KTH Electrical Engineering

Compressed Sensing Algorithms and Applications

DENNIS SUNDMAN

Licentiate Thesis in Telecommunications
Stockholm, Sweden 2012

Compressed Sensing
Algorithms and Applications

Copyright © 2012 by Dennis Sundman except where
otherwise stated. All rights reserved.

ISSN 1653-5146
ISBN 978-91-7501-269-8

Dennis Sundman
Communication Theory
School of Electrical Engineering
Royal Institute of Technology (KTH)
SE-100 44 Stockholm, Sweden
E-mail: denniss@kth.se

Printed by Universitetsservice US-AB.

Abstract

The theoretical problem of finding the solution to an underdetermined set of linear equations has for several years attracted considerable attention in the literature. This problem has many practical applications. One example of such an application is compressed sensing (CS), which has the potential to revolutionize how we acquire and process signals. In a general CS setup, few measurement coefficients are available and the task is to reconstruct a larger, sparse signal.

In this thesis we focus on algorithm design and selected applications for CS. The contributions of the thesis appear in the following order: (1) We study an application where CS can be used to relax the necessity of fast sampling for power spectral density estimation problems. In this application we show by experimental evaluation that we can gain an order of magnitude in reduced sampling frequency. (2) In order to improve CS recovery performance, we extend simple well-known recovery algorithms by introducing a look-ahead concept. From simulations it is observed that the additional complexity results in significant improvements in recovery performance. (3) For sensor networks, we extend the current framework of CS by introducing a new general network model which is suitable for modeling several CS sensor nodes with correlated measurements. Using this signal model we then develop several centralized and distributed CS recovery algorithms. We find that both the centralized and distributed algorithms achieve a significant gain in recovery performance compared to the standard, disconnected, algorithms. For the distributed case, we also see that as the network connectivity increases, the performance rapidly converges to the performance of the centralized solution.

Keywords: Compressed sensing, greedy pursuits, subspace pursuit, orthogonal matching pursuit, power spectral density estimation, distributed compressed sensing.

Acknowledgments

I want to express my sincere gratitude to Prof. Mikael Skoglund for welcoming me into his research group. In particular I want to thank Mikael for his guidance, showing me new directions whenever I have reached a dead end. For his encouragement and support Dr. Saikat Chatterjee has doubtlessly been the single most important source of inspiration for this thesis to come true. One of the first things Saikat told me was “never thank me”. Well, I am sorry dear friend, here it comes: thank you.

I have since the first day at the lab shared office with Ricardo Blasco Serrano. I would like to thank Ricardo for all the geeky talks about computers, our few but intense bike rides, and not least for proofreading the thesis. For further valuable feedback on the thesis I want to thank Dave Zachariah and Assoc. Prof. Mats Bengtsson. I would like to thank some of the people with whom I have had many interesting discussions regarding my work: Assoc. Prof. Carlo Fischione, Mattias Andersson and Nicolas Schrammar. I want to thank all my colleagues in the lab for providing a great working environment. In particular I would like to thank Annika Augustsson and Iréne Kindblom for taking care of the tricky administration business. I would also like to thank my coaches Magnus Vinblad and Frédéric Gabry for the many hours of running, discussions, and the challenges we have shared.

I wish to thank Assoc. Prof. Magnus Jansson for taking the time to act as opponent to this thesis.

I would like to thank Maria for the all the love and understanding you have given me over the past years. Finally, I would like to express my gratitude to my parents Kurre and Lotta and to my sister Tina for your endless love and encouragement, always being there whenever I need support in life.

Dennis Sundman
Stockholm, February 2012

Contents

List of Algorithms

BAOMP:	Backtracking Orthogonal Matching Pursuit
CoSaMP:	Compressive Sampling Matching Pursuit
DiFROMP:	Distributed Forward-Reverse Orthogonal Matching Pursuit[1]
DiOMP:	Distributed Orthogonal Matching Pursuit[1]
DiSP:	Distributed Subspace Pursuit[1]
FROMP:	Forward-Reverse Orthogonal Matching Pursuit[1]
JFROMP:	Joint Forward-Reverse Orthogonal Matching Pursuit[1]
JOMP:	Joint Orthogonal Matching Pursuit[1]
JSP:	Joint Subspace Pursuit[1]
LAOMP:	Look Ahead Orthogonal Matching Pursuit[1]
LAPP:	Look Ahead Parallel Pursuit[1]
OMP:	Orthogonal Matching Pursuit
POMP:	Projection-based Orthogonal Matching Pursuit
ROMP:	Regularized Orthogonal Matching Pursuit
SP:	Subspace Pursuit
StOMP:	Stagewise Orthogonal Matching Pursuit

[1]Developed in this thesis

Part I

Introduction

Chapter 1

Introduction

Compressed (or compressive) sensing (or sampling) (CS) is a framework for signal reconstruction from a measurement vector which is assumed smaller in size than the Nyquist-sampled signal vector, where this signal vector is inherently sparse (i.e., a majority of the signal-vector components are zero). The measurement acquisition process is described by a matrix multiplication with a fat sensing-matrix and thus the reconstruction problem is an under-determined system of linear equations. The challenge in CS is two-fold: firstly, how to produce the measurement vector in practice and secondly, based on knowing the measurement vector and measurement matrix, how to find the correct underlying sparse signal-vector. The theory behind CS is based on the observation that many natural signals, such as sound or images, can be well approximated with a sparse representation in some domain. For example, it turns out that most of the energy in a typical image signal is preserved within the $2\% - 4\%$ dominating wavelet coefficients [1].

The term compressed sensing was coined in the article 'Compressed Sensing' by Donoho [2]. Along with Donoho, other pioneering work to ground this area of research was done by Candès, Romberg and Tao [3]. CS is often considered as an underlying topic in the more general field sparse signal processing [4]. In sparse signal processing, one is usually interested in solving some detection or estimation problem, based on a measurement-vector whose size is not necessarily smaller than the size of the sparse signal-vector.

This introduction chapter is divided into four sections. In Section 1 we introduce the standard CS problem formulation; then we give some properties associated with the standard CS problem; we briefly discuss some signal acquisition; and end the section with conventional CS signal reconstruction techniques. In Section 2 we further describe the main tools of a class of signal reconstruction algorithms called greedy pursuits and present two examples of such algorithms: orthogonal matching pursuit and subspace pursuit. We end Section 2 by discussing performance measures used throughout the thesis. In Section 3, we introduce the distributed CS problem and provide some figures that can be used as references. Finally, in Section 4 we enlist the contributions from this thesis.

1 Compressed Sensing

Before we define the standard compressed sensing problem we introduce the l_0-"norm".

Definition 1 (l_0-norm)

$$\|\mathbf{x}\|_0 \triangleq \{\textit{The number of components in } \mathbf{x} \textit{ which fulfill } x_i \neq 0\}\,. \quad (1.1)$$

The l_0-norm does not satisfy the conditions of a true norm, which is easily verified by checking the required norm property *absolute homogenity*[1]. Nevertheless, we abuse the name l_0-norm. Furthermore, we say that a vector is sparse if the l_0-norm of the vector is much smaller than the vector dimensionality. More precisely, we say that an N-sized vector $\mathbf{x}$ is P-sparse if $\|\mathbf{x}\|_0 \leq P$, where $P \ll N$. Now, using the l_0-norm we are ready to define the standard CS problem.

Definition 2 (The standard CS problem) *Find the sparse signal-vector $\mathbf{x} \in \mathbb{R}^N$ provided the measurement-vector $\mathbf{y} \in \mathbb{R}^M$, the measurement-matrix $\mathbf{A} \in \mathbb{R}^{M \times N}$ and the under-determined set of linear equations as*

$$\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{e}, \quad (1.2)$$

where $\mathbf{e} \in \mathbb{R}^M$ represents some measurement noise $\mathbf{e} \sim \mathcal{N}(0, \sigma^2 \mathbf{I}_M)$ and $\mathbf{y} \in \mathbb{R}^M$ is the measurement vector. The number of non-zero components in $\mathbf{x}$ are $\|\mathbf{x}\|_0 = K$, where $K < M < N$.

Throughout this thesis we will assume that the measurement matrix $\mathbf{A}$ is constructed by first picking the individual elements independently $a_{m,n} \sim \mathcal{N}(0, 1)$ and then normalize each column $\mathbf{a}_n$ in $\mathbf{A}$ to unit length. This assumption is in general not necessary, but simplifies the notation. We will refer to the space in which $\mathbf{x}$ resides as the *signal space* ($\mathbb{R}^N$) and the space in which $\mathbf{y}$ resides as the *measurement space* ($\mathbb{R}^M$). In

[1]Absolute homogenity: $\forall \alpha \neq 0$ and $\forall \mathbf{x} \neq \mathbf{0}$, $\|\alpha \mathbf{x}\| = |\alpha| \|\mathbf{x}\|$.

this thesis, we only consider the case for real vectors and matrices, but it is usually straight-forward to extend the theory to the complex case.

In the standard CS problem, the reconstruction performance of $\mathbf{x}$ (based on $\mathbf{y}$ and $\mathbf{A}$) depends on $\mathbf{A}$ and $\mathbf{e}$. Assuming we construct the measurement device, we can usually choose $\mathbf{A}$. To help determine how good a matrix $\mathbf{A}$ is for the purpose of CS measurement there are some analytical properties associated with $\mathbf{A}$ available. In order to provide a solid background, we will present two fundamental properties associated with $\mathbf{A}$ often used in the literature, although we do not explicitly use these properties in the thesis. These two measurements are the restricted isometry property and the mutual coherence.

Restricted Isometry Property

In the attempt of finding some parameter describing how good the measurement matrix serves for the CS purpose, Candès and Tao introduced a concept referred to as the restricted isometry property (RIP) [5–7], here defined as:

Definition 3 (Restricted isometry property, RIP) *For all $\mathbf{x}$ so that $\|\mathbf{x}\|_0 \leq K$, it is said that $\mathbf{A}$ fulfills the RIP property with parameter δ_K, if δ_K is the smallest value satisfying:*

$$(1-\delta_K)\|\mathbf{x}\|_2^2 \leq \|\mathbf{A}\mathbf{x}\|_2^2 \leq (1+\delta_K)\|\mathbf{x}\|_2^2. \tag{1.3}$$

The measurement matrix $\mathbf{A}$ can be seen as a transformation of the signal from the signal space to the measurement space, where the measurement space is smaller than the signal space. For the variable $\mathbf{x}$, the RIP property is a characterization of how the Euclidean norm of $\mathbf{x}$ change by the transformation described by $\mathbf{A}$. On the other hand, by considering two variables $\mathbf{x}_1$ and $\mathbf{x}_2$, the RIP property can also be interpreted as a property on how the distance between $\mathbf{x}_1$ and $\mathbf{x}_2$ change by the transformation of $\mathbf{A}$. We give a visual interpretation by providing a sketchy example.

Example 1.1 (Transformation of distance) Consider two P-sparse variables $\mathbf{x}_1$ and $\mathbf{x}_2$ transformed noiselessly (i.e., $\mathbf{e} = \mathbf{0}$) by $\mathbf{A}$ into $\mathbf{y}_1$ and $\mathbf{y}_2$ as depicted in Figure 1.1. By constructing the K-sparse signal $\mathbf{x} = \mathbf{x}_1 - \mathbf{x}_2$ where $2P \geq K \geq 0$, and $\mathbf{y} = \mathbf{y}_1 - \mathbf{y}_2$ we find the following relation between the distances:

$$\frac{\|\mathbf{y}_1 - \mathbf{y}_2\|_2^2}{\|\mathbf{x}_1 - \mathbf{x}_2\|_2^2} = \frac{\|\mathbf{A}\mathbf{x}\|_2^2}{\|\mathbf{x}\|_2^2}. \tag{1.4}$$

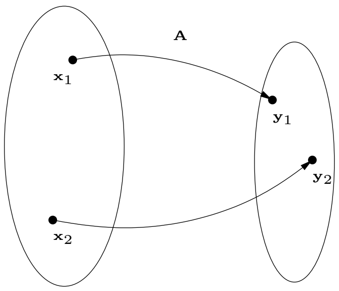


Figure 1.1: Transformation from the signal-space to the measurement-space

By comparing (1.4) with (1.3), we see that δ_K ($= \delta_{2P}$) provides for an upper- and lower bound of the change in Euclidean distance by the transformation $\mathbf{A}$. $\diamond$

For certain classes of matrices, assuming big values on N and M, there are statistical results on the parameter δ_K. For example, if $\mathbf{A}$ is drawn from an independent Gaussian distribution with normalized columns (as we have defined our $\mathbf{A}$), and if $M \geq CK \log(\frac{N}{K})/\epsilon^2$, then with high probability $\delta_K < \epsilon$ [7–9]. However, for a specific matrix $\mathbf{A}$, finding δ_K is a combinatorial problem which in practice often is too computationally demanding to solve.

In theory, RIP is a popular tool when characterizing the performance of different CS recovery algorithms because of its mathematical elegance. In practice however, performance analyses based on RIP turns out to be challenging because of the difficulty of finding δ_K for a given specific measurement-matrix. In a practical scenario, one can instead bound δ_K with the weaker mutual coherence.

Mutual Coherence

Another property of the measurement matrix is the mutual coherence, introduced by Donoho and Huo in [10]. We define it here as

Definition 4 (Mutual coherence) *Let $\mathbf{a}_i$ and $\mathbf{a}_j$ be two columns of $\mathbf{A}$. Then, the mutual coherence of $\mathbf{A}$ is*

$$\mu(\mathbf{A}) \triangleq \sup\{|\langle \mathbf{a}_i, \mathbf{a}_j \rangle| : \forall i, j, \text{ where } i \neq j\} \tag{1.5}$$

This measure is easier to calculate than the RIP parameter since the computational complexity scales exponentially with the number of columns in $\mathbf{A}$. The mutual coherence provides a bound on the RIP parameter: $\delta_K \leq (K-1)\mu$ [11]. Based on the mutual coherence and RIP we can now say something about how good a given matrix $\mathbf{A}$ is in terms of finding $\mathbf{x}$ in (1.2). If a solution-vector is provided, it comes natural to ask if we can tell whether this is the unique correct answer.

Uniqueness and Stability

For the noiseless case, there are ways to determine if a solution is unique. For the noisy case uniqueness is not possible to determine, in which case one instead talks about stability. We mention that these results exist without further going into detail. Instead we refer the interested reader to the book *Sparse and Redundant Representations* [4] by M. Elad.

1.1 Data Acquisition

First we mention that the standard CS problem is defined as a sub-sampling problem (Definition 2), where an already N-sized vector $\mathbf{x}$ is down-sampled to the M-sized vector $\mathbf{y}$. This setup is relevant in for example bio-engineering where the engineers and researchers deal with enormous data-sets [12], which may be infeasible to analyze in the signal space.

The real challenge in data acquisition for CS is to sample an analog signal directly into the compressed vector $\mathbf{y}$. This challenge is presently an active research area where the devices that acquire the samples are referred to as analog-to-information converters [13]. Some examples of analog-to-information converters are chirp sampling [14], Xampling [15], and random demodulator [16].

1.2 Data Reconstruction

Based on the knowledge of the measurement matrix $\mathbf{A}$ and measurement vector $\mathbf{y}$ as in (1.2), the task in the data reconstruction (or recovery) side of CS is to find the signal vector $\mathbf{x}$. There is no obvious way how to formulate this problem. Based on basic linear algebra knowledge, one first approach (which does not take the sparsity into account) could be trying to solve the least squares problem

$$\min_{\hat{\mathbf{x}}} \|\mathbf{A}\hat{\mathbf{x}} - \mathbf{y}\|_2^2. \tag{1.6}$$

However, since $\mathbf{A}$ is column-rank deficient, (1.6) has infinitely many solutions and it is not clear how to find the solution for our problem. Instead we try to take the knowledge of sparsity into account, by realizing that the number of non-zero elements in the solution vector $\mathbf{x}$ is $\|\mathbf{x}\|_0 \leq K$. Let us assume for now that we are dealing with a noiseless scenario (i.e., $\mathbf{e} = \mathbf{0}$). Then, we can attempt to solve the following problem

$$\min_{\hat{\mathbf{x}}} \|\hat{\mathbf{x}}\|_0 \ \text{ such that } \ \mathbf{A}\hat{\mathbf{x}} = \mathbf{y}. \tag{1.7}$$

If we have some knowledge about the noise variance, we can modify the problem so that it can handle a noisy case by relaxing the constraint $\mathbf{A}\hat{\mathbf{x}} = \mathbf{y}$ with $\|\mathbf{A}\hat{\mathbf{x}} - \mathbf{y}\|_2 \leq \epsilon$, where ϵ depends on the noise; we will come back to the noisy setting in the next section. The problem (1.7) is in some sense the central problem we always want to solve in CS. It turns out that solving (1.7) always provides the right solution $\mathbf{x}$ if $\delta_K < 1$, but solving this equation requires a combinatorial search. Instead, we will in the following two sections look at two classes of methods for finding an (approximate) solution to (1.7).

Convex Relaxation Methods

The convex relaxation methods are based on relaxing problem (1.7) by replacing the objective function of the problem by a convex function. We first consider the noiseless case and then introduce the noise. By replacing the l_0-norm with an l_1-norm (which is a true norm), we arrive at:

$$\min_{\hat{\mathbf{x}}} \|\hat{\mathbf{x}}\|_1 \ \text{ such that } \ \mathbf{A}\hat{\mathbf{x}} = \mathbf{y}. \tag{1.8}$$

This problem can be rewritten in a linear form which can be solved efficiently with existing linear (or convex) solvers. These convex solvers will find the same solution as (1.7) with high probability if $\mathbf{A}$ fulfills RIP with $\delta_K < \sqrt{2} - 1$ [5]. We now re-introduce the noise in our problem and see that by relaxing the constraints in (1.8) we get

$$\min_{\hat{\mathbf{x}}} \|\hat{\mathbf{x}}\|_1 \ \text{ such that } \ \|\mathbf{A}\hat{\mathbf{x}} - \mathbf{y}\|_2 \leq \epsilon. \tag{1.9}$$

This formulation was introduced in [17]. One problem with (1.9) is that ϵ, which is dependent on the noise, needs to be determined. When $\mathbf{e}$ is white Gaussian noise, [17] suggests to choose $\epsilon = \sigma\sqrt{M + \lambda\sqrt{2M}}$, where λ is some tuning parameter, for example 2. If there is no knowledge about the noise, one can instead further relax the problem to

$$\min_{\hat{\mathbf{x}}} \frac{1}{2}\|\mathbf{A}\hat{\mathbf{x}} - \mathbf{y}\|_2^2 + \gamma\|\hat{\mathbf{x}}\|_1, \tag{1.10}$$

where $\gamma > 0$, again, is a tuning parameter adjustable for different levels of sparsity. It is worth mentioning that finding the optimal solution to

(1.10) simultaneously for every value of γ can be done with the least angle regression stagewise (LARS) [18] algorithm. In the literature, the most frequently encountered problem formulations in presence of noise are (1.9) and (1.10) and they are usually referred to as basis pursuit denoising and least absolute shrinkage and selector operator (LASSO) [19], respectively. There exist other relaxations of problem (1.8) for example where the l_1-norm is placed as a constraint.

Greedy Pursuits

The greedy pursuits are algorithms that primarily try to solve the sought problem (1.7) by different linear algebraic tools. They operate by first trying to find the active indices which are those indices corresponding to non-zero values in $\mathbf{x}$, and then assigning the correct values to these indices. Most greedy pursuits present in the literature turn out to work efficiently even in the presence of noise. Since the greedy pursuits often rely on signal processing heuristics, it is in general harder to prove the performance of these algorithms compared to the convex relaxation based problems. Most contributions in this thesis are associated with greedy pursuit algorithms, which is why the full Section 2 is dedicated to a more detailed study of them. In particular, we study orthogonal matching pursuit [20] and subspace pursuit [21] in Section 2.3.

2 Greedy Pursuits

In this thesis, the main focus on recovery algorithms is the greedy pursuit (GP) class of algorithms. The GP algorithms are in general harder to analyze than the mathematically elegant convex-relaxation based methods. On the other hand, the GP algorithms have proven computationally efficient and easy to implement while providing similar (and sometimes better) performance than the convex relaxation based algorithms. To understand the GP algorithms we first define the support-set.

Definition 5 (Support-set) *The support-set $\mathcal{T}$ is a set of indices corresponding to the non-zero (active) components in the sparse vector* $\mathbf{x}$*:*

$$\mathcal{T} \triangleq \{i : x_i \neq 0\}. \tag{1.11}$$

Its complement is denoted by $\bar{\mathcal{T}}$

$$\bar{\mathcal{T}} \triangleq \{i : x_i = 0\}. \tag{1.12}$$

Clearly, $\mathcal{T} \cup \bar{\mathcal{T}} = \{1, 2, ..., N\}$ and $\mathcal{T} \cap \bar{\mathcal{T}} = \emptyset$. Furthermore, we can pick all the active components in $\mathbf{x}$ and place them one-by-one in a column-vector $\mathbf{x}_{\mathcal{T}} = \{x_i : x_i \neq 0\}$ which has size $\|\mathbf{x}\|_0 = |\mathcal{T}| \leq K$. Similarly, we can take the active column vectors in $\mathbf{A}$ and form the matrix $\mathbf{A}_{\mathcal{T}} = \{\mathbf{a}_i : x_i \neq 0\}$.

