des Sciences. Paris, France, ser. I, vol. 346, pp. 589–592 (2008)
10. Donoho, D.L.: Compressed sensing. IEEE Trans. Inf. Theor. **52**(4), 1289–1306 (2006)
11. Fornasier, M.: Theoretical foundations and numerical methods for sparse recovery. Radon Series on Computational and Applied Mathematics (2010)
12. Tropp, J.A., Gilbert, A.C., Strauss, M.J.: Algorithms for simultaneous sparse approximation: part i: Greedy pursuit. Signal Process. **86**(3), 572–588 (2006)
13. Tropp, J.A., Gilbert, A.C.: Signal recovery from random measurements via orthogonal matching pursuit. IEEE Trans. Inf. Theor. **53**, 4655–4666 (2007)
14. Needell, D., Vershynin, R.: Signal recovery from incomplete and inaccurate measurements via regularized orthogonal matching pursuit. IEEE J. Sel. Top. Signal Process. **4**(2), 310–316 (2010)
15. Needell, D., Tropp, J.A.: CoSaMP: iterative signal recovery from incomplete and inaccurate samples. Appl. Comput. Harmonic Anal. **26**(3), 301–321 (2008)
16. Baron, D., Sarvotham, S., Baraniuk, R.: Bayesian compressive sensing via belief propagation. IEEE Trans. Signal Process. **58**(1), 269–280 (2010)
17. Jafarpour, S., Xu, W., Hassibi, B., Calderbank, R.: Efficient and robust compressed sensing using optimized expander graphs. IEEE Trans. Inf. Theor. **55**(9), 4299–4308 (2009)
18. Ji, S., Xue, Y., Carin, L.: Bayesian compressive sensing. IEEE Trans. Signal Process. **56**(6), 2346–2356 (2008)
19. Blumensath, T., Davies, M.E.: Iterative thresholding for sparse approximations. J. Fourier Anal. Appl. **14**(5), 629–654 (2004)
20. Blumensath, T., Davies, M.E.: Iterative hard thresholding for compressed sensing. Appl. Comput. Harmonic Anal. **27**(3), 265–274 (2009)
21. Daubechies, I., Defrise, M., De Mol, C.: An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. Comm. Pure Appl. Math. **57**(11), 1413–1457 (2004)

22. Duarte, M.F., Wakin, M.B., Baron, D., Sarvotham, S., Baraniuk, R.G.: Measurement bounds for sparse signal ensembles via graphical models. IEEE Trans. Inf. Theor. **59**(7), 4280–4289 (2013)
23. Baron, D., Duarte, M.F., Wakin, M.B., Sarvotham, S., Baraniuk, R.G.: Distributed compressive sensing. arXiv preprint arXiv:0901.3403 (2009)
24. Hormati, A., Vetterli, M.: Distributed compressed sensing: sparsity models and reconstruction algorithms using annihilating filter. In: IEEE International Conference on Acoustics, Speech and Signal Processing, 2008, ICASSP 2008, pp. 5141–5144 (2008)
25. Cotter, S., Rao, B., Engan, K., Kreutz-Delgado, K.: Sparse solutions to linear inverse problems with multiple measurement vectors. IEEE Trans. Signal Process. **53**(7), 2477–2488 (2005)
26. Chen, J., Huo, X.: Theoretical results on sparse representations of multiple-measurement vectors. IEEE Trans. Signal Process. **54**(12), 4634–4643 (2006)

Chapter 3
Rate-Distortion Theory of Distributed Compressed Sensing

In this chapter, correlated and distributed sources without cooperation at the encoder are considered. For these sources, the best achievable performance in the rate-distortion sense of any distributed compressed sensing scheme is derived, under the constraint of high-rate quantization. Moreover, under this model we derive a closed-form expression of the rate gain achieved by taking into account the correlation of the sources at the receiver and a closed-form expression of the average performance of the oracle receiver for independent and joint reconstruction. Finally, we show experimentally that the exploitation of the correlation between the sources performs close to optimal and that the only penalty is due to the missing knowledge of the sparsity support as in (non-distributed) compressed sensing. Even if the derivation is performed in the large system regime, where signal and system parameters tend to infinity, numerical results show that the equations match simulations for parameter values of practical interest.

3.1 Introduction

Using CS as signal representation requires to cast the representation/coding problem in a rate-distortion (RD) framework, particularly regarding the rate necessary to encode the measurements. For single sources, this problem has been addressed by several authors. In [1], a RD analysis of CS reconstruction from quantized measurements was performed, when the observed signal is sparse. Instead, [2] considered the RD behavior of strictly sparse or compressible memoryless sources in their own domain. Fletcher et al. [3, 4] considered the cost of encoding the random measurements for single sources. More precisely, RD analysis was performed and it was shown that adaptive encoding, taking into account the source distribution, outperforms scalar quantization of random measurements at the cost of higher computational complexity. However, in the distributed context, adaptive encoding may loose

G. Coluccia et al., *Compressed Sensing for Distributed Systems*,
SpringerBriefs in Signal Processing, DOI 10.1007/978-981-287-390-3_3

the intercorrelation between the sources since it is adapted to the distribution of each single source or even the realization of each source.

On the other hand, the distributed case is more sophisticated. Not only one needs to encode a source, but also to design a scheme capable of exploiting the correlation among different sources. Therefore, distributed CS (DCS) was proposed in [5] and further analyzed in [6]. In those papers, an architecture for separate acquisition and joint reconstruction was defined, along with three different joint sparsity models (which where merged into a single formulation in the latter paper). For each model, necessary conditions were posed on the number of measurement to be taken on each source to ensure perfect reconstruction. An analogy between DCS and Slepian-Wolf distributed source coding was depicted, in terms of the necessary conditions about the number of measurements, depending on the sparsity degree of sources, and the necessary conditions on encoding rate, depending on conditional and joint entropy, typical of Slepian-Wolf theory. Moreover, it was shown that a distributed system based on CS could save up to 30 % of measurements with respect to separate CS encoding/decoding of each source. On the other hand, [5] extended CS in the acquisition part, but it was mainly concerned with the performance of perfect reconstruction, and did not consider the representation/coding problem, which is one of the main issues of a practical scheme and a critical aspect of CS.

In [7], a DCS scheme was proposed that takes into account the encoding cost, exploits both inter- and intracorrelations, and has low complexity. The main idea was to exploit the knowledge of side information (SI) not only as a way to reduce the encoding rate, but also in order to improve the reconstruction quality, as is common in the Wyner-Ziv context [8]. The proposed architecture applies the same Gaussian random matrix to information and SI sources, then quantizes and encodes the measurements with a Slepian-Wolf source code.

In this chapter, we study analytically the best achievable RD performance of any single-source and distributed CS scheme, under the constraint of high-rate quantization, providing simulation results that perfectly match the theoretical analysis. In particular, we provide the following contributions. First, we derive the asymptotic (in the rate and in the number of measurements) distribution of the measurement vector. Even if the analysis is asymptotic, we show that the convergence to a Gaussian distribution occurs with parameter values of practical interest. Moreover, we provide an analytical expression of the rate gain obtained exploiting intersource correlation at the decoder. Second, we provide a closed-form expression of the average reconstruction error using the oracle receiver, improving the results existing in literature, consisting only in bounds hardly comparable to the results of numerical simulations [9, 10]. The proof relies on recent results on random matrix theory [11]. Third, we provide a closed-form expression of the rate gain due to joint reconstruction from the measurements of multiple sources. We compare the results obtained by theory both with the ideal oracle receiver and with a practical algorithm [7], showing that the penalty with respect to the ideal receiver is due to the lack of knowledge of the sparsity support in the reconstruction algorithm. Despite this penalty, the theoretically

derived rate gain matches that obtained applying distributed source coding followed by joint reconstruction to a practical reconstruction scheme. With respect to [5, 6], we use information theoretic tools to provide an analytical characterization of the performance of CS and DCS, for a given number of measurements and set of system parameters.

The chapter is organized as follows. Some background information about source coding with side information at the decoder is given in Sect. 3.2. Analytical results are presented in Sects. 3.3 and 3.4. These results are validated via numerical simulations that are presented throughout the chapter.

3.2 Source Coding with Side Information at the Decoder

Source coding with SI at the decoder refers to the problem of compressing a source X when another source Y, correlated to X, is available at the decoder only. It is a special case of distributed source coding, where the two sources have to be compressed without any cooperation at the encoder.

For lossless compression, if X is compressed without knowledge of Y at its conditional entropy, i.e., $R_X > H(X|Y)$, it can be recovered with vanishing error rate exploiting Y as SI. This represents the asymmetric setup, depicted in Fig. 3.1, where source Y is compressed in a lossless way ($R_Y > H(Y)$) or otherwise known at the decoder. Therefore, the lack of SI at the encoder does not incur any compression loss with respect to joint encoding, as the total rate required by DSC is equal to $H(Y) + H(X|Y) = H(X, Y)$. The result holds for i.i.d. finite sources X and Y [12] but also for ergodic discrete sources [13], or when X is i.i.d. finite, Y is i.i.d. continuous and is available at the decoder [14, Proposition 19].

For lossy compression of i.i.d sources, [15] shows that the lack of SI at the encoder incurs a loss except for some distributions (Gaussian sources, or more generally Gaussian correlation noise). Interestingly, [14] shows that uniform scalar quantization followed by lossless compression incurs a suboptimality of 1.53 dB, in the high-rate regime. Therefore, practical solutions (see for example [16]) compress and decompress the data relying on an *inner lossless distributed codec*, usually referred to as Slepian-Wolf Code (SWC), and an *outer quantization-plus-reconstruction filter*.

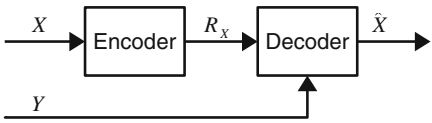

Fig. 3.1 The asymmetric distributed source coding setup

3.3 Rate-Distortion Functions of Single-Source Compressed Sensing

In this section, we derive the best achievable performance in the RD sense over all CS schemes, under the constraint of high-rate quantization. The distribution of the measurements derived in Theorem 3.1 allows to write a closed-form expression of the RD functions of the measurement vectors. In Theorem 3.2, we derive a closed-form expression of the average reconstruction error of the oracle receiver, which will use the results from Theorem 3.1 to present the RD functions of the reconstruction.

3.3.1 Single-Source System Model

Definition 3.1 (*Sparse vector*) The vector $x \in \mathbb{R}^n$ is said to be $(k, n, \sigma_\theta^2, \Psi)$-*sparse* if x is sparse in the domain defined by the orthogonal matrix $\Psi \in \mathbb{R}^{n\times n}$, namely: $x = \Psi\theta$, with $\|\theta\|_0 = k$, and if the nonzero components of θ are modeled as i.i.d. centered random variables with variance $\sigma_\theta^2 < \infty$. Ψ is independent of θ.

The sparse x vector is observed through a smaller vector of Gaussian measurements defined as

Definition 3.2 (*Gaussian measurement*) The vector y is called the $(m, n, \sigma_\Phi^2, \Phi)$-*Gaussian measurement* of $x \in \mathbb{R}^n$, if $y = \frac{1}{\sqrt{m}}\Phi x$, where the sensing matrix $\Phi \in \mathbb{R}^{m\times n}$, with $m < n$, is a random matrix[1] with i.i.d. entries drawn from a Gaussian $\mathsf{N}(0, \sigma_\Phi^2)$ with $\sigma_\Phi^2 < \infty$.

We denote as y_q the quantized version of y. To analyze the RD tradeoff, we consider the large system regime defined below.

Definition 3.3 (*Large system regime, overmeasuring and sparsity rates*) Let x be $(k, n, \sigma_\theta^2, \Psi)$-sparse. Let y be the $(m, n, \sigma_\Phi^2, \Phi)$-Gaussian measurement of x. The system is said to be in the *large system regime* if n goes to infinity, k and m are functions of n and tend to infinity as n does, under the constraint that the rates k/n and m/k converge to constants called *sparsity rate* (γ) and *overmeasuring rate* (μ) i.e.:

$$\lim_{n\to+\infty} \frac{k}{n} = \gamma, \quad \lim_{n\to+\infty} \frac{m}{k} = \mu > 1 \tag{3.1}$$

The sparsity rate is a property of the signal. Instead, the overmeasuring rate is the ratio of the number of measurements to the number of nonzero components, and is therefore a property of the system [1].

[1] This definition complies with the usual form $y = \Phi x$ where the variance σ_Φ^2 of the elements of Φ depends on m. Here, we wanted to keep σ_Φ^2 independent of system parameters.

3.3.2 Rate-Distortion Functions of Measurement Vector

The *information RD function* of an i.i.d. source X defines the minimum amount of information per source symbol R needed to describe the source under the distortion constraint D. For an i.i.d. Gaussian source $X \sim \mathsf{N}(0, \sigma_x^2)$, choosing as distortion metric the squared error between the source and its representation on R bits per symbol, the distortion satisfies

$$D_x(R) = \sigma_x^2 2^{-2R}. \tag{3.2}$$

Interestingly, the *operational RD function* of the same Gaussian source, with uniform scalar quantizer and entropy coding satisifies, in the high-rate regime:

$$\lim_{R\to+\infty} \frac{1}{\sigma_x^2} 2^{2R} D_x^{\mathsf{EC}}(R) = \frac{\pi \mathrm{e}}{6} \tag{3.3}$$

where EC stands for *entropy-constrained scalar quantizer*. This leads to a 1.53 dB gap between the information and the operational RD curves. The relation (3.3) can be easily extended to other types of quantization adapting the factor $\frac{\pi \mathrm{e}}{6}$ to the specific quantization scheme.

Theorem 3.1 (CS: Asymptotic distribution of Gaussian measurements and measurement RD function) *Let x be $(k, n, \sigma_\theta^2, \Psi)$-sparse. Let y be the $(m, n, \sigma_\Phi^2, \Phi)$-Gaussian measurement of x, s.t. $k < m < n$. Consider the large system regime with finite sparsity rate $\gamma = \lim_{n\to\infty} \frac{k}{n}$ and finite over-measuring rate $\mu = \lim_{n\to\infty} \frac{m}{k} > 1$. The Gaussian measurement converges in distribution to an i.i.d. Gaussian, centered random sequence with variance*

$$\sigma_y^2 = \frac{1}{\mu} \sigma_\Phi^2 \sigma_\theta^2. \tag{3.4}$$

Therefore, the information RD function satisfies

$$\lim_{n\to+\infty} \frac{1}{\sigma_y^2} 2^{2R} D_y(R) = 1, \tag{3.5}$$

where R is the encoding rate per measurement sample, and the entropy-constrained scalar quantizer achieves a distortion D_y^{EC} that satisfies

$$\lim_{R\to+\infty} \lim_{n\to+\infty} \frac{1}{\sigma_y^2} 2^{2R} D_y^{\mathsf{EC}}(R) = \frac{\pi \mathrm{e}}{6}. \tag{3.6}$$

Sketch of proof. The distribution of the Gaussian matrix Φ is invariant under orthogonal transformation. Thus, we obtain $y = \frac{1}{\sqrt{m}}\Phi\Psi\theta = \frac{1}{\sqrt{m}}U\theta$, where U is an i.i.d. Gaussian matrix with variance σ_Φ^2. Then, we consider a finite length subvector $y^{(m)}$ of y. From the multidimensional Central Limit theorem (CLT) [17, Theorem 7.18], $y^{(m)}$ converges to a Gaussian centered vector with independent components. Then, as $m \to \infty$, the sequence of Gaussian measurements converges to an i.i.d. Gaussian sequence. See [18, Appendix A] for the complete proof. □

Theorem 3.1 generalizes [1, Theorem 2], which derives the marginal distribution of the measurements, when the observed signal is directly sparse. Instead, Theorem 3.1 derives the *joint* distribution of the measurements and considers *transformed* sparse signals.

It has to be stressed that, even if the RD curves for measurement vectors do not have any "practical" direct use, they are required to derive the RD curves for the reconstruction of the sources, which can be found later in this section.

The validity of the RD functions derived for the measurements is shown in Fig. 3.2. The figure plots the quantization distortion of y, i.e., $\mathbb{E}\left[\frac{1}{m}\|y - y_q\|_2^2\right]$ versus the rate R, measured in bits per measurement sample (bpms). The distortion has been averaged over 10^4 trials, and for each trial different realizations of the sources, the sensing matrix and the noise have been drawn. Figure 3.2 shows two subfigures corresponding to different sets of signal and system parameters (signal length n, sparsity of the common component k_C and of the innovation component $k_{\mathsf{I},j}$, variance of the nonzero components $\sigma^2_{\theta_\mathsf{C}}$ and $\sigma^2_{\theta_{\mathsf{I},j}}$, respectively, and length of the measurement vector m). For each test, Ψ is the DCT matrix, each nonzero component of θ is drawn from a normal distribution, and $\sigma_\Phi^2 = 1$.

- The curve labeled as (HR)—standing for *high rate*—is the asymptote of the operational RD curves (3.19).
- The curve labeled as (Gauss.) corresponds to the distortion of a synthetic zero-mean Gaussian source with variance σ_y^2 as in (3.16) and quantized with a uniform scalar quantizer.
- The curve labeled as (sim.) is the simulated RD for a measurement vector obtained generating x according to Definition 3.1, measuring it with the sensing matrix Φ to obtain y and quantizing it with a uniform scalar quantizer.

For the (Gauss.) and (sim.) curves, the rate is computed as the *symbol* entropy of the samples y_q, quantized with a uniform scalar quantizer. Entropy has been evaluated computing the number of symbol occurrences over vectors of length 10^8.

First, we notice that the (HR) equation perfectly matches the simulated curves when $R > 2$, showing that the high-rate regime occurs for relative small values of R. Then, it can be noticed that (Gauss.) curves perfectly overlap the (sim.) ones, showing the validity of equation (3.16) and showing that the convergence to the Gaussian case occurs for low values of n, m, k_C, $k_{\mathsf{I},j}$, as was shown also in [19].

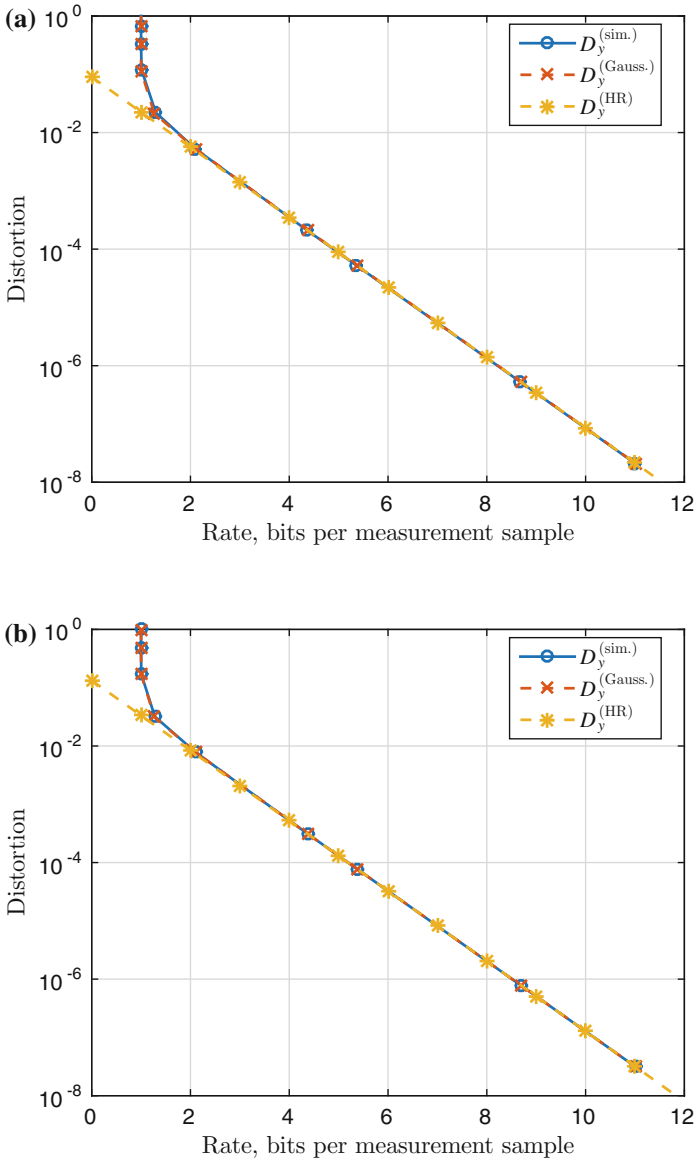

Fig. 3.2 Simulated versus theoretical rate-distortion functions of *measurement vectors* for two different sets of signal and system parameters. Single-source case. **a** $n = 512, k_{\mathsf{C}} = k_{\mathsf{I},j} = 8$, $\sigma^2_{\theta_{\mathsf{C}}} = 1, \sigma^2_{\theta_{\mathsf{I},j}} = 10^{-2}, m = 128$, **b** $n = 1024, k_{\mathsf{C}} = 16, k_{\mathsf{I},j} = 8, \sigma^2_{\theta_{\mathsf{C}}} = \sigma^2_{\theta_{\mathsf{I},j}} = 1, m = 256$

3.3.3 Rate-Distortion Functions of the Reconstruction

We now evaluate the performance of CS reconstruction with quantized measurements. The performance depends on the amount of noise affecting the measurements. In particular, the distortion $\|\widehat{x} - x\|_2^2$ is upper bounded by the noise variance up to a scaling factor [20, 21], i.e.,

$$\|\widehat{x} - x\|_2^2 \leq c^2 \varepsilon^2 \,, \tag{3.7}$$

where the constant c depends on the realization of the measurement matrix, since it is a function of the RIP constant. Since we consider the average[2] performance, we need to consider the worst case c and this upper bound will be very loose [22, Theorem 1.9].

Here, we consider the *oracle* estimator (see Sect. 2.1.2), which is the estimator knowing exactly the sparsity support $\mathscr{S} = \{i | \theta_i \neq 0\}$ of the signal x. For the oracle estimator, upper and lower bounds depending on the RIP constant can be found, for example in [9] when the noise affecting the measurements is white and in [10] when the noise is correlated. Unlike [9, 10], here the average performance of the oracle, depending on system parameters only, is derived exactly.

As we will show in the following sections, the characterization of the ideal oracle estimator allows to derive the reconstruction RD functions with results holding also when nonideal estimators are used.

Theorem 3.2 (CS: Reconstruction RD functions) *Let x be $(k, n, \sigma_\theta^2, \Psi)$-sparse. Let y be the $(m, n, \sigma_\Phi^2, \Phi)$-Gaussian measurement of x, s.t. $k + 3 < m < n$. Consider the large system regime with finite sparsity rate $\gamma = \lim_{n\to\infty} \frac{k}{n}$ and finite overmeasuring rate $\mu = \lim_{n\to\infty} \frac{m}{k} > 1$. R denotes the encoding rate per measurement sample. Assume reconstruction by the oracle estimator, when the support $\mathscr{S}$ of x is available at the receiver. The operational RD function of any CS reconstruction algorithm is lower bounded by that of the oracle estimator that satisfies*

$$D_x^{\mathsf{CS}}(R) \geq D_x^{\text{oracle}}(R) = \gamma \frac{\mu}{\mu - 1} \frac{1}{\sigma_\Phi^2} D_y(R) = \frac{\gamma}{\mu - 1} \sigma_\theta^2 2^{-2R}. \tag{3.8}$$

Similarly, the entropy-constrained RD function satisfies in the high-rate regime

$$D_x^{\mathsf{EC-CS}}(R) \geq D_x^{\mathsf{EC\ oracle}}(R) = \frac{\gamma}{\mu - 1} \sigma_\theta^2 \frac{\pi \mathrm{e}}{6} 2^{-2R}. \tag{3.9}$$

[2]The average performance is obtained averaging over all random variables i.e., the measurement matrix, the nonzero components θ and noise, as for example in [10].

Sketch of proof. A novel result about the expected value of a matrix following a generalized inverse Wishart distribution [11, Theorem 2.1] is used. This result can be applied to the distortion of the oracle estimator for finite length signals, depending on the expected value of the pseudo inverse of Wishart matrix [23]. The key consequence is that the distortion of the oracle only depends on the variance of the quantization noise and not on its covariance matrix. Therefore, our result holds even if the noise is correlated (for instance if vector quantization is used). Hence, this result applies to any quantization algorithm. This result improves those in [9, Theorem 4.1] and [10], where upper and lower bounds depending on the RIP constant of the sensing matrix are given, and it also generalizes [1, Section III.C], where a lower bound is derived whereas we derive the exact average performance. See [18, Appendix B] for the complete proof. □

It must be noticed that the condition $m > k + 3$ is not restrictive since in all cases of practical interest, $m > 2k$.

Figure 3.3 depicts the RD performance of the *oracle receiver*, in terms of reconstruction error, i.e., $\mathbb{E}\left[\frac{1}{n}||\widehat{x} - x||_2^2\right]$ versus the rate per measurement sample R. The distortion has been averaged over 10^4 trials, and for each trial different realizations of the sources, the sensing matrix and the noise have been drawn. For each test, Ψ is the DCT matrix, each nonzero component of θ is drawn from a normal distribution, and $\sigma_\Phi^2 = 1$. The figure shows two subfigures corresponding to different sets of signal and system parameters $(n, k_{\mathsf{C}}, k_{\mathsf{I},j}, \sigma_{\theta_{\mathsf{C}}}^2, \sigma_{\theta_{\mathsf{I},j}}^2, m)$. Each subfigure compares the RHS of Eq. (3.29) with the oracle reconstruction distortion from y_q versus the *symbol* entropy of the samples y_q, obtaining a match for $R > 2$. For the (sim.) curves, the rate is computed as the *symbol* entropy of the samples y_q, quantized with a uniform scalar quantizer. Entropy has been evaluated computing the number of symbol occurrences over vectors of length 10^8.

3.4 Rate-Distortion Functions of Distributed Compressed Sensing

In this section, the best achievable performance in the RD sense over all DCS schemes, under the constraint of high-rate quantization, is derived. Note that [7] (see Fig. 3.4) is one instance of such a scheme. Results about the distribution of the measurements in the distributed case are presented in Theorem 3.3. Hence, Theorem 3.4, will combine the results of Theorem 3.3 and previous section to derive the RD functions of the reconstruction in the distributed case.