Given a measurement vector, the main principle of GP algorithms is to detect (or estimate) the underlying support-set of a sparse signal-vector followed by evaluating the associated signal values. To estimate the support set and the associated signal values the GP algorithms use different linear algebraic tools. Among these tools, the two major tools are the matched filter (MF) detection and least squares (LS) estimation, which we will take a closer look at in Section 2.1 and Section 2.2, respectively. Among the many existing GP algorithms, two popular ones are orthogonal matching pursuit (OMP) and subspace pursuit (SP). We will describe these in detail in Section 2.3.

Among the GP algorithms, there are two main methods to estimate the support set. The first method operates by iteratively detecting support-set components to be added to the support-set estimate one at a time until the support-set is finally considered full. We refer to this class of algorithms as serial pursuit (S-pursuit) algorithms because of their strategy of serially choosing new components to the support-set estimate. OMP is a typical example of an S-pursuit algorithm. The second support-set estimation method is based on iteratively refining an estimate of the full support-set and to finally stop when the support set no longer improves by performing further iterations. We refer to this class of algorithms as parallel pursuit (P-pursuit) algorithms because of their strategy of choosing the elements of the support-set simultaneously (in parallel). SP is a typical example of a P-pursuit algorithm.

We will now turn our attention to the two major tools for the GP algorithms, namely the MF and the LS. Then, we will describe the two algorithms OMP and SP and end the section with some performance measurements discussion.

2.1 Matched Filter Detection

Assuming $\mathbf{A}$ has normalized columns, the matched filter (MF) detector is used as a tool for detecting active components in $\mathbf{x}$. It is referred to as MF because the filter is matched to the signal of interest for detection. In our model (1.2), the signals of interest turn out to be the column vectors in $\mathbf{A}$, which can be understood by the following relation:

$$\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{e} = \mathbf{A}_{\mathcal{T}}\mathbf{x}_{\mathcal{T}} + \mathbf{e} = \sum_{\forall i \in \mathcal{T}} \mathbf{a}_i x_i + \mathbf{e}, \tag{1.13}$$

where $\mathbf{a}_i$ is the i'th column in $\mathbf{A}$. It is clear from (1.13) that $\mathbf{y}$ is a weighted sum of the active column-vectors in $\mathbf{A}$ (and noise). The task of the matched filter is to detect if a column-vector (signal) i is present in $\mathbf{y}$ or not.

Definition 6 (Matched filter detection, MF) *Let $\mathbf{A}$ be a matrix with normalized columns $\mathbf{a}_i$ and $\mathbf{y}$ the measurement vector as in Definition 2. Then the matched filter detection is based on calculating the absolute correlation-vector*

$$|\mathbf{a}_i^T \mathbf{y}|, \ \forall i, \tag{1.14}$$

and selecting one or many indices corresponding to the largest components.

Based on the amplitude of the resulting vector, we select the column (signal) in $\mathbf{A}$ that is maximally correlated with $\mathbf{y}$. Let us study component i in the measurement space:

$$\mathbf{a}_i^T \mathbf{y} + \mathbf{a}_i^T \mathbf{e} = \underbrace{\|\mathbf{a}_i\|_2}_{=1} \|\mathbf{y}\|_2 \cos(\phi_i) + \mathbf{a}_i^T \mathbf{e}, \tag{1.15}$$

where ϕ_i is the angle between the vectors $\mathbf{a}_i$ and $\mathbf{y}$. This equation gives the geometrical interpretation that the MF will pick the index i corresponding to the vector $\mathbf{a}_i$ with the smallest angle ϕ_i (more precisely the smallest $\phi_i \pm \pi$, since cos is periodic with π), to $\mathbf{y}$ in the measurement space (assuming that $\mathbf{a}_i \mathbf{e} \approx \mathbf{a}_j \mathbf{e} \ \forall \ i, j$). The MF does not always work good in CS and to better understand the reason for this, we study component i from the MF in the signal space:

$$\begin{aligned} \mathbf{a}_i^T \mathbf{y} + \mathbf{a}_i^T \mathbf{e} &= \mathbf{a}_i^T \sum_j \mathbf{a}_j x_j + \mathbf{a}_i^T \mathbf{e} && (1.16) \\ &= \underbrace{\mathbf{a}_i^T \mathbf{a}_i}_{=1} x_i + \sum_{i \neq j} \mathbf{a}_i^T \mathbf{a}_j x_j + \mathbf{a}_i^T \mathbf{e} && (1.17) \\ &\approx x_i. && (1.18) \end{aligned}$$

Since $\mathbf{A}$ is not an orthogonal matrix, we notice that the second term in (1.17) (referred to as correlation noise) is not equal to 0. However, we argue that the correlation noise is small because $\mathbf{a}_i$ and $\mathbf{a}_j$ (for $i \neq j$) are independent, normalized, vectors with zero-mean Gaussian vector components. If also the measurement noise is small (third term in (1.17)) we see that a rough estimate of x_i is obtained directly from the MF. However, because of the two above mentioned noise terms the MF is in the GP algorithms usually not directly used as an estimator but instead as a detector for the support-set $\mathcal{T}$. Once $\mathcal{T}$ is known, instead the LS is used to estimate the active components in $\mathbf{x}$.

2.2 Least Squares Estimation

As mentioned earlier, least squares estimation (LS) is one of the main tools for many GP algorithms. The full standard CS problem could not be solved with LS because that problem represents an under-determined set of linear equations from which it is not clear which solution to choose. However, suppose one accurately detects the true support-set $\mathcal{T}$ of $\mathbf{x}$, then it is in the noiseless case straightforward to see that $\mathbf{A}\mathbf{x} = \mathbf{A}_{\mathcal{T}}\mathbf{x}_{\mathcal{T}}$. Here, $\mathbf{y} = \mathbf{A}_{\mathcal{T}}\mathbf{x}_{\mathcal{T}}$ represents an over-determined set of linear equations, to which LS can be used to find a unique solution. By limiting ourselves to find $\mathbf{x}_{\mathcal{T}}$, we can then reconstruct the full $\mathbf{x}$ by padding zeros in the remaining positions (i.e., $\mathbf{x}_{\bar{\mathcal{T}}} = \mathbf{0}$).

Definition 7 (Least squares estimation, LS) *The least squares estimation of the signal is the $\hat{\mathbf{x}}_{\mathcal{T}}$ minimizing the following problem*

$$\min_{\hat{\mathbf{x}}_{\mathcal{T}}} \|\mathbf{y} - \mathbf{A}_{\mathcal{T}}\hat{\mathbf{x}}_{\mathcal{T}}\|_2^2. \tag{1.19}$$

The LS is a quadratic problem and we can find the solution by setting the derivative with respect to $\hat{\mathbf{x}}_{\mathcal{T}}$ to zero

$$\begin{aligned} 0 &= \frac{\partial}{\partial \hat{\mathbf{x}}_{\mathcal{T}}} \|\mathbf{y} - \mathbf{A}_{\mathcal{T}}\hat{\mathbf{x}}_{\mathcal{T}}\|_2^2 & (1.20)\\ &= \frac{\partial}{\partial \hat{\mathbf{x}}_{\mathcal{T}}} \left\{ (\mathbf{y} - \mathbf{A}_{\mathcal{T}}\hat{\mathbf{x}}_{\mathcal{T}})^T (\mathbf{y} - \mathbf{A}_{\mathcal{T}}\hat{\mathbf{x}}_{\mathcal{T}}) \right\} & (1.21)\\ &= \frac{\partial}{\partial \hat{\mathbf{x}}_{\mathcal{T}}} \left\{ \mathbf{y}^T\mathbf{y} - \hat{\mathbf{x}}_{\mathcal{T}}^T\mathbf{A}_{\mathcal{T}}^T\mathbf{y} - \mathbf{y}^T\mathbf{A}_{\mathcal{T}}\hat{\mathbf{x}}_{\mathcal{T}} + \hat{\mathbf{x}}_{\mathcal{T}}^T\mathbf{A}_{\mathcal{T}}^T\mathbf{A}_{\mathcal{T}}\hat{\mathbf{x}}_{\mathcal{T}} \right\} & (1.22)\\ &= -2\mathbf{A}_{\mathcal{T}}^T\mathbf{y} + 2\mathbf{A}_{\mathcal{T}}^T\mathbf{A}_{\mathcal{T}}\hat{\mathbf{x}}_{\mathcal{T}}. & (1.23) \end{aligned}$$

Equation (1.23) is still a general result and the solution can be found using the Moore-Penrose inverse. In this thesis we are only interested in the case where $\mathbf{A}_{\mathcal{T}}$ has full column rank (with probability one since $a_{m,n} \sim \mathcal{N}(0,1)$), in which case the result can be obtained from

$$\hat{\mathbf{x}}_{\mathcal{T}} = \mathbf{A}_{\mathcal{T}}^{\dagger}\mathbf{y} = \left(\mathbf{A}_{\mathcal{T}}^T\mathbf{A}_{\mathcal{T}}\right)^{-1}\mathbf{A}_{\mathcal{T}}^T\mathbf{y}, \tag{1.24}$$

where we have that $\mathbf{A}_{\mathcal{T}}^{\dagger} = \left(\mathbf{A}_{\mathcal{T}}^T\mathbf{A}_{\mathcal{T}}\right)^{-1}\mathbf{A}_{\mathcal{T}}^T$. In practice, one does not have access to the true support-set $\mathcal{T}$ of $\mathbf{x}$ but instead an estimate $\hat{\mathcal{T}}$. However, if one indeed accurately detects the true support-set $\mathcal{T}$ and there is no noise present, then (1.24) gives the values of the active components in $\mathbf{x}$, i.e. $\hat{\mathbf{x}}_{\mathcal{T}} = \mathbf{x}_{\mathcal{T}}$. When this happens, (1.19) becomes strictly 0. Notice that by further multiplying the result with $\mathbf{A}_{\mathcal{T}}$, (i.e., $\mathbf{A}_{\mathcal{T}}\mathbf{A}_{\mathcal{T}}^{\dagger}\mathbf{y}$) gives the orthogonal projection of $\mathbf{y}$, denoted by $\mathbf{y}_p$, onto the space spanned by the columns of $\mathbf{A}_{\mathcal{T}}$.

Remark 1.1 In practice, solving a matrix inverse numerically often gives rise to rounding errors. An alternative practical approach is to do QR- or singular value decomposition of $\mathbf{A}$ with pivoting, which is more stable but requires higher computational complexity.

2.3 Algorithms

In this section we will take a closer look at the GP algorithms OMP and SP, which later serve as a platform for other algorithms developed in the thesis. We start by defining two central functions for clarity in the algorithmic notation as follows:

$$\begin{aligned} \texttt{resid}(\mathbf{y}, \mathbf{A}_{\mathcal{T}}) &\triangleq \mathbf{y} - \mathbf{y}_p, \quad \text{where} \quad \mathbf{y}_p = \mathbf{A}_{\mathcal{T}}\mathbf{A}_{\mathcal{T}}^{\dagger}\mathbf{y}, \qquad (1.25) \\ \texttt{max\_indices}(\mathbf{x}, k) &\triangleq \{\text{the set of indices corresponding to the} \\ &\qquad k \text{ largest amplitude components of } \mathbf{x}\}. \qquad (1.26) \end{aligned}$$

The second term in the right hand side of (1.25), $\mathbf{y}_p = \mathbf{A}_{\mathcal{T}}\mathbf{A}_{\mathcal{T}}^{\dagger}\mathbf{y}$, is the orthogonal projection of $\mathbf{y}$ onto the space spanned by the columns of $\mathbf{A}_{\mathcal{T}}$, as mentioned in the previous section. This is a direct result of the LS, which finds the values of the active components $\mathbf{x}_{\mathcal{T}}$.

Orthogonal Matching Pursuit

The orthogonal matching pursuit (OMP) was described by Pati, Rezaiifar and Krishnaprasad in 1993 [22]. In their paper, it is said that a prototype of the OMP algorithm first appeared in the statistics community at some point in the 1950s, where it was called stagewise regression. Since then, it is encountered in many different fields, for example machine learning, applied mathematics, and signal processing. The use of OMP for solving the standard CS problem was studied by Tropp and Gilbert in 2007 [20]. OMP is an example of an S-pursuit algorithm. By assuming a normalized $\mathbf{A}$, it can be described as in Algorithm 1.

In Algorithm 1, in the k'th *iteration* stage, OMP forms the MF detection, identifies the index corresponding to the largest amplitude (step 3), and adds this to the support-set estimate (step 4). It proceeds by computing the residual vector $\mathbf{r}_k$ by from $\mathbf{y}$ removing the projection-vector (i.e., $\mathbf{y} - \mathbf{A}_{\mathcal{T}}\mathbf{A}_{\mathcal{T}}^{\dagger}\mathbf{y}$) (step 5). The residual vector is the vector pointing from $\mathbf{y}_p = \mathbf{A}_{\mathcal{T}}\mathbf{A}_{\mathcal{T}}^{\dagger}\mathbf{y}$ to $\mathbf{y}$; if the $\mathcal{T}$ is accurately detected, $\mathbf{y}_p = \mathbf{0}$. Thus, the residual norm $\|\mathbf{r}\|_2$ is also used as a stopping-criterion since it should never grow, unless the algorithm picks very bad support-set components. This process is then repeated until a total of $K_{\max}$ components have been picked in the support-set at which point OMP halts.

Algorithm 1 : Orthogonal matching pursuit (OMP)

```
Input: $\mathbf{A}$, $K_{\max}$, $\mathbf{y}$
Initialization:
1: $k \leftarrow 0$
2: $\mathbf{r}_0 \leftarrow \mathbf{y}$, $\mathcal{T}_0 \leftarrow \emptyset$
Iteration:
1: repeat
2:     $k \leftarrow k + 1$
3:     $\tau_{\max} \leftarrow \texttt{max\_indices}\left(\mathbf{A}^T \mathbf{r}_{k-1}, 1\right)$
4:     $\mathcal{T}_k \leftarrow \mathcal{T}_{k-1} \cup \tau_{\max}$
5:     $\mathbf{r}_k \leftarrow \texttt{resid}(\mathbf{y}, \mathbf{A}_{\mathcal{T}_k})$
6: until $(k > K_{\max})$ or $(\|\mathbf{r}_k\|_2 > \|\mathbf{r}_{k-1}\|_2)$
Output:
1: $\hat{\mathcal{T}} \leftarrow \mathcal{T}_{k-1}$
2: $\hat{\mathbf{x}}$ such that $\hat{\mathbf{x}}_{\mathcal{T}_{k-1}} = \mathbf{A}^{\dagger}_{\mathcal{T}_{k-1}} \mathbf{y}$ and $\hat{\mathbf{x}}_{\bar{\mathcal{T}}_{k-1}} = \mathbf{0}$
3: $n_r \leftarrow \|\mathbf{r}_{k-1}\|_2$
```

Example 1.2 (A simple example of OMP) For better understanding the principle of OMP, we provide a simple noiseless example. Assume the following data is given to the algorithm:

$$\mathbf{A} = \begin{bmatrix} 0.4033 & 0.3257 & -0.0198 \\ 0.9150 & 0.9455 & -0.9998 \end{bmatrix}, \qquad \mathbf{y} = \begin{bmatrix} 0.9307 \\ -0.4271 \end{bmatrix}.$$

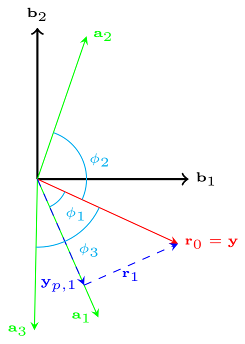
The corresponding signal- and measurement space are shown in Figure 1.2, where in Figure 1.2b, the sought signal $\mathbf{x}$ (in red) is shown as a reference. In Figure 1.2a, the column vectors $\mathbf{a}_1, \mathbf{a}_2$ and $\mathbf{a}_3$ from $\mathbf{A}$ are shown in green and the measurement vector $\mathbf{y}$ in red.

Starting OMP, the initialization phase of the algorithm is executed: $k \leftarrow 0$, $\mathbf{r}_0 \leftarrow \mathbf{y}$ and $\mathcal{T}_0 \leftarrow \emptyset$. It then proceeds to the first iteration:

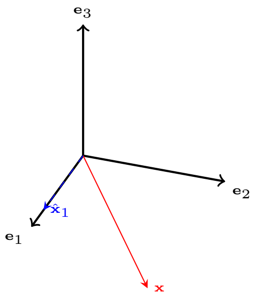
- Step 2 in Algorithm 1, $k \leftarrow k + 1$ gives $k = 1$.
- Step 3 in Algorithm 1, the residual vector $\mathbf{r}_0$ (which is $\mathbf{r}_0 = \mathbf{y}$ from the initialization) is correlated with every column-vector in $\mathbf{A}$:

$$\mathbf{A}^T \mathbf{r}_0 = \mathbf{A}^T \mathbf{y} = \begin{bmatrix} 0.4033 & 0.3257 & -0.0198 \\ 0.9150 & 0.9455 & -0.9998 \end{bmatrix}^T \begin{bmatrix} 0.9307 \\ -0.4271 \end{bmatrix} = \tag{1.27}$$

$$= \begin{bmatrix} \|\mathbf{y}\|_2 \cos(\phi_1) \\ \|\mathbf{y}\|_2 \cos(\phi_2) \\ \|\mathbf{y}\|_2 \cos(\phi_3) \end{bmatrix} = \begin{bmatrix} 0.7662 \\ -0.1007 \\ 0.4086 \end{bmatrix}.$$



(a) Measurement space $\mathbb{R}^2$



(b) Signal space $\mathbb{R}^3$

Figure 1.2: The signal- and measurement-space during the first iteration of OMP.

Consequently, the index corresponding to the maximum-in-amplitude value is chosen by `max_indices`(.,.) and found to be $\tau \leftarrow 1$. We can verify the result by studying Figure 1.2a, where we see that the index corresponding to the vector $\mathbf{a}_1$ gives the smallest angle ϕ_1.

- Step 4 in Algorithm 1, the support set becomes $\mathcal{T}_1 = \mathcal{T}_0 \cup \tau = \emptyset \cup 1 = \{1\}$.

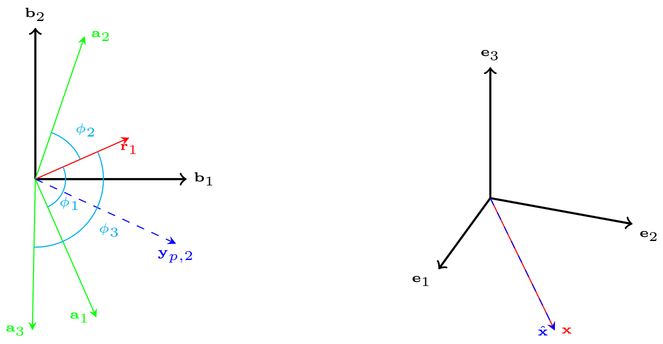
- Step 5 in Algorithm 1, the new residual-vector is achieved via `resid`(.,.). We show how it is done in two steps according to (1.25):

  (i) $\mathbf{y}_p = \mathbf{A}_\mathcal{T}\mathbf{A}_\mathcal{T}^\dagger \mathbf{y} = \mathbf{a}_1\mathbf{a}_1^\dagger \mathbf{y} = [0.3090, -0.7011]^T$, which is shown in Figure 1.2a as $\mathbf{y}_{p,1}$ ('1' because it is the first iteration).

  (ii) Now, the new residual is derived from $\mathbf{r}_1 = \mathbf{y} - \mathbf{y}_{p,1} = [0.6217, 0.2740]^T$ which is also shown in Figure 1.2a.

The first iteration by OMP is now finished and we can show how $\hat{\mathbf{x}}$ would look like in the signal space (referred to as $\hat{\mathbf{x}}_1$ in Figure 1.2b) if the algorithm stopped here. One point with $\hat{\mathbf{x}}_1$ at this stage is that we can verify that OMP found the dominating base vector $\mathbf{e}_1$ of $\mathbf{x}$. We now proceed to the second iteration in Figure 1.3.

- Step 2 in Algorithm 1, $k \leftarrow k + 1$ gives $k = 2$.



(a) Measurement space $\mathbb{R}^2$ (b) Signal space $\mathbb{R}^3$

Figure 1.3: The signal- and measurement-space in the second iteration of OMP.

- Step 3 in Algorithm 1, the residual vector $\mathbf{r}_1$ (from previous iteration) is correlated with every column-vector in $\mathbf{A}$:

$$\mathbf{A}^T\mathbf{r}_1 = \begin{bmatrix} 0.4033 & 0.3257 & -0.0198 \\ 0.9150 & 0.9455 & -0.9998 \end{bmatrix}^T \begin{bmatrix} 0.6217 \\ 0.2740 \end{bmatrix} = \\ = \begin{bmatrix} \|\mathbf{r}_1\|_2 \cos(\phi_1) \\ \|\mathbf{r}_1\|_2 \cos(\phi_2) \\ \|\mathbf{r}_1\|_2 \cos(\phi_3) \end{bmatrix} = \begin{bmatrix} 0.0000 \\ 0.4615 \\ -0.2862 \end{bmatrix}. \qquad (1.28)$$

  Note that the first element in (1.28) is 0 since $\mathbf{r}_1$ is orthogonal to $\mathbf{a}_1$. Thus, `max_indices`$(.,.)$ chooses $\tau \leftarrow 2$, which can be verified in Figure 1.3a as the index corresponding to the vector $\mathbf{a}_2$ (since ϕ_2 is the smallest angle).

- Step 4 in Algorithm 1, the support-set becomes $\mathcal{T}_2 = \mathcal{T}_1 \cup \tau = \{1\} \cup 2 = \{1, 2\}$.

- Step 5 in Algorithm 1, the new residual vector is achieved via `resid`$(.,.)$. We show how it is done in two steps according to (1.25):

  (i) $\mathbf{y}_p = \mathbf{A}_{\mathcal{T}}\mathbf{A}_{\mathcal{T}}^{\dagger}\mathbf{y} = [\mathbf{a}_1, \mathbf{a}_2][\mathbf{a}_1, \mathbf{a}_2]^{\dagger}\mathbf{y} = [0.9307, -0.4271]^T$, which is shown in Figure 1.3a as $\mathbf{y}_{p,2}$. Note that $\mathbf{y}_{p,2} = \mathbf{y}$ because $\mathbf{a}_1$ and $\mathbf{a}_2$ spans $\mathbb{R}^2$.

  (ii) Now, the new residual is derived from $\mathbf{r}_2 = \mathbf{y} - \mathbf{y}_{p,1} = [0.0000, 0.0000]^T$.

The second iteration of OMP is now finished and we note that the estimate $\hat{\mathbf{x}}$ is a perfect recovery of $\mathbf{x}$. The algorithm will according to the stopping criterion iterate one more time. But the result from this last iteration will be discarded according to the output of Algorithm 1. $\diamond$

OMP has similar performance properties as the convex relaxation based methods and this is studied by for example Tropp [23] and Donoho *et. al.* [24]. We now move on to describe the subspace pursuit GP algorithm.

Subspace Pursuit

The subspace pursuit (SP) algorithm was developed by Dai and Milenkovic [21] and published in 2009. This algorithm is a P-pursuit algorithm and according to the original paper with the underlying algorithmic principle borrowed from the A^* order-statistic algorithm [25]. Another algorithm, called CoSaMP [26] closely resembles SP and was developed simultaneously as the latter. The difference between these two algorithms lies in the extension phase, which we will get back to in the description of SP. Each iteration in SP requires more computational effort than each iteration in OMP, but it turns out that in practice SP iterates fewer times. By experimental evaluations it can be observed that the two GP algorithms SP and OMP provide for similar execution times and similar performance. We show the SP algorithm in Algorithm 2.

In the *initialization* phase of Algorithm 2, SP uses the $K_{\max}$ largest-in-amplitude components from the matched filter as a first estimate of the support-set $\mathcal{T}_0$. Also the iteration counter k is reset and the first residual $\mathbf{r}_0$ is calculated.

In the k'th *iteration*, SP evaluates the correlation filter, $\mathbf{A}^T\mathbf{r}_{k-1}$, identifies the indices corresponding to the $K_{\max}$ largest amplitudes and stores this in a temporary set, $\mathcal{T}_\Delta$ (step 3). Then, the union between the old support-set and $\mathcal{T}_\Delta$ is formed in step 4. We call this the extension phase and notice that the support-set $\mathcal{T}'$ is at most of size $2K_{\max}$[2]. The algorithm solves a LS problem with the selected indices of $\mathcal{T}'$ (requirement: $|\mathcal{T}'| \leq M$) and identifies a new support-set corresponding to the $K_{\max}$ largest amplitudes in $\tilde{\mathbf{x}}$ (step 5 and 6). The algorithm then computes the residual (step 7), which is used as a stopping-criterion for the iteration phase.

Example 1.3 (Support-set evolution in sp) The SP algorithm uses

[2]Here we note the difference to CoSaMP, where instead the algorithm picks $2K_{\max}$ components and thus, for CoSaMP, the support-set $\mathcal{T}'$ is at most of size $3K_{\max}$.

Algorithm 2 : Subspace pursuit SP

```
Input: $\mathbf{A}$, $K_{\max}$, $\mathbf{y}$
Initialization:
1: $\mathcal{T}_0 \leftarrow \texttt{max\_indices}\left(\mathbf{A}^T\mathbf{y}, K_{\max}\right)$
2: $\mathbf{r}_0 \leftarrow \texttt{resid}(\mathbf{y}, \mathbf{A}_{\mathcal{T}_0})$
3: $k \leftarrow 0$
Iteration:
1: **repeat**
2:     $k \leftarrow k + 1$
3:     $\mathcal{T}_\Delta \leftarrow \texttt{max\_indices}\left(\mathbf{A}^T\mathbf{r}_{k-1}, K_{\max}\right)$
4:     $\mathcal{T}' \leftarrow \mathcal{T}_{k-1} \cup \mathcal{T}_\Delta$
5:     $\tilde{\mathbf{x}}$ such that $\tilde{\mathbf{x}}_{\mathcal{T}'} = \mathbf{A}^\dagger_{\mathcal{T}'}\mathbf{y}$ and $\tilde{\mathbf{x}}_{\bar{\mathcal{T}}'} = \mathbf{0}$
6:     $\mathcal{T}_k \leftarrow \texttt{max\_indices}(\tilde{\mathbf{x}}, K_{\max})$
7:     $\mathbf{r}_k \leftarrow \texttt{resid}(\mathbf{y}, \mathbf{A}_{\mathcal{T}_k})$
8: **until** $(\|\mathbf{r}_k\|_2 \geq \|\mathbf{r}k - 1\|_2)$
9: $k \leftarrow k - 1$                                  ('Previous iteration count')
Output:
1: $\hat{\mathcal{T}} \leftarrow \mathcal{T}_k$
2: $\hat{\mathbf{x}}$ such that $\hat{\mathbf{x}}_{\mathcal{T}_k} = \mathbf{A}^\dagger_{\mathcal{T}_k}\mathbf{y}$ and $\hat{\mathbf{x}}_{\bar{\mathcal{T}}_k} = \mathbf{0}$
3: $n_r \leftarrow \|\mathbf{r}_k\|_2$
```

the same algebraic tools as OMP but in a different manner. In particular, instead of growing the support-set estimate with one parameter at a time, SP iteratively refines a full support-set estimate. Having a good understanding of the support-set evolution through the iterations of SP is very helpful in order to understand the SP algorithm literature. Thus, instead of showing a full example of the algorithm (as we did for OMP), we will instead give a schematic sketch in Figure 1.4 with explanation of how the support-set evolves from iteration $k-1$ to k.

- Figure 1.4, part 1: Based on the previous residual $\mathbf{r}_{k-1}$, which in turn depends on the previous support-set estimate $\mathcal{T}_{k-1}$, we find $K_{\max}$ new indices in a temporary support-set $\mathcal{T}_\Delta$. The corresponding function f_1 is

$$f_1(\mathcal{T}_{k-1}) = \texttt{max\_indices}(\mathbf{A}^T\texttt{resid}(\mathbf{y}, \mathbf{A}_{\mathcal{T}_{k-1}}), K_{\max}),$$

  which is done in steps 7 and 3 in the iteration phase of Algorithm 2.

- Figure 1.4, part 2: The previous support-set $\mathcal{T}_{k-1}$ is merged with $\mathcal{T}_\Delta$ to form a bigger intermediate support-set $\mathcal{T}'$. This corresponds to step 4 in the iteration phase of Algorithm 2.

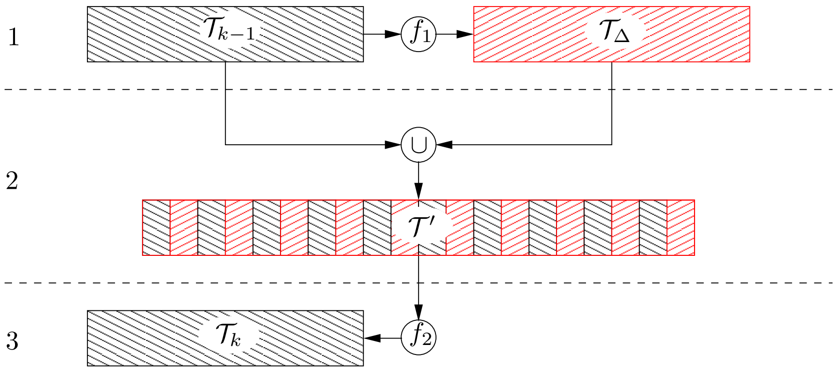


Figure 1.4: Support-set evolution in subspace pursuit

- Figure 1.4, part 3: We choose the $K_{\max}$ indices from $\mathcal{T}'$ that contributes to the largest-in-amplitude components in $\tilde{\mathbf{x}}$ and let these components be $\mathcal{T}_k$. The corresponding non-linear function f_2 is

$$f_2(\mathcal{T}') = \texttt{max\_indices}(\tilde{\mathbf{x}}, K_{\max}),$$

where $\tilde{\mathbf{x}}$ is chosen so that $\tilde{\mathbf{x}}_{\mathcal{T}'} = \mathbf{A}^{\dagger}_{\mathcal{T}'}\mathbf{y}$ and $\tilde{\mathbf{x}}_{\bar{\mathcal{T}}'} = \mathbf{0}$. This corresponds to step 5 and 6 in Algorithm 2. $\diamond$

Just as for the convex relaxation methods and OMP, substantial amount of effort has been put in evaluating the performance and requirements of SP. Already in the original paper careful analysis for both the noisy and noiseless case [21] was presented, but these results have further been improved, in for example [9]. These analyses provide for analytical performance based on requirements of noise-level, sparsity and RIP but it turns out that in practice they are difficult to use. In this thesis, we instead rely on carefully executed empirical results.

2.4 Performance Measures

In this thesis we will evaluate the performance of different algorithms by experimental evaluation. We take this engineering approach despite the fact that there are analytical methods (see Section 1) and performance bounds on the algorithms because it turns out that in practice, these analytical approaches generally provide too loose bounds. Furthermore, from an engineering perspective, perfect recovery is often not required. Instead one may want to be able to specify some required performance by a particular measure. We describe two performance measures that are repeatedly used throught the thesis. The first measure is the signal-to-reconstruction-error-ratio, which is measured in dB.

Definition 8 (Signal-to-reconstruction-error-ratio, srer) *This measure is in the literature sometimes also referred to as signal-to-reconstruction-noise-ration (*SRNR*).*

$$\text{SRER} = 10 \log \frac{\mathbb{E}\left\{\|\mathbf{x}\|_2^2\right\}}{\mathbb{E}\left\{\|\mathbf{x} - \hat{\mathbf{x}}\|_2^2\right\}}. \tag{1.29}$$

The second measure presented here is based on the insight that the GP algorithms work with the support-set as their main tools and thus the support-set distortion gives valuable information. We call this measure the average support-set cardinality error.

Definition 9 (Average support-set cardinality error, asce) *The* ASCE *is a performance measure that measures the distortion between $\mathcal{T}$ and $\hat{\mathcal{T}}$ as*

$$\text{ASCE} = \mathbb{E}\left\{1 - \frac{|\mathcal{T} \cap \hat{\mathcal{T}}|}{|\mathcal{T}|}\right\}. \tag{1.30}$$

Based on these performance measures, a performance request from a user could for example be average SRER $\geq$ 10 dB in a scenario where the signal-to-measurement-noise-ratio (SMNR) is 20 dB (SMNR $= 10 \log_{10} \frac{\mathbb{E}\{\|\mathbf{x}\|_2^2\}}{\mathbb{E}\{\|\mathbf{e}\|_2^2\}}$). Another performance request could be that in average 50% (i.e., ASCE $= 0.5$) of the support-set should accurately be determined. This is an engineering approach and we can not apply the present theory which only provides bounds for perfect recovery. Developing new theory that describe imperfect scenarios is challenging and we will in this thesis instead rely on experimental evaluation of ASCE and SRER.

3 Distributed Compressed Sensing

From the standard CS problem, we can form the distributed CS problem, which is a problem with jointly correlated data.

Definition 10 (The distributed cs problem) *For each sensor node l out of a total L nodes, find the sparse signal vector $\mathbf{x}_l \in \mathbb{R}^N$ provided the measurement-vector $\mathbf{y}_l \in \mathbb{R}^M$, the measurement-matrix $\mathbf{A}_l \in \mathbb{R}^{M \times N}$ and the under-determined set of linear equations*

$$\mathbf{y}_l = \mathbf{A}_l \mathbf{x}_l + \mathbf{e}_l, \qquad \forall l \in \{1, 2, ..., L\}, \tag{1.31}$$

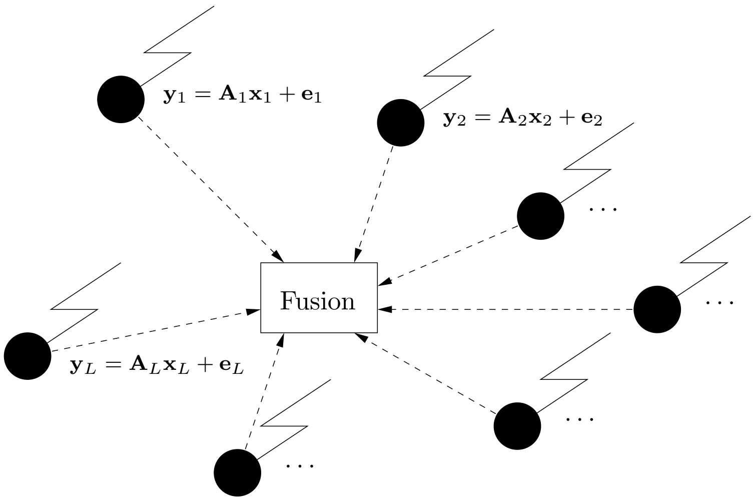

$\mathbf{y}_1 = \mathbf{A}_1\mathbf{x}_1 + \mathbf{e}_1$
$\mathbf{y}_2 = \mathbf{A}_2\mathbf{x}_2 + \mathbf{e}_2$
Fusion
$\mathbf{y}_L = \mathbf{A}_L\mathbf{x}_L + \mathbf{e}_L$
...

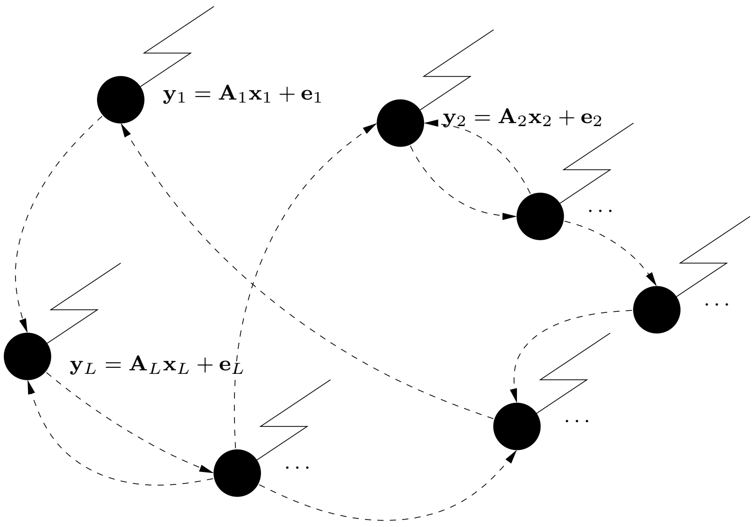
Figure 1.5: An example of a centralized network with a fusion center.

where $(e_i)_l \sim \mathcal{N}(0, \sigma_l)$ *represents the measurement noise. The number of non-zero components in* $\mathbf{x}_l$ *are* $\|\mathbf{x}_l\|_0 = K_l$*, where* $K_l < M_l < N_l$*.*

We further assume that $\mathbf{A}_l$ is constructed by first picking the elements $(a_{m,n})_l \sim \mathcal{N}(0, 1)$ and then normalize each column $(\mathbf{a}_n)_l$ in $\mathbf{A}_l$ to unit length. The idea of distributed CS is to improve the performance of the standard CS reconstruction performance by exchanging appropriate information among the nodes, so that some assumed correlation among the sensor nodes in Definition 10 can be exploited.

3.1 Centralized Solver

The most straight forward way of realizing a distributed CS problem is through a centralized solver, where a fusion center has knowledge of all data achieved at multiple sensor-nodes. A centralized solver will in general provide the best possible result since it has the most possible knowledge. The model can also be applied to a scenario where one single sensor is tracking an over time slowly varying signal. Then, every measurement is slightly different and can be modeled as different nodes in the network. Figure 1.5 shows a schematical picture for a typical centralized network.



$\mathbf{y}_1 = \mathbf{A}_1\mathbf{x}_1 + \mathbf{e}_1$
$\mathbf{y}_2 = \mathbf{A}_2\mathbf{x}_2 + \mathbf{e}_2$
$\mathbf{y}_L = \mathbf{A}_L\mathbf{x}_L + \mathbf{e}_L$
...
...
...
...

Figure 1.6: An example of a decentralized network.

3.2 Distributed Solver

In a distributed netwok there may not be any fusion center available. Then each node in a network may independently solve the distributed CS problem with (or without) help of its neighbors. This is attractive for self-arranging networks, for example cognitive radio networks [27,28]. These distributed nodes need in general more computational capability compared to the sensing nodes in a centralized network but in turn, there is no need of a central fusion center. Figure 1.6 shows a schematical picture for a typical distributed network.

4 Contributions

In this section we enlist the contributions of this thesis. The section is divided into three parts: First we present the included papers, then we present some new additional comparisons between material from Paper B and Paper C, not already present in the articles, and at last we list work that is not included in the thesis.

Paper A:

In a scenario where a cognitive radio unit wishes to transmit, it needs to know over which frequency bands it can operate. It can obtain this knowledge by estimating the power spectral density from a Nyquist-rate sampled signal. For wide-band signals sampling at the Nyquist rate is a major challenge and may be unfeasible. In this paper we accurately detect spectrum holes in sub-Nyquist frequencies without assuming wide sense stationarity in the compressed sampled signal. A novel extension to further reduce the sub-Nyquist samples is then presented by introducing a memory based compressed sensing that relies on the spectrum to be slowly varying.

[29] D. Sundman, S. Chatterjee, and M. Skoglund, "On the use of compressive sampling for wide-band spectrum sensing," in *IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)*, Luxor, Egypt, Dec. 2010, pp. 354–359

Paper B:

In this paper, we want to improve the recovery performance of the existing orthogonal matching pursuit (OMP) algorithm. To achieve a better estimate of the underlying support set progressively through iterations, we use a look ahead strategy. The choice of an atom in the current iteration is performed by checking its effect on the future iterations (look ahead strategy). Through experimental evaluations, the effect of look ahead strategy is shown to provide a significant improvement in performance. This new algorithm is referred to as look ahead orthogonal matching pursuit (LAOMP).

[30] S. Chatterjee, D. Sundman, and M. Skoglund, "Look ahead orthogonal matching pursuit," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Prague, Czech Republic, May 2011, pp. 4024–4027

Paper C:

We want to improve the reconstruction performance of the parallel pursuit algorithms SP and CoSaMP. In an iteration, standard parallel pursuit algorithms use a support-set expansion by a fixed number of coefficients, leading to restricted performance. To achieve a better performance, we develop a look ahead strategy that adaptively chooses the best number of coefficients in each extension. We develop a new algorithm which we call look ahead parallel pursuit, where a look ahead strategy is invoked on a minimal residual norm criterion. The new algorithm provides a trade-off between performance and complexity and is referred to as look ahead parallel pursuit (LAPP).

[31] D. Sundman, S. Chatterjee, and M. Skoglund, "Look ahead parallel pursuit," in *IEEE Swedish Communication Technologies Workshop (Swe-CTW)*, Stockholm, Sweden, Oct. 2011, pp. 114–117

Paper D:

For compressed sensing with jointly sparse signals, we present a new signal model and two new joint iterative-greedy-pursuit recovery algorithms. The signal model is based on the assumption of a jointly shared support-set and the joint recovery algorithms have knowledge of the size of the shared support-set. Through experimental evaluation, we show that the new joint algorithms provide significant performance improvements compared to regular algorithms which do not exploit a joint sparsity.