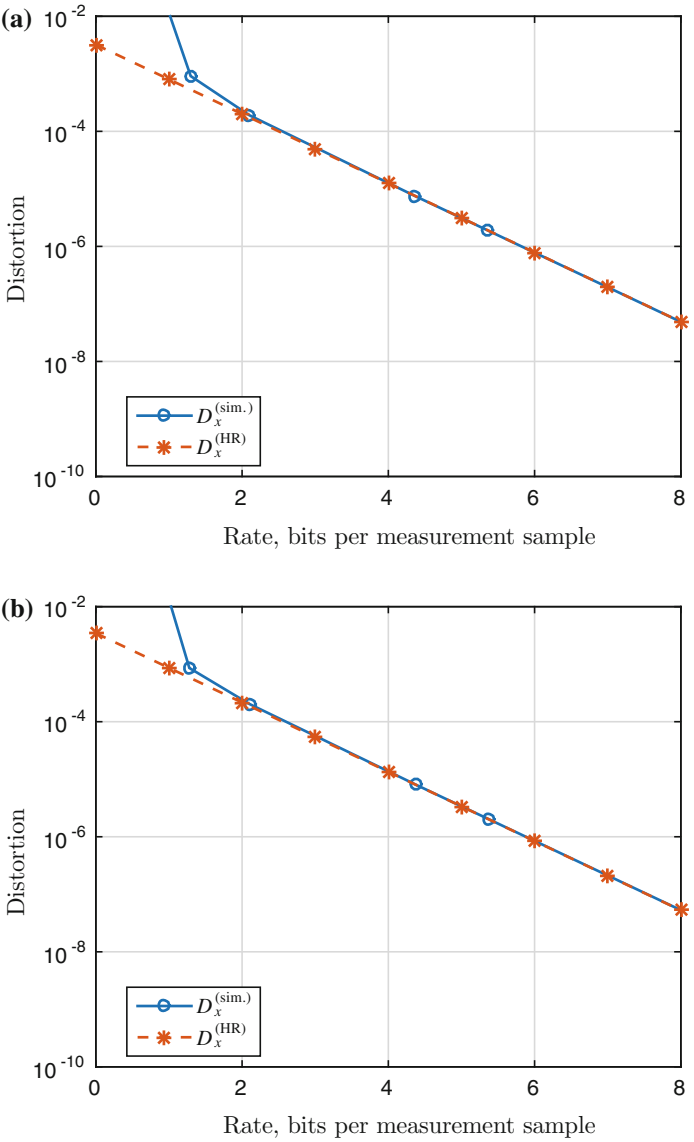

Fig. 3.3 Simulated versus theoretical rate-distortion functions of the *oracle reconstruction* for two different sets of signal and system parameters. Single-source cases. **a** $n = 512, k_{\mathsf{C}} = k_{\mathsf{I},j} = 8$, $\sigma^2_{\theta_{\mathsf{C}}} = 1, \sigma^2_{\theta_{\mathsf{I},j}} = 10^{-2}, m = 128$, **b** $n = 1024, k_{\mathsf{C}} = 16, k_{\mathsf{I},j} = 8, \sigma^2_{\theta_{\mathsf{C}}} = \sigma^2_{\theta_{\mathsf{I},j}} = 1, m = 256$

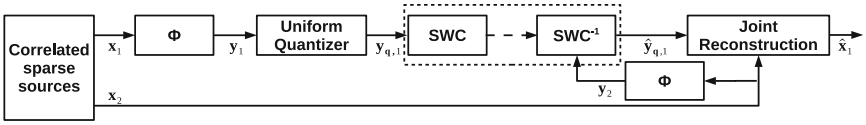

Fig. 3.4 The Distributed compressed sensing scheme in [7]

3.4.1 Distributed System Model

Definition 3.4 (*Correlated Sparse vectors*) J vectors $x_j \in \mathbb{R}^{n\times 1}$, $j \in \{1,\ldots,J\}$ are said to be $(\{k_{\mathsf{I},j}\}_{j=1}^{J}, k_{\mathsf{C}}, \{k_j\}_{j=1}^{J}, n, \sigma^2_{\theta_{\mathsf{C}}}, \{\sigma^2_{\theta_{\mathsf{I},j}}\}_{j=1}^{J}, \Psi)$—*sparse* if

1. Each vector $x_j = x_{\mathsf{C}} + x_{\mathsf{I},j}$ is the sum of a *common* component x_{C} shared by all signals and an *innovation* component $x_{\mathsf{I},j}$, which is unique to each signal x_j.
2. Both x_{C} and $x_{\mathsf{I},j}$ are sparse in the same domain defined by the orthogonal matrix $\Psi \in \mathbb{R}^{n\times n}$, namely: $x_{\mathsf{C}} = \Psi\theta_{\mathsf{C}}$ and $x_{\mathsf{I},j} = \Psi\theta_{\mathsf{I},j}$, with $||\theta_{\mathsf{C}}||_0 = k_{\mathsf{C}}$, $||\theta_{\mathsf{I},j}||_0 = k_{\mathsf{I},j}$ and $k_{\mathsf{C}}, k_{\mathsf{I},j} < n$.
3. The global sparsity of x_j is k_j, with $\max\left\{k_{\mathsf{C}}, k_{\mathsf{I},j}\right\} \le k_j \le k_{\mathsf{C}} + k_{\mathsf{I},j}$.
4. The nonzero components of θ_{C} and $\theta_{\mathsf{I},j}$ are i.i.d. centered random variables with variance $\sigma^2_{\theta_{\mathsf{C}}} < \infty$ and $\sigma^2_{\theta_{\mathsf{I},j}} < \infty$, respectively.

The correlation between the sources is modeled through a *common* component and their difference through an individual *innovation* component. This is a good fit for signals acquired by a group of sensors monitoring the same physical event in different spatial positions, where local factors can affect the innovation component of a more global behavior taken into account by this common component. Note that the Joint Sparsity Model-1 (JSM-1) [5] and the ensemble sparsity model (ESM) in [6], described in Sect. 2.3, are deterministic models. Instead, the sparse model (Definition 3.4) is probabilistic, since we look for the performance averaged over all possible realizations of the sources.

Focusing without loss of generality on the case $J = 2$, we assume that x_1 and x_2 are $\left(k_{\mathsf{I},1}, k_{\mathsf{I},2}, k_{\mathsf{C}}, k_1, k_2, n, \sigma^2_{\theta_{\mathsf{C}}}, \sigma^2_{\theta_{\mathsf{I},1}}, \sigma^2_{\theta_{\mathsf{I},2}}, \Psi\right)$-sparse. x_1 is the source to be compressed whereas x_2 serves as SI. y_1 and y_2 are the $(m, n, \sigma^2_{\Phi}, \Phi)$-Gaussian measurements of x_1 and x_2, and $y_{q,j}$ is the quantized version of y_j.

The large system regime becomes in the distributed case:

Definition 3.5 (*Large system regime, sparsity and overmeasuring rates, and overlaps*) Let the J vectors $x_j \in \mathbb{R}^{n\times 1}$, $\forall j \in \{1,\ldots,J\}$ be $\left(\{k_{\mathsf{I},j}\}_{j=1}^{J}, k_{\mathsf{C}}, \{k_j\}_{j=1}^{J}, n, \sigma^2_{\theta_{\mathsf{C}}}, \{\sigma^2_{\theta_{\mathsf{I},j}}\}_{j=1}^{J}, \Psi\right)$—sparse. For each j, let $y_j \in \mathbb{R}^{m\times 1}$ be the $(m, n, \sigma^2_{\Phi}, \Phi)$-*Gaussian measurement* of x_j. The system is said to be in the *large system regime* if:

1. n goes to infinity.
2. The other dimensions $k_{\mathsf{I},j}, k_{\mathsf{C}}, k_j, m$ are functions of n and tend to infinity as n does.
3. The following rate converges to a constant called *sparsity rate* as n goes to infinity:
$$\frac{k_j}{n} \to \gamma_j \ . \tag{3.10}$$
4. The following rate converges to a constant called *overmeasuring rate* as n goes to infinity:
$$\frac{m}{k_j} \to \mu_j > 1. \tag{3.11}$$
5. All following rates converge to constants called *overlaps* of the common and innovation components as n goes to infinity:
$$\frac{k_{\mathsf{C}}}{k_j} \to \omega_{\mathsf{C},j}, \ \frac{k_{\mathsf{I},j}}{k_j} \to \omega_{\mathsf{I},j} \ . \tag{3.12}$$

Note that $\max\left\{\omega_{C,j}, \omega_{I,j}\right\} \leq 1 \leq \omega_{C,j} + \omega_{I,j} \leq 2$.

3.4.2 Rate-Distortion Functions of Measurement Vector

The information RD function can also be derived for a pair $(X, Y) \sim \mathsf{N}(0, K_{xy})$ of i.i.d. jointly Gaussian distributed random variables with covariance matrix

$$K_{xy} = \begin{pmatrix} \sigma_x^2 & \rho_{xy}\sigma_x\sigma_y \\ \rho_{xy}\sigma_x\sigma_y & \sigma_y^2 \end{pmatrix}. \tag{3.13}$$

Interestingly, when the SI is available at both encoder and decoder or at the decoder only, the information RD function is the same:

$$D_{x|y}(R) = \sigma_x^2(1-\rho_{xy}^2)2^{-2R} = D_x(R+R^*), \tag{3.14}$$

where $R^* = \frac{1}{2}\log_2 \frac{1}{1-\rho_{xy}^2} \geq 0$ is the *rate gain*, measuring the amount of rate one saves by using the side information Y to decode X. This result holds for optimal vector quantizer [15] but also for scalar uniform quantizers [14, Theorem 8 and Corollary 9] by replacing D_x in (3.14) by the entropy-constrained distortion function $D_x^{\mathsf{EC}}(R)$, defined in (3.3).

To derive the RD curves for the reconstruction of the sources, we first generalize Theorem 3.1 and derive the asymptotic distribution of pairs of measurements.

Theorem 3.3 (Distributed CS: Asymptotic distribution of the pair of Gaussian measurements and measurement RD functions) *Let x_1 and x_2 be $\Big(k_{\mathsf{I},1}, k_{\mathsf{I},2}, k_{\mathsf{C}}, k_1, k_2, n, \sigma^2_{\theta_{\mathsf{C}}}, \sigma^2_{\theta_{\mathsf{I},1}}, \sigma^2_{\theta_{\mathsf{I},2}}, \Psi\Big)$-sparse. x_2 serves as SI for x_1 and is available at the decoder, only. Let y_1 and y_2 be the $(m, n, \sigma^2_{\Phi}, \Phi)$-Gaussian measurements of x_1 and x_2. Let (Y_1, Y_2) be the pair of random processes associated to the random vectors (y_1, y_2). In the large system regime, (Y_1, Y_2) converges to an i.i.d. Gaussian sequence with covariance matrix*

$$K_{12} = \begin{pmatrix} \sigma^2_{y_1} & \rho_{12}\sigma_{y_1}\sigma_{y_2} \\ \rho_{12}\sigma_{y_1}\sigma_{y_2} & \sigma^2_{y_2} \end{pmatrix}, \tag{3.15}$$

$$\sigma^2_{y_j} = \frac{\sigma^2_{\Phi}}{\mu_j}\left[\omega_{\mathsf{C},j}\sigma^2_{\theta_{\mathsf{C}}} + \omega_{\mathsf{I},j}\sigma^2_{\theta_{\mathsf{I},j}}\right] \tag{3.16}$$

$$\rho_{12} = \left[\left(1 + \frac{\omega_{\mathsf{I},1}}{\omega_{\mathsf{C},1}}\frac{\sigma^2_{\theta_{\mathsf{I},1}}}{\sigma^2_{\theta_{\mathsf{C}}}}\right)\left(1 + \frac{\omega_{\mathsf{I},2}}{\omega_{\mathsf{C},2}}\frac{\sigma^2_{\theta_{\mathsf{I},2}}}{\sigma^2_{\theta_{\mathsf{C}}}}\right)\right]^{-\frac{1}{2}}. \tag{3.17}$$

Let R be the encoding rate per measurement sample. When the SI is not used, the information RD function satisifies

$$\lim_{N\to+\infty} \frac{1}{\sigma^2_{y_1}} 2^{2R} D_{y_1}(R) = 1, \tag{3.18}$$

and the entropy-constrained scalar quantizer achieves a distortion $D^{\mathsf{EC}}_{y_1}$ that satisfies

$$\lim_{R\to+\infty} \lim_{N\to+\infty} \frac{1}{\sigma^2_{y_1}} 2^{2R} D^{\mathsf{EC}}_{y_1}(R) = \frac{\pi \mathrm{e}}{6}. \tag{3.19}$$

When the measurement y_2 of the SI is used at the decoder, the information RD function satisifies

$$\lim_{N\to+\infty} \frac{1}{\sigma^2_{y_1}} 2^{2(R+R^*)} D_{y_1|y_2}(R) = 1, \tag{3.20}$$

while the entropy-constrained scalar quantizer achieves a distortion $D^{\mathsf{EC}}_{y_1|y_2}$ that satisfies

$$\lim_{R\to+\infty} \lim_{N\to+\infty} \frac{1}{\sigma^2_{y_1}} 2^{2(R+R^*)} D^{\mathsf{EC}}_{y_1|y_2}(R) = \frac{\pi \mathrm{e}}{6}, \tag{3.21}$$

where

$$R^* = \frac{1}{2}\log_2 \frac{1}{1-\rho_{12}^2}\,. \tag{3.22}$$

Therefore, in the large system regime (and in the high-rate regime for entropy-constrained scalar quantizer), the measurement y_2 of the SI helps reducing the rate by R^* (3.22) bits per measurement sample:

$$D_{y_1|y_2}(R) = D_{y_1}(R + R^*). \tag{3.23}$$

Sketch of proof. We consider a vector of finite length $2m$, which contains the first m components of y_1 followed by the first m components of y_2. The vector can be seen as a sum of three components, where each component converges to a Gaussian vector from the multidimensional CLT [17, Theorem 7.18]. Finally, we obtain that (Y_1, Y_2) converges to an i.i.d. Gaussian process. Therefore, classical RD results for i.i.d. Gaussian sources apply. See [18, Appendix C] for the complete proof. □

Theorem 3.3 first states that the measurements of two sparse vectors converge to an i.i.d. Gaussian process in the large system regime. Then, lossy compression of the measurements is considered and the information and entropy-constrained rate-distortion functions are derived. It is shown that if one measurement vector is used as side information at the decoder, some rate can be saved, depending on the sparse source characteristics, only (see (3.22) and (3.17)).

The validity of the RD functions derived for the measurements is shown in Fig. 3.5. The figure plots the quantization distortion of y_1, i.e., $\mathbb{E}\left[\frac{1}{m}\|y_1 - y_{q,1}\|_2^2\right]$ when y_2 is or is not used as Side Information, versus the rate R, measured in bits per measurement sample (bpms). The distortion has been averaged over 10^4 trials, and for each trial different realizations of the sources, the sensing matrix and the noise have been drawn. Figure 3.5 shows two subfigures corresponding to different sets of signal and system parameters (signal length n, sparsity of the common component k_{C} and of the innovation component $k_{\mathsf{I},j}$, variance of the nonzero components $\sigma^2_{\theta_{\mathsf{C}}}$ and $\sigma^2_{\theta_{\mathsf{I},j}}$, respectively, and length of the measurement vector m). For each test, Ψ is the DCT matrix, each nonzero component of θ is drawn from a normal distribution, and $\sigma^2_{\Phi} = 1$. Each subfigure shows two families of curves, corresponding to the cases in which y_2 is (respectively, is not) used as SI. Each family is composed by 3 curves.

- The curve labeled as (HR)—standing for *high rate*—is the asymptote of the operational RD curves (3.19) (or (3.21)).
- The curve labeled as (Gauss.) corresponds to the distortion of a synthetic correlated Gaussian source pair with covariance matrix as in (3.15), where $\sigma^2_{y_j}$ is defined in (3.16) and $\rho_{y_1y_2}$ in (3.17), and quantized with a uniform scalar quantizer.
- The curves labeled as (sim.) are the simulated RD for a measurement vector pair obtained generating x_1 and x_2 according to Definition 3.4, measuring them with the same Φ to obtain y_1 and y_2 and quantizing y_1 and y_2 with a uniform scalar quantizer.

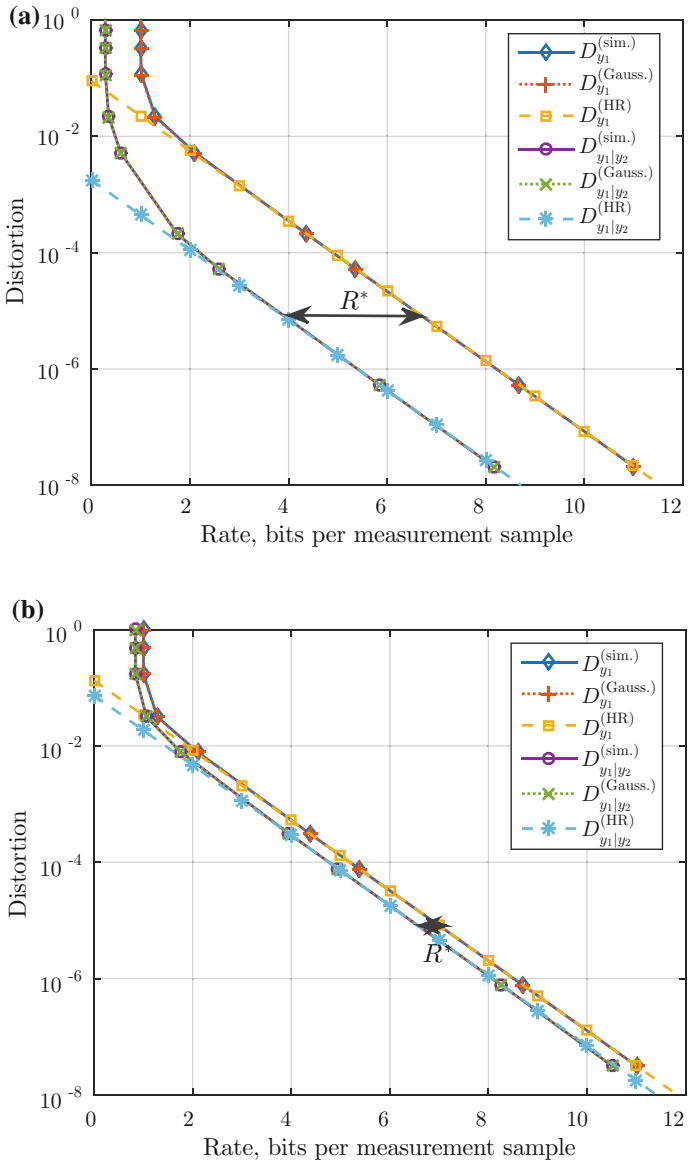

Fig. 3.5 Simulated versus theoretical rate-distortion functions of *measurement vectors* for two different sets of signal and system parameters. Single-source and distributed cases. **a** $n = 512$, $k_{\mathsf{C}} = k_{\mathsf{I},j} = 8, \sigma^2_{\theta_{\mathsf{C}}} = 1, \sigma^2_{\theta_{\mathsf{I},j}} = 10^{-2}, m = 128$, **b** $n = 1024, k_{\mathsf{C}} = 16, k_{\mathsf{I},j} = 8, \sigma^2_{\theta_{\mathsf{C}}} = \sigma^2_{\theta_{\mathsf{I},j}} = 1$, $m = 256$

For the (Gauss.) and (sim.) curves, the rate is computed as the *symbol* entropy of the samples $y_{q,1}$ (or the conditional *symbol* entropy of $y_{q,1}$ given y_2 in the distributed case), quantized with a uniform scalar quantizer. Entropy and conditional entropy have been evaluated computing the number of symbol occurrences over vectors of length 10^8.

First, we notice that the (HR) equation perfectly matches the simulated curves when $R > 2$, showing that the high-rate regime occurs for relative small values of R. Then, it can be noticed that (Gauss.) curves perfectly overlap the (sim.) ones, showing the validity of equation (3.17) and showing that the convergence to the Gaussian case occurs for low values of $n, m, k_{\mathsf{C}}, k_{\mathsf{l},j}$, as was shown also in [19]. It can be also shown that, being the sources of Fig. 3.5b much less correlated than the ones of Fig. 3.5a also the rate gain R^* is smaller.

3.4.3 Rate-Distortion Functions of the Reconstruction

We now derive the RD functions after reconstruction of the DCS scheme.

Theorem 3.4 (Distributed CS: Reconstruction RD functions) *Let x_1 and x_2 be $\left(k_{\mathsf{l},1}, k_{\mathsf{l},2}, k_{\mathsf{C}}, k_1 k_2 n, \sigma^2_{\theta_{\mathsf{C}}}, \sigma^2_{\theta_{\mathsf{l},1}}, \sigma^2_{\theta_{\mathsf{l},2}}, \Psi\right)$-sparse. x_2 serves as SI for x_1 and is available at the decoder, only. Let y_1 and y_2 be the $(m, n, \sigma^2_\Phi, \Phi)$-Gaussian measurements of x_1 and x_2, s.t. $k_1 + 3 < m <$. Let R be the encoding rate per measurement sample. The distortion[a] of the source x_1 is denoted as $D^{\mathsf{IR}}_{x_1}$ when the SI is not available at the receiver, $D^{\mathsf{IR}}_{x_1|y_2}$ when the measurements of the SI are available at the SWC decoder* (IR *stands for* independent reconstruction)*, and $D^{\mathsf{JR}}_{x_1|x_2}$ when the SI is used not only to reduce the encoding rate but also to improve the reconstruction fidelity* (JR *stands for* joint reconstruction)*. Then, when independent reconstruction is performed, the RD functions for x_1 satisfy, in the large system regime:*

$$D^{\mathsf{IR}}_{x_1}(R) \geq D^{\mathsf{IR}\ \text{oracle}}_{x_1}(R) = \gamma_1 \frac{\mu_1}{\mu_1 - 1} \frac{1}{\sigma^2_\Phi} D_{y_1}(R), \tag{3.24}$$

$$D^{\mathsf{IR}}_{x_1|y_2}(R) \geq D^{\mathsf{IR}\ \text{oracle}}_{x_1|y_2}(R) = \gamma_1 \frac{\mu_1}{\mu_1 - 1} \frac{1}{\sigma^2_\Phi} D_{y_1|y_2}(R), \tag{3.25}$$

Therefore, in the large system regime, the operational RD functions satisfy

$$D_{x_1}^{\mathsf{IR}}(R) \geq D_{x_1}^{\mathsf{IR\ oracle}}(R) = \gamma_1 \frac{\omega_{\mathsf{C},1}\sigma_{\theta_\mathsf{C}}^2 + \omega_{\mathsf{I},1}\sigma_{\theta_{\mathsf{I},1}}^2}{\mu_1 - 1} 2^{-2R}, \tag{3.26}$$

$$D_{x_1|y_2}^{\mathsf{IR}}(R) \geq D_{x_1|y_2}^{\mathsf{IR\ oracle}}(R) = \gamma_1 \frac{\omega_{\mathsf{C},1}\sigma_{\theta_\mathsf{C}}^2 + \omega_{\mathsf{I},1}\sigma_{\theta_{\mathsf{I},1}}^2}{\mu_1 - 1} 2^{-2(R+R^*)}, \tag{3.27}$$

$$D_{x_1|y_2}^{\mathsf{IR}}(R) = D_{x_1}^{\mathsf{IR}}(R + R^*), \tag{3.28}$$

where R^* is defined in (3.22). In the large system regime and in the high-rate regime, the entropy-constrained RD functions satisfy:

$$D_{x_1}^{\mathsf{IR\ EC}}(R) \geq \gamma_1 \frac{\omega_{\mathsf{C},1}\sigma_{\theta_\mathsf{C}}^2 + \omega_{\mathsf{I},1}\sigma_{\theta_{\mathsf{I},1}}^2}{\mu_1 - 1} \frac{\pi \mathrm{e}}{6} 2^{-2R}, \tag{3.29}$$

$$D_{x_1|y_2}^{\mathsf{IR\ EC}}(R) \geq \gamma_1 \frac{\omega_{\mathsf{C},1}\sigma_{\theta_\mathsf{C}}^2 + \omega_{\mathsf{I},1}\sigma_{\theta_{\mathsf{I},1}}^2}{\mu_1 - 1} \frac{\pi \mathrm{e}}{6} 2^{-2(R+R^*)}. \tag{3.30}$$

When joint reconstruction is performed, the RD functions for x_1 satisfy:

$$D_{x_1|x_2}^{\mathsf{JR}}(R) \geq D_{x_1|x_2}^{\mathsf{JR\ oracle}}(R) = \omega_{\mathsf{I},1}\gamma_1 \frac{\mu_1}{\mu_1 - \omega_{\mathsf{I},1}} \frac{1}{\sigma_\Phi^2} D_{y_1|y_2}(R), \tag{3.31}$$

where, in the large system regime,

$$D_{x_1|x_2}^{\mathsf{JR\ oracle}}(R) = \omega_{\mathsf{I},1}\gamma_1 \frac{\omega_{\mathsf{C},1}\sigma_{\theta_\mathsf{C}}^2 + \omega_{\mathsf{I},1}\sigma_{\theta_{\mathsf{I},1}}^2}{\mu_1 - \omega_{\mathsf{I},1}} 2^{-2(R+R^*)}, \tag{3.32}$$

and in the high-rate regime

$$D_{x_1|x_2}^{\mathsf{JR\ EC\ oracle}}(R) = \omega_{\mathsf{I},1}\gamma_1 \frac{\omega_{\mathsf{C},1}\sigma_{\theta_\mathsf{C}}^2 + \omega_{\mathsf{I},1}\sigma_{\theta_{\mathsf{I},1}}^2}{\mu_1 - \omega_{\mathsf{I},1}} \frac{\pi \mathrm{e}}{6} 2^{-2(R+R^*)}. \tag{3.33}$$

Finally,

$$D_{x_1|x_2}^{\mathsf{JR}}(R) \geq D_{x_1}^{\mathsf{IR}}(R + R^* + R^{\mathsf{JR}}), \tag{3.34}$$

$$\text{where } R^{\mathsf{JR}} = \frac{1}{2}\log_2\left[\frac{1}{\omega_{\mathsf{I},1}} \frac{\mu_1 - \omega_{\mathsf{I},1}}{\mu_1 - 1}\right] \tag{3.35}$$

and where R^* has been defined in (3.22). Therefore, when the SI is available at the decoder, it helps reducing the rate by $R^* + R^{\mathsf{JR}}$ bits per measurement sample.

[a] All the RD functions are operational referred to CS reconstruction algorithms, so the CS superscript is omitted not to overload the notation.

Sketch of proof. An oracle is considered in order to derive lower bounds. More precisely, it is assumed that the sparsity support of x_1 is known if independent reconstruction is performed and that also the support of the common component x_{C} is known if joint reconstruction is performed. The exact distortion of the oracles are derived, from which a closed-form expression of the rate gains are given. The complete proof can be found in [18, Appendix D]. □

As one would expect, when there is no innovation component ($\omega_{\mathsf{I},j} \to 0$), the distortion of the oracle is zero and the rate gain R^{JR} is largest (tends to infinity). On the contrary, when there is no common component ($\omega_{\mathsf{I},j} \to 1$), R^{JR} tends to zero. Moreover, even if the SI could be exploited also to enhance the quality of the dequantization of the unknown source, the gain due to Joint Dequantization becomes negligible in the high-rate region [7]. For this reason, Joint Dequantization is neglected in this analysis.

Figure 3.6 depicts the performance of the complete DCS scheme, in terms of reconstruction error, i.e., $\mathbb{E}\left[\frac{1}{n}\|\widehat{x}_1 - x_1\|_2^2\right]$ versus the rate per measurement sample R. The figure shows two subfigures corresponding to different sets of signal and system parameters (n, k_{C}, $k_{\mathsf{I},j}$, $\sigma^2_{\theta_{\mathsf{C}}}$, $\sigma^2_{\theta_{\mathsf{I},j}}$, m). Each subfigure shows three pairwise comparisons, namely

- First, it compares the RHS of equation (3.29) (IR HR—standing for *independent reconstruction high rate*) with the oracle reconstruction distortion from $y_{q,1}$ versus the *symbol* entropy of the samples $y_{q,1}$ (IR sim.—standing for *independent reconstruction simulated*), obtaining a match for $R > 2$.
- Second, it compares the RHS of equation (3.30) (IR HR) with the oracle reconstruction distortion from $y_{q,1}$ versus the conditional *symbol* entropy of $y_{q,1}$ given y_2 (IR sim.), obtaining a match for $R > 2.5$ and validating once more the evaluation of the rate gain R^* due to the SWC.
- Third, it compares the RHS of equation (3.33) (JR HR—standing for *joint reconstruction high rate*) with the ideal (knowing the sparsity support of the common component) oracle Joint Reconstruction distortion from $y_{q,1}$ and y_2 versus the conditional *symbol* entropy of $y_{q,1}$ given y_2 (JR sim.—standing for *joint reconstruction simulated*), obtaining a match for $R > 3$, validating the expression of the Rate Gain due to Joint Reconstruction given in (3.35).

Moreover, comparing Fig. 3.6a with Fig. 3.6b further aspects can be noticed. The sources of Fig. 3.6a, where $\frac{\sigma^2_{\theta_{\mathsf{I},j}}}{\sigma^2_{\theta_{\mathsf{C}}}} = 10^{-2}$ are more correlated than the ones of Fig. 3.6b, where $\frac{\sigma^2_{\theta_{\mathsf{I},j}}}{\sigma^2_{\theta_{\mathsf{C}}}} = 1$. Hence, the rate gain due to DSC, R^*, is bigger. On the other hand, the sources of Fig. 3.6b have a stronger common component, hence the rate gain due to Joint Reconstruction R^{JR} is more significant.

To conclude, this chapter has studied the best achievable performance in the RD sense over all single-source and DCS schemes, under the constraint of high-rate quantization. Closed-form expressions of the RD curves have been derived in the asymptotic regime, and simulations have shown that the asymptote is reached for relatively

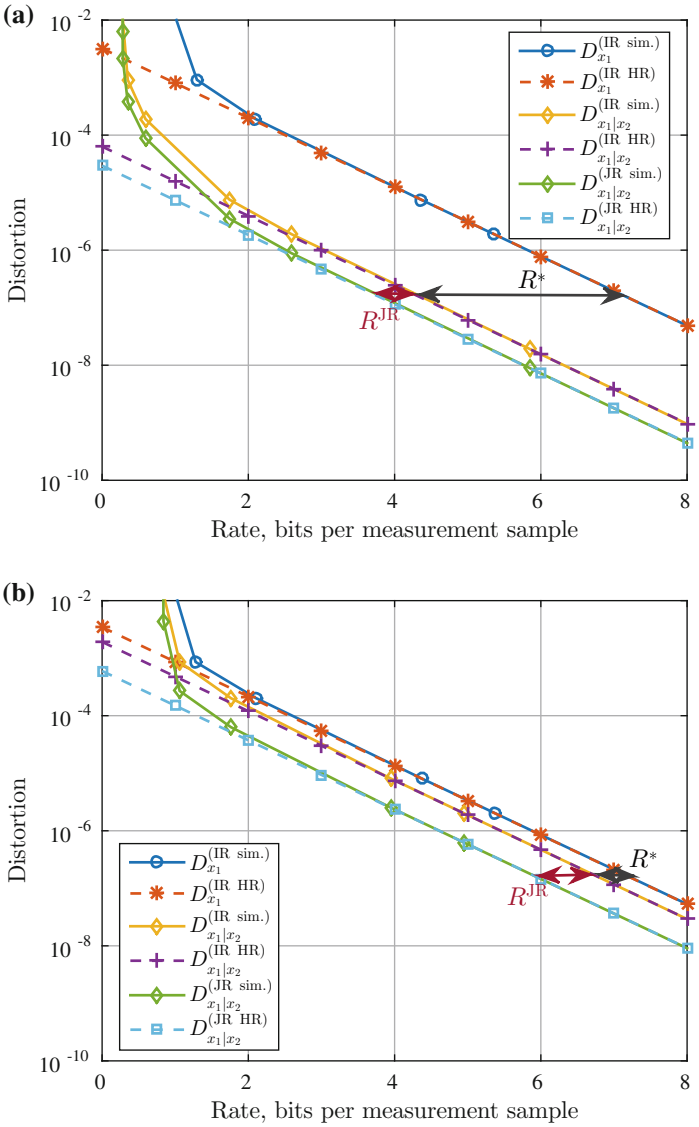

Fig. 3.6 Simulated versus theoretical rate-distortion functions of the *oracle reconstruction* for two different sets of signal and system parameters. Single-source and distributed cases. **a** $n = 512$, $k_{\mathsf{C}} = k_{\mathsf{I},j} = 8$, $\sigma^2_{\theta_{\mathsf{C}}} = 1$, $\sigma^2_{\theta_{\mathsf{I},j}} = 10^{-2}$, $m = 128$, **b** $n = 1024$, $k_{\mathsf{C}} = 16$, $k_{\mathsf{I},j} = 8$, $\sigma^2_{\theta_{\mathsf{C}}} = \sigma^2_{\theta_{\mathsf{I},j}} = 1$, $m = 256$

small number of measurements ($m \simeq 100$) and small rate ($R > 2$ bits/measurement sample). The RD curve computation is based on the convergence of the measurement vector to the multidimensional standard normal distribution. This generalizes [1, Theorem 2] that derives the marginal distribution of the measurement samples when the observed signal is directly sparse. We have then derived a closed-form expression of the highest rate gain achieved exploiting all levels of source correlations at the receiver, along with a closed-form expression of the average performance of the oracle receiver, using novel results on random Wishart matrix theory. Simulations showed that the scheme proposed in [7] almost achieves this best rate gain, and that the only penalty is due to the missing knowledge of the sparsity support as in single-source compressed sensing.

References

1. Dai, W., Milenkovic, O.: Information theoretical and algorithmic approaches to quantized compressive sensing. IEEE Trans. Commun. **59**, 1857–1866 (2011)
2. Weidmann, C., Vetterli, M.: Rate distortion behavior of sparse sources. IEEE Trans. Inf. Theory **58**(8), 4969–4992 (2012)
3. Fletcher, A., Rangan, S., Goyal, V.: On the rate-distortion performance of compressed sensing. In: IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), vol. 3. IEEE (2007)
4. Goyal, V., Fletcher, A., Rangan, S.: Compressive sampling and lossy compression. IEEE Sig. Process. Mag. **25**(2), 48–56 (2008)
5. Baron, D., Duarte, M.F., Wakin, M.B., Sarvotham, S., Baraniuk, R.G.: Distributed compressive sensing. arXiv preprint arXiv:0901.3403 (2009)
6. Duarte, M.F., Wakin, M.B., Baron, D., Sarvotham, S., Baraniuk, R.G.: Measurement bounds for sparse signal ensembles via graphical models. IEEE Trans. Inf. Theory **59**(7), 4280–4289 (2013)
7. Coluccia, G., Magli, E., Roumy, A., Toto-Zarasoa, V., et al.: Lossy compression of distributed sparse sources: a practical scheme. In: 2011 European Signal Processing Conference (EUSIPCO) (2011)
8. Liu, Z., Cheng, S., Liveris, A.D., Xiong, Z.: Slepian-Wolf coded nested lattice quantization for Wyner-Ziv coding: high-rate performance analysis and code design. IEEE Trans. Inf. Theory **52**(10), 4358–4379 (2006)
9. Davenport, M.A., Laska, J.N., Treichler, J.R., Baraniuk, R.G.: The pros and cons of compressive sensing for wideband signal acquisition: noise folding versus dynamic range. CoRR abs/1104.4842 (2011)
10. Laska, J.N., Baraniuk, R.G.: Regime change: bit-depth versus measurement-rate in compressive sensing. IEEE Trans. Sig. Process. **60**(7), 3496–3505 (2012)
11. Cook, R.D., Forzani, L.: On the mean and variance of the generalized inverse of a singular wishart matrix. Electron. J. Stat. **5**, 146–158 (2011)
12. Slepian, D., Wolf, J.: Noiseless coding of correlated information sources. IEEE Trans. Inf. Theory **19**(4), 471–480 (1973)
13. Cover, T.: A proof of the data compression theorem of Slepian and Wolf for ergodic sources (Corresp.). IEEE Trans. Inf. Theory **21**(2), 226–228 (1975)
14. Rebollo-Monedero, D., Rane, S., Aaron, A., Girod, B.: High-rate quantization and transform coding with side information at the decoder. Sig. Process. **86**(11), 3160–3179 (2006). November
15. Wyner, A.: The rate-distortion function for source coding with side information at the decoder-ii: general sources. Inf. Control **38**(1), 60–80 (1978)

16. Bassi, F., Kieffer, M., Weidmann, C.: Wyner-Ziv coding with uncertain side information quality. In: European Signal Processing Conference (EUSIPCO) (2010)
17. Khoshnevisan, D.: Probability. Graduate Studies in Mathematics, AMS, New York (2007)
18. Coluccia, G., Roumy, A., Magli, E.: Operational rate-distortion performance of single-source and distributed compressed sensing. IEEE Trans. Commun. **62**(6), 2022–2033 (2014). June
19. Coluccia, G., Roumy, A., Magli, E.: Exact performance analysis of the oracle receiver for compressed sensing reconstruction. In: 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE (2014), pp. 1005–1009
20. Candès, E.J., Romberg, J.K., Tao, T.: Stable signal recovery from incomplete and inaccurate measurements. Commun. Pure Appl. Math. **59**(8), 1207–1223 (2006)
21. Candès, E.J.: The restricted isometry property and its implications for compressed sensing. Compte Rendus de l'Academie des Sciences. Paris, France, ser. I **346**, 589–592 (2008)
22. Eldar, Y.C., Kutyniok, G., (eds.): Compressed Sensing: Theory and Applications. Cambridge University Press, Cambridge (2012)
23. Díaz-García, J.A., Gutiérrez-Jáimez, R.: Distribution of the generalised inverse of a random matrix and its applications. J. Stat. Plan. Inf. **136**(1), 183–192 (2006)

Chapter 4
Centralized Joint Recovery

This chapter describes algorithms for distributed compressed sensing that take into account the correlation among the sources at various stages of the compressed sensing process. The nodes acquire measurements of a set of signals obeying a specific joint sparsity model, while a centralized fusion center collects the measurements of the entire network and jointly processes them to reconstruct the acquired signals. This kind of mechanism has the advantage that there is no need for communication among the nodes, they only need to transmit their measurements to the fusion center, which will handle the most complex part of the process, i.e., the joint reconstruction.

We first describe the baseline algorithms of the pioneering work about DCS [1]. Then, we analyze a selection of the algorithms appeared in literature in the following years. Finally, we compare the performance of the proposed algorithms in different settings and scenarios.

We focus on the JSM-1 and JSM-3 models, since the JSM-2 can be cast as a MMV problem, as described in Sect. 2.3. Anyway, references to algorithms tailored for the JSM-2 model can be found in [1, Sect. 5.2.2] and [2]. Instead, as an example of a recovery strategy for a signal ensemble not obeying the JSM model, we refer the reader to [3]. Joint sparsity models similar to the previously described ones were also proposed in [4], along with recovery strategies that bear some similarities with the algorithms reviewed in this chapter.

4.1 Baseline Algorithms

4.1.1 Recovery Strategy for JSM-1: γ-Weighted ℓ_1-Norm Minimization

The algorithm [1, Sect. 5.1.3] is able to reconstruct an ensemble of J signals, even if, as it will be shown, in the case of large J complexity issues may arise. In fact, the proposed technique solves the distributed problem as a "big" single-source one, by

G. Coluccia et al., *Compressed Sensing for Distributed Systems*,
SpringerBriefs in Signal Processing, DOI 10.1007/978-981-287-390-3_4

properly combining the signals and measurements belonging to the various sensors as a single quantity. Moreover, it considers signals that are sparse in the sensing domain, i.e., $\Psi = I$. Finally, it takes into account a distributed system where each sensor uses its own sensing matrix, i.e., $\Phi_i \neq \Phi_j, \quad i, j \in [1, \ldots, J]$.

Hence, the following "merged" quantities are defined

$$X = \begin{bmatrix} x_{\mathsf{C}} \\ x_{\mathsf{I},1} \\ x_{\mathsf{I},2} \\ \vdots \\ x_{\mathsf{I},J} \end{bmatrix}, \quad \widehat{X} = \begin{bmatrix} \hat{x}_{\mathsf{C}} \\ \hat{x}_{\mathsf{I},1} \\ \hat{x}_{\mathsf{I},2} \\ \vdots \\ \hat{x}_{\mathsf{I},J} \end{bmatrix}, \quad Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_J \end{bmatrix} \quad \text{and} \quad \widetilde{\Phi} = \begin{bmatrix} \Phi_1 & \Phi_1 & 0 & \cdots & 0 \\ \Phi_2 & 0 & \Phi_2 & \cdots & 0 \\ \vdots & \vdots & & \ddots & \vdots \\ \Phi_J & 0 & \cdots & & \Phi_J \end{bmatrix}. \tag{4.1}$$

Then, the joint estimation of the signals $\widehat{X}$ is obtained as in Algorithm 1.

Algorithm 1 γ-weighted ℓ_1-norm minimization

Input: The measurements $\{y_j\}$, the sensing matrices $\{\Phi_j\}$, the coefficients γ_{C} and $\{\gamma_j\}$, $j = 1, \ldots, J$

1: Build Y and $\widetilde{\Phi}$ as in (4.1)

2: $\widehat{X} \leftarrow \arg \min_{x_{\mathsf{C}}, \{x_{\mathsf{I},j}\}} \gamma_{\mathsf{C}} \|x_{\mathsf{C}}\|_1 + \sum_{j=1}^{J} \gamma_j \|x_{\mathsf{I},j}\|_1 \quad \text{s.t.} \quad Y = \widetilde{\Phi} X$

3: Extract $\hat{x}_{\mathsf{C}}$ from $\widehat{X}$ as in (4.1)

4: **for** $j \leftarrow 1, \ldots, J$ **do**

5: Extract $\hat{x}_{\mathsf{I},j}$ from $\widehat{X}$ as in (4.1)

6: $\hat{x}_j \leftarrow \hat{x}_{\mathsf{C}} + \hat{x}_{\mathsf{I},j}$

7: **end for**

Output: The signal estimation $\{\hat{x}_j\}$

Note that the coefficients γ_{c}, and $\{\gamma_j\}$ are positive scalars which need to be numerically optimized. In [1], it is stated that in the symmetric case where $k_i = k_j$ and $m_i = m_j$, then one can set $\gamma_i = \gamma_j = 1$ and then numerically optimize γ_{C}, only. This is one of the main drawbacks of this solution. The other drawback is that the problem tends to get rapidly unmanageable under the computational complexity point of view. In fact, since the complexity of the ℓ_1 norm minimization is roughly cubic with the signal length n, in the simple case of two signals $J = 2$ the signal to be reconstructed X is $3n$-samples long; hence the computational complexity of the reconstruction is roughly 27 times the one required for the single-source problem. For large values of J, the complexity increases by a factor J^3.

Notice that Algorithm 1 can be easily extended to signals which are sparse in a generic basis $\Psi \neq I$. The only modification required is to replace each block Φ_j of $\widetilde{\Phi}$ with the product $\Phi_j \Psi$.

As for the performance, the results of [1] (restricted for complexity issues to small signals) show that for a signal ensemble with a strong common component, hence, with a strong intracorrelation, the number of required measurements can be reduced by 30 % when compared to separate reconstruction. This gain rapidly drops when the correlation between signals diminishes, i.e., when the innovation components dominate the signals with respect to the common components.

4.1.2 Recovery Strategies for JSM-3

As described in Sect. 2.3, the design of a recovery system for the JSM-3 model is more challenging, since a single signal, being nonsparse by itself, would not be recoverable using CS techniques. But, as stated by [1, Corollary 3], in the distributed setting with joint reconstruction, it is sufficient that the overall number of measurements available is enough to capture enough information about the nonsparse common component to be able to reconstruct the entire ensemble. The principle behind the corollary, which is common to many practical reconstruction schemes, but also to algorithms working in different scenarios—see for example [5]—is the following. If the common component of the ensemble x_{C} was known, then its contribution could be subtracted from the unknown signal's measurements

$$y_j - \Phi_j x_{\mathsf{C}} = \Phi_j x_{\mathsf{I},j}$$

to reconstruct the sparse innovation signals from their compressive measurements. Since in general x_{C} is unknown, it needs to be estimated from the overall measurements available at the joint decoder, using standard tools. For this reason, it is necessary that all the sensors use different sensing matrices, i.e., $\Phi_i \neq \Phi_j, \quad i, j \in [1, \ldots, J]$ in order to be able to estimate the nonsparse n-samples long vector x_{C} from the set of measurements y_j.

4.1.2.1 Transpose Estimation of Common Component (TECC)

The algorithm described below [1, Sect. 5.3.1] represents a prototype algorithm and assumes that the elements of the sensing matrices Φ_j are distributed as i.i.d. Gaussian $\mathsf{N}(0, \sigma_j^2)$. First, the following quantities need to be defined:

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_J \end{bmatrix} \quad \text{and} \quad \widetilde{\Phi} = \begin{bmatrix} \frac{1}{m_1 \sigma_1^2} \Phi_1 \\ \frac{1}{m_2 \sigma_j^2} \Phi_2 \\ \vdots \\ \frac{1}{m_J \sigma_J^2} \Phi_J \end{bmatrix}. \tag{4.2}$$

Algorithm 2 Transpose Estimation of Common Component

Input: The measurements $\{y_j\}$, the sensing matrices $\{\Phi_j\}$, $j = 1, \ldots, J$

1: Estimate the common component as $\hat{x}_{\mathsf{C}} \leftarrow \frac{1}{J}\widetilde{\Phi}^{\top} Y$
2: **for** $j \leftarrow 1, \ldots, J$ **do**
3: Cancel the contribution of the common component from the measurements: $\tilde{y}_j \leftarrow y_j - \Phi_j \hat{x}_{\mathsf{C}}$
4: Reconstruct the innovation components, only $\hat{x}_{\mathsf{I},j} \leftarrow \arg\min_x \|x\|_1 \quad \text{s.t.} \quad \tilde{y}_j = \Phi_j x$
5: $\hat{x}_j \leftarrow \hat{x}_{\mathsf{C}} + \hat{x}_{\mathsf{I},j}$
6: **end for**

Output: The signal estimation $\{\hat{x}_j\}$

Then, Algorithm 2 is applied to reconstruct the signals. By "prototype" it is intended that Algorithm 2 sets a principle, but both the estimation of the common component and the reconstruction of each individual innovation component can be performed using any known algorithm.

4.1.2.2 Alternating Common and Innovation Estimation (ACIE)

This algorithm [1, Sect. 5.3.2] iteratively refines the estimation of the common and innovation components at each step. The principle is that a rough estimation of the common component may help obtaining a first estimation of the sparsity support of the innovation components, which in turn may help improving the estimation of the common component, and so on. The process is described in Algorithm 3.

First, the following quantities are defined. $\widehat{\mathscr{S}}_{\mathsf{I},j}$ is the estimation of the sparsity support of the innovation component of the j-th source. $\Phi_{j,\widehat{\mathscr{S}}_{\mathsf{I},j}}$ is the $m \times \left|\widehat{\mathscr{S}}_{\mathsf{I},j}\right|$ matrix obtained by keeping the columns of Φ_j indexed by $\widehat{\mathscr{S}}_{\mathsf{I},j}$. Q_j is a $m \times (m - \left|\widehat{\mathscr{S}}_{\mathsf{I},j}\right|)$ orthonormal basis for the null space of $\Phi^{\mathsf{T}}_{j,\widehat{\mathscr{S}}_{\mathsf{I},j}}$. Moreover,

$$\widetilde{Y} = \begin{bmatrix} \tilde{y}_1 \\ \tilde{y}_2 \\ \vdots \\ \tilde{y}_J \end{bmatrix} \quad \text{and} \quad \widetilde{\Phi} = \begin{bmatrix} \widetilde{\Phi}_1 \\ \widetilde{\Phi}_2 \\ \vdots \\ \widetilde{\Phi}_J \end{bmatrix}. \tag{4.3}$$

As stated in [1] and [7], when $\widehat{\mathscr{S}}_{\mathsf{I},j} = \mathscr{S}_{\mathsf{I},j}$, i.e., when the estimation of the sparsity support of the innovation component is exact, the vector $\tilde{y}_j$ from line 5 will exactly consist of the measurements of the common component, only. If $\widehat{\mathscr{S}}_{\mathsf{I},j} = \mathscr{S}_{\mathsf{I},j}$ holds for every j and the number of available measurements is big enough, then the estimation of the common component will be perfect. Simulation results in [1] show that, if J is large enough, the signals in the ensemble can be recovered in most cases. Moreover, it is shown that for small J the performance degrade as m grows large, due to the misestimation of the common component.

Algorithm 3 Alternating Common and Innovation Estimation

Input: The measurements $\{y_j\}$, the sensing matrices $\{\Phi_j\}$, $j = 1, \ldots, J$. The desired number of iterations L.

Initialize $\widehat{\mathscr{I}}_{\mathsf{I},j} \leftarrow \emptyset$, $j = 1, \ldots, J$ and $\ell \leftarrow 0$

Estimate and refine the common component and innovation sparsity supports:

for $\ell \leftarrow 1, \ldots, L$ **do**

  for $j \leftarrow 1, \ldots, J$ **do**

    Evaluate $\Phi_{j,\widehat{\mathscr{I}}_{\mathsf{I},j}}$ and Q_j from it (see above)

    $\tilde{y}_j \leftarrow Q_j^\top y_j$ and $\widetilde{\Phi}_j \leftarrow Q_j^\top \Phi_j$

  end for

  Assemble $\widetilde{Y}$ and $\widetilde{\Phi}$ as in (4.3)

  Estimate the common component: $\hat{x}_{\mathsf{C}} \leftarrow \widetilde{\Phi}^\dagger \widetilde{Y}$

  for $j \leftarrow 1, \ldots, J$ **do**

    Subtract the contribution of the estimated common component from the measurements: $\hat{y}_{\mathsf{I},j} \leftarrow y_j - \Phi_j \hat{x}_{\mathsf{C}}$

    Refine the innovation support estimation $\widehat{\mathscr{I}}_{\mathsf{I},j}$. For example, run OMP [6] using $\hat{y}_{\mathsf{I},j}$ and Φ_j as inputs.

  end for

end for

for $j \leftarrow 1, \ldots, J$ **do**

  Estimate the innovation components as $\hat{x}_{\mathsf{I},j} \leftarrow \Phi^\dagger_{j,\widehat{\mathscr{I}}_{\mathsf{I},j}} \left(y_j - \Phi_j \hat{x}_{\mathsf{C}}\right)$

  $\hat{x}_j \leftarrow \hat{x}_{\mathsf{C}} + \hat{x}_{\mathsf{I},j}$

end for

Output: The signal estimation $\{\hat{x}_j\}$

Similar to algorithm 3 is the algorithm presented in [8], which is a generalization of the one presented in [9], but for the JSM-1 scenario. It tries to solve the issue of the γ-weighted ℓ_1-norm minimization algorithm, i.e., the numerical optimization of the coefficients. The algorithm, named *Extraction and Sparse Recovery*, performs the estimation of the common and innovation components using two iterative stages, where each stage consists of an extraction phase and a sparse recovery phase.

4.2 Texas Hold'em

The first "practical" algorithm for the reconstruction of a JSM-1 modeled signal ensemble was proposed in [10]. It is a two-stage reconstruction algorithm. In the first stage, the common component is recovered. Then, in the second stage the innovation components are recovered, by subtracting the contribution of the common component to the measurements, recovering then the innovation component, only. The name of the algorithm stems from the poker Texas Hold'em terminology, where "hold'em" refers to a variant of poker games where community cards (shared among all the players) are used. In this algorithm, indeed, each measurement vector y_j is decomposed into two partitions, y_j^{c} and y_j^{h}, of length m_{c} and m_{h}, respectively, such that $m_{\mathsf{c}} + m_{\mathsf{h}} = m$. As in poker, the former, y_j^{c}, represent the

community measurements, which are shared with the entire network. The latter, y_j^{h}, instead, are the *hold* measurements, which are retained by each sensor. Denote with $\Phi\mathsf{c}_j$ and $\Phi\mathsf{h}_j$ matrices obtained by partitioning the rows of Φ_j in the same way y is partitioned into y_j^{c} and y_j^{h}. Moreover, assume that each sensor uses the same Φ_j^{c}, namely, $\Phi_j^{\mathsf{c}} = \Phi^{\mathsf{c}}$ for each $j = 1, \ldots, J$.

Algorithm 4 Averaged Community Texas Hold'em

Input: The measurements $\{y_j\}$, the sensing matrices $\{\Phi_j\}$, $j = 1, \ldots, J$.

1: Evaluate the average measurements $\bar{y} \leftarrow \frac{1}{J}\sum_{j=1}^{J} y_j^{\mathsf{c}}$
2: Recover the common component: $\hat{x}_{\mathsf{C}} \leftarrow \arg\min_x \|x\|_1 \quad \text{s.t.} \quad \bar{y} = \Phi^{\mathsf{c}} x$
3: **for** $j \leftarrow 1, \ldots, J$ **do**
4: Subtract the contribution of the estimated common component from the measurements: $\hat{y}_{\mathsf{I},j} \leftarrow y_j - \Phi_j \hat{x}_{\mathsf{C}}$
5: Recover the innovation component $\hat{x}_{\mathsf{I},j} \leftarrow \arg\min_x \|x\|_1 \quad \text{s.t.} \quad \hat{y}_{\mathsf{I},j} = \Phi x$
6: $\hat{x}_j \leftarrow \hat{x}_{\mathsf{C}} + \hat{x}_{\mathsf{I},j}$
7: **end for**

Output: The signal estimation $\{\hat{x}_j\}$

Notice that Algorithm 4 can be easily extended to signals which are sparse in a generic basis $\Psi \neq I$. The only modification required is to replace each matrix Φ_j with the product $\Phi_j\Psi$. Moreover, the recovery steps 2 and 5 can be performed with recovery algorithms different than the Basis Pursuit. The principle of the averaging step 1 is that, if J is big enough, the contribution of the innovation components will be averaged out. This implies that the recovery of the common component (and hence, of the innovation compoennt) will be affected by a systematic error due to the residual contribution of the innovation components into $\bar{y}$.

The partitioning into community and hold measurements can be explained with the aim of minimizing the intersensor communications. Only the community measurements are transmitted to the fusion center, which evaluates the common component and sends it back to the sensors, which are then able to recover their own innovation components independently.

Algorithm 4 can be adapted also to the JSM-3 model, provided that the community measurements are enough to recover the nonsparse common component, following strategies similar to the ones described in Sect. 4.1.2.

4.3 Algorithms Exploiting Side Information

In this section, we show algorithms exploiting the fact that one signal in the ensemble is perfectly known and, hence, serves as Side Information. This hypothesis is not unrealistic since, in case of sparse signals (as in JSM-1), one can think of one sensor taking $M > m$ measurements, with M large enough to recover the signal by itself

without cooperation with the desired accuracy. In this case, the overhead can be estimated as $oh_{\mathrm{SI}} = \frac{M-m}{mJ}$.

Under the JSM-3 model, instead, one signal in the ensemble can be acquired uncompressed, or compressed using a standard technique, e.g., transform coding. However, as the number of nodes increases, the overhead due to Side Information becomes negligible. The savings in the number of measurements to be acquired by the other nodes outweigh the small overhead due to the acquisition of Side Information, thus making the framework interesting even for the JSM-3 model, where the Side Information signal is not sparse.

Several examples of "standard" algorithms, modified when some form of Side Information is available, can be found in literature, not necessarily related to the JSM model or, in general, to the distributed case. As an example, refer to [11], where the OMP algorithm was modified in order to exploit the (imperfect) knowledge of the signal sparsity support.

4.3.1 The Intersect *and* Sort *algorithms*

The algorithms described in this section were proposed in [12], and hold strictly for signals ensembles modeled as the JSM-1 model. The principle behind both algorithms, depicted in Fig. 4.1, is the following. Suppose that we are able to obtain an estimation of the unknown j-th source, or of, at least, its sparsity support. Then, comparing the estimated unknown sparsity support with the one of the Side Information, one can estimate the sparsity support of the common component, then estimating the common component as the Side Information vector, limited to the elements indexed by this estimated sparsity support. The difference between the algorithms relies in how the comparison between the Side Information and the estimated unknown source is performed. For the *Intersect* algorithm (Algorithm 5), the estimated sparsity support of the common component is given by the intersection between the sparsity supports of the Side Information and of the estimated unknown source. On the other hand, for the *Sort* algorithm (Algorithm 6), the estimated sparsity support of the common component is obtained by sorting by magnitude the coefficients of the estimated unknown source, in decreasing order. Then, they are scanned in order and their index is picked, if they belong also to the sparsity support of the Side Information.