[32] D. Sundman, S. Chatterjee, and M. Skoglund, "Greedy pursuits for compressed sensing of jointly sparse signals," in *EURASIP European Signal Processing Conference (EUSIPCO)*, Barcelona, Spain, Aug. 2011, pp. 368–372

Paper E:

In distributed compressed sensing, we consider the setup where several sensor nodes are connected through a decentralized (distributed) network; the network can have an arbitrarily connected topology. For this setup, we first introduce a new signal model that helps to exploit correlations between the underlying signals acquired at multiple sensor nodes. Based on this signal model, we then develop new distributed greedy pursuit algorithms for the distributed setup. Incorporating appropriate modifications, we extend the existing orthogonal matching pursuit (OMP) and subspace pursuit (SP) algorithms for designing two new distributed algorithms. We also develop a new greedy pursuit algorithm which we use for designing a third distributed algorithm. Through simulations, we evaluate the new algorithms and show that the algorithms can provide a performance close to that of a centralized (joint) solution.

[33] D. Sundman, S. Chatterjee, and M.Skoglund, "Greedy pursuits for distributed compressed sensing," *IEEE Transactions on Signal Processing*, 2012, Submitted February 2012

4.1 Comparison between look-ahead algorithms

During the work of producing this thesis it was realized that a performance and complexity comparison between the two algorithms LAOMP (Paper B) and LAPP (Paper C) is of interest. Refer to the corresponding papers for further information regarding these algorithms. As a reference we also provide results for SP and OMP. We compare the performance by ASCE and SRER and the complexity by execution times. We first define the fraction of measurements which is used in the simulations:

$$\alpha = \frac{M}{N}. \tag{1.32}$$

Using α, the testing is performed as follows:

1. Given the signal parameter N, choose an α (such that M is an integer).
2. Randomly generate an $M \times N$ sensing matrix $\mathbf{A}$ where the components are drawn independently from an i.i.d. Gaussian source (i.e. $a_{m,n} \sim \mathcal{N}\left(0, \frac{1}{M}\right))$ and then scale the columns of $\mathbf{A}$ to unit-norm.
3. Generate a support-set $\mathcal{T}$ of cardinality K. The support-set is uniformly chosen from $\{1, 2, ..., N\}$.
4. Randomly generate a sparse signal vector $\mathbf{x}$ with non-zero components determined by the support-set in step 3. The non-zero

components in the vector are chosen independently from a Gaussian source for the Gaussian signals, and chosen as 1 for the binary signal.

5. Compute the measurement vector $\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{w}$, where $\mathbf{w}$ is standard i.i.d. Gaussian noise.
6. Apply the CS algorithms on the data $\mathbf{y}$, $\mathbf{A}$.

In the simulation procedure above, a number Q different sensing matrices $\mathbf{A}$ are created. For each sensing matrix, P data vectors $\mathbf{x}$ are generated. In total, we will average over $Q \cdot P$ data to evaluate the performance.

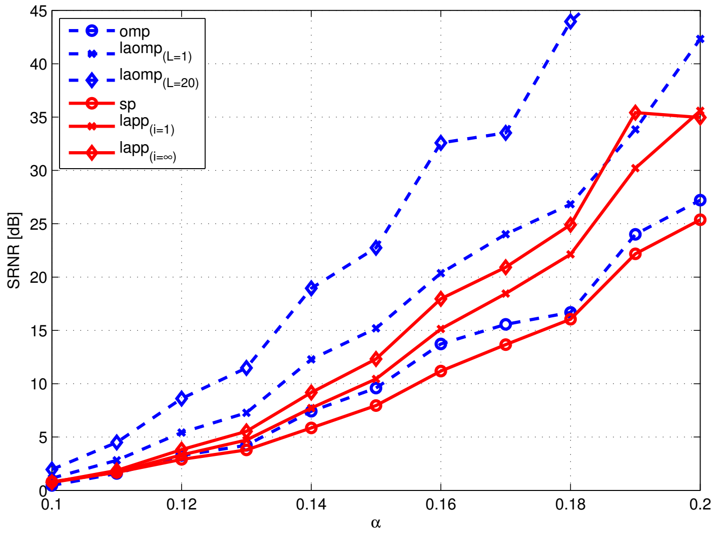
Simulation results

For the plots presented here, we have chosen: $N = 500$, $K = 20$. We have chosen the number of matrices $\mathbf{A}$ to 100 ($Q = 100$) and the number of data-sets $\mathbf{x}$ to 100 ($P = 100$), giving a total number of $Q \cdot P = 10000$ data for evaluating each data point.

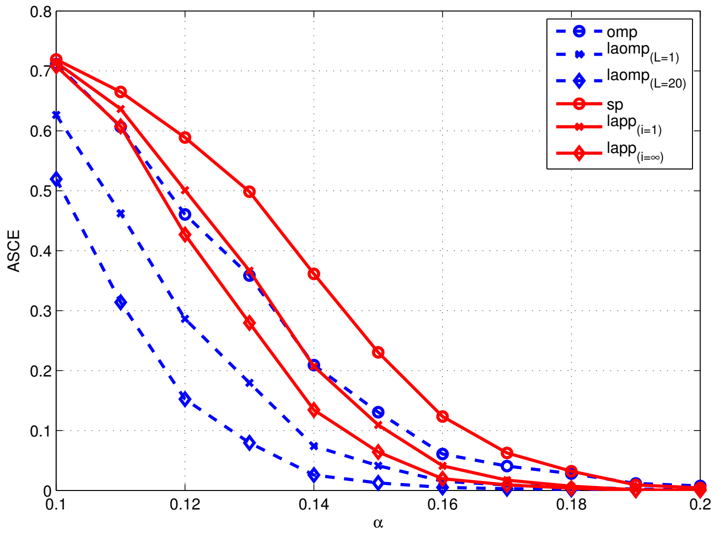
Performance comparison Figure 1.7 shows SRER and ASCE results for a Gaussian sparse signal in a clean measurement condition. In Figure 1.7a we see the red solid lines as the parallel algorithms (SP and LAPP) and the dotted blue as the serial (OMP and LAOMP). It is clear that the two different look-ahead strategies (LAOMP and LAPP) provide better performance than both SP and OMP. From these curves, LAOMP with $L = 20$ clearly performs the best. For example, at $\alpha = 0.16$, OMP is about 2 dB better than SP, while LAPP$_{(i=\infty)}$ performs about 4 dB better than OMP and LAOMP$_{(L=20)}$ performs a whole 15 dB better than LAOMP$_{(L=20)}$. Similar trends in performance are observed in Figure 1.8 for the noisy measurement condition with SMNR = 20 dB. It is, however, interesting to notice that in Figure 1.8 the gap between the LAOMP and LAPP algorithms is much smaller.

Next we provide the performance results for binary sparse signal. Figure 1.9 shows the results at clean measurement condition. In this case the most interesting observation is that LAOMP, which previously performed the best, now at $L = 2$ does not even beat SP and that LAPP for both $i = 1$ and $i = \infty$ beats all other algorithms. For example, in Figure 1.9a at $\alpha = 0.22$, we see that SP beats OMP with about 8 dB, LAOMP$_{(L=20)}$ beats SP with about 4 dB and that LAPP$_{(i=\infty)}$ beats LAOMP$_{(L=20)}$ with about 3 dB. Similar trends in performance are observed in Figure 1.10

Simulation complexity We now study the complexity of the different algorithms OMP, SP LAOMP and LAPP. From [34] and Paper C, we can

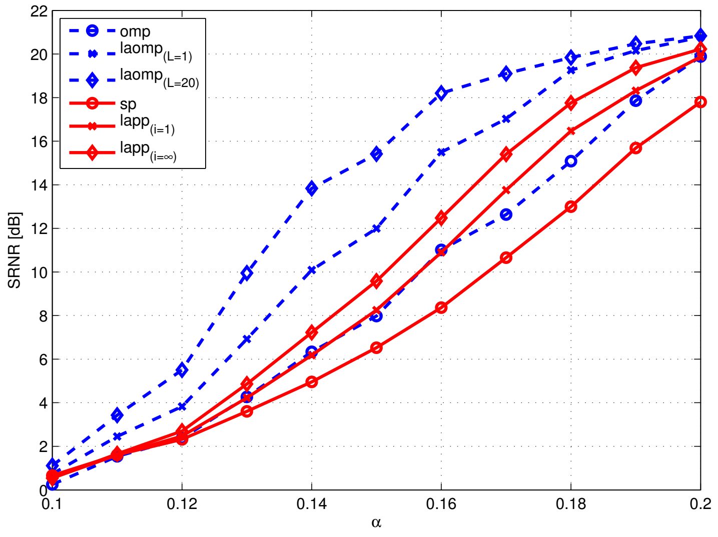


(a) SRER (in dB) vs α

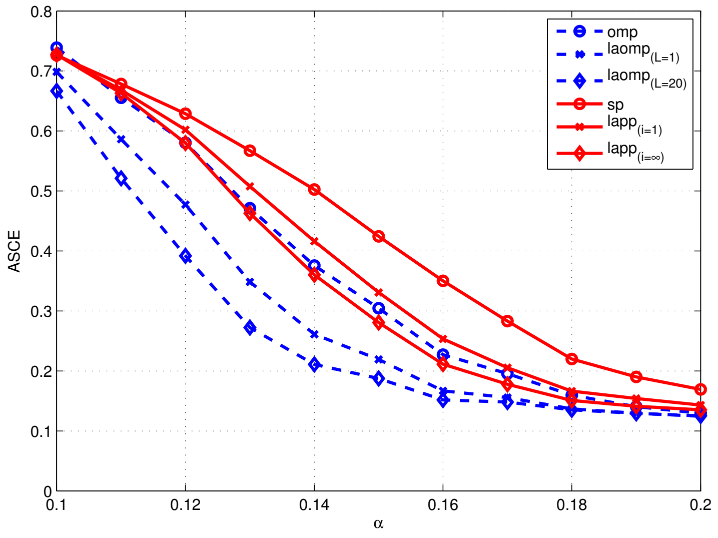


(b) ASCE vs α

Figure 1.7: OMP, SP LAOMP and LAPP algorithms for **Gaussian** sparse signals in **clean measurement** condition.

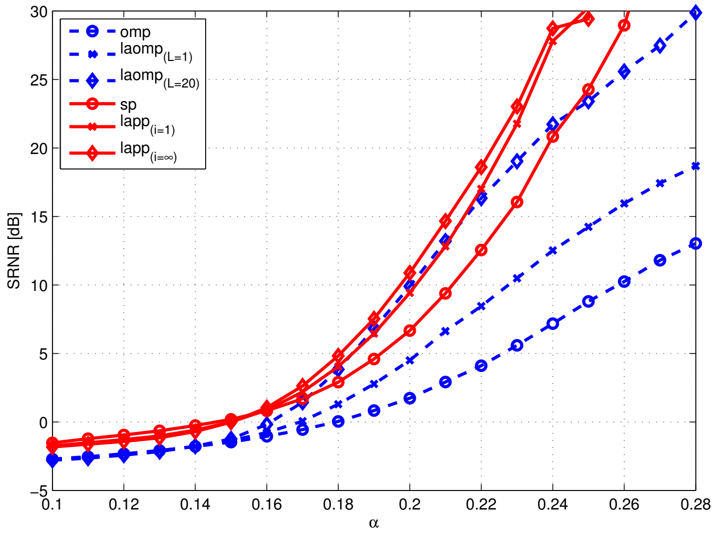


(a) SRER (in dB) vs α

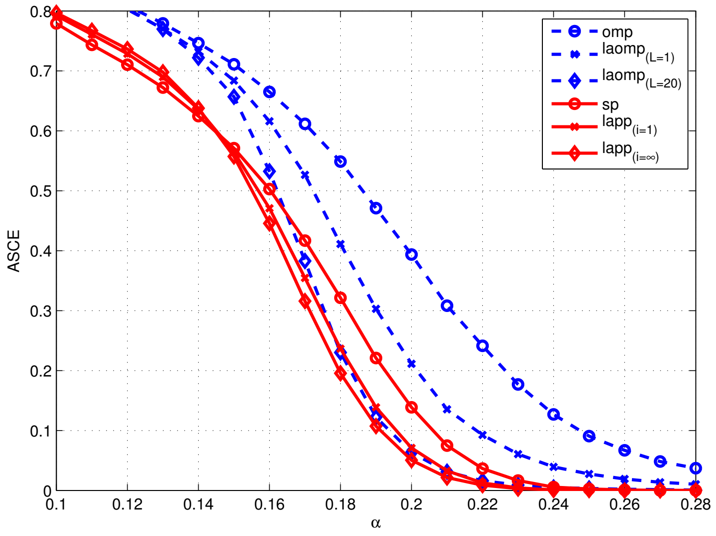


(b) ASCE vs α

Figure 1.8: OMP, SP LAOMP and LAPP algorithms for **Gaussian** sparse signals in **noisy measurement**, where SMNR = 20 dB.

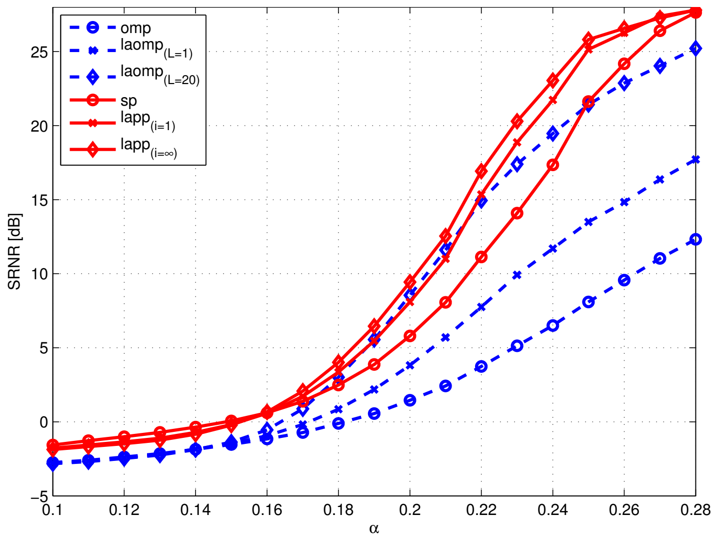


(a) SRER (in dB) vs α

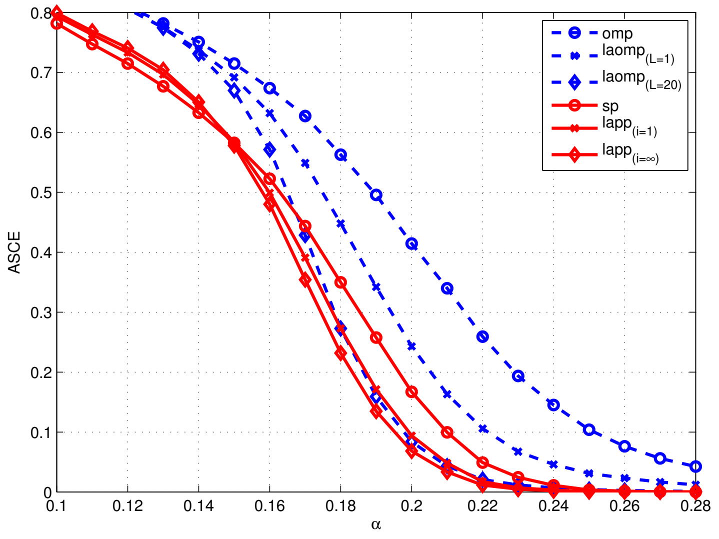


(b) ASCE vs α

Figure 1.9: OMP, SP LAOMP and LAPP algorithms for **binary** sparse signals in **clean measurement** condition.



(a) SRER (in dB) vs α



(b) ASCE vs α

Figure 1.10: OMP, SP LAOMP and LAPP algorithms for **binary** sparse signals in **noisy measurement**, where SMNR = 20 dB.

directly enlist the theoretical worst-case performances of the different algorithms:

- OMP: $\mathcal{O}(K(MN + K^2 + KM) \sim \mathcal{O}(KMN)$
- LAOMP$_{(L=2)}$: $\mathcal{O}(K^2(MN + K^2 + KM)) \sim \mathcal{O}(K^2MN)$
- LAOMP$_{(L=K)}$: $\mathcal{O}(K^3(MN + K^2 + KM)) \sim \mathcal{O}(K^3MN)$
- SP: $\mathcal{O}(K(MN + K^M)) \sim \mathcal{O}(KMN)$
- LAPP$_{(i=1)}$: $\mathcal{O}(K^2(MN + K^M)) \sim \mathcal{O}(K^2MN)$
- LAPP$_{(i=\infty)}$: $\mathcal{O}(K^3(MN + K^M)) \sim \mathcal{O}(K^3MN)$

Notice that the look-ahead algorithms scale poorly in only K, which is desirable since K in general is considered to be very small compared to M and N. We now provide a running time comparison between these algorithms. This running-time comparison provides a rough idea about computational resource. The running time results are shown in Table 1.6. In this case, we performed simulations for Gaussian sparse signal at SMNR $= 20$ dB for different values of α and N. We notice from the table that all the look-ahead algorithms require order of magnitude longer execution time compared to the standard algorithms, and that in all cases LAOMP$_{(L=K)}$ is the computationally most demanding algorithm. Observe that the parameter L changes for LAOMP$_{(L=K)}$ the last set of simulations. We can also clearly observe the trend from the theoretical performances, that all algorithms scale well in N, while they scale bad in K, which is the typical trait for greedy pursuit algorithms. Another interesting trend is that for higher α, it seems that SP and LAPP provides for less computational resource, which is due to the more dynamic convergence criterion.

4.2 Material Not Included in the Thesis

During the course of study leading to this thesis, the author has had the opportunity to work with a wide range of people. The cooperation has produced challenging research which has resulted in some publications which are outside the scope of this thesis. For the interested reader, we provide references to these articles in two following subsections. In the first subsection, we briefly mention a cooperation among the labs Signal processing, Communication theory, Automatic control and Mechanics department, who decided to participate in the competition Grand cooperative driving challenge. In the second subsection, we list additional work produced in collaboration with Dr. Chatterjee, Dr. Vehkaperä and Prof. Skoglund.

Table 1.6: Running time comparison between OMP, LAOMP, SP and LAPP algorithms for varying signal- and measurement-size.

α	OMP	LAOMP$_{(L=2)}$	LAOMP$_{(L=K)}$	SP	LAPP$_{(i=1)}$	LAPP$_{(i=\infty)}$
$N = 500,\ K = 20$						
0.14	0.042	0.854	8.049	0.034	2.287	4.361
0.20	0.047	0.908	8.576	0.041	2.274	3.348
0.26	0.049	0.962	9.090	0.038	2.135	2.676
$N = 1000,\ K = 20$						
0.08	0.051	1.035	9.713	0.040	2.875	5.102
0.14	0.061	1.224	11.51	0.052	2.771	3.537
0.20	0.075	1.474	13.89	0.054	2.857	3.394
$N = 500,\ K = 40$						
0.26	0.136	3.799	72.57	0.1126	15.11	31.20
0.29	0.1471	4.013	77.22	0.1296	15.74	28.03
0.33	0.1586	4.244	81.34	0.1274	15.77	22.87

Grand Cooperative Driving Challenge

Together with the Swedish truck manufacturer Scania, KTH participated in the competition Grand cooperative driving challenge (GCDC) provided by TNO in the Netherlands. In the competition there were nine teams from six different countries present, and the KTH-Scania team called Scoop placed fourth. The objective of GCDC was for each participating team to independently of the other teams develop the best autonomously driving vehicle by exploiting vehicle-sensors, GPS data and wireless communication where the wireless communication was performed among the different teams and road-side-units (a road side unit can be for example a traffic light or speed sign) over a predefined communication protocol. Autonomous driving requires careful signal processing, automatic control and network communication. In the pursuit of further experience and possible applications of CS, the author of this thesis contributed to the Scoop team by supervising two master thesis students and by directly implementing several features in the system. So far, this work has resulted in two articles:

[35] A. Alam, F. Asplund, S. Behere, M. Björk, L. Garcia Alonso, F. Khaksari, A. Khan, J. Kjellberg, K.-Y. Liang, R. Lyberger, J. Mårtensson, J.-O. Nilsson, H. Pettersson, S. Pettersson, E. Stålklinga, D. Sundman, M. Törngren, and D. Zachariah, "Cooperative driving according to scoop," Tech. Rep., Royal Institute of Technology (KTH), Stockholm, Sweden, June 2011, Also presented at RTiS 2011, Västerås

[36] A. Alam, S. Behere, A. Khan, J. Kjellberg K.-Y. Liang, J. Mårtensson, H. Pettersson, and D. Sundman, "The development of a cooperative heavy-duty vehicle for the gcdc 2011: Team scoop," *IEEE Transactions on Intelligent Transportation Systems*, 2012, Submitted December 2011

Further research on compressed sensing

Here we enlist additional work which falls outside the scope of this thesis. The journal article [34] is not included because it is an extension of Paper B already included in the thesis. Articles [37–39] are not included because the author of the thesis did not contribute significantly. The article [40] is a conference paper version of the journal presented in Paper E in the thesis.