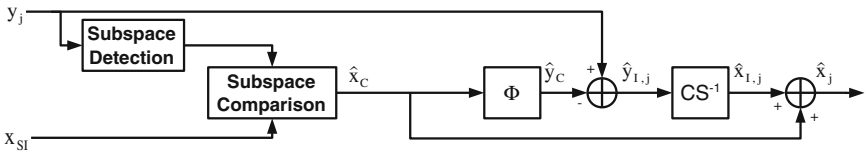

Fig. 4.1 Block diagram of the joint reconstruction stage

Algorithm 5 Intersect algorithm

Input: The measurements $\{y_j\}$, the sensing matrices $\{\Phi_j\}$, $j = 1, \ldots, J$, the Side Information x_{SI}, a threshold t

1: Side Information sparsity support $\mathscr{S}_{\mathsf{SI}} \leftarrow \{i | (x_{\mathsf{SI}})_i \neq 0\}$
2: **for** $j \leftarrow 1, \ldots, J$ **do**
3: Coarsely estimate the j-th unknown source $\tilde{x}_j \leftarrow \arg\min_x \|x\|_1 \quad \text{s.t.} \quad y_j = \Phi_j x$
4: Unknown source sparsity support estimation $\widehat{\mathscr{S}}_j \leftarrow \{i | \left|(\tilde{x}_j)_i\right| > t\}$
5: Estimate the sparsity support of the common component $\widehat{\mathscr{S}}_{\mathsf{C}} \leftarrow \mathscr{S}_{\mathsf{SI}} \cap \widehat{\mathscr{S}}_j$
6: Estimate the common component as $(\hat{x}_{\mathsf{C}})_i \leftarrow (x_{\mathsf{SI}})_i$ if $i \in \widehat{\mathscr{S}}_{\mathsf{C}}$ and $(\hat{x}_{\mathsf{C}})_i \leftarrow 0$ otherwise
7: Subtract the contribution of the estimated common component from the measurements: $\hat{y}_{\mathsf{I},j} \leftarrow y_j - \Phi_j \hat{x}_{\mathsf{C}}$
8: Recover the innovation component $\hat{x}_{\mathsf{I},j} \leftarrow \arg\min_x \|x\|_1 \quad \text{s.t.} \quad \hat{y}_{\mathsf{I},j} = \Phi x$
9: $\hat{x}_j \leftarrow \hat{x}_{\mathsf{C}} + \hat{x}_{\mathsf{I},j}$
10: **end for**

Output: The signal estimation $\{\hat{x}_j\}$

It has to be noticed that both algorithms were proposed in a $J = 2$ settings, where one source was unknown and the other one served as Side Information. Hence, in this generalization the Side Information is used pairwise with each unknown source. In particular for the Intersect algorithm, one could think to estimate the sparsity support of the common component as the intersection of the sparsity supports of all the sources in the ensemble.

Both algorithms assume that there is a high probability that the common component and the innovation compontents have disjoint sparsity supports. Their weak point is that they rely on an initial estimation of the unknown source, based on its measurements, only. This reconstruction may completely fail if m is not big enough to provide a reconstruction of acceptable quality. But it has to be noticed that this first estimation does not need to be perfect in order to obtain a better quality as an output of the algorithm. The role of the parameters t and K is the following. As for t, it sets a threshold below which to consider a coefficent as a zero. This may be helpful if the algorithm is run in a noisy setting. In a noiseless setting, it can be trivially set to 0. As for K, it should be a rough estimate of the sparsity of the common component, so that the set $\widehat{\mathscr{S}}_{\mathsf{C}}$ has a sparsity of K.

As a final remark, both algorithms can be adapted to the case in which signals are sparse in a generic basis $\Psi \neq I$. The only modification required is to replace each Φ_j with the product $\Phi_j \Psi$. It has to be reported that in the paper where these algorithms were proposed, [12], it was required that the sensing matrix was the same for each source. Generally speaking, this is not mandatory and in [12] it was required to preserve the elementwise correlation between the measurement vectors, as in Chap. 3.

Algorithm 6 Sort algorithm

Input: The measurements $\{y_j\}$, the sensing matrices $\{\Phi_j\}$, $j = 1, \ldots, J$, the Side Information x_{SI}, a parameter K

Side Information sparsity support $\mathscr{S}_{\mathsf{SI}} \leftarrow \{i | (x_{\mathsf{SI}})_i \neq 0\}$
for $j \leftarrow 1, \ldots, J$ **do**
  $\widehat{\mathscr{S}_{\mathsf{C}}} \leftarrow \emptyset$
  Coarsely estimate the j-th unknown source $\tilde{x}_j \leftarrow \arg\min_x \|x\|_1 \quad \text{s.t.} \quad y_j = \Phi_j x$
  while $\left|\widehat{\mathscr{S}_{\mathsf{C}}}\right| \leq K$ **do**
    $i \leftarrow \arg\max_i \left|(\tilde{x}_j)_i\right|$
    if $i \in \mathscr{S}_{\mathsf{SI}}$ **then**
      $\widehat{\mathscr{S}_{\mathsf{C}}} \leftarrow \widehat{\mathscr{S}_{\mathsf{C}}} \cup i$
    end if
    $(\tilde{x}_j)_i \leftarrow 0$
  end while
  Estimate the common component as $(\hat{x}_{\mathsf{C}})_i \leftarrow (x_{\mathsf{SI}})_i$ if $i \in \widehat{\mathscr{S}_{\mathsf{C}}}$ and $(\hat{x}_{\mathsf{C}})_i \leftarrow 0$ otherwise
  Subtract the contribution of the estimated common component from the measurements: $\hat{y}_{\mathsf{I},j} \leftarrow y_j - \Phi_j \hat{x}_{\mathsf{C}}$
  Recover the innovation component $\hat{x}_{\mathsf{I},j} \leftarrow \arg\min_x \|x\|_1 \quad \text{s.t.} \quad \hat{y}_{\mathsf{I},j} = \Phi x$
  $\hat{x}_j \leftarrow \hat{x}_{\mathsf{C}} + \hat{x}_{\mathsf{I},j}$
end for

Output: The signal estimation $\{\hat{x}_j\}$

4.3.2 Algorithms Based on Difference of Innovations

This family of algorithms, proposed in [13], are based on the reconstruction of the *difference* signal between the unknown signal and the Side Information. The requirement is then that all the signals are acquired using the same sensing matrix, namely, $\Phi_j = \Phi$. The resulting signal will be then the measurement of the difference of innovation components, as the common one will be canceled out by the subtraction

$$\begin{aligned} y_{\mathsf{D},j} &= y_j - y_{\mathsf{SI}} \\ &= \Phi x_j + \Phi x_{\mathsf{SI}} \\ &= \Phi \left(x_{\mathsf{C}} + x_{\mathsf{I},j}\right) - \Phi \left(x_{\mathsf{C}} + x_{\mathsf{I},\mathsf{SI}}\right) \\ &= \Phi \left(x_{\mathsf{I},j} - x_{\mathsf{I},\mathsf{SI}}\right) \\ &= \Phi x_{\mathsf{D},j} \,. \end{aligned} \tag{4.4}$$

Then, the difference signal $x_{\mathsf{D},j}$ will be recovered from y_{D}, using any reconstruction algorithm, and added to the Side Information to recover x_j

$$\begin{aligned} \hat{x}_j &= x_{\mathsf{SI}} + \hat{x}_{\mathsf{D},j} \\ &= x_{\mathsf{C}} + x_{\mathsf{I},\mathsf{SI}} + \hat{x}_{\mathsf{I},j} - \hat{x}_{\mathsf{I},\mathsf{SI}} \,, \end{aligned}$$

where the difference $x_{\mathsf{I},\mathsf{SI}} - \hat{x}_{\mathsf{I},\mathsf{SI}}$ is ideally 0. The block diagram of the algorithm is depicted in Fig. 4.2.

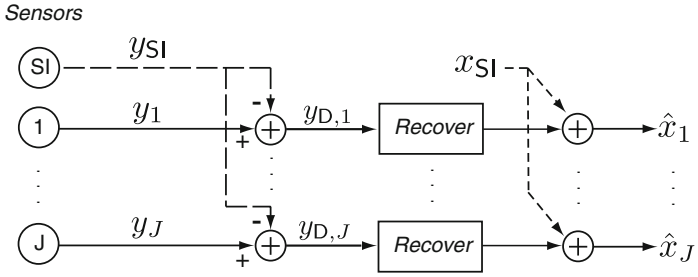

Fig. 4.2 Difference of Innovations algorithm

Two aspects emerge from (4.4). The first is that the common component is *exactly* removed from the measurements. This constitutes the biggest advantage with respect to the Texas Hold'em algorithm described in Sect. 4.2. The second aspect is that, in general, the difference signal $x_{\mathsf{D},j}$ is *less sparse* than the individual innovation component $x_{\mathsf{I},j}$. This means that this kind of algorithms is expected to be performant when the innovation components are significantly sparser that the common component (as a rule of thumb, when $k_{\mathsf{C}} \geq 2k_{\mathsf{I},j}$). The procedure is summarized in Algorithm 7 and its block diagram is depicted in Fig. 4.2.

Algorithm 7 Difference of Innovations algorithm

Input: The measurements $\{y_j\}$, the sensing matrix Φ, $j = 1, \ldots, J$, the Side Information x_{SI}
1: **for** $j \leftarrow 1, \ldots, J$ **do**
2: Compute the measurements of the difference signal $y_{\mathsf{D},j} \leftarrow y_j - y_{\mathsf{SI}}$
3: Recover $x_{\mathsf{D},j}$ as $\hat{x}_{\mathsf{D},j} \leftarrow \arg\min_x \|x\|_1 \quad \text{s.t.} \quad y_{\mathsf{D},j} = \Phi x$
4: Recover x_j as $\hat{x}_j \leftarrow x_{\mathsf{SI}} + \hat{x}_{\mathsf{D},j}$
5: **end for**
Output: The signal estimation $\{\hat{x}_j\}$

One important fact about the difference of innovation algorithm is that it can be readily implemented in a parallel architecture, since each reconstruction is independent from each other.

The second algorithm proposed in [13] attempts at overcoming the drawbacks of the difference of innovations algorithm and of the Texas Hold'em algorithm. It is labeled as *Texas Difference of Innovations*, and it is schematically represented in Fig. 4.3. From the Texas Hold'em algorithm, it inherits the idea of averaging the measurement vectors of the unknown signals. On the contrary, instead of explicitly estimating the common component, it combines the average of the measurements with the Side Information, in the following way. On one hand, it uses the average of the measurements, which is itself an estimation of the measurements of the common component, and subtracts it from the measurements of the Side Information, to obtain the estimate of the measurements of the innovation of the Side Information, namely

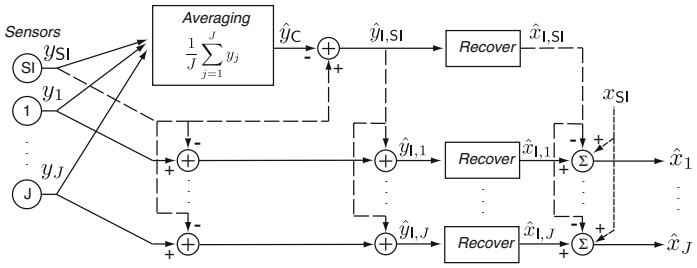

Fig. 4.3 Texas Difference of Innovations algorithm

$$\hat{y}_{\mathsf{C}} = \frac{1}{J}\sum_{j=1}^{J} y_j \tag{4.5}$$

$$\hat{y}_{\mathsf{I,SI}} = y_{\mathsf{SI}} - \hat{y}_{\mathsf{C}} \,. \tag{4.6}$$

The output of this branch will be the estimate of the innovation component of the Side Information, which will be recovered from $\hat{y}_{\mathsf{I,SI}}$.

In parallel, on each branch corresponding to each unknown signal, the difference of innovation procedure of (4.4) is performed. Adding to the output of (4.4) the previously estimated measurement of the innovation component of the Side Information, it is possible to obtain an estimate of the innovation component of the unknown source, namely,

$$\begin{aligned} \hat{y}_{\mathsf{I},j} &= y_j - y_{\mathsf{SI}} + \hat{y}_{\mathsf{I,SI}} \\ &= y_{\mathsf{I},j} - y_{\mathsf{I,SI}} + \hat{y}_{\mathsf{I,SI}} \,, \end{aligned} \tag{4.7}$$

where the difference $\hat{y}_{\mathsf{I,SI}} - y_{\mathsf{I,SI}}$ is ideally zero, which can be separately recovered to obtain the innovation component of each unknown signal. Combining the Side Information, the estimate of its innovation component and the one of the unknown signal, the estimate of the entire unknown signal will be obtained

$$\begin{aligned} \hat{x}_j &= x_{\mathsf{SI}} - \hat{x}_{\mathsf{I,SI}} + \hat{x}_{\mathsf{I},j} \\ &= \hat{x}_{\mathsf{C}} + \hat{x}_{\mathsf{I},j} \end{aligned} \tag{4.8}$$

The complete procedure is reported in Algorithm 8.

As reported in [13], the Texas Difference of Innovations algorithm is affected by the same error floor as the Texas algorithm of Sect. 4.2, due to the averaging procedure. However, since this error floor decreases as $\frac{1}{\sqrt{J}}$, it is likely that with a large ensemble, the performance is limited by different sources of noise.

Algorithm 8 Texas Difference of Innovations algorithm

Input: The measurements $\{y_j\}$, the sensing matrix Φ, $j = 1, \ldots, J$, the Side Information x_{SI}

1: Estimate the measurements of the common component $\hat{y}_{\mathsf{C}} \leftarrow \frac{1}{J} \sum_{j=1}^{J} y_j$
2: Estimate the measurements of the innovation component of the Side Information $\hat{y}_{\mathsf{I,SI}} \leftarrow y_{\mathsf{SI}} - \hat{y}_{\mathsf{C}}$
3: Recover the innovation component of the Side Information $\hat{x}_{\mathsf{SI}} \leftarrow \arg\min_x \|x\|_1 \quad \text{s.t.} \quad \hat{y}_{\mathsf{I,SI}} = \Phi x$
4: **for** $j \leftarrow 1, \ldots, J$ **do**
5: Compute the measurements of the difference signal $y_{\mathsf{D},j} \leftarrow y_j - y_{\mathsf{SI}}$
6: Estimate the measurements of the innovation component of the unknown signal $\hat{y}_{\mathsf{I},j} \leftarrow y_{\mathsf{D},j} + \hat{y}_{\mathsf{I,SI}}$
7: Recover the innovation component of the unknown signal $\hat{x}_{\mathsf{I},j} \leftarrow \arg\min_x \|x\|_1 \quad \text{s.t.} \quad y_{\mathsf{I},j} = \Phi x$
8: Recover x_j as $\hat{x}_j \leftarrow x_{\mathsf{SI}} - \hat{x}_{\mathsf{I,SI}} + \hat{x}_{\mathsf{I},j}$
9: **end for**

Output: The signal estimation $\{\hat{x}_j\}$

4.4 Performance Comparison

In the following, we report some comparative performance figures. They compare the performance of some of the algorithms described in this section in the JSM-1 and JSM-3 scenarios.

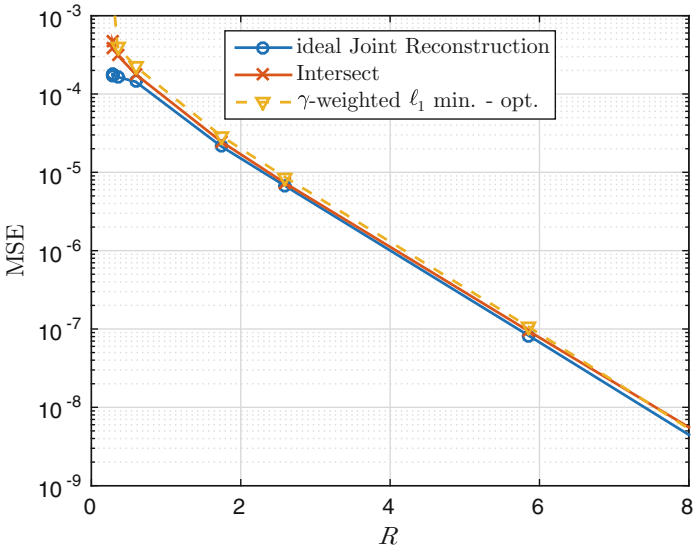

Fig. 4.4 JSM-1. Mean square error versus measurement quantization rate. $n = 512$, $k_{\mathsf{C}} = k_{\mathsf{I},j} = 8$, $\sigma^2_{\Phi} = 1$, $\sigma^2_{x_{\mathsf{C}}} = 1$, $\sigma^2_{x I_j} = 10^{-2}$ and $m = 128$

In [12, Fig. 3], it was shown that in a JSM-1 signal scenario, sort and intersect algorithms perform quite close to each other, with a slight penalty with respect to the ideal case. Here, we first compare in Fig. 4.4 the performance of the intersect algorithm and of the γ-weighted ℓ_1 norm minimization (properly modified to take into account the presence of a perfectly known source) with an oracle scheme knowing perfectly the sparsity support of the common component. The scenario is a noisy setting where the source of noise is the quantization of the measurements with a uniform quantizer using R bits per measurement sample. The comparison is performed in terms of Mean Square Error versus R. The length of the signals is $n = 512$, the sparsity of the common component and of the innovation components is $k_{\mathsf{C}} = k_{\mathsf{I},j} = 8$, while the nonzeros are modeled as zeromean Gaussian random variables with unit variance. The number of measurements is $m = 128$. It can be noticed that the intersect algorithm perform very close to the ideal Joint Reconstruction and slightly better (and will less computational requirements) than the γ-weighted ℓ_1-norm minimization.

Then, in Fig. 4.5 we report the performance of various algorithms in terms of Mean Square Error versus the number of measurements per signal m. Simulations have been performed using sensing matrices with Gaussian i.i.d. entries, with zero mean and unit norm columns. The model for the ensemble is the JSM-1 model, where the nonzeros of the common and the innovation components are generated as zeromean Gaussians with unit variance. Quantization noise is introduced by quantizing each entry of the measurements with R bits. The signal ensemble counts $J = 100$ signals. The sparsity of the common component is $k_{\mathsf{C}} = 20$, while the innovation components are $k_{\mathsf{I},j} = 5$-sparse. The length of the signals is $n = 256$. For what concerns the

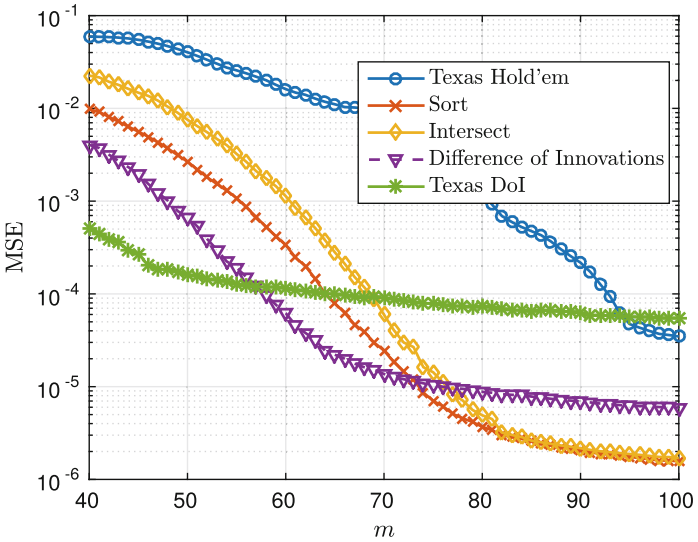

Fig. 4.5 JSM-1. Mean square error versus number of measurements m. $J = 100$, $k_{\mathsf{C}} = 20$, $k_{\mathsf{I},j} = 5$, $n = 256$, $R = 8$ bps

Texas Hold'em algorithm, the entire set of measurements is considered as *community measurements*, hence contributing to the average.

From Fig. 4.5, it can be noticed that the Texas Hold'em algorithm, the only one in the pool not relying on Side Information, is the one that performs worst. As we have stated in Sect. 4.3, the overhead related to the Side Information becomes negligible when the signal ensemble dimension J is large. When few measurements are available, corresponding to the left-hand side of the figure, the Texas DOI algorithm is the one with best performance. DOI algorithm, in fact, performs worse than Texas DOI when few measurements are available, since it needs to reconstruct the difference signal, which is less sparse than the innovation component of the single signal. Also sort and intersect algorithms have poor performance when m is low, since they rely on an initial estimate of the unknown signal to recover the sparsity support of the common component. On the other hand, when m grows large, Texas DOI shows its typical error floor, due to averaging, while the performance of DOI, Sort and Intersect improve, because they can fully exploit the presence of the Side Information. For some m, the performance of Sort and Intersect overcome the one of DOI, as the estimation of the common component improves, so they have to recover the innovation component, which is sparser than the difference signal recovered by DOI.

On the other hand, Fig. 4.6 reports the performance of various algorithms in terms of Mean Square Error versus the number of measurements per signal m for a JSM-3 modeled signal ensemble. The nonsparse x_c component contains i.i.d. Gaussian random variables with zero mean and unit variance. The performance of DOI and

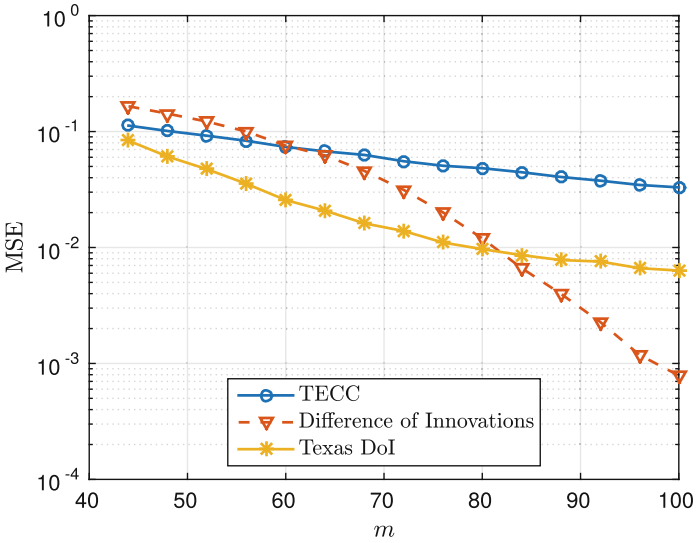

Fig. 4.6 JSM-3. Mean square error versus number of measurements m. $J = 100, k_{l,j} = 5, n = 256$, $R = 8$ bps

Texas DOI are compared with the TECC algorithm described in Sect. 4.1.2.1, where the latter uses a different sensing matrix for each source. From Fig. 4.6 it can be noticed that the Texas DOI algorithm has the best performance when the number of measurements m is low. In this regime, Texas DOI show slightly worse performance than TECC. When m grows, the performance of TECC improves but slower than DOI, while the one of Texas DOI tends to even out, because of the error floor due to the averaging. Hence, after a certain threshold, DOI is the best performing algorithm.

References

1. Baron, D., Duarte, M.F., Wakin, M.B., Sarvotham, S., Baraniuk, R.G.: Distributed compressive sensing. arXiv preprint arXiv:0901.3403 (2009)
2. Mishali, M., Eldar, Y.C.: Reduce and boost: recovering arbitrary sets of jointly sparse vectors. IEEE Trans. Signal Process. **56**(10), 4692–4702 (2008)
3. Fornasier, M., Rauhut, H.: Recovery algorithms for vector-valued data with joint sparsity constraints. SIAM J. Numer. Anal. **46**(2), 577–613 (2008)
4. Hormati, A., Vetterli, M.: Distributed compressed sensing: sparsity models and reconstruction algorithms using annihilating filter. In: IEEE International Conference on Acoustics, Speech and Signal Processing 2008, pp. 5141–5144. ICASSP 2008, IEEE (2008)
5. Coluccia, G., Kuiteing, S.K., Abrardo, A., Barni, M., Magli, E.: Progressive compressed sensing and reconstruction of multidimensional signals using hybrid transform/prediction sparsity model. IEEE J. Emerg. Sel. Top. Circuits Syst. **2**(3), 340–352 (2012)
6. Tropp, J.A., Gilbert, A.C.: Signal recovery from random measurements via orthogonal matching pursuit. IEEE Trans. Inf. Theory **53**, 4655–4666 (2007)
7. Wakin, M.B., Duarte, M.F., Sarvotham, S., Baron, D., Baraniuk, R.G.: Recovery of jointly sparse signals from few random projections. In: NIPS (2005)
8. Xu, W., Lin, J., Niu, K., He, Z.: A joint recovery algorithm for distributed compressed sensing. Trans. Emerg. Telecommun. Technol. **23**(6), 550–559 (2012)
9. Davenport, M.A., Boufounos, P.T., Baraniuk, R.G.: Compressive domain interference cancellation. Technical report, DTIC Document (2009)
10. Schnelle, S., Laska, J., Hegde, C., Duarte, M., Davenport, M., Baraniuk, R.: Texas hold 'em algorithms for distributed compressive sensing. In: IEEE ICASSP. pp. 2886–2889 (2010)
11. Stankovic, V., Stankovic, L., Cheng, S.: Compressive image sampling with side information. In: 2009 16th IEEE International Conference on Image Processing (ICIP), pp. 3037–3040 (2009)
12. Coluccia, G., Magli, E., Roumy, A., Toto-Zarasoa, V., et al.: Lossy compression of distributed sparse sources: a practical scheme. In: 2011 European Signal Processing Conference (EUSIPCO) (2011)
13. Valsesia, D., Coluccia, G., Magli, E.: Joint recovery algorithms using difference of innovations for distributed compressed sensing. In: 2013 Asilomar Conference on Signals, Systems and Computers, IEEE, pp. 414–417 (2013)

Chapter 5
Distributed Recovery

This chapter surveys a few basic algorithms for distributed reconstruction from compressive measurements in a network of nodes, which may be sensors or nodes that collect measurements from different sensors. This estimation problem can be recast into an optimization problem where a convex and separable loss function should be minimized subject to sparsity constraints. The goal of the network is to handle distributed sparse estimation. Clearly, to achieve such a goal, the nodes must share, at least partially, their estimation. A single node typically has limited memory and processing capability, therefore cooperation is the key to compensate for this lack and achieve satisfactory performance. Cooperation, however, raises the problem of communication among nodes, which is known to be the largest consumer of the limited energy of a node, compared to other functions such as sensing and computation. Particular attention is devoted to energy efficiency, in terms of transmissions and memory requirements.

5.1 Introduction

In this chapter, we consider the problem of in-network processing and recovery in DCS [1]. In the distributed setting, a network is considered with J nodes that individually store data

$$y_v = \Phi_v x + e_v,$$

where $x \in \Sigma_k \subseteq \mathbb{R}^n$, $\Phi_v \in \mathbb{R}^{m \times n}$, and $e_v \in \mathbb{R}^m$ is a bounded perturbation term and $v \in \{1, \ldots, J\}$. The most used paradigm can be summarized as follows: nodes transmit data $(y_v, \Phi_v)_{v=1}^J$ to a central processing unit that performs joint estimation of the signal $x \in \mathbb{R}^n$ with $\Phi = (\Phi_1^\top, \ldots, \Phi_J^\top)^\top$ and $y = (y_1^\top, \ldots, y_J^\top)^\top$. A drawback of this model is that, particularly in large-scale networks, gathering all data to a central processing unit may be inefficient, as a large number of hops have to be

G. Coluccia et al., *Compressed Sensing for Distributed Systems*,
SpringerBriefs in Signal Processing, DOI 10.1007/978-981-287-390-3_5

taken, requiring a significant amount of energy for communication over the wireless channel. Moreover, this may also introduce delays, severely reducing the network performance. In other applications, nodes providing private data may not be willing to share them, but only to build the learning results. This occurs, for example, in classification [2], one of the fundamental problems in machine learning, or in data fitting in statistics [3], where the training data consist of sensitive information (such as medical data, or data flow across the Internet). We also notice that parallel computing can be recast in this distributed context. Even if the data are centrally available, we could be interested in decentralizing them on the cores of a Graphics Processing Unit (GPU) cores and threads in order to distribute the computational load and accelerate the estimation procedure (see [4, 5]).