[34] S. Chatterjee, D. Sundman, M. Vehkaperä, and M.Skoglund, "Projection-based and look-ahead strategies for atom selection," *IEEE Transactions on Signal Processing*, vol. 60, pp. 634–647, Feb. 2012

[37] S. Chatterjee, D. Sundman, and M. Skoglund, "Statistical post-processing improves basis pursuit denoising performance," in *IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)*, Luxor, Egypt, Dec. 2010, pp. 23–27

[38] S. Chatterjee, D. Sundman, and M. Skoglund, "Robust matching pursuit for recovery of gaussian sparse signal," in *IEEE Digital Signal Processing Workshop and IEEE Signal Processing Education Workshop (DSP/SPE)*, Sedona, Arizona, Jan. 2011, pp. 420–424

[39] S. Chatterjee, D. Sundman, and M. Skoglund, "Hybrid greedy pursuit," in *EURASIP European Signal Processing Conference (EUSIPCO)*, Barcelona, Spain, Aug. 2011, pp. 343–347

[40] D. Sundman, S. Chatterjee, and M. Skoglund, "A greedy pursuit algorithm for distributed compressed sensing," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Kyoto, Japan, Mar. 2012, Accepted

References

[1] E.J. Candès and M.B. Wakin, "An introduction to compressive sampling," *IEEE Electron Device Letters*, vol. 25, pp. 21–30, Mar. 2008.

[2] D.L. Donoho, "Compressed sensing," *IEEE Transactions on Information Theory*, vol. 52, pp. 1289–1306, Apr. 2006.

[3] E.J. Candès, J. Romberg, and T. Tao, "Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information," *IEEE Transactions on Information Theory*, vol. 52, pp. 489–509, Feb. 2006.

[4] M. Elad, *Sparse and Redundant Representations: From Theory to Applications in Signal and Image Processing*, Springer, Haifa, Israel, 2010.

[5] E. J. Candès, "The restricted isometry property and its implications for compressed sensing," *Comptes Rendus Mathematique*, vol. 346, pp. 589–592, May 2008.

[6] E.J. Candès and T. Tao, "Near-optimal signal recovery from random projections: Universal encoding strategies?," *IEEE Transactions on Information Theory*, vol. 52, pp. 5406–5425, Dec. 2006.

[7] E.J. Candès and T. Tao, "Decoding by linear programming," *IEEE Transactions on Information Theory*, vol. 51, pp. 4203–4215, Dec. 2005.

[8] M. Rudelson and R. Vershynin, "Sparse reconstruction by convex relaxation: Fourier and gaussian measurements," in *40th Annual Conference on Information Sciences and Systems*, Princeton, New Jersey, Mar. 2006, pp. 207–212.

[9] R. Giryes and M. Elad, "Rip-based near-oracle performance guarantees for sp, cosamp, and iht," *IEEE Transactions on Signal Processing*, vol. PP, 2011, To appear.

[10] D.L. Donoho and X. Huo, "Uncertainty principles and ideal atomic decomposition," *IEEE Transactions on Information Theory*, vol. 47, pp. 2845–2862, Nov. 2001.

[11] Z. Ben-Haim, Y.C. Eldar, and M. Elad, "Coherence-based performance guarantees for estimating a sparse vector under random noise," *IEEE Transactions on Signal Processing*, vol. 58, pp. 5030–5043, Oct. 2010.

[12] T.T. Wu, Y. F. Chen, T. Hastie, E. Sobel, and K. Lange, "Genome-wide association analysis by lasso penalized logistic regression," *Bioinformatics*, vol. 25, pp. 714–721, Mar. 2009.

[13] R. Agarwal and S.R. Sonkusale, "Input-feature correlated asynchronous analog to information converter for ecg monitoring," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 5, pp. 459–467, Oct. 2011.

[14] L. Applebaum, S. D. Howard, S. Searle, and R. Calderbank, "Chirp sensing codes: Deterministic compressed sensing measurements for fast recovery," *Applied and Computational Harmonic Analysis*, vol. 26, pp. 283290, Mar. 2009.

[15] M. Mishali, Y.C. Eldar, O. Dounaevsky, and E. Shoshan, "Xampling: Analog to digital at sub-nyquist rates," *IET Circuits, Devices and Systems*, vol. 5, pp. 8–20, Jan. 2011.

[16] J.A. Tropp, J.N. Laska, M.F. Duarte, J.K. Romberg, and R.G. Baraniuk, "Beyond nyquist: Efficient sampling of sparse bandlimited signals," *IEEE Transactions on Information Theory*, vol. 56, pp. 520–544, Jan. 2010.

[17] E. J. Candès, J.K. Romberg, and T. Tao, "Stable signal recovery from incomplete and inaccurate measurements," *Communications on Pure and Applied Mathematics*, vol. 59, pp. 1207–1223, Aug. 2006.

[18] B. Efron, T. Hastie, L. Johnstone, and R. Tibshirani, "Least angle regression," *Annars of Statistics*, vol. 32, pp. 407–499, June 2004.

[19] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society (Series B)*, vol. 58, pp. 267–288, 1996.

[20] J.A. Tropp and A.C. Gilbert, "Signal recovery from random measurements via orthogonal matching pursuit," *IEEE Transactions on Information Theory*, vol. 53, pp. 4655–4666, Dec. 2007.

[21] W. Dai and O. Milenkovic, "Subspace pursuit for compressive sensing signal reconstruction," *IEEE Transactions on Information Theory*, vol. 55, pp. 2230–2249, May 2009.

[22] Y.C. Pati, R. Rezaiifar, and P.S. Krishnaprasad, "Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition," in *Annual Asilomar Conference on Signals, Systems, and Computers*, Pacific Grove, California, Nov. 1993, pp. 40–44.

[23] J.A. Tropp, "Just relax: convex programming methods for identifying sparse signals in noise," *IEEE Transactions on Information Theory*, vol. 52, pp. 1030–1051, Mar. 2006.

[24] D.L. Donoho, M. Elad, and V.N. Temlyakov, "Stable recovery of sparse overcomplete representations in the presence of noise," *IEEE Transactions on Information Theory*, vol. 52, pp. 6–18, Jan. 2006.

[25] Y.S. Han, C.R.P. Hartmann, and Chih-Chieh Chen, "Efficient priority-first search maximum-likelihood soft-decision decoding of linear block codes," *IEEE Transactions on Information Theory*, vol. 39, pp. 1514–1523, Sept. 1993.

[26] D. Needell and J. A. Tropp, "Cosamp: Iterative signal recovery from incomplete and inaccurate samples," *Applied and Computational Harmonic Analysis*, vol. 26, pp. 301–321, Apr. 2009.

[27] J. Mitola III and G.Q. Maguire Jr., "Cognitive radio: making software radios more personal," *IEEE Personal Communications Magazine*, vol. 6, pp. 13–18, Aug. 1999.

[28] S. Haykin, "Cognitive radio: brain-empowered wireless communications," *IEEE Journal on Selected Areas in Communications*, vol. 23, pp. 201–220, Feb. 2005.

[29] D. Sundman, S. Chatterjee, and M. Skoglund, "On the use of compressive sampling for wide-band spectrum sensing," in *IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)*, Luxor, Egypt, Dec. 2010, pp. 354–359.

[30] S. Chatterjee, D. Sundman, and M. Skoglund, "Look ahead orthogonal matching pursuit," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Prague, Czech Republic, May 2011, pp. 4024–4027.

[31] D. Sundman, S. Chatterjee, and M. Skoglund, "Look ahead parallel pursuit," in *IEEE Swedish Communication Technologies Workshop (Swe-CTW)*, Stockholm, Sweden, Oct. 2011, pp. 114–117.

[32] D. Sundman, S. Chatterjee, and M. Skoglund, "Greedy pursuits for compressed sensing of jointly sparse signals," in *EURASIP European Signal Processing Conference (EUSIPCO)*, Barcelona, Spain, Aug. 2011, pp. 368–372.

[33] D. Sundman, S. Chatterjee, and M.Skoglund, "Greedy pursuits for distributed compressed sensing," *IEEE Transactions on Signal Processing*, 2012, Submitted February 2012.

[34] S. Chatterjee, D. Sundman, M. Vehkaperä, and M.Skoglund, "Projection-based and look-ahead strategies for atom selection," *IEEE Transactions on Signal Processing*, vol. 60, pp. 634–647, Feb. 2012.

[35] A. Alam, F. Asplund, S. Behere, M. Björk, L. Garcia Alonso, F. Khaksari, A. Khan, J. Kjellberg, K.-Y. Liang, R. Lyberger, J. Mårtensson, J.-O. Nilsson, H. Pettersson, S. Pettersson, E. Stålklinga, D. Sundman, M. Törngren, and D. Zachariah, "Cooperative driving according to scoop," Tech. Rep., Royal Institute of Technology (KTH), Stockholm, Sweden, June 2011, Also presented at RTiS 2011, Västerås.

[36] A. Alam, S. Behere, A. Khan, J. Kjellberg K.-Y. Liang, J. Mårtensson, H. Pettersson, and D. Sundman, "The development of a cooperative heavy-duty vehicle for the gcdc 2011: Team scoop," *IEEE Transactions on Intelligent Transportation Systems*, 2012, Submitted December 2011.

[37] S. Chatterjee, D. Sundman, and M. Skoglund, “Statistical post-processing improves basis pursuit denoising performance,” in *IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)*, Luxor, Egypt, Dec. 2010, pp. 23–27.

[38] S. Chatterjee, D. Sundman, and M. Skoglund, “Robust matching pursuit for recovery of gaussian sparse signal,” in *IEEE Digital Signal Processing Workshop and IEEE Signal Processing Education Workshop (DSP/SPE)*, Sedona, Arizona, Jan. 2011, pp. 420–424.

[39] S. Chatterjee, D. Sundman, and M. Skoglund, “Hybrid greedy pursuit,” in *EURASIP European Signal Processing Conference (EUSIPCO)*, Barcelona, Spain, Aug. 2011, pp. 343–347.

[40] D. Sundman, S. Chatterjee, and M. Skoglund, “A greedy pursuit algorithm for distributed compressed sensing,” in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Kyoto, Japan, Mar. 2012, Accepted.

Part II

Included papers

Paper A

On the use of Compressive Sampling for Wide-band Spectrum Sensing

Dennis Sundman, Saikat Chatterjee, and Mikael Skoglund

Published in
10'th IEEE International Symposium on Signal Processing and Information Technology

Issue Date:	*15-18 Dec. 2010*
On page(s):	*354 - 359*
Location:	*Luxor, Egypt*
Print ISBN:	*978-1-4244-9992-2*
INSPEC Accession Number:	*11823127*
Digital Object Identifier:	*10.1109/ISSPIT.2010.5711810*

©2010 IEEE
The layout has been revised

On the use of Compressive Sampling for Wide-band Spectrum Sensing

Dennis Sundman, Saikat Chatterjee, and Mikael Skoglund

Abstract

In a scenario where a cognitive radio unit wishes to transmit, it needs to know over which frequency bands it can operate. It can obtain this knowledge by estimating the power spectral density from a Nyquist-rate sampled signal. For wide-band signals sampling at the Nyquist rate is a major challenge and may be unfeasible. In this paper we accurately detect spectrum holes in sub-Nyquist frequencies without assuming wide sense stationarity in the compressed sampled signal. A novel extension to further reduce the sub-Nyquist samples is then presented by introducing a memory based compressed sensing that relies on the spectrum to be slowly varying.

Keywords: Compressive sampling, Cognitive radio, power spectrum estimation, sub-Nyquist sampling.

1 Introduction

Compressive Sampling (CS) [1] has attracted much attention in the current literature due to its promise of realizing sub-Nyquist sampling. A signal can be reconstructed from samples obtained at sampling rate lower than the Nyquist rate if the signal is sparse in some domain. Recently CS [2,3] has been employed in wide-band spectrum sensing in Cognitive Radio (CR) [4] applications because the spectrum has an underlying sparsity. In practice, at a given time, the full wide-band spectrum is inherently sparse in the sense that only a few users occupy some of many available frequency bands.

In a CR scenario we are interested in sensing wide-band signals in order to detect vacant frequency bands. A straightforward approach to detect the unoccupied frequency bands can be based on evaluating the Power Spectral Density (PSD) from Nyquist rate samples. However, wide-band signal sampling at Nyquist rate is a major challenge.

An autocorrelation based method of estimating the PSD from CS samples has been proposed in [3, 5]. The connection between an analog signal and the CS samples is established through using an analog-to-information converter (AIC) [6], [7]. An AIC takes an analog signal as input and gives the CS samples as output.

In the standard CS set-up, CS samples and Nyquist rate samples are connected via a linear transformation where the sensing matrix is random in nature [8].

In this paper we deal with the practical problem of estimating the autocorrelation coefficients of a signal from CS samples. The PSD can then be obtained as the Fourier transform of the autocorrelation coefficients. Due to the random sensing matrix, the signal obtained from the CS samples is in general not wide sense stationary (WSS). We note that stationarity was assumed in [3], although autocorrelation coefficients for non-WSS signals have no physical meaning.

We follow the guide-lines of [3] and notice that it is possible to sample in a significantly lower rate than the Nyquist rate in the task of reconstructing the PSD (with a for detection purposes acceptable degree of accuracy). In our approach, instead of estimating the autocorrelation coefficients from a CS sample vector (i.e. a vector consisting of successive CS samples), a correlation matrix is estimated from several CS sample vectors. Effectively, this strategy may lead to acquiring more total samples than the Nyquist rate samples. However, we no longer need the assumption of WSS and although the total number of samples is increased the actual sampling rate can be reduced significantly.

We further present a novel extension of the CS PSD estimation by assuming that the spectrum is slowly time-varying in the sense that the change in spectrum between two neighboring time-instants is small. This assumption is motivated by the fact that there are static components in the frequency spectrum; broadcast TV and radio, mobile traffic, wireless Internet traffic, etc.. According to this assumption, we notice that the difference between two spectrum realizations taken close in time is likely to be sparser than each spectrum by itself. We propose to benefit from this increased sparsity by taking fewer number of samples while still reconstructing the current spectrum without loss of accuracy. We are able to do this by introducing a memory in the CS process. Since we are only interested in knowing which frequency bands are free or occupied we realize detection by thresholding.

This paper is organized as follows. We discuss CS and how it can be applied for wide-band spectrum estimation in section 2. In section 3 we discuss the issue of detection from a spectrum estimation and we present our alternative method which takes into account the aforementioned memory of the CS process. The performance of the proposed method is verified through simulations in section 4. Section 5 concludes this paper.

2 CS Based Wide-band Spectrum Sensing

This section starts with a short definition of notation. We follow with a subsection describing the general CS method and then a description how this can be used for wide band spectrum sensing.

Italic letters, i.e. M, will refer to constants. Bold face capital letters, i.e. $\mathbf{\Phi}$, denotes matrices and bold face lower-case letters $\mathbf{x}$ denotes vectors. The notation * denotes, in case of a scalar the complex conjugate and in case of a matrix the Hermitian operator. The p-norm is defined as $\|\mathbf{x}\|_p = (\sum_{\forall i} |x_i|^p)^{1/p}$. The (i, j)'th element of a matrix $\mathbf{\Phi}$ is denoted by $\{\mathbf{\Phi}\}_{i,j}$.

2.1 Compressed Sensing

In the Compressed Sensing (CS) framework one assumes that an N-dimensional signal $\mathbf{s}$ consists of samples presented at the Nyquist rate. The signal is sparse in a transform domain. Let us denote the $N \times N$ transform matrix $\mathbf{\Psi}$, in which the signal is sparse. A K-sparse signal has at most K non-zero components. In other words, if $\mathbf{s}$ is K-sparse in the transform domain $\mathbf{\Psi}$, then $\|\mathbf{\Psi}\mathbf{s}\|_0 \leq K$, where $\|\cdot\|_0$ refers to the number of non-zero coefficients.

To obtain the CS samples from an analog signal a hardware device commonly known as AIC [6] is used. Assume that sampling at the Nyquist rate yields N samples. The purpose of the AIC is then to collect M samples such that $M < N$ according to the CS procedure, from which a Nyquist sampled signal then can be reconstructed using CS reconstruction. Several AIC architectures have been proposed in the literature, e.g. random demodulator [9], random filtering and random convolution [10–12]. In particular, the random demodulator has shown promising results in acquiring CS samples at a sampling rate exponentially lower than the Nyquist-rate.

A common interpretation of an AIC is as sampling done at Nyquist-rate followed by an $M \times N$ (down-)sampling matrix, $\mathbf{\Phi}$. For reconstruction to be possible, M has to be sufficiently large. According to E.

Candés and T. Tao if M is chosen as

$$M \gtrsim K \log_{10} N, \tag{1}$$

the signal can be recovered with high probability if the elements in $\mathbf{\Phi}$ are selected independently at random [13] (also see [8,14]). A matrix $\mathbf{\Phi}$, with its elements chosen in this way (and satisfying the aforementioned inequality of M), is said to obey the Restricted Isometric Property (RIP), which is necessary for reconstruction of

$$\mathbf{y} = \mathbf{\Phi}\mathbf{s}. \tag{2}$$

The reconstruction procedure takes the form of a convex problem:

$$\hat{\mathbf{s}} = \arg\min_{\mathbf{s}} \|\Psi\mathbf{s}\|_1 \qquad \text{s.t.} \qquad \mathbf{y} = \mathbf{\Phi}\hat{\mathbf{s}}, \tag{3}$$

or equivalently,

$$\hat{\mathbf{z}} = \arg\min_{\mathbf{z}} \|\mathbf{z}\|_1 \qquad \text{s.t.} \qquad \mathbf{y} = \Psi\mathbf{\Phi}^{-1}\hat{\mathbf{z}}, \tag{4}$$

where $\mathbf{z}$ is K-sparse. For practical convenience the latter form is used in this paper. The convex problem can be solved by different linear programming techniques, e.g. basis pursuit [15] or iterative greedy algorithms [16]. All convex problems in this paper are solved using the cvx toolkit [17].

2.2 Wide band spectrum Sensing

The CS technique relies on sparsity; the sparser signal the more efficient is the method. It is realized that the edge spectrum $\mathbf{z}_x$ (derivative of the PSD) is sparser than the spectrum alone. The PSD is obtained from the autocorrelation coefficients via the Fourier transform. Denoting the edge spectrum by $\mathbf{z}_x$ we can form the relation between the edge spectrum and the autocorrelation coefficients, analogously to [2], as

$$\mathbf{r}_x = (\Gamma\mathcal{W}\mathcal{F})^{-1}\mathbf{z}_x, \tag{5}$$

where $\mathcal{W}$ is a $2N \times 2N$ wavelet smoothing matrix, $\mathcal{F}$ is the $2N \times 2N$ Discrete Fourier Transform (DFT) matrix and Γ is the approximated, by first-order difference, derivative implemented as the $2N \times 2N$ matrix

$$\Gamma = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ -1 & 1 & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & -1 & 1 & 0 \\ 0 & \cdots & 0 & -1 & 1 \end{bmatrix} \tag{6}$$

For simplicity we will denote $\mathbf{G} = (\Gamma \mathcal{W} \mathcal{F})^{-1}$.

The spectrum sensing scheme presented in [2] is based on full ADC sampling i.e. according to the Nyquist rate. This approach is extended in [3] to the case where we only have access to the CS-sampled signal

$$\mathbf{y} = \mathbf{\Phi x}. \tag{7}$$

We are now left with the task of finding a linear transform that takes us from the CS-sampled signal to the autocorrelation coefficients $\mathbf{r}_x$. A linear relation, based on the correlation matrix relation $\mathbf{R}_y = \mathbf{\Phi R}_x \mathbf{\Phi}^*$ is found

$$\mathbf{r}'_y = \mathbf{\Phi_{II} r}_x, \tag{8}$$

where $\mathbf{r}'_y$ is defined as the first row (reversed) concatenated with the first column (without the first component)

$$\mathbf{r}'_y \triangleq [0 \ \{\mathbf{R}_y\}_{1,N} \cdots \{\mathbf{R}_y\}_{1,1} \ \{\mathbf{R}_y\}_{2,1} \cdots \{\mathbf{R}_y\}_{N,2}]^T \tag{9}$$

and $\mathbf{r}_x$ is defined as

$$\mathbf{r}_x \triangleq [0 \ r_x(-N+1) \cdots r_x(0) \cdots r_x(N-1)]^T \tag{10}$$

where the zeros are added for implementation convenience. The vector $\mathbf{r}'_y$ is defined from the correlation matrix rather than the autocorrelation coefficients because $\mathbf{\Phi}$ does, in general, not preserve WSS in the measurements.

Remark 1.2 The linear sensing operation (7), can be realized as filtering with the filter described by the matrix $\mathbf{\Phi}$. This filter corresponds to a time-varying filter. It is well known [18] that time-varying filters do not preserve the WSS property in signals.

The main difference between the approach in [3] and the one presented here lies in the definition of $\mathbf{r}'_y$. The relation matrix $\mathbf{\Phi_{II}}$ is the same as in [3] and is defined as

$$\mathbf{\Phi_{II}} = \begin{bmatrix} \overline{\mathbf{\Phi}}\mathbf{\Phi}_1 & \overline{\mathbf{\Phi}}\mathbf{\Phi}_2 \\ \mathbf{\Phi\Phi}_3 & \mathbf{\Phi\Phi}_4 \end{bmatrix} \tag{11}$$

where we denote the (i,j)'th element of $\mathbf{\Phi}$ by $\phi_{i,j}$, then the $M \times N$ matrix $\overline{\mathbf{\Phi}}$ has its (i,j)'th element given by

$$\{\overline{\mathbf{\Phi}}\}_{i,j} = \begin{cases} 0 & i = 1, j = 1, ..., N \\ \phi^*_{M+2-i,j} & i \neq 1, j = 1, ..., N \end{cases} \tag{12}$$

The $N \times N$ matrices $\mathbf{\Phi}_1$, $\mathbf{\Phi}_2$, $\mathbf{\Phi}_3$, and $\mathbf{\Phi}_4$ are

$$\begin{aligned}
\mathbf{\Phi}_1 &= \text{hankel}([\mathbf{0}_{N\times 1}], [0\ \phi_{1,1}\ \cdots\ \phi_{1,N-1}]),\\
\mathbf{\Phi}_2 &= \text{hankel}([\phi_{1,1}\ \cdots\ \phi_{1,N}], [\phi_{1,N}\ \mathbf{0}_{1\times(N-1)}]),\\
\mathbf{\Phi}_3 &= \text{toeplitz}([\mathbf{0}_{N\times 1}], [0\ \phi^*_{1,N}\ \cdots\ \phi^*_{1,2}]),\\
\mathbf{\Phi}_4 &= \text{toeplitz}([\phi^*_{1,1}\ \cdots\ \phi^*_{1,N}], [\phi^*_{1,1}\ \mathbf{0}_{1\times(N-1)}]).
\end{aligned}$$

Here hankel$(\mathbf{c}, \mathbf{r})$ denotes a Hankel matrix (i.e., symmetric and constant across the anti-diagonals) whose first column is $\mathbf{c}$ and whose last row is $\mathbf{r}$, toeplitz$(\mathbf{c}, \mathbf{r})$ denotes a Toeplitz matrix (i.e., symmetric and constant across the diagonals) whose first column is $\mathbf{c}$ and whose first row is $\mathbf{r}$. The vector $\mathbf{0}_{N\times 1}$ is a column of N zeros and $\mathbf{0}_{1\times N-1}$ is a row of $N-1$ zeros.

A straight-forward method to estimate $\mathbf{r}'_y$ is to first estimate $\mathbf{R}_y$ and then pick the corresponding elements from the matrix as in (9). We now have linear relations between the CS-sampled signal and the edge spectrum. The convex problem of estimating the edge spectrum becomes

$$\hat{\mathbf{z}}_x = \arg\min_{\mathbf{z}} \|\mathbf{z}\|_1 \qquad \text{s.t.} \qquad \mathbf{r}'_y = \mathbf{\Phi}_{\mathbf{II}}\mathbf{G}\hat{\mathbf{z}}_x. \tag{13}$$

Finally, the estimate of the PSD can be achieved from the edge spectrum as [2]

$$S(n) = \sum_{k=1}^{n} \hat{z}(k) \qquad n = 1, ..., 2N-1. \tag{14}$$

3 Spectral detection with/without memory

In this section we investigate how to use the estimate of the PSD in (14) in order to determine which frequency bands are available for CR transmission. First we formulate our detection problem [18] and introduce a method for setting the threshold, then we introduce the memory.

3.1 Wide band spectral detection based on the PSD

Having access to the estimate (14) we consider how to use this information to determine which bands in $S(n)$ that are free for transmission. Firstly, we define the wide-band PSD estimate vector

$$\begin{aligned}
\mathbf{s} &= \left[\ S(1)\quad S(2)\quad \cdots\quad S(2N-1)\ \right]^T\\
&= \left[\ \mathbf{s}_1^T\quad \mathbf{s}_2^T\quad \cdots\quad \mathbf{s}_B^T\ \right]^T,
\end{aligned}$$

where $\mathbf{s}_i$ is a vector representing the PSD over the known frequency sub-band, i, and B is the total number of sub-bands. In practice the number of components in each vector $\mathbf{s}_i$ does not need to be the same. Now we define the band-vector $\mathbf{b}$ as

$$\mathbf{b} = \begin{bmatrix} b_1 & b_2 & \cdots & b_B \end{bmatrix}$$

where $b_i = \sum_{\forall j} s_i(j)$ is the sum over all components in each frequency band and can be thought of as the total energy in that band. In order to find the occupied bands in $\mathbf{b}$, we determine a threshold value γ, above which the band is considered occupied and below which it is considered vacant. We sort $\mathbf{b}$ in descending order and call the sorted vector $\mathbf{b}_\mathrm{s}$. Then take the difference between each consecutive element in $\mathbf{b}_\mathrm{s}$ and locate the maximum value

$$\max \begin{bmatrix} b_{\mathrm{s},1} - b_{\mathrm{s},2} & b_{\mathrm{s},2} - b_{\mathrm{s},3} & \cdots & b_{\mathrm{s},B-1} - b_{\mathrm{s},B} \end{bmatrix}$$

Let the maximum value be $b_{\mathrm{s},i-1} - b_{\mathrm{s},i}$, then $b_{\mathrm{s},i-1}$ is chosen as the threshold value γ. A requirement for the threshold value to be meaningful is that at least one band has to be occupied. It is also required that the total energy from the weakest occupied band contains "much" more energy compared to a vacant frequency band than it does to all the other occupied bands.

We are now ready to write the frequency band detection problem as

$$b_k \underset{\mathcal{H}_0}{\overset{\mathcal{H}_1}{\gtrless}} \gamma \qquad\qquad k = 1, 2, ..., B.$$

Where $\mathcal{H}_1$ and $\mathcal{H}_0$ denote the hypothesis of a primary signal being present and absent, respectively. The choice of γ does not effect the ability of the system to perform at sub-Nyquist rates, even in the memory-based scenario described later. It threshold value only effects the probability of miss or false alarm in the sub-bands of the current PSD.

3.2 No Memory

We now introduce our proposed method of estimation of PSD. Assume an AIC that satisfies the RIP (1) gives M measurements. We estimate the correlation matrix of $\mathbf{y}$, as

$$\hat{\mathbf{R}}_y = \frac{1}{L} \sum_{l=1}^{L} \mathbf{y}_l \mathbf{y}_l^*, \tag{15}$$

where L needs to be sufficiently large to achieve a good quality estimate. We obtain an estimate of the vector $\hat{\mathbf{r}}'_y$ according to (9), and solve the convex problem (13).

If we require L samples for the estimation of $\hat{\mathbf{R}}_y$ and $\mathbf{y}$ has the size M, this procedure requires a total number of LM samples for each realization of the power spectrum. The total number of samples LM may not be smaller than a conventional sampling of N samples, but these samples can be picked at an (exponentially [9]) lower sampling rate, according to the aforementioned demodulator.

3.3 Memory

In practical CR implementations it is of great interest to reduce the total number of samples LM required in estimation since acquiring each sample has an associated cost in terms of energy.

The method proposed in section 3.2 relies on the accuracy in estimating $\hat{\mathbf{R}}_y$. We see in (15) that reducing L sacrifices precision in the estimation process. Therefore we limit ourselves to reducing M which is the number of CS samples in each CS vector $\mathbf{y}$.

We propose to use a memory-based spectrum estimation method that exploits the fact that the PSD is likely to be slow-varying nature. The slow-varying nature comes from the fact that there are many static transmissions in our surroundings. Examples of transmissions that clearly can be considered static are broadcast TV and radio. Moreover, in CR environments where the freed bands only need to be determined every second, mobile phone calls and wireless Internet can also be considered as static.

Let C be a value describing the changes in the K-sparse edge spectrum from one time instance to the following as follows. Assume that $\mathbf{z}_1$ is K-sparse, the next instance edge spectrum $\mathbf{z}_2$ is also K-sparse and $\mathbf{z}_2 - \mathbf{z}_1$ is K_2 sparse, then $C = K_2/K$. By this definition, C can take any value between 0 and 2. For the method to be useful it is necessary that $C < 1$. From (1) we then notice that the number of CS samples M_2 when making use of the memory can, for the CK-sparse signal, be chosen as CM_1. We call this method a memory based compressed sensing. The convex problem at hand then becomes

$$\hat{\mathbf{z}}_x = \arg\min_{\mathbf{z}} \|\mathbf{z} - \hat{\mathbf{z}}_{\text{old}}\|_1 \qquad \text{s.t.} \qquad \mathbf{r}'_y = \mathbf{\Phi}_{\mathbf{II}}\mathbf{G}\hat{\mathbf{z}}_x, \tag{16}$$

By assuming the change is no more than a specified C (which we assume < 1), we are able to repeat this memory based algorithm consecutively a number of times. Full CS sampling without memory requires M_1 samples per estimate. In contrast, the improved estimation method based on the system memory requires only $M_2 = CM_1$ samples per estimate. Repeat this algorithm, taking consecutive CS samples M_2 based on the memory a number of times. The drawback is that if an error occurs, it is likely to remain in the memory until we take a full CS sampling at M_1 again. By applying the memory CS sampling a number of Q times, we then need to take a total of $LM + LMQC = LM(1 + QC)$ numbers of samples. If $C < 1$ then the number of samples required is smaller with the memory than the method without memory which has a total of $LM+LMQ = LM(1+Q)$ number of samples. We quantify the reduction in frequency by comparing the number of required CS measurements for the memory based algorithm with memory Q to the non-memory case. For this purpose we define the average reduction fraction as

$$R_f = \frac{L(M_1 + QM_2)}{L(M_1 + QM_1)} = \{M_2 = CM_1\} = \frac{1 + QC}{1 + Q}. \tag{17}$$

This fraction will become smaller, hence better, as Q increases.

How to choose C depends on how small changes the CR user wants to track. It seems reasonable that the CR user wants to track quite small changes in the spectrum, i.e. choosing C small, which in comparison to the standard method gives a big improvement to the proposed algorithm.

On the contrary, unless there are frequently significant changes in the spectrum, most likely due to big changes in the surroundings, the memory Q can also be chosen large, which further reduces the number of total samples.

4 Simulations

4.1 Signal Modeling

A challenge in simulating the proposed algorithm is obtaining realistic data to be sampled by the AIC. We are interested in the power spectrum which means that the signal needs to be WSS. One can also assume that in practice the signal can be modeled as ergodic. These properties of the signal will however not remain after the sampling.

As mentioned earlier, the AIC can be thought of as a Nyquist sampling followed by the application of a measurement matrix $\mathbf{\Phi}$. Since we do not have access to a real time-continuous signal, we instead construct a discrete signal $\mathbf{x}$ and apply $\mathbf{\Phi}$ directly to that.

To form a signal with the given PSD, we first form a 80-tap AR-filter designed from the Yule-walker equations [19]. By filtering white noise through this filter, we obtain a signal $\mathbf{x}$ with the desired properties. The power spectrum of this signal will look like the top picture in Fig. 3. The oscillating effect in the figure is due to Gibbs phenomenon and can be reduced with larger number of taps in the filter.

4.2 Parameters and Simulation set-up

In the following simulations we have set the fraction C of change for the power spectrum to $C = 1/3$ (see Section 3.3). The value $C = 1/3$ is chosen because it is easy to implement in this particular realization. As an example of wide-band spectrum our range of interest is 0 - 5 GHz. This would require a Nyquist-sampling frequency of 10 GHz in the traditional approach. We have chosen $B = 20$ frequency bands equally spaced over the spectrum because it is a reasonable range to show in the plots. We need about $N = 400$ to get a good resolution of this spectrum. Because of the Gibbs phenomenon and the smoothing matrix $\mathcal{W}$, we get a sparsity in the edge spectrum $K \approx \|\mathbf{z}_x\|_0 \approx 77$. From (1) we get $M = 200$ which gives that we instead of a required 10 GHz sampling rate can do it at $\frac{200}{400} f_s = 5$ GHz. This improvement is rather modest due to the relatively small N. In a real-world scenario, however, it is likely that N is chosen as a larger value because of the need of higher spectrum-resolution and since M only grows logarithmically in N, potential gains are much larger.

As argued before, to get a good estimation of $\hat{\mathbf{R}}_y$, we need a sufficiently large L. By testing we find that $L = 10000$ is required.

If the first sample needs $M_1 = 200$, the samples using memory need $M_2 = CM_1$. Since $C = 1/3$, $M_2 = \frac{1}{3}200 = 67$. This gives a further reduction in the sampling frequency $f_s = \frac{67}{200} < 2$ GHz.

In this paper we define an error to occur when the algorithm detects a frequency band as occupied when it is vacant or vice versa. In Fig. 1, we see the memory Q plotted against the error (in percent), averaged from 1000 runs. We have chosen to plot the curves for $M_2 = 50 < 67$ (which is insufficient for our reconstruction) and $M_2 = 100$ (which should be sufficient for full recovery). In the following section we notice that indeed, considerable savings are obtained while maintaining full or nearly full performance.

4.3 Analysis of the simulation results

By examining Fig. 1, we notice that by taking the full $M = 200$ at each sample, i.e. setting the memory $Q = 0$, we get no errors at all. By applying the proposed algorithm with memory, we see that a reduction of M_2 down to 100 is possible without sacrificing any detection performance. We also notice that by choosing $M_2 = 50$, which is insufficient for full recovery, errors occurs. The errors increase with the memory factor Q, as expected since an error in the detection is based on the total energy in each frequency band and, hence, is likely to remain in each step until a new full spectrum, sampled without utilizing old memory is estimated.

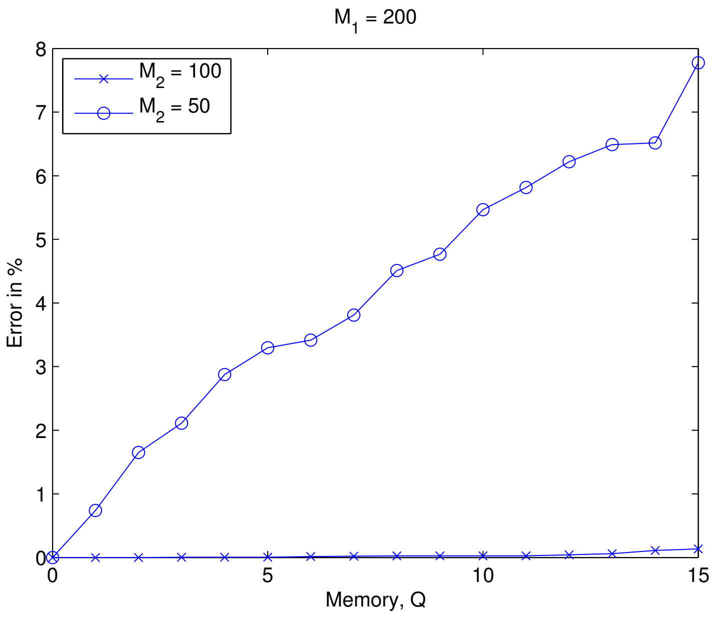


Figure 1: Errors with different memory. $Q = 0$ corresponds to taking full CS sample at each step

This is true for both $M_2 = 50$ and $M_2 = 100$, but it shows clearer in $M_2 = 50$.

Fig. 2 shows how the reduction fraction varies with the memory assumed in our model. As one would expect, higher memory orders allow for larger reduction in terms of samples and frequency rate. This corresponds to the first equality in (17). The second equality in the same equation is achieved by setting $M_2 = \frac{1}{3}M_1$ which would give a curve between the two curves plotted in the figure. Notice that the lower curve gives rise to errors in the process and the upper curve is close to error free.

In Fig. 3 we have represented the PSD estimated from $\mathbf{x}$ (top) and the corresponding reconstruction based on CS samples (bottom). Gibbs phenomenon is readily observed in the upper figure. With a larger number of taps in the filter, the step response would be faster, resulting in a sparser edge spectrum, as also discussed in section 4.2. If we compare the original signal with the reconstruction we observe a slight shift to the left in the latter with respect to the former. This shift is a natural effect of the implementation of the smoothing matrix $\mathcal{W}$. The detector

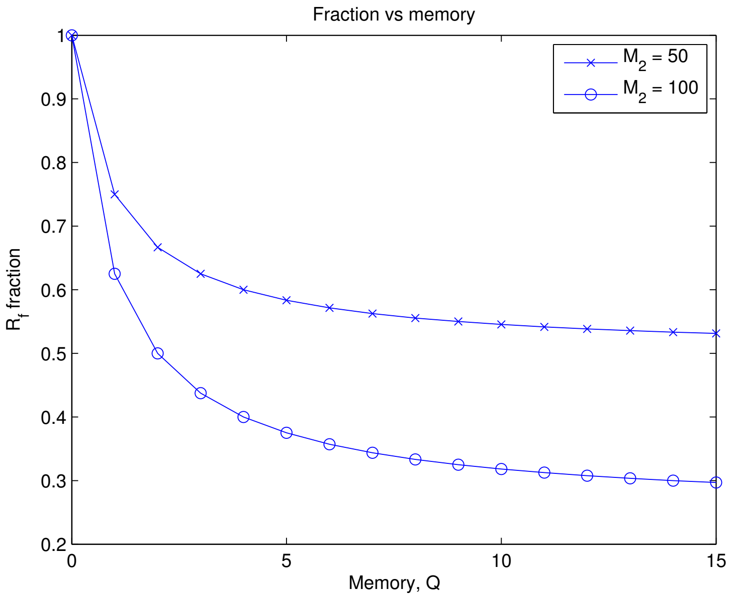


Figure 2: The sample fraction gain at different memory. At $Q = 0$ we take full CS sample at each iteration, i.e. no sampling improvement due to memory.

should be aware of this shift and compensate for it appropriately. The rough outline of the reconstructed spectrum is mainly due to the estimation of the correlation matrix, rather than the CS process. However, since we are interested in detecting if there is a transmission in a certain frequency band this poses no practical problem.

Finally we remark on the trade-off in our memory based method. Larger memory allows for reduced frequencies but may impose errors in the detection. Also, as the memory grows, the robustness to sudden unexpected significant changes in the spectrum clearly decreases.

5 Conclusion

We have presented a method for estimating the power spectrum of a signal from CS samples intended for users with limited sampling capability, i.e. CR. The novelty of our method is that it does not rely on WSS assumptions of the CS sampled signal. Compared to traditional

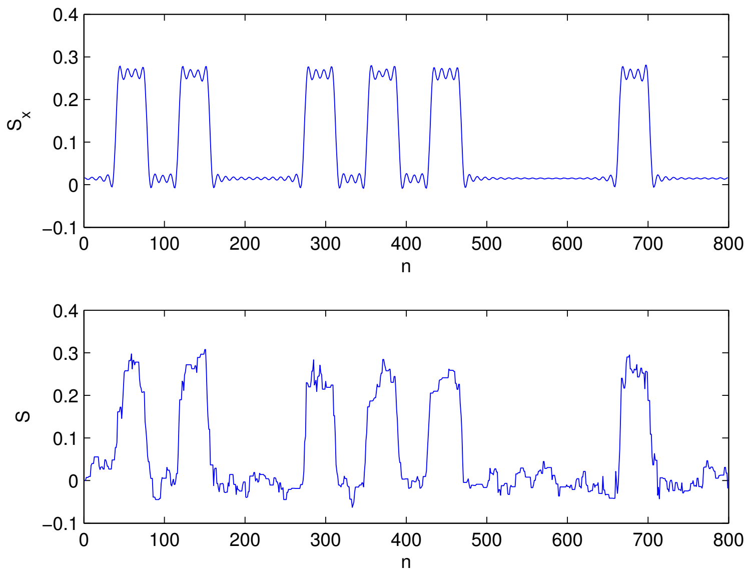


Figure 3: top: PSD estimated from full Nyquist sampling, bottom: a typical CS-reconstruction from M=200 samples.

sampling at Nyquist rate, our suggested method may require more total samples for accurate estimation. However, these samples can be taken at a significantly lower rate than the Nyquist rate.

Our method extends to the case of slowly time-varying PSD. Further reductions is then possible in sampling rate and total number of samples are possible by employing an estimation technique based on memory. Our simulation results show that the proposed memory based PSD estimation method is a promising alternative in scenarios where full-rate Nyquist sampling is not possible.