Specifically, we assume that no fusion center is available and we consider networks formed by nodes that can store a limited amount of information, perform a low number of operations, and communicate under some constraints. Our aim is to study how to achieve compressed acquisition and recovery leveraging these seemingly scarce resources, with no computational support from an external processor.

A common element to many distributed applications is the necessity of limiting computations and memory usage, as the nodes generally consist of devices with scarce capabilities in this sense. An evident example is given by sensor networks: sensors require a lot of energy to perform computations and are generally endowed with small memories of the order of a few kB. Also in parallel computing this issue is crucial, as the memory is known to be a bottleneck to the system performance.

So far, distributed recovery has received less attention than the centralized problem and the literature is very recent [2, 6–9]. These first contributions propose natural ways to distribute known centralized methods, and obtain interesting results in terms of convergence and estimation performance. However they do not consider the problem of the insufficient computation and memory resources. In particular, in [6] a distributed pursuit recovery algorithm is proposed, assuming that every node v knows the matrix Φ_v of every other node. This estimation scheme is clearly unpractical in large-scale networks, since individual nodes do not have the capacity to store and process a large number of these matrices. Distributed basis pursuit algorithms for sparse approximations when the measurement matrices are not globally known have been studied in [7–9]. In these algorithms, sensors collaborate to estimate the original vector, and, at each iteration, they update this estimate based on communication with their neighbors in the network.

In this chapter, we describe in detail three families of methods: distributed subgradient methods (DSM, [10]), alternating direction method of multipliers (ADMM, [11]), and distributed iterative soft thresholding algorithms (DISTA, see [12–15]) for Lasso estimation problems. We compare these algorithms via extensive simulations. In particular, we show that distributed algorithms are satisfactory in the following sense: when the product of the number of nodes in the network times the number of data for each unit exceeds a given threshold, accurate estimation is achieved. Moreover, the total number of available data required for the estimation is comparable to that required by joint estimation. This implies that decentralization is not a drawback.

In the mentioned algorithms, each node keeps an approximation of the original signal and updates it iteratively according to both its own measurements and local information received from its neighbors. Although these methods are of significant interest, as they do away with a centralized processing unit, they still suffer from one or more of the following limitations [16]:

(a) *Synchronous updates*: the nodes are assumed to send and process signals synchronously [12, 15, 17, 18].
(b) Selection of the *gradient stepsize*: we distinguish between constant and diminishing stepsizes. Constant stepsizes do not guarantee convergence [18] or guarantee convergence only to neighborhoods of the optimal solution [15]. Larger stepsizes imply larger neighborhoods, while smaller ones produce slower convergence. On the other hand, diminishing stepsizes can ensure convergence, but a suitable design of them is very difficult and the convergence rate is always slow [19].
(c) *Spanning tree* and *multi-hop transmissions*: the nodes first generate a spanning tree, which implies that they must be aware of the network's structure, in terms of position of the root and of their own roles (parents or children) with respect to their neighbors. Moreover, a routing protocol is necessary and multi-hop communication occurs at each iteration [20].
(d) *Bandwidth and energy consumption*: these algorithms are generally not optimized for metrics that are important in a distributed setting, most notably, bandwidth and energy consumption.
(e) *Computational complexity:* these techniques often have a high computational cost as they require every node to solve a convex optimization problem in each iteration. Such computational capacity may not be available in low-power sensor networks.

Since a single node has limited memory and processing capability, a cooperation is the key to compensate for this lack and achieve satisfactory performance, as proved in many works, e.g., [12, 15, 20]. Cooperation, however, raises the problem of communication among nodes, which can be expensive from different viewpoints. For example, if we consider a territorial monitoring wireless sensor network, the nodes may be deployed at large distance or in unfavorable conditions, which makes communication uneconomical.

We describe algorithms that reduce the number of necessary information exchanges [13, 14, 20] and we compare them, in terms of transmissions, memory requirements, and energy efficiency. These methods use time-varying communication, which allows us to limit the communication load, and also to overcome synchronization issues. These methods are able to achieve performance equivalent to that of centralized schemes if properly designed. In this chapter we compare different algorithms in terms of accuracy of the reconstructed signals and number of transmitted values required to obtain a certain accuracy.

Finally, we extend the discussion to the joint sparsity models and describe how to design efficient distributed algorithms for the recovery of multiple correlated signals.

5.2 Problem Setting

Before presenting the mathematical formulation of the problem, we fix some notations and review some basic definitions of graph theory. An undirected graph is a pair $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ is the set of vertices (or nodes), and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges with the property $(i, j) \in \mathcal{E}$ implies $(j, i) \in \mathcal{E}$. In this chapter, we use the convention that $(i, i) \in \mathcal{E}$ for all $i \in \mathcal{V}$. A path in a graph is a sequence of edges which connects a sequence of vertices. In an undirected graph $\mathcal{G}$, two vertices u and v are called connected if there exists a path from u to v. A graph is said to be connected if every pair of vertexes in the graph is connected. A graph is said to be regular when each vertex is connected to the same number of nodes. Using the convention that each node has a self-loop, we call d-regular a graph in which each vertex is connected to exactly d nodes different from itself. A matrix with nonnegative elements P is said to be stochastic if $\sum_{j \in \mathcal{V}} P_{ij} = 1$ for every $i \in \mathcal{V}$. Equivalently, P is stochastic if $P1 = 1$, where 1 is the vector of ones. The matrix P is said to be adapted to a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ if $P_{v,w} = 0$ for all $(w, v) \notin \mathcal{E}$. We finally define the neighborhood of v as the set $\mathcal{N}_v$ that contains all $w \in \mathcal{V}$ such that $P_{v,w} \neq 0$. According to our notation, $v \in \mathcal{N}_v$ as $(v, v) \in \mathcal{E}$.

In this chapter, we consider a network represented by an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. $\mathcal{V}$ is the set of J nodes, labeled as $v \in \mathcal{V} = \{1, \ldots, J\}$, and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of available communication links. The set $\mathcal{V}$ can represent sensors or processing units that collect measurements of a common sparse signal $x^\star \in \mathbb{R}^n$ of the form

$$y_v = \Phi_v x^\star, \qquad v \in \mathcal{V} \tag{5.1}$$

where $y_v \in \mathbb{R}^m$ are the local data and $\Phi_v \in \mathbb{R}^{m \times n}$ is the local sensing matrix.

If the data stored by all nodes (y_v, Φ_v) were available at once in a single central processing unit, an estimation of the signal $x^\star$ would be provided by the solution of the following constraint satisfaction problem:

$$\begin{cases} y = \Phi x \\ x \in \Sigma_k \end{cases} \tag{5.2}$$

where

$$\Phi = \begin{pmatrix} \Phi_1 \\ \Phi_2 \\ \ldots \\ \Phi_J \end{pmatrix}, \qquad y = \begin{pmatrix} y_1 \\ y_2 \\ \ldots \\ y_J \end{pmatrix} \tag{5.3}$$

As already observed in Chap. 2, if for every index set $\Gamma \subseteq \{1, \ldots, n\}$ with $|\Gamma| = 2k$ the columns of Φ associated with Γ are linearly independent, then $x^\star$ is the unique solution to (5.2) (see also [21]). From now on, we make also the following assumption.

Assumption 1 The following conditions hold:

1. every $2k$ columns of Φ are linearly independent;
2. there exists $\overline{v} \in \mathscr{V}$ such that every k columns of $\Phi_{\overline{v}}$ are linearly independent.

Another way to express this condition is via the spark (spark(Φ) is the minimum number of linearly dependent columns). Thus, Φ must satisfy spark$(\Phi) > 2k$ and there exists $\overline{v} \in \mathscr{V}$ such that spark$(\Phi_{\overline{v}}) > k$. When $k \leq m$, Assumption 1.2 guarantees that the number of solutions to the linear system $\Phi_{\overline{v}} x_{\overline{v}} = y_{\overline{v}}$ is finite. This condition is rather mild in the sense that if the components of $\Phi_{\overline{v}}$ are independently randomly generated from a continuous distribution, then it will be satisfied with probability one.

In a distributed scenario, it is reasonable to assume that (y_v, Φ_v) are available only at the v-th node. Nodes seek to estimate the signal $x^\star \in \mathbb{R}^n$ starting from their own sets of measurements under the assumption that it is sparse. It should be noted that the presence of the common variable x in the optimization problem (5.2) imposes a coupling among the nodes in the network. Then any distributed algorithm requires collaboration of sensor nodes that are assumed to be able to leverage *local communication* and share *partial information* with other neighbors.

We need to specify two main features in order to describe the in-network estimation system:

- the *consensus-based optimization model*, specifying the overall cost function that nodes are cooperatively minimizing by individually minimizing their own local objective functions;
- the *communication and processing model* describing the evolution of the nodes' estimates and specifying the interactions among the nodes (e.g., how often the nodes communicate, the confidence level on their own information and the information received from the neighbors, etc.).

The models are discussed in the following sections.

5.2.1 Consensus-Based Optimization Model

Here, we describe the general principles behind distributed sparse recovery via optimization. Then we will discuss some specific examples.

If the data stored by all nodes (y_v, Φ_v) were available at once in a single central processing unit, an estimation of the vector $x^\star$ would be provided by optimization problem of the form:

$$\min_{x \in \mathbb{R}^n} f(x) \tag{5.4}$$

where $f(x) = f(x; y, \Phi)$ is a separable loss function, i.e., it can be decomposed as a sum of J terms

$$f(x) := \sum_{v \in \mathscr{V}} f_v(x),$$

where we refer to $f_v(x)$ as the v-th term in the cost function that is only known to node v. Local function $f_v(x)$ indicates how much a vector $x \in \mathbb{R}^n$ is consistent with the local measurements vector y_v and can also encode constraints by assigning $f_v(x) = +\infty$ when a constraint is violated. The exchange of the loss function or of its gradient may require the transmission of a large amount of data and is not practical in most cases. Instead, the nodes must coordinate and communicate other information such as their current estimates x_v or gradients of $f_v(x)$ evaluated at a particular value.

The consensus-based optimization can be viewed as a technique to recast the optimization problem into a separable form that facilitates distributed implementation. For a recent overview on consensus algorithms, we refer to [22] and reference therein. The goal is to split the problem into simpler subtasks that can be executed locally at each node. Let us replace the global variable x in (5.4) with local variables $\{x_v\}_{v \in \mathscr{V}} \in \mathbb{R}^n$. The problem can be rewritten as follows:

$$\begin{aligned} &\min_{x_1, \ldots, x_J \in \mathbb{R}^n} \sum_{v \in \mathscr{V}} f_v(x_v) \\ &\text{s.t.} \quad x_v = x_w \quad \forall (v, w) \in \mathscr{E} \,. \end{aligned} \tag{5.5}$$

This problem is generally known as *global consensus problem*, since the constraint imposes a coupling among the nodes or, equivalently, an agreement among all the local variables.

Proposition 5.1 *If $\mathscr{G}$ is a connected graph, then the optimization problems in* (5.4) *and* (5.5) *are equivalent, in the sense that any solution of* (5.4) *is a minimizer for* (5.5) *and vice versa.*

A variety of algorithms have been proposed to perform sparse recovery. They differ essentially for the choice of loss functions $f_v(x)$ and for the constraints that generally are relaxed. In the following, we give some examples. For simplicity of exposition, we describe some examples of consensus-based sparse recovery problems under the following assumption.

Assumption 2 The graph of communication is

1. connected;
2. regular, that is, all nodes $v \in \mathscr{V}$ have degree $d_v = d$.

5.2.1.1 Consensus-Based Distributed Lasso

As a first example, we consider the Lasso optimization problem

$$\min_{x\in\mathbb{R}^n} f(x) := \min_{x\in\mathbb{R}^n} \frac{1}{2}\sum_{v\in\mathscr{V}}\left[\|y_v - \Phi_v x\|_2^2 + 2\lambda\|x\|_1\right], \tag{5.6}$$

for some $\lambda > 0$. It is well known that for problems in which the number of variables n exceeds the number of observations m the function $f(x)$ is not strictly convex, and hence it may not have a unique minimum. Sufficient conditions guaranteeing the uniqueness of the solution of (5.6) are derived in [23]. We make the following assumption throughout the chapter.

Assumption 3 The problems in (5.6) admits a unique solution x_{Lasso}.

Example 5.1 (Consensus constraints relaxation) The consensus-based reformulation of (5.6) can be expressed as

$$\begin{aligned}&\min_{x_1,\ldots,x_J\in\mathbb{R}^n} \frac{1}{2}\sum_{v\in\mathscr{V}}\left[\|y_v - \Phi_v x_v\|_2^2 + \frac{2\lambda}{J}\|x_v\|_1\right], \\ &\text{s.t. } x_v = \overline{x}_w, \quad \forall w \in \mathscr{N}_v, \forall v \in \mathscr{V},\end{aligned} \tag{5.7}$$

where $\overline{x}_w = \frac{1}{d+1}\sum_{u\in\mathscr{N}_w} x_u$. The following can be easily guessed.

Proposition 5.2 *If $\mathscr{G}$ is a connected graph, then the optimization problem in (5.7) is equivalent to (5.6), in the sense that any solution of (5.6) is a minimizer for (5.7) and vice versa.*

Proof Since $v \in \mathscr{N}_v$, from the constraints in (5.7) we have $x_v = \overline{x}_v$ and $x_w = \overline{x}_v$ for all $w \in \mathscr{N}_v$. This implies by transitivity that $x_v = x_w$ for all $w \in \mathscr{N}_v$. If the graph is connected, then there exists a path connecting every pair of vertexes. We can conclude that $x_v = x$ for any $v \in \mathscr{N}_v$, in which case the cost function (5.7) reduces to the one in (5.6).

Relaxing the consensus constraints in (5.7) and minimizing the functional $\mathscr{F} : \mathbb{R}^{n\times J} \longmapsto \mathbb{R}^+$ defined by

$$\mathscr{F}(x_1,\ldots,x_J) := \frac{1}{2}\sum_{v\in\mathscr{V}}\left[\|y_v - \Phi_v x_v\|_2^2 + \frac{2\lambda}{J}\|x_v\|_1 + \frac{1}{\tau(d+1)}\frac{1-q}{q}\sum_{w\in\mathscr{N}_v}\|\overline{x}_w - x_v\|_2^2\right] \tag{5.8}$$

for some $q \in (0, 1)$, each node seeks to estimate the sparse vector $x^\star$ and to enforce agreement with the estimates calculated by other nodes in the network. It should also be noted that $\mathscr{F}(x,\ldots,x) = f(x)$.

The following theorem ensures that, under certain conditions on the parameter τ, the problem of minimizing the functions in (5.8) is well-posed. Moreover, the necessary optimality conditions of (5.8) are derived and the relationships with the original problem (5.6) is discussed.

Let $\Gamma : \mathbb{R}^{n\times J} \longmapsto \mathbb{R}^{n\times J}$ be the operator defined as

$$(\Gamma X)_v = \eta_\alpha \left[(1-q)(X(P^\top)^2)_v + q(x_v + \tau \Phi_v^\top (y_v - \Phi_v x_v)) \right] \tag{5.9}$$

where $v \in \mathscr{V}$, $\alpha = q\lambda\tau/J$, $P_{v,w} = 1/(d+1)$ if $(v,w) \in \mathscr{E}$ and $P_{v,w} = 0$ otherwise, and

$$\eta_\alpha(s) := \begin{cases} \mathrm{sgn}(s)(|s| - \alpha) & \text{if } |s| \geq \alpha \\ 0 & \text{otherwise.} \end{cases} \tag{5.10}$$

Theorem 5.1 (Characterization of minima) *If $\tau < \|\Phi_v\|_2^{-2}$ for all $v \in \mathscr{V}$, the set of minimizers of the function $\mathscr{F}$, defined in (5.8), is not empty and coincides with the set*

$$\mathrm{Fix}(\Gamma) := \{Z \in \mathbb{R}^{n\times J} : \Gamma Z = Z\}.$$

Theorem 5.1 is proved in [15] through variational techniques. The following theorem states that q can be interpreted as a temperature; as q decreases, estimates x_v's associated with adjacent nodes become increasingly correlated. This fact suggests that if q is sufficiently small, then each vector $\widehat{x}_v^q$ can be used as an estimate of the vector $x^\star$. The proof of Theorem 5.2 is reported in [15].

Theorem 5.2 *Let us denote as $\{\widehat{x}_v^q\}_{v\in\mathscr{V}}$ a minimizer of $\mathscr{F}(x_1, \ldots, x_J)$ in (5.8). Under Assumption 2*

$$\lim_{q\to 0} \widehat{x}_v^q = x_{\text{Lasso}}, \quad \forall v \in \mathscr{V}.$$

Example 5.2 (Consensus-based quadratic programming distributed Lasso) We consider now the following alternative that relies on a dual formulation of (5.6):

$$\min_{\{x_v\}_{v\in\mathscr{V}}, \{\breve{z}_v^w, \breve{z}_v^w\}_{(v,w)\in\mathscr{E}}} \frac{1}{2} \sum_{v\in\mathscr{V}} \left[\|y_v - \Phi_v x_v\|_2^2 + \frac{2\lambda}{J} \|x_v\|_1 \right], \tag{5.11}$$

$$\text{s.t. } x_v = \breve{z}_v^w, \; x_w = \breve{z}_v^w, \; \breve{z}_v^w = \breve{z}_v^w \quad \forall w \in \mathscr{N}_v, \forall v \in \mathscr{V}.$$

where $\check{z}$ and $\check{\check{z}}$ are auxiliary variables that yield an alternative representation of the constraint set in (5.5).

Proposition 5.3 *If $\mathscr{G}$ is a connected graph, then the optimization problem in (5.6) is equivalent to (5.11), in the sense that any solution of (5.11) is a minimizer for (5.6) and vice versa.*

The quadratically augmented Lagrangian function (see [9, 11, 24] for background) is defined by

$$\begin{aligned}\mathscr{L}(x,z,\omega) &:= \mathscr{L}(\{x_v\}_{v\in\mathscr{V}}, \{\check{\omega}_v^w\}_{(v,w)\in\mathscr{E}}, \{\hat{\omega}_v^w\}_{(v,w)\in\mathscr{E}}, \{\check{z}_v^w\}_{(v,w)\in\mathscr{E}}, \{\check{\check{z}}_v^w\}_{(v,w)\in\mathscr{E}}) \\ &= \sum_{v\in\mathscr{V}} \left\{ \frac{1}{2}\|y_v - \Phi_v x_v\|_2^2 + \frac{\lambda}{J}\|x_v\|_1 + \frac{\rho}{2}\sum_{w\in\mathscr{N}_v}\left[\|x_v - \check{z}_v^w\|_2^2 + \|x_w - \check{\check{z}}_v^w\|_2^2\right] \right. \\ &\qquad \left. + \sum_{w\in\mathscr{N}_v}\left[(\check{\omega}_v^w)^\top(x_v - \check{z}_v^w) + (\hat{\omega}_v^w)^\top(x_w - \check{\check{z}}_v^w)\right] \right\}, \end{aligned} \tag{5.12}$$

where $\{\omega_v^w\}_{(v,w)\in\mathscr{E}}$ are the Lagrange multipliers associated to the constraints. The constraints $\check{z}_v^w, \check{\check{z}}_v^w \in C_z = \{z : \check{z}_v^w = \check{\check{z}}_v^w, (v,w) \in \mathscr{E}\}$ have not been dualized, and $\rho > 0$ is a preselected parameter. It should be noted that the quadratic terms in (5.12) guarantee that $\mathscr{L}$ is strictly convex with respect to the variables (x, z). Moreover, the augmented Lagrangian function has a saddle point, i.e., there exists a point $(x_\star, z_\star, \omega_\star)$ such that

$$\mathscr{L}(x_\star, z_\star, \omega) \le \mathscr{L}(x_\star, z_\star, \omega_\star) \le \mathscr{L}(x, z, \omega_\star)$$

The related dual problem is highly decomposable and it decouples into J subproblems:

$$\begin{aligned} \max_{\{\check{\omega}_v^w, \hat{\omega}_v^w\}_{(v,w)\in\mathscr{E}}} & \; g(\{\check{\omega}_v^w, \hat{\omega}_v^w\}_{(v,w)\in\mathscr{E}}) \\ g(\{\check{\omega}_v^w, \hat{\omega}_v^w\}_{(v,w)\in\mathscr{E}}) = \min_{\{x_v, \check{z}_v^w, \check{\check{z}}_v^w\}_{(v,w)\in\mathscr{E}}} & \; \mathscr{L}(\{x_v, \check{\omega}_v^w, \hat{\omega}_v^w, \check{z}_v^w, \check{\check{z}}_v^w\}_{(v,w)\in\mathscr{E}}) \end{aligned} \tag{5.13}$$

Assuming that strong duality holds [24], the solution of the primal and dual problems are the same and a primal optimal solution is retrieved from a dual optimal point.

A related estimation scheme is provided in [9] by the following optimization problem

$$\begin{aligned} \min_{\{x_v, z_v\}_{v\in\mathscr{V}}} & \; \frac{1}{2}\sum_{v\in\mathscr{V}}\left[\|y_v - \Phi_v\gamma_v\|^2 + \frac{2\lambda}{J}\|x_v\|_1\right] \\ \text{s.t.} \quad & x_v = \gamma_v, \; v \in \mathscr{V} \\ & x_v = \check{z}_v^w, \; x_w = \check{\check{z}}_v^w, \; \check{z}_v^w = \check{\check{z}}_v^w \quad \forall w \in \mathscr{N}_v, \forall v \in \mathscr{V}. \end{aligned} \tag{5.14}$$

Introducing the Lagrange multipliers $\hat{\omega}$ associated to the additional constraint in (5.14) and iterating the procedure above the following augmented Lagrangian function can be used for estimation

$$\begin{aligned}\mathscr{L}_r(x,z,\omega) &:= \mathscr{L}(\{x_v,\hat{\omega}_v\}_{v\in\mathscr{V}},\{\check{\omega}_v^w\}_{(v,w)\in\mathscr{E}},\{\check{\omega}_v^w\}_{(v,w)\in\mathscr{E}},\{\check{z}_v^w\}_{(v,w)\in\mathscr{E}},\{\check{z}_v^w\}_{(v,w)\in\mathscr{E}}) \\ &= \sum_{v\in\mathscr{V}}\left\{\frac{1}{2}\|y_v-\Phi_v x_v\|_2^2+\frac{\lambda}{J}\|x_v\|_1+\frac{\rho}{2}\sum_{w\in\mathscr{N}_v}\left[\|x_v-\check{z}_v^w\|_2^2+\|x_w-\check{z}_v^w\|_2^2\right]\right. \\ &\quad \left.+\sum_{w\in\mathscr{N}_v}\left[(\check{\omega}_v^w)^\top(x_v-\check{z}_v^w)+(\check{\omega}_v^w)^\top(x_w-\check{z}_v^w)\right]\right\} \\ &\quad +\sum_{v\in\mathscr{V}}\left[\hat{\omega}_v^\top(x_v-\gamma_v)+\frac{\rho}{2}\|x_v-\gamma_v\|_2^2\right] \end{aligned} \tag{5.15}$$

where $\{\omega_v^w\}_{(v,w)\in\mathscr{E}}$ are the Lagrange multipliers associated to the constraints. The constraints $\check{z}_v^w,\check{z}_v^w\in C_z=\{z:\check{z}_v^w=\check{z}_v^w,(v,w)\in\mathscr{E}\}$ have not been dualized, and $\rho>0$ is a preselected parameter.

5.2.1.2 Consensus-Based Sparsity Constrained Least Squares

We now consider the function $f(x):=\sum_{v\in\mathscr{V}}f_v(x)=\sum||y_v-\Phi_v x||_2^2$. If the data stored by all nodes (y_v,Φ_v) were available at once in a single central processing unit, an estimation of the signal $x^\star$ can be provided by the solution of the following optimization problem:

$$\min_{x\in\mathbb{R}^n} f(x) \qquad \text{s.t.} \quad x\in\Sigma_k, \tag{5.16}$$

A key property of the considered model is that the function f under the Assumption 1 is convex over canonical sparse subspaces, but they are not necessarily convex everywhere, as often assumed in literature of distributed optimization [17, 25–27]. Let us replace the global variable x in (5.16) with local variables $\{x_v\}_{v\in\mathscr{V}}$. We rewrite the distributed problem as follows:

$$\begin{aligned}&\min_{x_1,\ldots,x_J\in\mathbb{R}^n}\sum_{v\in\mathscr{V}}f_v(x_v), \\ &\text{s.t.}\begin{cases}x_v=x_w, & \forall w\in\mathscr{N}_v,\forall v\in\mathscr{V},\\ x_v\in\Sigma_k.\end{cases}\end{aligned} \tag{5.17}$$

In [14] the problem (5.5) is relaxed and the minimization of the following function is considered $\mathscr{F}:\mathbb{R}^{n\times J}\longmapsto\mathbb{R}^+$ defined as follows

$$\mathscr{F}(X) := \sum_{v\in\mathscr{V}} \left[f_v(x_v) + \frac{1}{4\tau(d+1)} \sum_{w\in\mathscr{N}_v} \|x_v - x_w\|^2 \right] \quad \text{s.t. } x_v \in \Sigma_k, \quad \forall v \in \mathscr{V} \tag{5.18}$$

where $X = (x_1, \ldots, x_J)$. By minimizing $\mathscr{F}$, each node seeks to estimate the sparse solution to (5.16) and to enforce agreement with the estimates calculated by other nodes in the network. It should also be noted that $\mathscr{F}(x1^\top) = f(x)$.

We discuss necessary optimality conditions of (5.18) and the relations to the original problem (5.16).

Definition 5.1 $Z \in \Sigma_k^J$ is called a τ-stationary point of (5.18) if it satisfies the following condition $\forall v \in \mathscr{V}$:

$$z_v = \sigma_k\left(\overline{z}_v - \tau\nabla f_v(z_v)\right).$$

The following proposition gives another representation of τ-stationary points. The proof is omitted for brevity but can be easily derived from Definition 5.1.

Proposition 5.4 *Z is a τ-stationary point of (5.18) if and only if $\forall v \in \mathscr{V}$ we have $z_v \in \Sigma_k$ and*

$$\begin{cases} \overline{z}_v^j - \tau\nabla_j f_v(z_v) = z_v^j & \text{if } j \in \operatorname{supp}(z_v) \\ |\overline{z}_v^j - \tau\nabla_j f_v(z_v)| \le r^k(z_v) & \text{if } j \notin \operatorname{supp}(z_v). \end{cases} \tag{5.19}$$

Theorem 5.3 *Suppose that Assumptions 1 and 2 hold.*

1. *If $Z^\star$ is an optimal solution of (5.18), then $Z^\star$ is a τ-stationary point for any $\tau < \frac{1}{(d+1)L}$ with $L = \max_{v\in\mathscr{V}} \|\Phi_v\|_2^2$;*
2. *Any τ-stationary point of (5.18) is a local minimum for (5.18).*

The first part of Theorem 5.3 proves that, under a suitable assumption, the τ-stationarity is only a necessary but not a sufficient condition for optimality. This means that τ-stationary points are only local minima of (5.18).

Let us state the relation between minimizers of (5.18) and of (5.16).