Acknowledgements

The authors would like to thank Dr. Michael Grant and Professor Stephen Boyd, Stanford University, for making the cvx toolkit for solving convex problems available to the public. Furthermore the authors would like to thank Professor Peter Händel, Royal Institute of Technology, for fruitful discussions regarding signal processing and Ricardo

Blasco Serrano for his helpful comments.

References

[1] D.L. Donoho, "Compressed sensing," *IEEE Trans. Inf. Theory,*, vol. 52, no. 4, pp. 1289 –1306, April 2006.

[2] Zhi Tian and G.B. Giannakis, "Compressed sensing for wideband cognitive radios," in *IEEE Int. Conf. Acoustics, Speech and Signal Proc., 2007. ICASSP 2007.*, 15-20 2007, vol. 4, pp. IV–1357 –IV–1360.

[3] Y.L. Polo, Ying Wang, A. Pandharipande, and G. Leus, "Compressive wide-band spectrum sensing," in *IEEE Int. Conf. Acoustics, Speech and Signal Proc., 2009. ICASSP 2009.*, 19-24 2009, pp. 2337 –2340.

[4] Joseph Mitola, *Cognitive Radio: An Integrated Agent Architecture for Software Defined Radio*, Ph.D. thesis, Royal Institute of Technology (KTH), Stockholm, May 2000.

[5] Ying Wang, A. Pandharipande, Y.L. Polo, and G. Leus, "Distributed compressive wide-band spectrum sensing," *Inf. Theory and Applications Workshop,*, pp. 178 –183, Feb 2009.

[6] S. Kirolos, T. Ragheb, J. Laska, M.E. Duarte, Y. Massoud, and R.G. Baraniuk, "Practical issues in implementing analog-to-information converters," in *The 6th Int. Workshop System-on-Chip for Real-Time Applications*, Dec. 2006, pp. 141 –146.

[7] Jason Laska, Sami Kirolos, Yehia Massoud, Richard Baraniuk, Anna Gilbert, Mark Iwen, and Martin Strauss, "Random sampling for analog-to-information conversion of wideband signals," in *IEEE Dallas/CAS Workshop Design, Applications, Integration and Software,*, Oct. 2006, pp. 119 –122.

[8] E.J. Candes and T. Tao, "Near-optimal signal recovery from random projections: Universal encoding strategies?," *IEEE Trans. Inf. Theory,*, vol. 52, no. 12, pp. 5406 –5425, Dec 2006.

[9] J.A. Tropp, J.N. Laska, M.F. Duarte, J.K. Romberg, and R.G. Baraniuk, "Beyond nyquist: Efficient sampling of sparse bandlimited signals," *IEEE Trans. Inf. Theory,*, vol. 56, no. 1, pp. 520 –544, Jan 2010.

[10] J.A. Tropp, M.B. Wakin, M.F. Duarte, D. Baron, and R.G. Baraniuk, "Random filters for compressive sampling and reconstruction," *Acoustics, Speech and Signal Proc., 2006. ICASSP 2006 Proceedings. 2006 IEEE Int. Conf. on*, vol. 3, pp. III –III, May 2006.

[11] Waheed U. Bajwa, Jarvis D. Haupt, Gil M. Raz, Stephen J. Wright, and Robert D. Nowak, "Toeplitz-structured compressed sensing matrices," *Statistical Signal Proc., 2007. SSP '07. IEEE/SP 14th Workshop on*, pp. 294 –298, Aug 2007.

[12] J. Romberg, “Sensing by random convolution,” *Computational Advances in Multi-Sensor Adaptive Processing, 2007. CAMPSAP 2007. 2nd IEEE Int. Workshop on*, pp. 137 –140, Dec 2007.

[13] E.J. Candes and T. Tao, “Decoding by linear programming,” *IEEE Trans. Inf. Theory,*, vol. 51, no. 12, pp. 4203 – 4215, Dec. 2005.

[14] E. J. Candes and J. Romberg, “Sparsity and incoherence in compressive sampling,” *Inverse Problems*, vol. 23, no. 3, pp. 969–985, Jun. 2007.

[15] Scott Shaobing Chen, David L. Donoho, Michael, and A. Saunders, “Atomic decomposition by basis pursuit,” *SIAM Journal Scientific Computing*, vol. 20, pp. 33–61, 1998.

[16] Chinh La and M.N. Do, “Tree-based orthogonal matching pursuit algorithm for signal reconstruction,” *Image Processing, 2006 IEEE Int. Conf. on*, pp. 1277 –1280, Oct 2006.

[17] M. Grant and S. Boyd, “CVX: Matlab software for disciplined convex programming, version 1.21,” http://cvxr.com/cvx, Jul. 2010.

[18] H. L. Van Trees, *Detection, Estimation, and Modulation Theory, Part I*, Wiley-Interscience, 1 edition, Sep. 2001.

[19] P. Stoica and R. Moses, *Spectral Analysis of Signals*, Pearson Prentice Hall, 2005.

Paper B

Look Ahead Orthogonal Matching Pursuit

Saikat Chatterjee, Dennis Sundman, and Mikael Skoglund

Published in
36th International Conference on Acoustics, Speech and Signal Processing

Issue Date:	*22-27 May 2011*
On page(s):	*4024 - 4027*
Location:	*Prague, Czech Republic*
E-ISBN:	*978-1-4577-0537-3*
Print ISBN:	*978-1-4577-0538-0*
Digital Object Identifier:	*10.1109/ICASSP.2011.5947235*

©2011 IEEE
The layout has been revised

Look Ahead Orthogonal Matching Pursuit

Saikat Chatterjee, Dennis Sundman, and Mikael Skoglund

Abstract

For compressive sensing, we endeavor to improve the recovery performance of the existing orthogonal matching pursuit (OMP) algorithm. To achieve a better estimate of the underlying support set progressively through iterations, we use a look ahead strategy. The choice of an atom in the current iteration is performed by checking its effect on the future iterations (look ahead strategy). Through experimental evaluations, the effect of look ahead strategy is shown to provide a significant improvement in performance.

Keywords: Compressive sensing, orthogonal matching pursuit, basis pursuit, subspace pursuit.

1 Introduction

The standard CS problem [1] is based on a sparse signal model and uses an under-determined system of linear equations. In the literature, a variety of CS reconstruction algorithms have been developed based on convex relaxation [2]- [3], non-convex [4]- [5] and iterative greedy search [6]- [10] strategies. Among these methods, convex relaxation based methods have attracted much attention due to their theoretical elegance and provable recovery performance. In practice, convex relaxation based methods are computationally intensive. On the other hand, the iterative greedy search methods are of lower complexity and hence their use may be practically viable in solving large-dimensional CS problems. From a measurement vector, the main principle of the iterative greedy search methods is the estimation of the underlying support set of a sparse vector. The support set is the set of indices corresponding to non-zero elements of a sparse vector. To estimate the support set, iterative greedy search methods use linear algebraic tools (such as the matched filter and least square solution) iteratively. Through minimizing a measure of fitting residual, iterative greedy search methods find a better estimate of support set. An important iterative greedy method is the orthogonal matching pursuit (OMP) [7] that minimizes the least square residual iteratively. The OMP has some variants such as the stage-wise OMP [8] and the regularized OMP [9].

Starting with an initial null set, the OMP algorithm constructs the support set of a sparse vector *progressively* through iterations. In the k'th iteration, a new intermediate support set of cardinality k is formed through adding a new element to the previously formed intermediate support set of cardinality $(k-1)$. The new element is found through using a matched filter. Given a sparsity level K, the OMP algorithm usually performs K iterations to form a K-element support set.

In the standard OMP, choice of a new element in the current iteration does not depend on the final performance of the OMP algorithm when all the iterations would have been finished. Therefore, the choice of a new element in an iteration is blind to its effect on the future iterations. In this paper, we improve the existing OMP to overcome this shortcoming using a *look ahead* strategy. For the proposed method, an element is chosen by evaluating its effect on the final performance measure in the sense of minimizing the fitting residual at the end of all future iterations. In the current iteration, several potential elements are chosen through the use of a matched filter. Each of the potential elements are tested independently, and then one element among them is selected and inducted to the intermediate support set. Among the potential elements, the selected element provides the minimum fitting residual after executing all the future iterations (look ahead strategy). We refer to the new method as the look ahead OMP (LAOMP). Through experimental evaluations, we compare the LAOMP algorithm vis-a-vis the OMP, subspace pursuit (SP) [10] and convex relaxation based basis pursuit (BP) [2] algorithms.

Notations: Let $\mathbf{A} \in \mathbb{R}^{M \times N}$, $\mathbf{x} \in \mathbb{R}^{N}$, and $I \subset \{1, 2, \ldots, N\}$. The matrix $\mathbf{A}_I \in \mathbb{R}^{M \times |I|}$ consists of the columns of $\mathbf{A}$ with indices $i \in I$, and $\mathbf{x}_I \in \mathbb{R}^{|I|}$ is composed of the components of $\mathbf{x}$ indexed by $i \in I$. We also denote $(.)^t$ and $(.)^\dagger$ as transpose and pseudo-inverse respectively.

2 Compressive Sensing and Orthogonal Matching Pursuit

Let us state the standard CS problem where we acquire a K-sparse signal $\mathbf{x} \in \mathbb{R}^N$ via the linear measurements

$$\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{w}, \tag{1}$$

where $\mathbf{A} \in \mathbb{R}^{M \times N}$ is a matrix representing the sensing system, $\mathbf{y} \in \mathbb{R}^M$ represents a vector of measurements and $\mathbf{w} \in \mathbb{R}^M$ is additive noise representing measurement errors. A K-sparse signal vector consists of at most K non-zero scalar components. With the setup of $K < M < N$ (under-determined system of linear equations), the task is to reconstruct $\mathbf{x}$ from $\mathbf{y}$ as $\hat{\mathbf{x}}$. Naturally the objective is to strive for a reduced number of measurements as well as achieving good reconstruction quality. Note that, in practice, we may wish to acquire a signal $\mathbf{x}$ that is sparse in a known orthonormal basis and the concerned problem can be easily recast as (1). A column of $\mathbf{A}$ is also called an 'atom' in the literature. We note that the measurement $\mathbf{y}$ is the linear combination of at most K atoms.

For the signal vector $\mathbf{x} = [x_1, x_2, \ldots, x_N]^t$, let the support set $I_{\mathbf{x}} \subset \{1, 2, \ldots, N\}$ denote the set of indices of the non-zero components of $\mathbf{x}$, i.e., $I_{\mathbf{x}} = \{i : x_i \neq 0\}$. For a K-sparse vector $\mathbf{x} \in \mathbb{R}^N$, $|I_{\mathbf{x}}| = \|\mathbf{x}\|_0 \leq K$. In this paper, we assume that $|I_{\mathbf{x}}| = K$. Denoting the i'th column (atom) of the measurement matrix $\mathbf{A}$ as $\mathbf{a}_i$, note that

$$\mathbf{y} = \sum_{i \in I_{\mathbf{x}}} x_i \mathbf{a}_i + \mathbf{w} = \mathbf{A}_{I_{\mathbf{x}}} \mathbf{x}_{I_{\mathbf{x}}} + \mathbf{w}. \tag{2}$$

From $\mathbf{y}$, if the underlying support set (contains the indices of atoms that are linearly combined) can be identified, then finding the non-zero values of $\mathbf{x}$ is a relatively easier task to perform (as $K < M$, we can use least square solution). Therefore, a better estimate of support set leads to a better reconstruction performance.

Next we consider the question of how to construct the sensing matrix $\mathbf{A} \in \mathbb{R}^{M \times N}$. While it is possible to obtain deterministic constructions of $\mathbf{A} = \{a_{i,j}\}$ holding a specific structure, at present the most efficient designs (i.e., those requiring minimum number of rows) rely on random matrix constructions where the $a_{i,j}$'s are assumed realizations of independent and identically distributed (i.i.d.) random variables. A practical method is to draw $a_{i,j}$'s independently from a Gaussian source (i.e., $a_{i,j} \sim \mathcal{N}\left(0, \frac{1}{M}\right)$) and then to scale the columns of $\mathbf{A}$ as unit-norm.

Knowing preliminaries of CS, let us discuss the OMP algorithm. We summarize the main steps of the OMP algorithm below.

```
    (OMP for CS Recovery)
Input: $\mathbf{A} = [\mathbf{a}_1\ \mathbf{a}_2 \ldots \mathbf{a}_N]$, measurement $\mathbf{y}$, sparsity level $K$.
Initialization: $\hat{\mathbf{x}}_0 = \mathbf{0}$, $\mathbf{r}_0 = \mathbf{y}$, $I_0 = \emptyset$, counter $k = 0$.
  repeat
    $k = k + 1$;
    $i_k$ = index of the highest amplitude component
          of $\mathbf{A}^t r_{k-1}$ such that $i_k \not\subset I_{k-1}$;
    $I_k = I_{k-1} \cup i_k$;
    $\hat{\mathbf{x}}_{I_k} = \mathbf{A}^\dagger_{I_k}\mathbf{y}$; $\hat{\mathbf{x}}_{\overline{I}_k} = \mathbf{0}$;
    $\mathbf{r}_k = \mathbf{y} - \mathbf{A}_{I_k}\hat{\mathbf{x}}_{I_k}$;
  until $(\|\mathbf{r}_k\|_2 > \|\mathbf{r}_{k-1}\|_2)$ or $(k > K)$
  $k = k - 1$;
Output: $\hat{\mathbf{x}}$, satisfying $\hat{\mathbf{x}}_{I_k} = \mathbf{A}^\dagger_{I_k}\mathbf{y}$ and $\hat{\mathbf{x}}_{\overline{I}_k} = \mathbf{0}$.
```

The OMP algorithm starts with an initial residual $\mathbf{r}_0 = \mathbf{y}$. At the k'th iteration stage, it forms the 'matched filter' $\mathbf{A}^t\mathbf{r}_{k-1}$, identifies the coordinate (corresponding to an atom) with highest amplitude, solves a LS problem with the selected coordinates, subtracts the LS fit and produces a new residual. Given the sparsity level K, the algorithm usually executes K iterations and forms a support set of cardinality K at the end.

3 Look Ahead OMP

Based on the algorithmic structure of OMP, we develop look ahead OMP (LAOMP) where the choice of an index in the current iteration is carried out according to its future effect on minimizing the residual norm. For developing the LAOMP algorithm, we first need the following algorithm to describe.

```
    (Look Ahead Residual)
Input: $\mathbf{A}$, $\mathbf{y}$, $K$, previous support set $I$, new chosen index $i$.
Assumption: $i \not\subset I$, $\hat{\mathbf{x}} \in \mathbb{R}^N$.
Initialization: $I_k = I \cup i$, $\hat{\mathbf{x}}_{I_k} = \mathbf{A}^\dagger_{I_k}\mathbf{y}$, $\hat{\mathbf{x}}_{\overline{I}_k} = \mathbf{0}$, $\mathbf{r}_k = \mathbf{y} - \mathbf{A}_{I_k}\hat{\mathbf{x}}_{I_k}$, counter $k = |I_k|$.
  repeat
    $k = k + 1$;
    $i_k$ = index of the highest amplitude component
          of $\mathbf{A}^t r_{k-1}$ such that $i_k \not\subset I_{k-1}$;
    $I_k = I_{k-1} \cup i_k$;
```

```
        $\hat{\mathbf{x}}_{I_k} = \mathbf{A}^{\dagger}_{I_k}\mathbf{y}$; $\hat{\mathbf{x}}_{\overline{I}_k} = \mathbf{0}$;
        $\mathbf{r}_k = \mathbf{y} - \mathbf{A}_{I_k}\hat{\mathbf{x}}_{I_k}$:
    until ($\|\mathbf{r}_k\|_2 > \|\mathbf{r}_{k-1}\|_2$) or ($k > K$)
    $k = k - 1$;
Output: $\mathbf{r} = \mathbf{y} - \mathbf{A}_{I_k}\hat{\mathbf{x}}_{I_k}$ where $\hat{\mathbf{x}}_{I_k} = \mathbf{A}^{\dagger}_{I_k}\mathbf{y}$.
```

The above algorithm can be seen as a look ahead part of the OMP algorithm. Given the sparsity level K, an intermediate support set I and a new choice of an atom index i, the algorithm usually finds a K-element support set at the end and returns the final LS residual. Therefore, for a given intermediate support set I, the final effect of the choice of a new atom in an iteration can be evaluated through using this look ahead part.

Using the above algorithm, let us define the following function.

Function 1 *(Look ahead residual) Let* $\mathbf{y} \in \mathbb{R}^M$, $\mathbf{A} \in \mathbb{R}^{M \times N}$ *and* K *is the sparsity level. Suppose that the previously chosen intermediate support set is* I *and the index chosen in the current iteration is* i. *Then the output residual* $\mathbf{r} \in \mathbb{R}^M$ *using the following algorithmic function is*

$$\mathbf{r} = \text{look ahead resid}\,(\mathbf{y}, \mathbf{A}, K, I, i) \tag{3}$$

where the above function exactly executes the algorithm "Look Ahead Residual".

Using the above definition, the main steps of the LAOMP algorithm are summarized below.

```
    (LAOMP for CS Recovery)
Input: $\mathbf{A}$, $\mathbf{y}$, $K$, look ahead parameter $L \leq K$.
Initialization: $\hat{\mathbf{x}}_0 = \mathbf{0}$, $\mathbf{r}_0 = \mathbf{y}$, $I_0 = \emptyset$, outer loop counter $k = 0$, inner
loop counter $l = 0$.
Define: $\mathbf{n} = [n_1\, n_2 \ldots n_L]^t$
        $\mathbf{j} = [j_1\, j_2 \ldots j_L]^t$.
  repeat
    $k = k + 1$;
    $\mathbf{j}$ = indices of the $L$ highest amplitude components
         of $\mathbf{A}^t r_{k-1}$ such that $\mathbf{j} \not\subset I_{k-1}$;
    for $l = 1$ to $L$ do
      $\mathbf{rr}$ = look ahead resid $(\mathbf{y}, \mathbf{A}, K, I_{k-1}, j_l)$;
      $n_l = \|\mathbf{rr}\|_2$;                    (norm of final residual if $j_l$ is used)
    end for
    $l$ = index of the lowest component of $\mathbf{n}$;
    $i_k = j_l$;                   (chosen index leads to lowest final residual)
```

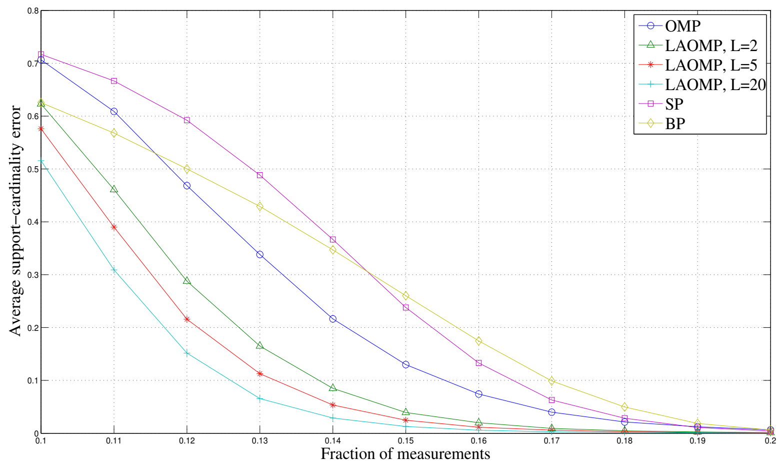


Figure 1: For Gaussian sparse signal in clean condition: Average support-cardinality error (ASCE) versus fraction of measurements (α).

```
        $I_k = I_{k-1} \cup i_k$;
        $\hat{\mathbf{x}}_{I_k} = \mathbf{A}_{I_k}^{\dagger}\mathbf{y}$; $\hat{\mathbf{x}}_{\overline{I}_k} = \mathbf{0}$;
        $\mathbf{r}_k = \mathbf{y} - \mathbf{A}_{I_k}\hat{\mathbf{x}}_{I_k}$;
    **until** $(\|\mathbf{r}^k\|_2 > \|\mathbf{r}^{k-1}\|_2)$ or $(k > K)$
    $k = k - 1$;
Output: $\hat{\mathbf{x}}$, satisfying $\hat{\mathbf{x}}_{I^k} = \mathbf{A}_{I^k}^{\dagger}\mathbf{y}$ and $\hat{\mathbf{x}}_{\overline{I}^k} = \mathbf{0}$.
```

The LAOMP algorithm is based on the OMP algorithm to construct the support set progressively. In the LAOMP algorithm, for the choice of a new atom, we take into account its effect on the final performance in the sense of minimizing the residual. We fix an integer parameter L which is referred to as the look ahead parameter. For the k'th iteration, L best choices of new atoms are found using a matched filtering operation. Then, for each choice of L atoms, we allow the algorithm to execute till it selects K atoms. We choose the best atom among L atoms that results in the minimum norm of fitting residual. The chosen atom is added to the intermediate support set I_{k-1} to form the new support set I_k of cardinality k. The rest of the algorithm remains same as that of like the standard OMP. We note that the algorithm must stops when the number of iterations exceeds K. Like most of the iterative greedy pursuit algorithms (such as OMP, SP, etc.), we assume that the apriori knowl-

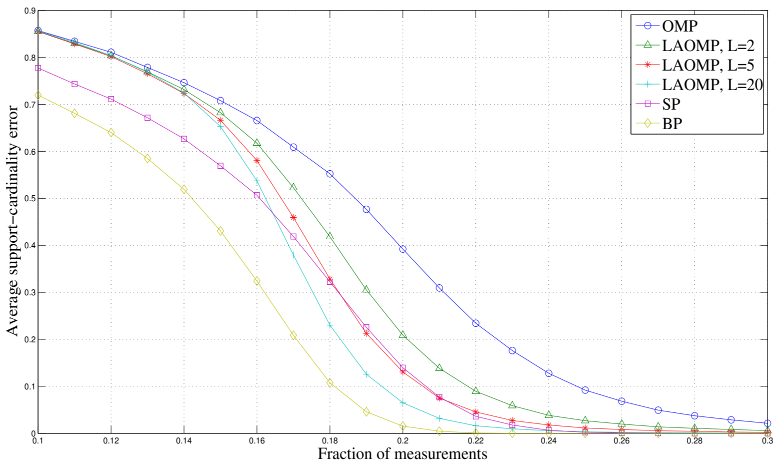


Figure 2: For zero-one sparse signal in clean condition: Average support-cardinality error (ASCE) versus fraction of measurements (α).

edge of the sparsity level K is available. Also note that, as the value of look ahead parameter L increases, better performance can be expected, but at the expense of higher complexity. For $L = 1$, the performance of LAOMP algorithm is the same as like the OMP algorithm.