Theorem 5.4 *Let us denote as $\widehat{X}_v^\tau$ the minimizer of $\mathscr{F}(X)$ in (5.18). If $\mathscr{G}$ is connected, then $\lim_{\tau\to 0} \widehat{X}^\tau = x_\star 1^\top$, where $x_\star$ is the minimizer of (5.16).*

Theorem 5.4 guarantees that parameter τ can be interpreted as a temperature; as τ decreases, estimates x_v's associated with adjacent nodes become increasingly

correlated. This suggests that if τ is sufficiently small, then each vector $\widehat{x}_v^{\tau}$ can be used as an approximation of the optimal solution $x_\star$ of (5.16).

Before presenting distributed algorithms for sparse estimation problems, we recall that they are tailored for specific optimization problems in the sense that the dynamics will depend on the loss function we choose for the signal estimation. *The task of choosing the loss function is tricky and we show in next sections that different choices may define distributed algorithms with different features in terms of energy consumption, memory, and computational complexity.*

5.2.2 Communication and Processing Model

In the distributed algorithms for sparse recovery, each node in the network is assumed to update and share data at discrete times $\{T_t\}_{t\in\mathbb{N}}$. We label nodes' estimates and any other information at time T_t by index $t \in \mathbb{N}$ (e.g., we use $x_v(t) \in \mathbb{R}^n$ to denote the estimate provided by node $v \in \mathcal{V}$ at time T_t). The rules that govern these dynamics are generally based on the model proposed by Tsitsiklis et al. in [28, 29] for distributed computation. The set of neighbors of node v consists in the nodes w that are communicating directly with node v through the directed link (w, v). Then the node v combines its own estimate with the information received from the neighbors. We mean that a communication step has occurred if all nodes in the network receive the estimates from all the neighbors.

We can distinguish two classes of distributed algorithms:

- *synchronous algorithms* where the nodes are active at any time and are assumed to send, process information and perform computations synchronously. It is also assumed that there is no communication delay in the transmission of messages.
- *asynchronous algorithms* where the nodes are not required to be aware of a common clock and to communicate to the neighbors at each time instance. Moreover, nodes may be allowed to sleep for some of the time and to perform computations asynchronously.

The distributed methods can be also categorized into other two categories according to the number of nested loops they have: the *single-looped algorithms* and the *double-looped algorithms*.

- the *single-looped algorithms* have one communication step per iteration and, in each iteration, every node solves a local optimization problem;
- the *double-looped algorithms* consist of an inner loop and an outer loop. In each iteration of the inner loop, the nodes transmit data with their neighbors and multiple loops are needed to solve the local optimization problem.

The distributed algorithms we consider in this chapter can differ also for the updating rules. They can include:

- a rule on the weights that are used when a node combines its information with the information received from the neighbors.

- a connectivity rule ensuring the spread of information over the network.
- a rule on the frequency at which a node communicates its information to the neighbors.
- an assumption on the topology of the communication graph that can be static or time varying.

5.2.3 Distributed Algorithms for Lasso Estimation Problem

Algorithms for distributed sparse recovery (with no central processing unit) in sensor networks have been proposed in the literature in the last few years. In this section, we present different synchronization schemes that require that nodes periodically exchange estimations and perform computations. We distinguish three classes:

1. algorithms based on the decentralization of subgradient methods for convex optimization (DSM, [10, 25, 27]);
2. consensus and distributed ADMM ([11, 30, 31]);
3. distributed proximal-gradient methods (DISTA/DIHTA, [15]).

In particular, we review theoretical guarantees of convergence and we compare them in terms of performance, complexity, and memory usage.

5.2.3.1 Distributed Subgradient Method

Most optimization algorithms developed for solving problem (5.6) are first-order methods, i.e., methods that use gradient or subgradient information of the local objective functions. These algorithms are computationally cheap and naturally lead to distributed implementations over networks [10, 27]. The main idea behind distributed algorithms is to use consensus as a mechanism for spreading information through the network. In particular, each node, starting from an initial estimate, say $x_v(0) = 0$ for all $v \in \mathscr{V}$, updates it by first combining the estimates received from its neighbors, then moving in the direction of a negative subgradient step, in order to minimize its objective function. More formally,

$$x_v(t+1) = \sum_{w \in \mathscr{V}} P_{v,w} x_w + r_v(t) \Phi_v^\top (y_v - \Phi_v x_v(t)) - r_v(t) \alpha \operatorname{sgn}(x_v(t)) \qquad (5.20)$$

where $P_{v,w} = 1/(d+1)$ if $(v, w) \in \mathscr{E}$ and $P_{v,w} = 0$ otherwise. In this setting, the vector $x_v(t)$ is the estimate of an optimal solution of the problem (5.6) provided by node $v \in \mathscr{V}$ at the time t.

The distributed subgradient method, tabulated in Algorithm 1, is a simple algorithm. Although this method looks like the distributed gradient method for differentiable convex functions, it is not a descent method of a consensus-based loss function.

Algorithm 1 DSM-Lasso

Input: (y_v, Φ_v), $\alpha = \lambda/J$
$x_v(0) = x_0$, iterate
for $t \in \{1, 2, \ldots\}$ **do**
Transmit $x_v(t-1)$ to neighbors in $\mathcal{N}_v$
Update $x_v(t)$ according to (5.20)
end for

The subgradient method was originally developed in [28]. We refer to [10, 26, 29] for an overview of distributed subgradient methods.

Several different step size rules are adopted in literature for $r_v(t)$. We distinguish the following choices:

- constant step size $r_v(t) = r$ for all $v \in \mathcal{V}$, i.e., independent of the time t (see [10]);
- square summable but not summable sequence $\{r_v(t)\}_{t\in\mathbb{N}}$, i.e.,

$$r_v(t) \in (0, 1), \quad \sum_{t=0}^{\infty} r_v(t)^2 < \infty, \quad \sum_{t=0}^{\infty} r_v(t) = \infty$$

  (for example $r_v(t) = a/(b + t)$, with $a \geq 0$ and $b \geq 0$, see [17]);
- nonsummable diminishing step size rules, i.e., the sequence $\{r_v(t)\}_{t\in\mathbb{N}}$, is such that

$$r_v(t) \in (0, 1), \quad \lim_{t\to\infty} r_v(t) = 0, \quad \sum_{t=0}^{\infty} r_v(t) = \infty.$$

  A typical example is $r_v(t) = a/\sqrt{t}$, with $a > 0$.

The protocol in (5.20) with constant stepsize $r_t = r$ is not guaranteed to converge and can (and often does) oscillate. The convergence of (5.20) can be achieved by considering the related "stopped" model (see page 56 in [26]), whereby the nodes stop computing the subgradient at some time, but they keep exchanging their information and averaging their estimates only with neighboring messages for subsequent time. However, the tricky point of such techniques is the optimal choice of the number of iterations to stop the computation of the subgradient. Moreover, the final point has not a variational characterization and the accuracy depends on the time the computation on the gradient is stopped.

In [25] a distributed version of DSM is proposed for the minimization of a separable convex function over a convex set. It is worth noticing that these types of algorithms cannot be applied to LS problems under sparsity constraints (5.16), whose feasible set is not convex.

The diminishing stepsize rules allow to force the convergence to a consensus but requires to fix an initial time and is not feasible in case of time-variant input: introducing a new input would require some resynchronization.

5.2.3.2 Alternating Direction Method of Multipliers

The consensus ADMM [11] is a method for solving problems in which the objective and the constraints are distributed across multiple processors. The problem in (5.6) is tackled by introducing dual variables ω_v and minimizing the augmented Lagrangian (5.13) in a iterative way with respect to the primal and dual variables. More recently, distributed versions of ADMM that just require local communications have been proposed in the literature [9] for solving Lasso problems.

The ADMM for quadratic programming distributed Lasso is an iterative algorithm that include three steps per iteration. The augmented function (5.12) is considered and then

1. it is minimized with respect to $\{x_v\}_{v\in\mathscr{V}}$ variables, considering the variables $\{\omega_{(v,w)}, z_{(v,w)}\}_{(v,w)\in\mathscr{E}}$ as fixed parameters;
2. it is minimized with respect to $\{z_{v,w}\}_{v\in\mathscr{E}}$ variables, considering the other variables fixed;
3. finally, the Lagrange multipliers $\{\omega_{v,w}\}_{v\in\mathscr{E}}$ are updated via gradient one step of the ascent algorithm [24].

It can be shown [9] that, if the Lagrange multipliers are initialized to zero, the dynamics of ADMM can be simplified and each node does not need to keep track of all multipliers $\{\omega_{(v,w)}\}_{(v,w)\in\mathscr{E}}$, but only to update an aggregate vector $\{\mu_v(t)\}$. Summarizing, each node stores two n-dimensional vectors $\{\mu_v(t), x_v(t)\}$ and transmits at each iteration, the local estimates $\{x_v(t)\}$ to the neighbors. Starting from $x_v(0) = 0$ and $\mu_v(0) = 0$, all nodes $v \in \mathscr{V}$ perform the following operations:

$$\mu_v(t+1) = \mu_v(t) + \rho \sum_{w\in\mathscr{N}_v} (x_v(t) - x_w(t)) \tag{5.21}$$

$$x_v(t+1) = \underset{x_v}{\operatorname{argmin}} \left\{ \frac{1}{2}\|y_v - \Phi_v x_v\|_2^2 + \frac{\lambda}{J}\|x_v\|_1 + \mu_v^\top(t+1)x_v + \rho \sum_{w\in\mathscr{N}_v} \left\| x_v - \frac{x_v(t) + x_w(t)}{2} \right\|_2^2 \right\} \tag{5.22}$$

The algorithm is tabulated in Algorithm 2. The local optimization in (5.22) is well defined and can be solved using standard optimization for QP problems [24].

Algorithm 2 DQP-Lasso

Input: (y_v, Φ_v), $\alpha = \lambda/J > 0$
Initialization: $x_v(0) = 0$, and $\mu_v(0) = 0$, iterate
for $t \in \{1, 2, \ldots\}$ **do**
Transmit $x_v(t-1)$ to neighbors in $\mathscr{N}_v$
Update $\mu_v(t)$ according to (5.21)
Update $x_v(t)$ according to (5.22)
end for

As proved in [9], Algorithm 2 produces for all $v \in \mathscr{V}$ a sequence of local estimates $\{x_v(t)\}_{t\in\mathbb{N}}$ converging to the Lasso solution as $t \to \infty$.

Theorem 5.5 *If $\mathscr{G}$ is connected, then for any value of the penalty coefficient $\rho > 0$ the Algorithm 2 generates sequences $\{x_v(t)\}_{t\in\mathbb{N}}$ for every $v \in \mathscr{V}$ such that*

$$\lim_{t\to\infty} x_v(t) = x_{\text{Lasso}}$$

Although Theorem 5.5 guarantees the convergence of the algorithm, the computation at each step of (5.22) has polynomial complexity in the signal length n. However, cost of the nodes is a hard constraint that has to be taken into account when designing a distributed algorithm. For example, in a sensor network, due to limitation on battery life of sensors, it is necessary to reduce energy consumption as much as possible and, consequently, to reduce the complexity of the operations performed locally. In [9] different techniques to reduce the computational complexity of (5.22) are proposed. For example, coordinate descent algorithms can be used to reduce the operational complexity if the Lasso-type subproblem in (5.22). However, these methods can not be run in parallel and need the nodes to update the estimate in a cyclic order. This requirement makes the algorithm less flexible.

Another version of the DQP-Lasso is proposed in [9] that yields local updates in closed-form. This algorithm is developed to solve the optimization in (5.14) and entails the following steps for each $t \in \mathbb{N}$: each node transmits the local estimate to the neighbors that use them to evaluate the dual price vector and calculate the new estimate via coordinate descent and thresholding operations. Formally, choosing some $\rho > 0$, the update equations would typically be performed according to the following rules:

$$\mu_v(t+1) = \mu_v(t) + \rho \sum_{w\in\mathscr{N}_v} (x_v(t) - x_w(t)) \tag{5.23}$$

$$\omega_v(t+1) = \omega_v(t) + \rho(x_v(t) - z_v(t)) \tag{5.24}$$

$$x_v(t+1) = \frac{1}{\rho(2d+1)} \eta_{\frac{\lambda}{J}} (\rho z_v(t) - \mu_v(t+1) - \omega_v(t+1) + \rho \sum_{w\in\mathscr{N}_w} (x_v(t) + x_w(t))) \tag{5.25}$$

$$z_v(t+1) = [\rho I + \Phi_v^\top \Phi_v]^{-1} (\Phi_v^\top y_v + \rho x_v(t+1) + \omega_v(t+1)) \tag{5.26}$$

Referring to [9] for the detailed derivation, the ADMM algorithm for the optimization of the augmented Lagrangian function in (5.15) is tabulated in Algorithm 3.

The convergence of Algorithm 3 is stated in the following theorem and rigorously proved in [9].

Algorithm 3 D-Lasso

Input: (y_v, Φ_v)
Initialization: $x_v(0) = 0$, $\mu_v(0) = 0$, iterate
for $t \in \{1, 2, \ldots\}$ **do**
Transmit $x_v(t-1)$ to neighbors in $\mathcal{N}_v$
Update $\mu_v(t)$ according to (5.23)
Update $\omega_v(t)$ according to (5.24)
Update $x_v(t)$ according to (5.25)
Update $z_v(t)$ according to (5.26)
end for

Theorem 5.6 *If $\mathcal{G}$ is connected, then for any value of the penalty coefficient $\rho > 0$ the Algorithm 3 generates sequences $\{x_v(t)\}_{t\in\mathbb{N}}$ for every $v \in \mathcal{V}$ such that*

$$\lim_{t\to\infty} x_v(t) = x_{\text{Lasso}}$$

5.2.3.3 Distributed Iterative Soft Thresholding Algorithm (DISTA)

In [15] *distributed iterative thresholding algorithms* are proposed as low complexity techniques that require very little memory, while achieving performance close to that of the ADMM estimation [9]. These algorithms build on the seminal work of Daubechies et al. and are related to the literature on distributed computation and estimation, which has attracted recent interest in the scientific community [10, 25, 27], and whose main goal is to design distributed iterative algorithms to cooperatively minimize (5.8) in an iterative, distributed way. The key idea is as follows.

The algorithm is parametrized by a stochastic transition matrix P which is adapted to the graph. All nodes v store two messages at each time $t \in \mathbb{N}$, $x_v(t)$ and $\overline{x}_v(t)$. Starting from $x_v(0) = x_0$ for all $v \in \mathcal{V}$, the update is performed in an alternating fashion. More specifically, the update consists of two stages; for convenience, the first stage is identified with even times $t \in 2\mathbb{N}$, and the second stage with odd times $t \in 2\mathbb{N} + 1$, so that one complete iteration spans two time units. At even time $t \in 2\mathbb{N}$, each node $v \in \mathcal{V}$ receives the estimates $x_w(t)$ for each $w \in \mathcal{N}_v$, which is communicating with v, and $\overline{x}_v(t+1)$ is obtained by a convex combination of these estimates. At odd time $t \in 2\mathbb{N} + 1$, each node receives the vectors $\overline{x}_w(t)$ from their neighbors, the estimate $x_v(t+1)$ is then obtained applying the thresholding operator to a convex combination of the received messages and of the subgradient step. The coefficients of the convex combination are obtained through the matrix P and the temperature parameter $q \in (0, 1)$. For simplicity, the nodes compute simply the average of the received messages giving equal weight to each of them,

i.e., $P_{v,w} = 1/(d+1)$ if $(v,w) \in \mathscr{E}$ and $P_{v,w} = 0$ otherwise, and setting $q = 1/2$. The thresholding operation is soft as described in (5.10).

Starting from $x_v(0) = x_0$, all nodes perform the following operations:

- $t \in 2\mathbb{N}$, $v \in \mathscr{V}$,

$$\overline{x}_v(t+1) = \sum_{w \in \mathscr{V}} P_{v,w} x_w(t), \qquad x_v(t+1) = x_v(t)$$

- $t \in 2\mathbb{N}+1$, $v \in \mathscr{V}$,

$$\overline{x}_v(t+1) = \overline{x}_v(t),$$
$$x_v(t+1) = \eta_\alpha \left[(1-q) \sum_{w \in \mathscr{V}} P_{v,w} \overline{x}_w(t) + q\Big(x_v(t) + \tau \Phi_v^\mathsf{T}(y_v - \Phi_v x_v(t))\Big) \right].$$

In other words, in each iteration the nodes transmit data with their neighbors and the double-stage consensus brings the estimates of the nodes close together before performing the inexact centralized proximal-gradient method.

The proposed method defines a distributed protocol: each node only needs to be aware of its neighbors and no further information about the network topology is required. It should be noted that if $J = 1$, DISTA coincide with IST (see Algorithm 2).

The dynamics of the algorithm can be rewritten as follows: Let

$$X(t) = (x_1(t), \ldots, x_J(t)), \quad \overline{X}(t) = X(t)P^\top, \quad t \in \mathbb{N}.$$

The updates of DISTA can thus be rewritten as

$$X(t+1) = \Gamma X(t) \tag{5.27}$$

Notice that this recursive formula joins in one step the operations that was split into two steps previously, but the dynamics is actually the same. $X(0)$ can be arbitrarily initialized. More precisely, the pattern is described in Algorithm 4.

Algorithm 4 DISTA

Input: (y_v, Φ_v), $\alpha = q\lambda\tau/J > 0$, $\tau > 0$
$x_v(0) = 0$, iterate
for $t \in \mathbb{N}$, $v \in \mathscr{V}$ **do**
Transmit $x_v(t-1)$ to neighbors in $\mathscr{N}_v$
Update $\overline{x}_v(t-1) = \sum_{w \in \mathscr{V}} P_{v,w} x_w(t)$
Transmit $\overline{x}_v(t-1)$ to neighbors in $\mathscr{N}_v$
Update $x_v(t) = \eta_{1,\alpha}[(1-q)\sum_{w \in \mathscr{V}} P_{v,w}\overline{x}_w(t) + q(x_v(t) + \tau\Phi_v^\mathsf{T}(y_v - \Phi_v x_v(t)))]$
end for

Theorem 5.1 guarantees that, under suitable conditions, the minima of the cost functions defined in (5.8) coincide with the fixed points of the map that rules the dynamics of DISTA. In this section, we present the theoretical results regarding the convergence of the proposed algorithm. More precisely, Theorem 4 defines sufficient conditions to guarantee the convergence of DISTA to a fixed point. It follows that DISTA converges to a minimum of (5.8).

Theorem 5.7 (DISTA convergence) *If* $\tau < \|\Phi_v\|_2^{-2}$ *for all* $v \in \mathscr{V}$, *DISTA produces a sequence* $\{X(t)\}_{t\in\mathbb{N}}$ *such that*

$$\lim_{t\to\infty} \left\| X(t) - X^\star \right\|_F = 0$$

where $X^\star \in \mathrm{Fix}(\Gamma)$.

It is worth mentioning that Theorem 5.7 does not imply necessarily the convergence of the algorithm to a consensus. The local estimates do not necessary coincide at convergence. However, the disagreement among the nodes is controlled by temperature parameter q. The consensus can be achieved by letting q go to zero as suggested by Theorem 5.2 or considering the related "stopped" model [10], whereby the nodes stop computing the subgradient at some time, but they keep exchanging their information and averaging their estimates only with neighboring messages for subsequent time. It should be noted that in the literature several consensus-based estimation algorithms have been proposed, which do not reach consensus but where the agreement can be induced by using a temperature parameter, e.g., [25, 32].

DISTA can be interpreted as a majorization-minimization algorithm. In fact, in order to minimize $\mathscr{F}(x_1, \ldots, x_J)$ in (5.8), a surrogate objective function [33] is considered:

$$\mathscr{F}^{\mathscr{S}}(X, C, B) := \frac{1}{2} \sum_{v\in\mathscr{V}} \Bigg(q \|\Phi_v x_v - y_v\|_2^2 + \frac{2\lambda}{J} \|x_v\|_1 + \frac{1-q}{(d+1)\tau} \sum_{w\in\mathscr{N}_v} \|x_v - c_w\|_2^2 + \frac{q}{\tau} \|x_v - b_v\|_2^2 - q \|\Phi_v(x_v - b_v)\|_2^2 \Bigg) \tag{5.28}$$

where $C = (c_1, \ldots, c_N) \in \mathbb{R}^{n\times N}$, $B = (b_1, \ldots, b_N) \in \mathbb{R}^{n\times N}$.

It should be noted that, defining $\overline{X} = XP^\mathsf{T}$,

$$\mathscr{F}^{\mathscr{S}}(X, \overline{X}, X) = \mathscr{F}(X)$$

and that if $\tau < \|\Phi_v\|_2^{-2}$ for all $v \in \mathscr{V}$ then this surrogate objective function is a majorization of $\mathscr{F}$ [34]. The optimization of (5.28) can be computed by minimizing with respect to each x_v separately.

The proof is rather technical and is based on the fact that DISTA produces sequences that do not increase the cost function $\mathscr{F}$:

$$\mathscr{F}(X(t+1)) \leq \mathscr{F}^{\mathscr{S}}(X(t+1), \overline{X}(t+1), X(t)) \leq \mathscr{F}^{\mathscr{S}}(X(t+1), \overline{X}(t), X(t)) \leq \mathscr{F}(X(t)).$$

5.2.3.4 Discussion

In this section, we compare the algorithms for distributed Lasso in terms of complexity, communication, memory requirements, and accuracy level.

In particular, we show that, compared to DSM-Lasso, DISTA requires equal memory storage and communication cost but is extremely faster. This implies that the total number of communications is minimized. Indeed, DISTA is only slightly suboptimal with respect to D-Lasso or DQP-Lasso. On the other hand, compared to D-Lasso, it requires a much lower memory usage and reduce the computationally complexity of DQP-Lasso. These features make DISTA suitable also for low-energy environments such as wireless sensor networks. For what concerns communication, all the proposed algorithms are comparable. A detailed discussion is given below.

Memory Requirements

DSM-Lasso, DQP-Lasso, and DISTA require to store a similar amount of real values. More precisely, we need to store $O(n)$ real values. In D-Lasso, instead, the inversion of the $n \times n$ matrices $(\Phi_v^\top \Phi_v + \rho I)$ represents a bottleneck for the implementation. Although the inversion of the matrices can be performed offline, the storage of that matrix may be prohibitive for a low-power node with a small amount of available memory. Specifically, for the D-Lasso each node has to store $O(n^2)$ real values. Just as a practical example, nodes with 16kB of RAM are widely used for wireless sensor networks, e.g. as STM32 F3-series microcontrollers with Contiki operating system. As the static memory occupied by D-Lasso and DISTA is almost the same, we neglect it along with the memory used by the operating system (the total is of the order of hundreds of byte). Using a single-precision floating-point format, 2^{12} real values can be stored in 16 kB. Therefore, even assuming just one measurement per node ($m = 1$), D-Lasso can handle signals with length of some tens, while DISTA and DSM-Lasso up to thousands.

Communication Cost and Computational Complexity

Let us consider a complete network with $J = 10$ sensors. Each sensor acquires a sparse signal, represented in vector form as $x^\star \in \mathbb{R}^{150}$. The sampling is performed at a rate below the Nyquist rate, using random linear projections as suggested by the CS theory [35]. One can represent the measurements $y_v \in \mathbb{R}^{10}$ as

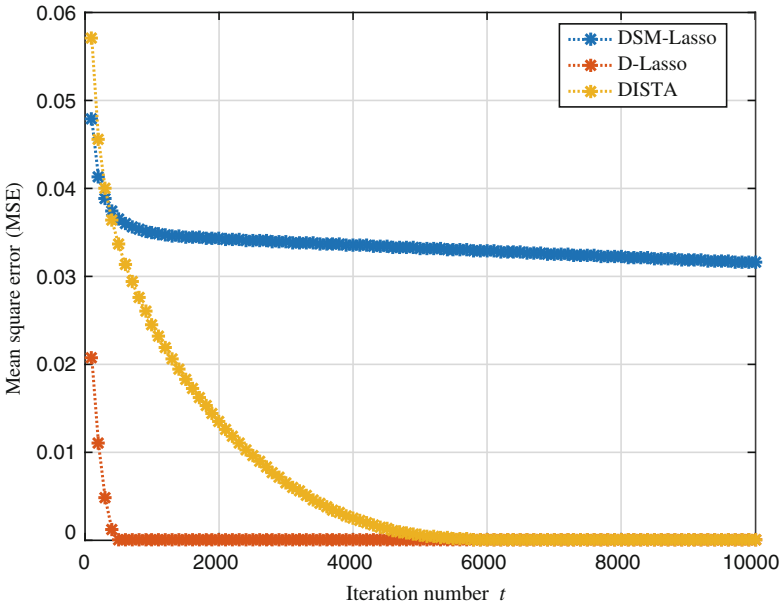

Fig. 5.1 Evolution of the mean-square error as a function of the iterations

$$y_v = \Phi_v x^\star \qquad v \in \{1, ..., 10\}.$$

For purpose of illustration, the signal to be recovered is generated choosing $k = 15$ nonzero components uniformly at random among the $n = 150$ elements and drawing the amplitude of each nonzero component from a Gaussian distribution $\mathsf{N}(0, 1)$. The sensing matrices $(\Phi_v)_{v\in\{1,...,10\}}$ are sampled from the Gaussian ensemble with 10 rows, 150 columns, zero mean and variance $\frac{1}{10}$. The DISTA parameters are set equal to $\lambda = 0.01$ $\tau = 0.1$ and $q = 1/2$; the parameter in the implementation of D-Lasso is $\rho = 0.1$.

In Fig. 5.1, we compare the time of convergence: as known, DSM has problems of slowness and thousands of steps are not sufficient to converge. DISTA, instead, is significantly faster, hence feasible. It does not reach the quickness of D-Lasso, which however has the price of the inversion of a $n \times n$ matrix at each node to start the algorithm.

The number of iterations for convergence is not sufficient to compare the different algorithms and we need to take into account the number of transmitted values per iteration and the number of computations at each time step.

DSM-Lasso, DQP-Lasso, and D-Lasso require that each node transmits at each communication step n real values to its neighbors. In DISTA, instead, the size of the transmitted vector is $2n$. As will be shown below, the DSM-Lasso is extremely slow and it turns out to have a higher communication cost if compared to DISTA and to DQP-Lasso and D-Lasso. It should also be noted that at each iteration DQP-Lasso

and D-Lasso are much more complex since at each iteration (5.22) has polynomial complexity $O(n^3)$. In this sense, DISTA combines a cost that is linear in n and linear updates of the variables. Moreover, it can be shown that the resulting DISTA strategy provably converges starting from any initial condition, and therefore does not require any specific procedure for initialization as it happens in ADMM versions (DQP-Lasso and D-Lasso) of distributed Lasso. It is worth remarking that the amount of data transmitted at every communication step might decrease with the iterations. In fact, since the signal $x^\star$ is sparse, the estimates provided by the nodes tend to be sparse, allowing a possible compression of the transmitted data (e.g., just transmit the nonzero entries).

The interested reader can refer to [15] for a more detailed discussion on communication cost and complexity.

Performance: Centralized Versus Distributed Reconstruction

Let us consider a network of interconnected sensors modeled as a graph $\mathscr{G} = (\mathscr{V}, \mathscr{E})$. Each sensor acquires a sparse signal, represented in vector form as $x \in \mathbb{R}^n$. The sampling is performed at a rate below the Nyquist rate, using random linear projections as suggested by the CS theory [35]. One can represent the measurements $y_v \in \mathbb{R}^m$ (with $m \ll n$) as

$$y_v = \Phi_v x^\star.$$

The asymptotic properties of the centralized Lasso estimator have been studied in different papers. These include the bias, the weak support consistency, and the estimation consistency.

DSM-Lasso algorithm tabulated in (5.20) with constant stepsize $r_t = r$ is not guaranteed to converge and can oscillate. The convergence of (5.20) can be achieved by considering the related "stopped" model (see page 56 in [26]), whereby the nodes stop computing the subgradient at some time. As already noted, the final point has not a variational characterization and the accuracy depends on the time the computation on the gradient is stopped. On the other hand, from Theorem 5.5 and 5.6, distributed implementation of ADMM for Lasso are guaranteed to converge to the Lasso solution x_{Lasso} of (5.6). It can be deduced that all asymptotic properties of centralized Lasso carry over to its distributed counterpart DQP-Lasso and D-Lasso. In this paragraph, we show that DISTA achieves extremely good performance and there is no significant loss compared to the centralized Lasso. Extensive simulations show that DISTA is satisfactory in the following sense: when the product of the number of nodes in the network times the number of data for each unit exceeds a given threshold, accurate estimation is achieved. Moreover, the total number of available data required for the estimation is comparable to that required by joint estimation. This implies that decentralization is not a drawback.

For purpose of illustration, the signal to be recovered is generated choosing k nonzero components uniformly at random among the $n = 150$ elements and drawing the amplitude of each nonzero component from a Gaussian distribution $\mathsf{N}(0, 1)$.

The sensing matrices $(\Phi_v)_{v\in\mathscr{V}}$ are sampled from the Gaussian ensemble with m rows, n columns, zero mean, and variance $\frac{1}{m}$. This is known to be a suitable choice from compressed sensing theory [35]. The parameters λ and τ are set equal to 0.01 $\tau = 0.1$ and $q = 1/2$, respectively. We now conduct a series of experiments for different architectures and for a variety of total number of measurements. Here, we are interested in the performance of the algorithms as a function of the number of data to store in the memory, which we try to minimize, and of the size of the graph. For each n, we vary the number of measurements m per node and the number of nodes in the network. For each (n, m, J) 3-tuple, we repeat the following procedures 50 times.

The measurements are then taken according to the model in (5.1). We use the so-called metropolis random walk construction for P (see [36]) which amounts to the following: if $i \neq j$,

$$P_{ij} = \begin{cases} 0 & \text{if } (i,j) \notin \mathscr{E} \\ (\max\{\deg(i)+1, \deg(j)+1\})^{-1} & \text{if } (i,j) \in \mathscr{E} \end{cases}$$

where $\deg(i)$ denotes the degree (the number of neighbors) of unit i in the graph $\mathscr{G}$. In particular, we consider the following topologies:

1. *Complete graph*: $P_{ij} = \frac{1}{J}$ for every $i, j = 1, \ldots, J$.
2. *Ring graph*: J sensors are disposed on a circle, and each node communicates with its first neighbor on each side (left and right). The corresponding circulant symmetric matrix P is given by $P_{ij} = \frac{1}{3}$ for every $i = 2, \ldots, J-1$, and $j \in \{i-1, i, i+1\}$; $P_{11} = P_{12} = P_{1J} = \frac{1}{3}$; $P_{J1} = P_{JJ-1} = P_{JJ} = \frac{1}{3}$; $P_{ij} = 0$ elsewhere.
3. *Random geometric graph*: sensors are assumed to be deployed uniformly at random in a square $[0, 1] \times [0, 1]$, and communication is allowed between sensors with distance below a certain radius (here we fix the radius to 0.75).

We define a success the case when

$$\sum_{v\in\mathscr{V}} \|x^\star - x_v^{DISTA}\|_2^2 / (nJ) < 10^{-4}$$

where $x^\star$ is the original signal to be recovered and x_v^{DISTA} is the estimate a provided by node v by using DISTA. In Figs. 5.2, 5.3, and 5.4 the probability of success is depicted as a function of the number of measurements over complete, ring, and random geometric (radius 0.75) topologies, respectively.

The color of the cell in the figures reflects the empirical success rate: white denotes perfect reconstruction in all the experiments, while black represents no success occurrence. It should be noted that the number of total measurements mJ which are sufficient for successful estimation is constant: the red curve collects the points $(m, |\mathscr{V}|)$ such that $m|\mathscr{V}| = 70$, which turns out to be a sufficient value to obtain good estimation (probability greater then 0.95). We observe that the performance of DISTA is not strongly affected by the graph topology. One could expect worse results with

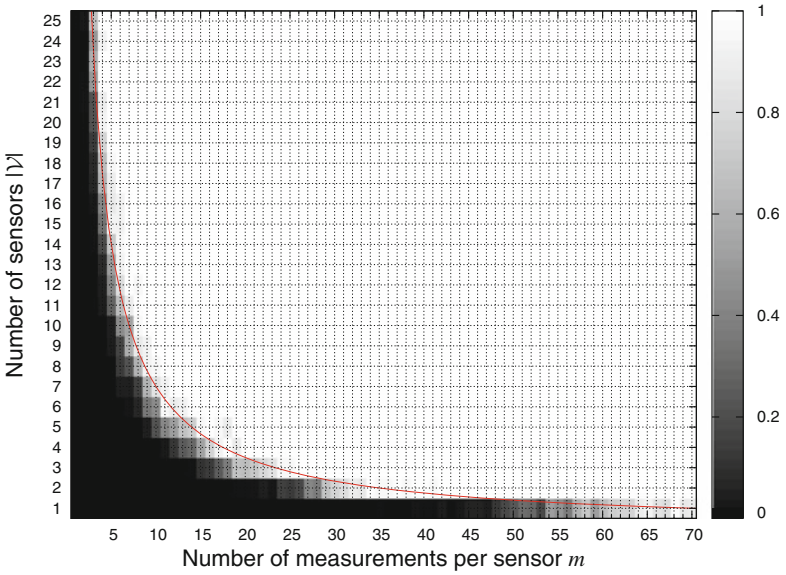

Fig. 5.2 DISTA performance (noise-free case, $n = 150$, $k = 15$): probability of success over a complete graph

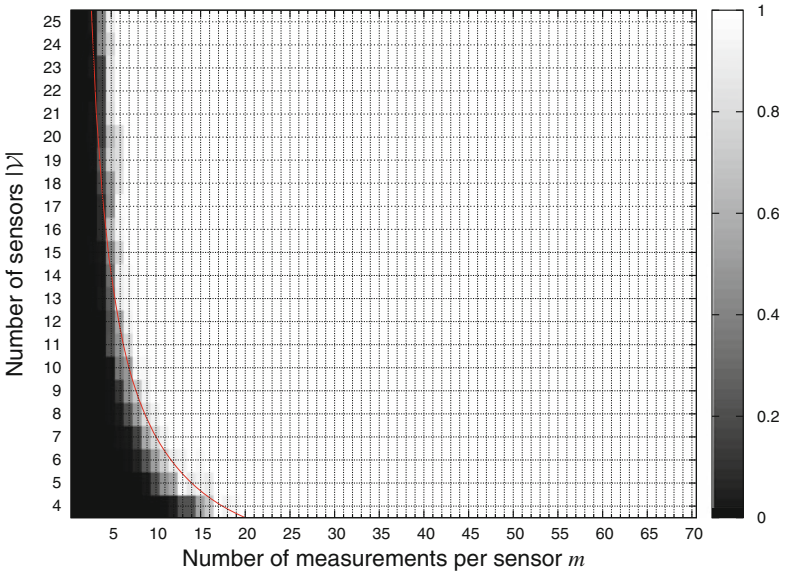

Fig. 5.3 DISTA performance (noise-free case, $n = 150$, $k = 15$): probability of success over a ring graph

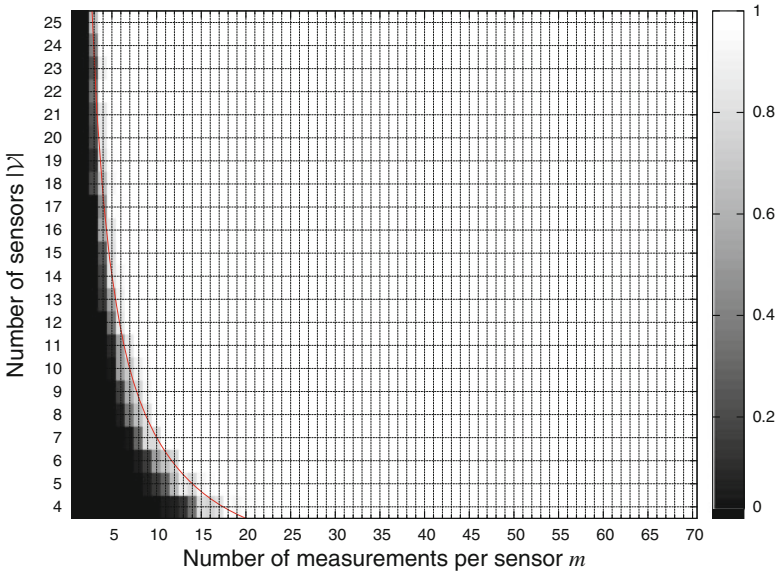

Fig. 5.4 DISTA performance (noise-free case, $n = 150$, $k = 15$): probability of success over a random geometric graph (radius 0.75)

less-connected graphs, since low connectivity may cause problems of scarce communication and slowness. However, our results show that only a slight degradation affects DISTA over the ring, while the behavior is very similar for the complete and random geometric graph. This also highlights that no loss occurs due to nonregularity of the graph.

In Fig. 5.5 the probability of success of DISTA, D-Lasso, and DSM-Lasso are compared as a function of the number of measurements per node. The curves are depicted for different numbers of sensors. We notice that the number of measurements needed for success by DISTA is smaller with respect to DSM. On the other hand, DISTA has performance close to the optimal ADMM: we obtain an almost perfect match in the curves obtained in the same scenario.

5.2.4 *Energy Saving Algorithms: Distributed Sparsity Constrained Least Squares*

Although the algorithms presented in the previous section provide good estimates of the unknown signal in terms of the MSE, they are not optimized for metrics that are relevant in distributed scenarios. Leaving aside the memory occupancy that can be critical in sensor networks, [8] shows that D-Lasso outperforms the other algorithms in terms of transmitted values. However, as already noted, the nodes transmit at each

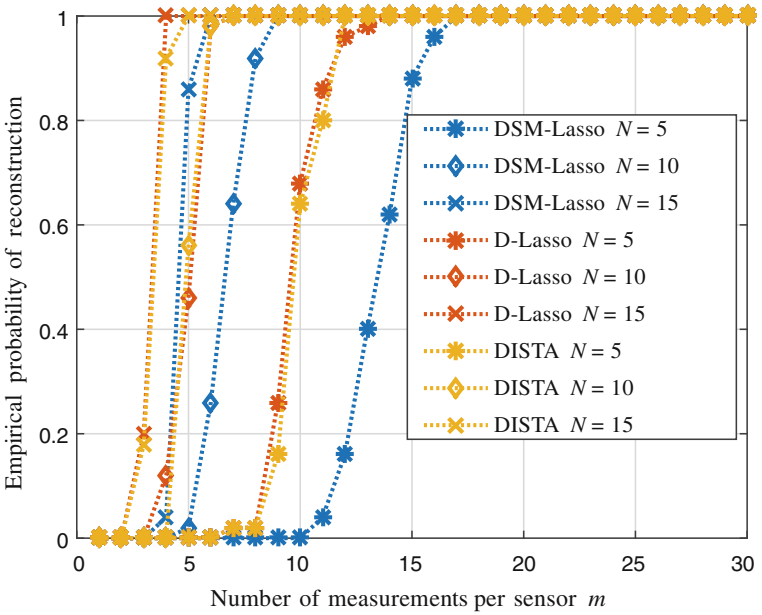

Fig. 5.5 Probability of success of DSM, D-Lasso, and DISTA over a complete graph, $n = 150$, $k = 15$

iteration an n-vector to every neighbor. These vectors may not be sparse for many iterations and, consequently, the energy consumption can be very high. In this section, we review alternative approach based on the sparsity constrained least squares that reduce energy consumption.

5.2.4.1 Distributed Iterative Hard Thresholding

Distributed approaches to problems of kind (5.16) have drawn much attention recently. In particular, [20] addresses (5.16) over static and time-varying networks and proposes two protocols based on Iterative Hard Thresholding (IHT).

The first method is distributed iterative hard thresholding (DIHT) that can be outlined as follows. Given a network, every node stores an identical copy of the signal estimate $x(t)$. Then one node r is chosen and a spanning tree is built fixing r as root; then, iterative communication is activated from the root toward the leaves and vice versa so that

(a) r broadcasts the estimate of the signal;
(b) the other nodes receive this estimate and transmit back information to r to update it.

More precisely, given the current estimate $x(t)$, each node v receives it and computes

$$z_v(t) = \nabla f_v(x(t)) = -\Phi_v^\top (y_v - \Phi_v x(t)). \tag{5.29}$$

Then, starting from the leaves the n-vectors $z_v(t)$ are sent back and accumulated so that r receives their sum and use it (along with $z_r(t)$) to update $x(t)$ to

$$x(t+1) = \sigma_k \left(x(t) - \tau \sum_{v=1}^{J} z_v(t) \right) \tag{5.30}$$

through hard thresholding. Pseudocode for DIHT is tabulated in Algorithm 5.

Algorithm 5 DIHT

Algorithm executed by root r

Input: (y_r, Φ_r)
  Initialize $x(0) = x_0$
  for $t \in \{1, 2, \ldots\}$ **do**
    Update $z_r(t) = -\Phi_r^\top (y_r - \Phi_r x(t-1))$
    Transmit $x(t)$ to children $v \in \mathscr{C}(r)$
    Receive $s_v(t)$ from each $v \in \mathscr{C}(r)$
    Update $s_r(t) = \sum_{v \in \mathscr{C}_r} s_v(t) + z_r(t)$
    Update $x(t) = \sigma_k \left(x(t) - \tau s_r(t) \right)$
  end for

Algorithm executed by nodes $v \neq r$

Input: (y_v, Φ_v), iterate
  for $t \in \{1, 2, \ldots\}$ **do**
    Receive $x(t-1)$ from parent
    Update $x(t) = x(t-1)$
    Update $z_v(t) = -\Phi_v^\top (y_v - \Phi_v x(t))$
    Transmit $x(t)$ to children $\ell \in \mathscr{C}(v)$
    Receive $s_w(t-1)$ from children $w \in \mathscr{C}(v)$
    Update $s_v(t) = \sum_{w \in \mathscr{C}_v} s_w(t) + z_v(t)$
    Transmit $s_v(t)$ to parent
  end for

The estimate provided by the network $x(t)$ evolves in the same way as the estimate given by the centralized IHT in Algorithm (1). Thus, DIHT has the same convergence guarantees as the centralized IHT. In [20], this method has been shown to work efficiently and overcome other distributed methods in terms of convergence times and number of required transmitted values. Its main limitation is in the imposed hierarchy, specifically, an exclusion of node r (due, for example, to a failure) would seriously disrupt the process, as it is the only node storing all the necessary information to pursue the recovery.

The necessity of a spanning tree is removed in the second method proposed in [20], known as CB-DIHT and based on diffusive consensus. In CB-DIHT, instead of summing the gradients from the leaves to the root of a spanning tree, local means of the gradients are computed by each node as in consensus procedures. In order to do this, all the nodes should receive from a prescribed node r the current estimate $x(t)$

through a diffusive procedure. This idea can be used even when the topology of the network is time varying, provided that some connectivity conditions are respected. Interestingly, CB-DIHT is identical to IHT except that at each iteration the gradient is approximated [20, Proposition 4.4].

5.2.4.2 Asynchronous Hard Thresholding, Broadcast Hard Thresholding, and Gossip Hard Thresholding

We now describe another family of distributed algorithms, all based on the idea that each node performs an IHT reconstruction procedure, but adjusts its own estimate based on knowledge of the estimates of its neighbors. The communication protocols can be of different kind, leading to a variety of different algorithms. Here, we study three cases that we name asynchronous hard thresholding (AHT), broadcast hard thresholding (BHT), and gossip hard thresholding (GHT). These methods are iterative, hence stopping criteria should be defined. As stopping criteria is not explicitly specified but the algorithm can be stopped when numerical convergence is achieved, that is, when the distance between the estimates of two successive iterates is below a fixed threshold.

The AHT algorithm is reported in Algorithm 6.

Algorithm 6 AHT

Input: Data (y_v, Φ_v), sparsity guess k
1: Initialization: $x_v(0) = 0$, $\tau > 0$
2: **for** $t = 0, 1, \ldots, StopIter$ **do**
3: Selection of $v \in \mathcal{V}$
4: $x_v(t+1) = \sigma_k \left[\frac{1}{d+1} \sum_{w \in \mathcal{N}_v \cup \{v\}} x_w(t) + \tau \Phi_v^\top (y_v - \Phi_v x_v(t)) \right]$
5: $x_h(t+1) = x_h(t)$ for any $h \neq v$
6: **end for**

At each iteration step, a node is selected and communicates with its neighbors to receive their estimates. After communication, the selected node performs the following operations:

(a) gradient computation of its loss function;
(b) average of the received neighbors' estimates (included itself);
(c) combined gradient step using (a) and (b);
(d) best k-term approximation of (c).

The procedure is iterated until a stopping criterion is met. In Algorithm 6, the pseudocode is reported. An example of a network with four nodes is presented in Fig. 5.6: when node 3 is initially selected, it receives the estimates from its neighbors and updates its estimate.

In the BHT procedure (Algorithm 7), the communication protocol is reversed with respect to AHT.

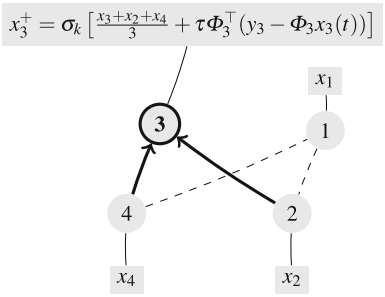

Fig. 5.6 Example of a network with four nodes: dynamics of AHT when node 3 is initially selected: node 3 gets information from its neighbor and updates its estimate

Algorithm 7 BHT

1: Initialization: $x_v(0) = 0 \in \mathbb{R}^n$, $\tau > 0$
2: **for** $t = 0, 1, \ldots, StopIter$ **do**
3: Selection of $v \in \mathscr{V}$
4: $x_w(t+1) = \sigma_k\left[\frac{1}{2}(x_v(t) + x_w(t)) + \tau\Phi_w^\top(y_w - \Phi_w x_w(t))\right]$ for any $w \in \mathscr{N}_v$
5: $x_h(t+1) = x_h(t)$ for any $h \notin \mathscr{N}_v \cup \{v\}$
6: **end for**

At each iteration step, one node $v \in \mathscr{V}$ is selected and sends its estimation to the neighbors $w \in \mathscr{N}_v$. After communication, each node $w \in \mathscr{N}_v$ updates its status performing the following operations:

(a) computation of the gradient of its loss function;
(b) average between its estimate x_w and the received estimate x_v;
(c) combination of the gradient step using (a) and (b);
(d) best k-term approximation of (c).

The procedure is iterated until a stopping criterion is met. Figure 5.7 shows the dynamics of the algorithm when node 3 in a network of four nodes is activated.

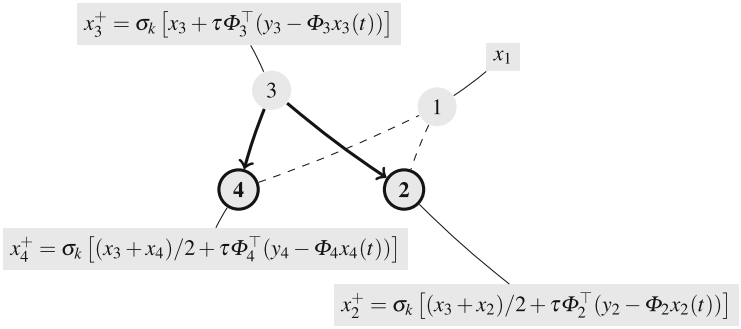

Fig. 5.7 Example of a network with four nodes: dynamics of BHT when node 3 is initially selected: node 3 broadcasts its estimate and makes the update, and its neighbors (node 2 and node 4) update their own estimate

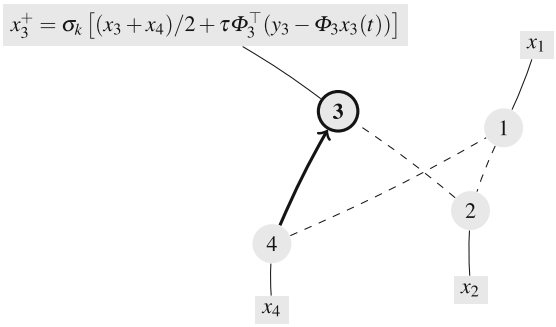

Fig. 5.8 Example of a network with 4 nodes: dynamics of GHT when edge (3,4) is initially selected: node 3 receives the estimate provided by node 4 and makes the update

In GHT, only one communication link is used at each iteration step. One node is selected and woken up and, in turn, selects one neighbor, receives its estimate, and performs the update as in Algorithm 8. The procedure is iterated until a stopping criterion is reached. The pseudocode is tabulated in Algorithm 8.

Algorithm 8 GHT

1: Initialization: $x_v(0) = 0 \in \mathbb{R}^n$ for any $v \in \mathscr{V}$, $\tau > 0$
2: **for** $t = 0, 1, \ldots, StopIter$ **do**
3: Selection of $(v, w) \in \mathscr{E}$
4: $x_v(t+1) = \sigma_k\left[\frac{x_v(t)+x_w(t)}{2} + \tau\Phi_v^\top(y_v - \Phi_v x_v(t))\right]$
5: $x_h(t+1) = x_h(t)$ for any $h \neq v$
6: **end for**

We have not specified a rule for the selection of node or edge at each iteration yet. Two different scenarios, which are formally described in the following definitions, can be considered.

Definition 5.2 (*Randomly persistent communication network*) A network of J nodes is said to be a randomly persistent communicating network if there exists a J-upla $(p_1, \ldots p_J)$ such that $p_v > 0$, for all $v \in \mathscr{V}$, $\sum_{v\in\mathscr{V}} p_v = 1$, and such that, for all $t \in \mathbb{N}$, $\mathbb{P}[\Omega_{v,t}] = p_v$, where $\Omega_{v,t}$ is the event

$$\Omega_{v,t} = \{\text{node } v \text{ makes the update at iteration } t.\}$$

Definition 5.3 (*Uniformly persistent communication network*) A network of J nodes is said to be a uniformly persistent communicating network if there exists a positive integer number $T > 0$ such that, for all $t \in \mathbb{N}$, each node makes the update perform at least once within the iteration-interval $[t, t+T)$.

An illustrative single example where perfect recovery is achieved is shown in Fig. 5.9. We have set the parameters $n = 150$ and $k = 15$. The nonzero components' positions are chosen uniformly at random; the amplitude of each nonzero component

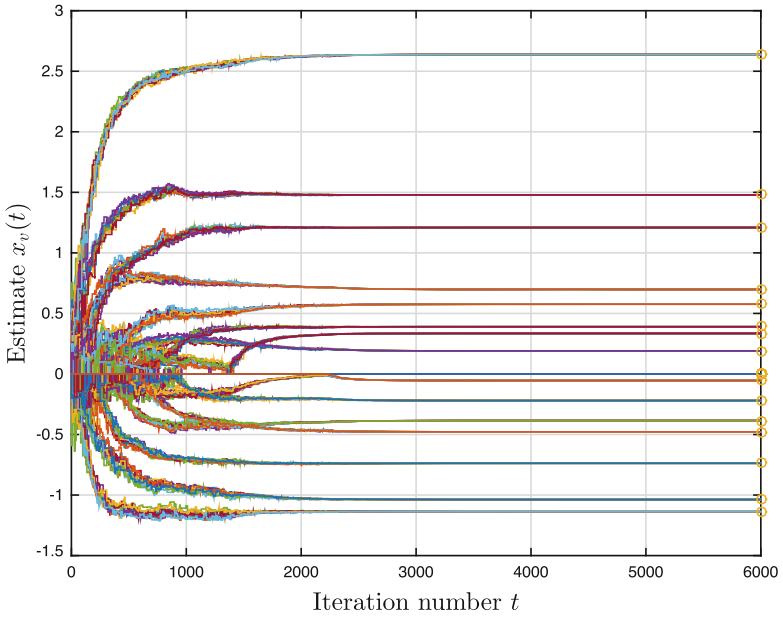

Fig. 5.9 Noise-free compressed sensing, $n = 150$, $k = 15$, $m = 10$, $J = 10$, complete graph: GHT converges to $x^\star$ (whose components are marked by circles)

is drawn from a Gaussian distribution $\mathsf{N}(0, 1)$. The sensing matrices $\Phi_v \in \mathbb{R}^{m \times n}$ are sampled from the Gaussian ensemble: $\Phi_v^{ij} \sim \mathsf{N}(0, 1/m)$, $\forall v \in \mathcal{V}$. GHT is implemented, over a complete graph, with $m = 10$, $J = 10$. Clearly, $m = 10$ is not sufficient for individual recovery, but collaboration among the 10 nodes allows to get it, in a reasonable number of iterations.

5.2.4.3 Discussion

We conclude the presentation of the algorithms with some remarks on their requirements in terms of computational complexity, memory storage, and communication cost.

Computational and Communication Requirements

Concerning the memory usage, all algorithms (DIHT, AHT, GHT, and BHT) require $O(n)$ storage at each node: we need to save $2k$ real values of the solution estimation (the nonzero components and their positions) at each node, in addition to the measurements y_v, the sensing matrix Φ_v and the necessary information to compute the gradient of its local loss function $\nabla f_v(x_v(t)) = -\Phi_v^\top (y_v - \Phi_v x_v(t))$. The

computational complexity per iteration requires only matrix vector multiplication. Therefore, the local computation is much simpler than the double- and single-looped algorithms that require each node to solve a convex optimization problem in every iteration (see DQP-Lasso in previous section for example).

The considered algorithms differ for communication cost per iteration. We recall that, for simplicity, we assume that all the nodes $v \in \mathscr{V}$ have degree $d_v = d$. At each iteration of DIHT, a k-sparse vector is sent down the tree and n-vectors corresponding to the gradient of the local loss functions are aggregated up the tree. Since a tree has $J-1$ edges, $2k(J-1)$ total values are sent per iteration for the broadcast. For the convergecast, the node sends a single message to its parent in the tree, for a total of $(2k+n)(J-1)$ messages per iteration. In this computation, we have to take into account the number of messages required to create a breadth-first spanning tree over the network, rooted at node r, using a distributed algorithm (see for example). This generally requires $2|\mathscr{E}| - (J-1) = J(d-1)+1$ messages. We analyze now the communication cost per iteration of AHT, BHT, and GHT. For GHT, only one communication link is used at each iteration step, while for AHT and BHT the number of active links is equal to d. The use of best k-term approximation is an advantage for what concerns the transmission of the current estimate at each iteration step, as the number of real values that have to be sent is reduced from n to $2k$ for each communication link. The number of updating nodes (that is, nodes that perform computations to update their own estimations) is d for BHT, and just 1 for both AHT and GHT. The number of nodes sending messages is d for AHT, and 1 for BHT and GHT. In Table 5.1, AHT, BHT, and GHT are compared in terms of computational effort, memory, and communication requirements.

It is clear that the total number of usages of communication links or sent values depend on the total number of iterations required to reach the convergence. We present some results of numerical tests about DIHT, AHT, BHT, and GHT.

We start by presenting some numerical results in the noise-free compressed sensing framework. In all the simulations we observe that convergence to the true signal can be achieved by AHT, BHT, and GHT, provided that the number of measurements is large enough (but keeping the number of measurements per node smaller than the number sufficient for individual reconstruction).

Table 5.1 Computational/communication requirements per iteration

	DIHT	AHT	BHT	GHT
Storage	$O(n)$	$O(n)$	$O(n)$	$O(n)$
Active comm. links	$J-1$	d	d	1
Sent values	$(2k+n)(J-1)$	$2kd$	$2kd$	$2k$
Updating nodes	J	1	$d+1$	1
Sender nodes	J	d	1	1

We assume that each node can in turn acquire $m = 10, 15, 20$ measurements and we study how the recovery accuracy varies at the increasing of the number of nodes. On one hand, adding nodes will augment the total number of measurements, which is expected to improve the recovery; on the other hand, a larger network may cause some degradation due to greater decentralization.

In Fig. 5.10, we show that the good effect prevails, that is, increasing the total number of measurements mJ (namely, the network size, having fixed m) the performance accuracy improves. More precisely, in Fig. 5.10 we show the results over two different topologies: ring (all the nodes communicates with two neighbors), and random geometric (we assign a uniformly random position to each node in the square $[0, 1] \times [0, 1]$, and we let communication between nodes with distance below a certain radius, in this case 0.75, [37]). The ring topology represents the least connected, regular case, while with the random geometric topology we explore the nonregular framework (for which we do not have theoretical guarantees). The algorithms are stopped at time T such that $\sum_{v\in\mathcal{V}} \left\| x_v^{T-1} - x_v^T \right\|_2^2 < 10^{-15}$ or after $T = 2 \times 10^5$ iterations, whichever occurred first [20, Section V.C].

The graphs show the rate of success (we declare a success when the accuracy condition $\frac{\sum_{v\in\mathcal{V}} \left\| x_v^T - x^\star \right\|_2^2}{J \|x^\star\|_2^2} < 10^{-4}$ holds) as a function of mJ for AHT, BHT, and GHT. All the results are obtained by averaging over 500 different runs. For these experiments, for each node $v \in \mathcal{V}$, we choose a $\tau_v = J^{-1} \|\Phi_v\|_2^{-2}$. In all the figures, we draw the curve for the (centralized) IHT as a benchmark.

Observing Fig. 5.10, we conclude that BHT tends to work better in the few measurements regime, immediately followed by GHT. AHT is a bit less reliable, in particular for $m = 10$ and ring topology. This can be explained with the presence of many stationary points when the number of measurements is low, among which the search of the true signal is even more difficult over few connected networks that do not support much collaboration. We finally notice that the gap with IHT is not dramatic, and that a 95 % of success is achieved by all the algorithms, over the different topologies, for $mJ \geq 120$ (except for AHT with $m = 10$ in the ring topology).

Number of Transmitted Values for Convergence

We now study the efficiencies of AHT, BHT, and GHT in terms of number of transmitted values necessary to achieve convergence (by transmitted value we mean a real scalar sent over a communication link in the network). As communications are typically energy expensive, we aim to keep that number as low as possible. To the best of our knowledge, among the possible distributed approaches to problem (5.16), DIHT [20] is the most efficient in terms of transmitted values: in [20, Section V], some tests are proposed that compare DIHT to CB-DIHT, D-ADMM, and subgradient methods, and the outcomes attest its higher performance. Those tests were conducted on a set of sparse problems selected from the Sparco dataset [38]; here, we consider a couple of those Sparco problems, with the same network topologies and parameters, and we show that AHT, BHT, and GHT overcome DIHT.

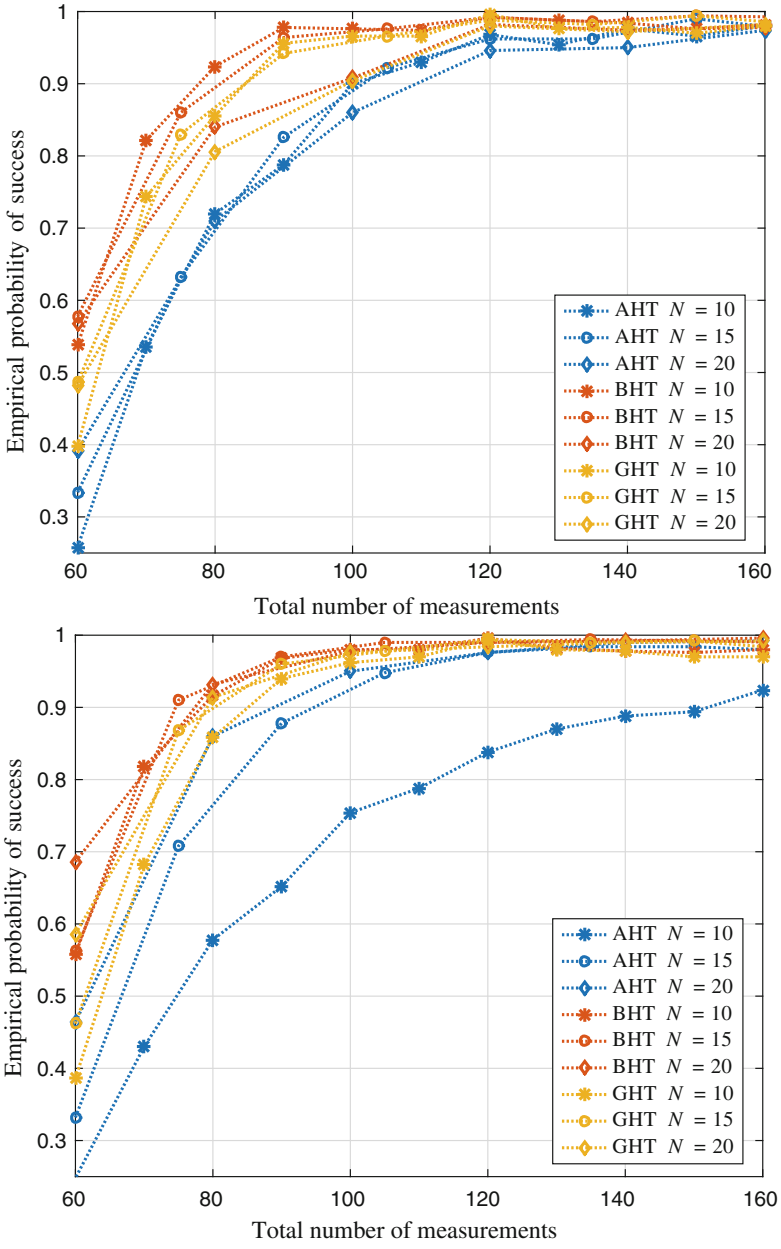

Fig. 5.10 Noise-free compressed sensing: probability of success over a ring (*left*) and over a random geometric graph with radius 0.75 (*right*) as a function of mJ

Table 5.2 Sparco problems' setting

Problem	n	m	$\lvert\mathcal{V}\rvert$	k
Sparco 902	1000	4	50	3
Sparco 7	2560	15	40	20

We repeat the experiments of [20]. More precisely, we consider the following problems: (a) Sparco 7, in which a sign spike signal is compressed through a Gaussian matrix; (b) Sparco 902, in which a signal sparse in the DCT domain has to be recovered (see [38]). Signals' lengths, number of measurements, and network specifications that we assume (see Table 5.2) are taken from [20, Table1]). We consider Erdos–Rényi (ER) and random geometric graphs (Geo), respectively, with connectivity parameters 0.25, 0.75 and 0.5, 0.75. The setting that we consider is the same as in [20, Section V], the only differences being in the different realizations for the random graphs and in the number of instances: our results are averaged over 100 instances, while in [20] 5 runs were performed.

In Table 5.3, we present our experimental results of the implementation of AHT, BHT, GHT, and DIHT on Sparco 7 and Sparco 902. As in [20, Sect. V.C], (a) algorithms are stopped at the time T such that

$$\sqrt{\frac{\sum_{v\in\mathcal{V}} \left\|x_v^T - x^\star\right\|_2^2}{J\,\|x^\star\|_2^2}} < \mathsf{tol},$$

with $\mathsf{tol} = 10^{-2}$ or $\mathsf{tol} = 10^{-5}$; (b) $x^\star$ is the true original signal or the stationary point to which the algorithm converges. In the next, we will show that not only AHT, but also BHT and GHT are always convergent in this setting, which makes unnecessary to fix an upper limit of iterations.

As in [20], in the implementation of DIHT we have considered $\tau = \frac{1}{2.01}$. The results that we obtain (in which we neglect the communications necessary to build the spanning tree) are substantially consistent with those presented in the original paper. It should be noted that the number of sent values for DIHT does not depend on the given topology, as communications are performed on the spanning tree which always has $J-1$ edges. The number of transmitted values is then $(J+1)(2k+n)T$, where T is the number of iterations to get convergence, $2k$ is the number of values required to diffuse the current k-sparse estimate from the root to the leaves, while n is the length of the nonsparse gradients that are accumulated and sent from the leaves to the root.

Concerning AHT, BHT, and GHT, we have fixed $\tau = 0.01$. For the Sparco 7 problem, no particular initialization is required, and the initial estimates are fixed to zero. For Sparco 902, instead, we assume that before starting the iterative procedure, each node v computes $x_v(0) = \sigma_k\left[\sum_{w\in\mathcal{N}_v} \tau\Phi_w^\mathsf{T} y_w\right]$ (if connectivity is high, the sum can be reduced over a selection of neighbors). This initialization has been experimentally

Table 5.3 Sparco tests: total number of sent values to converge. BHT and GHT converge to the true signal $x^\star$, while AHT converges to a stationary point (which is indicated by *)

Accuracy →	10^{-2}				10^{-5}			
	BHT mean	**BHT** min - max	**GHT** mean	**GHT** min - max	**BHT** mean	**BHT** min - max	**GHT** mean	**GHT** min - max
	Sparco 902							
ER $p = 0.25$	1.67×10^4	15768 - 18108	1.71×10^4	16110 - 18300	4.27×10^4	41616 - 43362	4.31×10^4	39492 - 45054
ER $p = 0.75$	2.72×10^4	25944 - 29652	2.77×10^4	26994 - 28482	5.07×10^4	48414 - 54300	5.24×10^4	50826 - 53580
Geo $d = 0.5$	2.30×10^4	19104 - 26070	2.30×10^4	20634 - 24456	4.72×10^4	44364 - 50586	4.87×10^4	45408 - 50424
Geo $d = 0.75$	2.63×10^4	24168 - 28998	2.68×10^4	26244 - 27372	4.94×10^4	44676 - 53694	5.14×10^4	50250 - 52398
	Sparco 7							
ER $p = 0.25$	1.82×10^5	165360 - 188760	1.57×10^5	133200 - 182440	3.79×10^5	359240 - 391840	3.67×10^5	341040 - 395600
ER $p = 0.75$	1.89×10^5	178600 - 203360	1.73×10^5	159960 - 184560	3.74×10^5	362560 - 396040	3.56×10^5	340960 - 372440
Geo $d = 0.5$	1.66×10^5	157640 - 173080	1.42×10^5	125040 - 152600	3.57×10^5	347880 - 372840	3.38×10^5	314560 - 352480
Geo $d = 0.75$	1.76×10^5	160240 - 188480	1.70×10^5	156640 - 178760	3.62×10^5	334400 - 383760	3.57×10^5	344360 - 366080

(continued)

Table 5.3 (continued)

Accuracy →	10^{-2}			10^{-5}		
	AHT* mean	**AHT*** min - max	**DIHT**	**AHT*** mean	**AHT*** min - max	**DIHT**
	Sparco 902					
ER $p = 0.25$	4.58×10^{4}	36690 - 58254	2.32×10^{6}	3.02×10^{5}	290874 - 316698	5.67×10^{6}
ER $p = 0.75$	3.71×10^{5}	337524 - 402828	2.32×10^{6}	1.23×10^{6}	1204188 - 1254288	5.67×10^{6}
Geo $d = 0.5$	1.78×10^{5}	145758 - 233646	2.32×10^{6}	7.02×10^{5}	585984 - 785532	5.67×10^{6}
Geo $d = 0.75$	3.79×10^{5}	332460 - 434796	2.32×10^{6}	1.25×10^{6}	1122006 - 1333182	5.67×10^{6}
	Sparco 7					
ER $p = 0.25$	4.98×10^{5}	375320 - 623280	6.39×10^{6}	2.44×10^{6}	2252680 - 2632720	1.39×10^{7}
ER $p = 0.75$	1.70×10^{6}	1611440 - 1761160	6.39×10^{6}	6.80×10^{6}	6655040 - 6932000	1.39×10^{7}
Geo $d = 0.5$	8.29×10^{5}	607840 - 1002960	6.39×10^{6}	4.43×10^{6}	3680560 - 4894320	1.39×10^{7}
Geo $d = 0.75$	1.94×10^{6}	1655360 - 2444440	6.39×10^{6}	7.54×10^{6}	6838160 - 8961760	1.39×10^{7}

proved to speed up the convergence. The total number of sent values is evaluated as $2kT(\sum_{t=1}^{T} l(t) + I)$, where T is the total number of iterations to get convergence, $l(t)$ is the number of used links at each time step (this number depends on the nodes' degree for AHT and BHT, while is simply 1 for GHT) and I is the number of used links in case of initialization. Notice that, as a difference from DIHT, only k-sparse vectors are transmitted (from which the coefficient $2k$), as each node performs hard thresholding before transmitting.

We remind that in principle BHT and GHT may not converge, and in particular may not converge to $x^\star$; however, for Sparco 7 and 902 considered in Table 5.3, we always observe convergence to $x^\star$, no matter which communication topology is assumed. In the table, we show the mean, minimum, and maximum number of sent values we have obtained over 100 runs. We point out that BHT, GHT, and DIHT converge to the true signal, while AHT converges to a stationary point different from $x^\star$ (this fact is highlighted by a "*" in Table 5.3).

In Table 5.1, we can appreciate the gain obtained by AHT, BHT, and GHT with respect to DIHT: in the 100 runs we consider, the number of transmitted values is always smaller using AHT, BHT, and GHT, which are then expected to outperform also CB-DIHT, D-ADMM, and subgradient methods, according to the results in [20].

We finally remark that AHT and BHT require less-transmitted values over less-connected topologies. This means that a limited collaboration, which reduces the number of used links at each iteration step, does not necessarily slow down the total algorithms' dynamics.

5.3 Beyond Single Source Estimation: Distributed Recovery of Correlated Signals

In previous sections, we focused on the distributed reconstruction of a *common signal* acquired in a networked system. Obviously, the main goal of the nodes in the network was to reach a consensus on the estimate of the signal.

In many applications, however, it is of interest to consider an ensemble of multiple correlated signals sensed by independent nodes in the networks. Algorithms for centralized reconstruction of signals that obey JSM-1, JSM-2, and JSM-3 correlation models have been already reviewed in Chap. 4. In literature, the problem of in-network reconstruction of multiple correlated sparse signals has been recently considered in [39, 40] and reference therein.

We are not interested in reviewing in detail the algorithms proposed in literature but we put the emphasis on the methodology used to design efficient and low-complex algorithms for distributed reconstruction. For this reason we start with a simple model of correlation, e.g. JSM-1, and we provide a simple recipe to follow: recast the problem as a sparse optimization problem, replicate the unknown variables, and solve the consensus-based optimization problem.

We start by recalling the notational convention of the JSM-1 problem. Given a network represented by a connected graph $\mathscr{G} = (\mathscr{V}, \mathscr{E})$, the nodes acquire measurements of the form

$$y_v = \Phi_v x_v, \qquad v \in \mathscr{V}$$

where $\Phi_v \in \mathbb{R}^{m \times n}$ is the sensing matrix of the v-th node, the observed signal follow the JSM-1 model

$$x_v = (x_{\mathsf{C}} + x_{\mathsf{I},v}), \qquad v \in \mathscr{V}$$

where x_{C} and $x_{\mathsf{I},v}$ are both sparse unknown terms with respective sparsities k_{C} and $k_{\mathsf{I},v}$. As already noted in Chap. 2, within the classical compressed sensing framework, the JSM-1 corresponds to a compressed sensing problem with one source

$$\begin{cases} y = Ax \\ x_{\mathsf{C}} \in \Sigma_{k_{\mathsf{C}}}, \\ x_{\mathsf{I},v} \in \Sigma_{k_{\mathsf{I},v}}, \ v \in \mathscr{V} \end{cases} \tag{5.31}$$

where $x = (x_{\mathsf{C}}^\top \ x_{\mathsf{I},1}^\top \ x_{\mathsf{I},2}^\top \ \dots \ x_{\mathsf{I},J}^\top)^\top$,

$$A = \begin{pmatrix} \Phi_1 & \Phi_1 & 0 & \dots & & \dots & 0 \\ \Phi_2 & 0 & \Phi_2 & 0 & \dots & \dots & 0 \\ & & & & \dots & & \\ \Phi_J & 0 & \dots & & & 0 & \Phi_J \end{pmatrix}, \qquad y = \begin{pmatrix} y_1 \\ y_2 \\ \dots \\ y_J \end{pmatrix} \tag{5.32}$$

If the measurements were available in a central processing unit, a reasonable estimation of the signals can be obtained by solving the following optimization problem:

$$\min_{x_{\mathsf{C}}, x_{\mathsf{I},v}} \frac{1}{2} \sum_{v=1}^{J} \|y_v - \Phi_v(x_{\mathsf{C}} + x_{\mathsf{I},v})\|_2^2 + \lambda_{\mathsf{C}} \|x_{\mathsf{C}}\|_1 + \sum_{v \in \mathscr{V}} \lambda_v \|x_{\mathsf{I},v}\|_1 \tag{5.33}$$

where λ_{C}, λ_v are scalar parameters related to the sparsity levels of the common signal and innovations. Following the consensus-based optimization, (5.31) can be recast into a separable form, that is prone to distributed implementation. The common variable z_{C} is replaced with local variables $\{\zeta_v\}_{v \in \mathscr{V}}$ and the optimization problem (5.33) is rewritten in the following form

$$\min_{\xi_v, z_{\mathsf{I},v}} \frac{1}{2} \sum_{v=1}^{J} \left[\|y_v - \Phi_v(\zeta_v + z_{\mathsf{I},v})\|_2^2 + 2\frac{\lambda_{\mathsf{C}}}{J} \|\zeta_v\|_1 + 2\lambda_v \|z_{\mathsf{I},v}\|_1 \right] \tag{5.34}$$

$$\text{s.t.} \qquad \zeta_v = \zeta_w, \qquad (v, w) \in \mathscr{E}.$$

Starting from this separable problem, a variety of algorithms can be proposed by using distributed subgradient method, iterative thresholding algorithms, or alternating direction method of multipliers.

References

1. Baron, D., Duarte, M.F., Wakin, M.B., Sarvotham, S., Baraniuk, R.G.: Distributed compressive sensing. arXiv preprint arXiv:0901.3403 (2009)
2. Forero, P.A., Cano, A., Giannakis, G.B.: Consensus-based distributed support vector machines. J. Mach. Learn. Res. **99**, 1663–1707 (2010)
3. Chaudhuri, K., Monteleoni, C.: Privacy-preserving logistic regression. In Koller, D., Schuurmans, D., Bengio, Y., Bottou, L. (eds.) NIPS, Curran Associates, Inc. pp. 289–296 (2008)
4. Krüger, J., Westermann, R.: Linear algebra operators for GPU implementation of numerical algorithms. ACM Trans. Graph. **22**(3), 908–916 (2003)
5. Owens, J.D., Luebke, D., Govindaraju, N., Harris, M., Krüger, J., Lefohn, A., Purcell, T.J.: A survey of general-purpose computation on graphics hardware. Comput. Graph. Forum **26**(1), 80–113 (2007)
6. Sundman, D., Chatterjee, S., Skoglund, M.: A greedy pursuit algorithm for distributed compressed sensing. In: IEEE ICASSP, pp. 2729–2732 (2012)
7. Mota, J., Xavier, J., Aguiar, P., Puschel, M.: Basis pursuit in sensor networks. In: IEEE ICASSP, pp. 2916–2919 (2011)
8. Mota, J., Xavier, J., Aguiar, P., Puschel, M.: Distributed basis pursuit. IEEE Trans. Signal Proc. **60**(4), 1942–1956 (2012)
9. Mateos, G., Bazerque, J.A., Giannakis, G.B.: Distributed sparse linear regression. IEEE Trans. Signal Proc. **58**(10), 5262–5276 (2010)
10. Nedić, A., Ozdaglar, A., Parrillo, P.A.: Constrained consensus and optimization in multi-agent networks. IEEE Trans. Autom. Control **55**, 922–938 (2010)
11. Boyd, S., Parikh, N., Chu, E., Peleato, B., Eckstein, J.: Distributed optimization and statistical learning via the alternating direction method of multipliers. Found. Trends Mach. Learn. **3**(1), 1–122 (2011)
12. Ravazzi, C., Fosson, S.M., Magli, E.: Distributed soft thresholding for sparse signal recovery. In: 2013 IEEE Global Communications Conference (GLOBECOM), pp. 3429–3434 (2013)
13. Ravazzi, C., Fosson, S.M., Magli, E.: Energy-saving gossip algorithm for compressed sensing in multi-agent systems. In: 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, pp. 5097–5101 (2014)
14. Ravazzi, C., Fosson, S.M., Magli, E.: Randomized algorithms for distributed nonlinear optimization under sparsity constraints. IEEE Trans. Signal Process. 1–14 (2014)
15. Ravazzi, C., Fosson, S., Magli, E.: Distributed iterative thresholding for ell_0/ell_1—regularized linear inverse problems. IEEE Trans. Inf. Theory **61**(4), 2081–2100 (2015)
16. Lu, J., Tang, C.Y., Regier, P.R., Bow, T.D.: Gossip algorithms for convex consensus optimization over networks. IEEE Trans. Autom. Control **56**(12), 2917–2923 (2011)
17. Sundhar Ram, S., Nedić, A., Veeravalli, V.V.: Distributed stochastic subgradient projection algorithms for convex optimization. J. Optim. Theory Appl. 516–545 (2010)
18. Sundhar Ram, S., Nedić, A., Veeravalli, V.V.: Asynchronous gossip algorithm for stochastic optimization: Constant stepsize analysis. Recent Advances in Optimization and its Applications in Engineering. Springer, Berlin, pp. 51–60 (2010)
19. Dimakis, A.G., Kar, S., Moura, J.M.F., Rabbat, M.G., Scaglione, A.: Gossip algorithms for distributed signal processing. Proc. IEEE **98**(11), 1847–1864 (2010)
20. Patterson, S., Eldar, Y., Keidar, I.: Distributed compressed sensing for static and time-varying networks. IEEE Trans. Signal Process. **62**(19), 4931–4946 (2014)

21. Candès, E.J.: The restricted isometry property and its implications for compressed sensing. Compte Rendus de l'Academie des Sciences. Paris, France, ser. I 346, pp. 589–592 (2008)
22. Olfati-Saber, R., Fax, J.A., Murray, R.M.: Consensus and cooperation in networked multi-agent systems. In: Proceedings of the IEEE (2007)
23. Tibshirani, R.J.: The Lasso problem and uniqueness. Electron. J. Stat. **7**, 1456–1490 (2013)
24. Boyd, S., Vandenberghe, L.: Convex Optimization. Cambridge University Press, New York (2004)
25. Sundhar Ram, S., Nedić, A., Veeravalli, V.V.: Distributed subgradient projection algorithm for convex optimization. In: Proceedings of IEEE ICASSP, pp. 3653–3656 (2009)
26. Nedic, A., Ozdaglar, A.: Distributed subgradient methods for multi-agent optimization. IEEE Trans. Autom. Control **54**(1), 48–61 (2009)
27. Lobel, I., Ozdaglar, A.: Distributed subgradient methods for convex optimization over random networks. IEEE Trans. Autom. Control **56**(6), 1291–1306 (2011)
28. Tsitsiklis, J.N.: Problems in decentralized decision making and computation. PhD thesis, Department of EECS, MIT (1984)
29. Tsitsiklis, J.N., Bertsekas, D.P., Athans, M.: Distributed asynchronous deterministic and stochastic gradient optimization algorithms. IEEE Trans. Autom. Control **31**(9), 803–812 (1986)
30. Wei, E., Ozdaglar, A.: Distributed alternating direction method of multipliers. In: IEEE 51st Annual Conference on Decision and Control (CDC), pp. 5445–5450 (2012)
31. Mota, J., Xavier, J., Aguiar, P., Puschel, M.: D-ADMM: a communication-efficient distributed algorithm for separable optimization. IEEE Trans. Signal Proc. **61**(10), 2718–2723 (2013)
32. Moallemi, C., Van Roy, B.: Consensus propagation. IEEE Trans. Inf. Theory **52**(11), 4753–4766
33. Lange, K., Hunter, D., Yang, I.: Optimization transfer using surrogate objective functions. J. Comput. Graph. Stat. **9**, 1–20 (2006)
34. Lange, K.: Optimization. Springer, New York (2004)
35. Candès, E.J., Romberg, J.K., Tao, T.: Stable signal recovery from incomplete and inaccurate measurements. Commun. Pure. Appl. Math. **59**(8), 1207–1223 (2006)
36. Xiao, L., Boyd, S., Lall, S.: Distributed average consensus with time-varying metropolis weights. http://web.stanford.edu/~boyd/papers/avg_metropolis.html (2006)
37. Penrose, M.: Random Geometric Graphs (Oxford Studies in Probability). Oxford University Press, Oxford (July 2003)
38. van den Berg, E., Friedlander, M., Hennenfent, G., Herrmann, F., Saab, R., Yılmaz, Ö.: Sparco: A testing framework for sparse reconstruction. Technical Report TR-2007-20, Department Computer Science, University of British Columbia, Vancouver (2007)
39. Fosson, S.M., Matamoros, J., Antón-Haro, C., Magli, E.: Distributed support detection of jointly sparse signals. In: 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 6434–6438 (May 2014)
40. Matamoros, J., Fosson, S.M., Magli, E., Antón-Haro, C.: Distributed ADMM for in-network reconstruction of sparse signals with innovations. In: 2014 IEEE Global Conference on Signal and Information Processing (GlobalSIP), pp. 429–433 (2014)

Chapter 6
Conclusions

Compressed sensing has attracted a huge interest in the scientific community, with hundreds of papers published in conferences and journals. The equivalent distributed problem has been studied comparatively less, but has an extremely large application potential in large-scale distributed systems. This book has summarized the state of the art of CS for distributed systems, covering both theory and algorithms. Intentionally, the material included in this book does not cover all available techniques, but only those that are more suitable in a distributed scenario, with an emphasis on low power techniques. As a consequence, very general techniques which have proved to be very successful in many applications but are not always applicable in a sensor network scenario have been left out of this book. The interested reader will find a lot of related material available.

CS for distributed systems is reaching a stage where it is now becoming mature. However, since it involves a more sophisticated system model than conventional CS, a lot of work still has to be done in the areas of sensing and reconstruction, as well as other communication and processing aspects. There are many things that distributed nodes can do using CS, besides just using CS as an efficient representation. As nodes cooperate to reconstruct a signal, they could also cooperate in order to extract information from that signal, or more precisely, from its linear measurements. Processing signals in the compressed domain has been proved possible in the classical case, but it is a much less studied problem in the distributed case. Yet, there is a large potential because not only the intranode, but also the internode statistics may be inferred through collaboration. CS security is another area that may certainly benefit from more research. The security properties of random projections point to an intriguing new class of encryption techniques. The possibility to consider these security properties in a more general distributed scenario paves the way for secure distributed multiparty computation based on CS. At the same time, considering the relation between CS and channel coding, it is easy to see that there are relations between DCS and network coding, which might be exploited to design new network coding schemes or new DCS schemes alike. These ideas only scratch the surface of what can be done applying CS to distributed systems, and we hope that this book will stimulate researchers to tackle innovative CS problems in order to advance this field even more.

G. Coluccia et al., *Compressed Sensing for Distributed Systems*,
SpringerBriefs in Signal Processing, DOI 10.1007/978-981-287-390-3_6