A natural observation is that the proposed LAOMP algorithm is computationally demanding. If the OMP complexity is $\mathcal{O}(C)$, then the LAOMP complexity can be seen loosely as $\mathcal{O}\left(\frac{KLC}{2}\right)$. To keep a manageable complexity, we impose that $L \leq K$. In our experimental results, we show the effects of several choices of L.

4 Experiments and Results

We performed computer simulations in order to compare the performance of four CS reconstruction algorithms: LAOMP, OMP, subspace pursuit (SP) [10] and basis pursuit (BP) [2]. Among these methods, BP is a convex relaxation based method and the other three methods are iterative greedy search methods. The BP simulation code (matlab) is taken from the l_1-magic toolbox [11]. For LAOMP, we used several L to check the trade-off between complexity and reconstruction performance. We first discuss the reconstruction performance measure and experimental setups, and then show the experimental results. For space limitation in the paper, we only report the results for clean measurements (i.e. $\mathbf{w} = \mathbf{0}$).

4.1 Performance measure and experimental setups

For a K-sparse signal vector $\mathbf{x}$, the support set was denoted as $I_{\mathbf{x}}$ with cardinality K. Let us denote the support set of reconstructed vector $\hat{\mathbf{x}}$ as $I_{\hat{\mathbf{x}}}$. We assume that $\hat{\mathbf{x}}$ is also a K-sparse signal vector, i.e. $|I_{\hat{\mathbf{x}}}| = K$. Among the four CS reconstruction methods that we compare, note that BP is the only method that does not provide a solution vector with a strict sparsity level of K. For the BP, we assume that the support set is constructed from its solution by considering the K highest amplitude coefficients. To compare the methods, we consider to use the distortion $d(I_{\mathbf{x}}, I_{\hat{\mathbf{x}}}) = 1 - (|I_{\mathbf{x}} \cap I_{\hat{\mathbf{x}}}|/K)$ to measure errors [12]. Considering a large number of realizations (data vectors), we can compute the average of $d(I_{\mathbf{x}}, I_{\hat{\mathbf{x}}})$. We define the average support-cardinality error (ASCE) as follows

$$ASCE = \mathcal{E}\left\{d(I_{\mathbf{x}}, I_{\hat{\mathbf{x}}})\right\} = 1 - \frac{1}{K}\mathcal{E}\left\{|I_{\mathbf{x}} \cap I_{\hat{\mathbf{x}}}|\right\}. \tag{4}$$

The ASCE is used as the performance evaluation measure.

Next we discuss experimental setups. In a CS setup, all sparse signal vectors are expected to be exactly reconstructed if the number of measurements is more than a certain threshold value. However, the computational complexity to test this uniform reconstruction ability is exponentially high. Instead, for empirical testing, we can devise a strategy that can compute ASCE for random measurement matrix ensemble. Let us define the fraction of measurements (FoM)

$$\alpha = \frac{M}{N}. \tag{5}$$

Using α, steps of the testing strategy are listed as follows:

1. For given values of the parameters K and N, choose α such that number of measurements M is an integer.

2. Randomly generate an $M \times N$ sensing matrix where the components are drawn independently from a Gaussian source (i.e., $a_{i,j} \sim \mathcal{N}\left(0, \frac{1}{M}\right)$) and then to scale the columns of $\mathbf{A}$ as unit-norm.

3. Randomly generate a set of K-sparse data where the support set $I_{\mathbf{x}}$ is chosen uniformly over the set $\{1, 2, \ldots, N\}$. Let we denote the size of data as S (i.e. the number of signal vectors is S). The

non-zero components of $\mathbf{x}$ are independently drawn by either of the following two methods.

(a) The non-zero components are drawn independently from a standard Gaussian source. This type of signal is referred to as Gaussian sparse signal.

(b) The non-zero components are set to ones. This type of signal is referred to as zero-one sparse signal.

Note that the Gaussian sparse signal is compressible in nature. That means, in the descending order, the sorted amplitudes of a sparse signal vector's components decay fast with respect to the sorted indices. This decaying trend corroborates with several natural signals (for example, wavelet coefficients of an image). On the other hand, a zero-one sparse signal is not compressible in nature, but of special interest for comparative study, since it represents a particularly challenging case for OMP-type of reconstruction strategies [7], [10].

4. For each data, compute the measurement $\mathbf{y} = \mathbf{A}\mathbf{x}$ and apply the CS reconstruction methods independently.
5. Repeat steps 2-4 for a given times (let T times). Then evaluate the CS performance evaluation measure ASCE (averaging over ST data).
6. Repeat steps 1-5 for a new α.

This testing strategy can be performed for any chosen K and N.

4.2 Experimental results

Using $N = 500$, $K = 20$, $S = 100$ and $T = 100$, we performed experiments. That means, we used 500-dimensional sparse signal vectors with sparsity level $K = 20$. Such a 4% sparsity level is chosen in accordance with real life scenarios, such as most of the energy of an image signal in the wavelet domain is concentrated within $2 - 4\%$ coefficients. We used 100 realizations of $\mathbf{A}$ ($T = 100$). For each realization of $\mathbf{A}$, we used 100 signal vectors that are randomly generated ($S = 100$). Then, we incremented α from a lower limit to a higher limit using some small step-sizes (with the constraint that corresponding M is an integer for a value of α). Therefore, for each CS method at a chosen α, the performance is evaluated through averaging over $100 \times 100 = 10000$ realizations. For LAOMP, we used $L = 2$, 5 and 20, to check the trade-off between complexity and reconstruction performance.

For Gaussian sparse signal, Fig. (1) shows the ASCE performance of CS reconstruction methods. We show the results for the range of α from 0.1 to 0.2. This range of α corresponds to the range of M from $M = 50$ to $M = 100$. As expected, we observe that LAOMP performs better with the increase of look ahead parameter L. Note that the LAOMP with $L = 2$ significantly improves the performance compared to the OMP method. Comparing all the CS methods for clean measurements, we note that the LAOMP performs the best for the Gaussian sparse signal.

Next we show the ASCE performance of CS reconstruction methods for zero-one sparse signal in Fig. (2). In this case, we show the results for the range of α from 0.1 to 0.3. Compared to the OMP, SP provides better performance, and BP is found to be the best. The LAOMP algorithm (with all chosen L) performs better than the OMP. This result is encouraging due to fact that the reconstruction of zero-one sparse signal is a challenging case for OMP. For $\alpha > 0.16$, the LAOMP with $L = 5$ provides similar performance to the SP. In that range, the LAOMP with $L = 20$ is found to outperform the SP.

5 Conclusions

In the algorithmic structure of OMP, we show that the choice of an atom in the current iteration can be performed by evaluating its effect on the final performance. The use of look ahead approach allows us to overcome shortcomings of using a matched filter partially. At the expense of higher computation, such an approach is shown to provide a significant improvement compared to the standard OMP for both Gaussian and zero-one sparse signals.

References

[1] E.J. Candes and M.B. Wakin, “An introduction to compressive sampling,” *IEEE Signal Proc. Magazine*, vol. 25, pp. 21-30, March 2008.

[2] S.S. Chen, D.L. Donoho, and M.A. Saunders, “Atomic decomposition by basis pursuit,” *SIAM J. Scientif. Comput.*, vol. 20, no. 1, pp. 33-61, 1998.

[3] E.J. Candes and T. Tao, “Decoding by linear programming,” *IEEE Trans. Info. Theory*, vol. 51, no. 12, pp. 4203-4215, 2005.

[4] S. Ji, Y. Xue, and L. Carin, “Bayesian compressive sensing,” *IEEE Trans. Signal Proc.*, vol. 56, no. 6, pp. 2346-2356, 2008.

[5] D.P. Wipf and B.D. Rao, “Sparse Bayesian learning for basis selection,” *IEEE Trans. Signal Proc.*, vol. 52, no. 8, pp. 2153-2164, 2004.

[6] S.G. Mallat and Z. Zhang, “Matching pursuits with time-frequency dictionaries,” *IEEE Trans. Signal Proc.*, vol. 41, no. 12, pp. 3397-3415, 1993.

[7] J.A. Tropp and A.C. Gilbert, “Signal recovery from random measurements via orthogonal matching pursuit,” *IEEE Trans. Info. Theory*, vol. 53, no. 12, pp. 4655-4666, 2007.

[8] D.L. Donoho, et. al., “Sparse solution of underdetermined linear equations by stagewise orthogonal matching pursuit,” http://www-stat.stanford.edu/∼donoho/reports.html, March 2006.

[9] D. Needell and R. Vershynin, “Uniform uncertainty principle and signal recovery via regularized orthogonal matching pursuit,” *J. Found Comput. Math.*, vol. 9, pp. 317-334, 2009.

[10] W. Dai and O. Milenkovic, “Subspace pursuit for compressive sensing signal reconstruction,” *IEEE Trans. Info. Theory*, vol. 55, no. 5, pp. 2230-2249, 2009.

[11] E.J. Candes and J. Romberg, “l_1-MAGIC: Recovery of sparse signal via convex programming,” A document in l_1-Magic toolbox shared online, http://www.acm.caltech.edu/l1magic/, Oct 2005.

[12] G. Reeves and M. Gastpar, “A note on optimal support recovery in compressed sensing,” *Proc. Asilomar conf.*, pp. 1576-1580, 2009.

Paper C

Look Ahead Parallel Pursuit

Dennis Sundman, Saikat Chatterjee, and Mikael Skoglund

Published in
1st IEEE Swedish Communication Technologies Workshop

Issue Date:	*19-21 Oct. 2011*
On page(s):	*114 - 117*
Location:	*Stockholm, Sweden*
E-ISBN:	*978-1-4577-1876-2*
Print ISBN:	*978-1-4577-1877-9*
INSPEC Accession Number:	*12377627*
Digital Object Identifier:	*10.1109/Swe-CTW.2011.6082477*

©2011 IEEE
The layout has been revised

Look Ahead Parallel Pursuit

Dennis Sundman, Saikat Chatterjee, and Mikael Skoglund

Abstract

We endeavor to improve compressed sensing reconstruction performance of parallel pursuit algorithms. In an iteration, standard parallel pursuit algorithms use a support-set expansion by a fixed number of coefficients, leading to restricted performance. To achive a better performance, we develop a look ahead strategy that adaptively chooses the best number of coefficients. We develop a new algorithm which we call look ahead parallel pursuit, where a look ahead strategy is invoked on a minimal residual norm criterion. The new algorithm provides a trade-off between performance and complexity.

Keywords: Compressed sensing, greedy pursuit algorithms.

1 Introduction

Compressed sensing (CS) [1, 2] refers to a linear under-sampling problem, where the underlying sampled signal is inherently sparse. In the general compressed sensing problem, sampling (or data acquisition) is computationally light but the reconstruction is computationally heavy. This stands in contrast to regular sampling and compression, where the computational burden is placed in the data acquisition side of the system. Because of this shift of burden, CS has the potential to be used in many applications. For example, in sensor networks where the computational capacity at each node is small [3] and in bio-genetic sequencing where scientists deal with enormous data-sets [4]. CS is also useful in brain-scanning where every sample is expensive to acquire [5] and in wide-band spectrum estimation where very high frequency sampling is hardware resource limited [6]. The challenge in CS is to provide for a trade-off between reconstruction and complexity. In the literature, there are three main classes of reconstruction algorithms: convex relaxation, non-convex and greedy-pursuit.

The greedy-pursuits, although they are in general hard to analyze, have lately received growing attention due to their computational efficiency. From a measurement vector, the main principle of greedy-pursuit algorithms is to estimate the underlying support-set of a sparse vector followed by evaluating the associated signal values. The support-set is the set of indices corresponding to non-zero elements of a sparse vector. To estimate the support set and the associated signal values, the greedy-pursuit algorithms use linear algebraic tools, e.g. the matched filter and least squares solution. Prominent examples of such algorithms are orthogonal matching pursuit (OMP) [7], subspace pursuit (SP) [8] and CoSaMP [9]. There have been attempts to improve on these algorithms, mainly in terms of complexity, e.g. (StOMP) [10], but also in terms of recovery performance, e.g. (LAOMP) [11]. There are two main methods to estimate the support set; either to slowly find one (or more) components at a time to finally stop when the support-set is deemed full, or to iteratively refine an estimate of the full support set and stop when the support-set is deemed not to improve by more iterations. We call the algorithms based on iterative refining of the support set as parallel pursuit algorithms (PP) because of their strategy of choosing the elements of the support set simultaneously (or parallely). Typical examples of PP algorithms are SP and CoSaMP.

In this paper, we construct a new algorithm, called look ahead parallel pursuit (LAPP) which can be tuned by a user-defined parameter. A higher value of this parameter provides a better performance at a cost of increased complexity. In order to develop the LAPP algorithm, we identify the common nature of SP and CoSaMP, and then come up with general PP algorithm. In the iterative process for the PP algorithms SP and CoSaMP, both algorithms first increase the size of the support-set estimate with a constant number of coefficients. Then, the most prominent coefficients of this enlarged support-set are kept and the least prominent coefficients discarded. The LAPP is based on the PP algorithmic architecture, but instead of increasing the support-set with a predetermined, constant, factor, the number of increased components are chosen dynamically in each step according to a look-ahead strategy. Through simulations we show that LAPP performs better than both SP and CoSaMP.

Notations: Let a matrix be denoted as $\mathbf{A} \in \mathbb{R}^{M \times N}$ and a vector as $\mathbf{x} \in \mathbb{R}^{M \times N}$. $\mathcal{T}$ is the support-set of $\mathbf{x}$, which is defined in the next section. $\mathbf{A}_{\mathcal{T}}$ is the sub-matrix consisting of the columns in $\mathbf{A}$ corresponding to the elements in a set $\mathcal{T}$. Similarly $\mathbf{x}_{\mathcal{T}}$ is a vector formed by the components of $\mathbf{x}$ that are indexed by $\mathcal{T}$. The pseudo-inverse of $\mathbf{A}$ is denoted as $\mathbf{A}^\dagger$ and the matrix transpose as $\mathbf{A}^T$.

2 Compressed Sensing, SP and CoSaMP

In the standard CS problem, we acquire K-sparse information data $\mathbf{x} \in \mathbb{R}^{N \times 1}$ via the linear measurements

$$\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{w}, \tag{1}$$

where $\mathbf{A} \in \mathbb{R}^{M \times N}$ is a matrix representing the sampling procedure, $\mathbf{w} \in \mathbb{R}^{M \times 1}$ represents standard iid Gaussian measurement noise $w_i \sim \mathcal{N}(0, \sigma_w)$ and $\mathbf{y}$ is the measurement data. In general, the measured signal may not be sparse directly, but is sparse under some linear transform, for example the Fourier or wavelet transform. If we have the non-sparse signal $\mathbf{z}$ and the linear transform $\mathcal{F}$, we require that $\mathcal{F}\mathbf{z}$ is sparse. We can then let $\mathbf{A} = \mathbf{B}\mathcal{F}$ and assume equation (1) without loss of generality.

From the measurement data $\mathbf{y}$ and the measurement matrix $\mathbf{A}$, the standard CS problem boils down to finding $\mathbf{x}$. Next, we will describe how $\mathbf{x}$ can be found using the SP or CoSaMP algorithm.

2.1 SP and CoSaMP

SP and CoSaMP are two similar algorithms that were developed independently of each-other. We focus on explaining the SP algorithm and then describes what differs between SP and CoSaMP.

To simplify algorithmic notation, we introduce four helping functions: `resid`, `max_indices`, `update` and PP.

$$\texttt{resid}(\mathbf{y}, \mathbf{A}_\mathcal{T}) \triangleq \mathbf{y} - \mathbf{A}_\mathcal{T}\mathbf{A}_\mathcal{T}^\dagger\mathbf{y}, \tag{2}$$

Given an active support-set $\mathcal{T}$, the first function `resid` computes the residual vector from the real measurement vector to the measurement vector projected on the active set. The second function, `max_indices` is defined as:

$$\begin{aligned}\texttt{max\_indices}(\mathbf{x}, k) \triangleq \{&\text{the set of indices corresponding} \\ &\text{to the } k \text{ largest amplitude components of } \mathbf{x}\}.\end{aligned} \tag{3}$$

From a support-set $\mathcal{T}_0$ of cardinality K, `update` uses a matched filter on a residual vector to find K_grow new indices. It extends the starting support-set with these new indices and creates $\mathcal{T}'$. The algorithm then projects the measurement data $\mathbf{y}$ onto the space described by the span of

Function 3 : update$(\mathbf{y}, \mathbf{A}, \mathcal{T}_0, \mathbf{r}_0, K, K_{\text{grow}})$

```
1: $\mathcal{T}' \leftarrow \mathcal{T}_0 \cup \texttt{max\_indices}(\mathbf{A}^T\mathbf{r}_0, K_{\text{grow}})$
2: $\hat{\mathbf{x}} \in \mathbb{R}^N$   such that   $\hat{\mathbf{x}}_{\mathcal{T}'} = \mathbf{A}^{\dagger}_{\mathcal{T}'}\mathbf{y}$ and $\hat{\mathbf{x}}_{\bar{\mathcal{T}}'} = \mathbf{0}$
3: $\mathcal{T} \leftarrow \texttt{max\_indices}(\hat{\mathbf{x}}, K)$
4: $\mathbf{r} \leftarrow \texttt{resid}(\mathbf{y}, \mathbf{A}_{\mathcal{T}})$
Output: $[\mathcal{T}, \mathbf{r}]$
```

$\mathbf{A}_{\mathcal{T}'}$, from which it picks the the K indices corresponding to the largest amplitude. Unless the algorithm has already converged, $\mathcal{T}, \mathbf{r}$ will be different from $\mathcal{T}_0, \mathbf{r}_0$.

Function 4 : PP$(\mathbf{y}, \mathbf{A}, \mathcal{T}_0, \mathbf{r}_0, K, K_{\text{grow}}, i)$

```
1: $k \leftarrow 0$
2: repeat
3:    $k \leftarrow k + 1$
4:    $[\mathcal{T}_k, \mathbf{r}_k] \leftarrow \texttt{update}(\mathbf{y}, \mathbf{A}, \mathcal{T}_{k-1}, \mathbf{r}_{k-1}, K, K_{\text{grow}})$
5: until $(\|\mathbf{r}_k\|_2 \geq \|\mathbf{r}_{k-1}\|_2)$ or $(k = i + 1)$
6: $k \leftarrow k - 1$                                   (‘Previous iteration count’)
Output:
1: $\hat{\mathcal{T}} \leftarrow \mathcal{T}_k$
2: $\hat{\mathbf{x}} \in \mathbb{R}^N$   such that   $\hat{\mathbf{x}}_{\mathcal{T}_k} = \mathbf{A}^{\dagger}_{\mathcal{T}_k}\mathbf{y}$ and $\hat{\mathbf{x}}_{\bar{\mathcal{T}}_k} = \mathbf{0}$
3: $\mathbf{r} \leftarrow \mathbf{r}_k$
```

Given a starting support-set $\mathcal{T}_0$, a residual vector $\mathbf{r}_0$, PP repeatedly calls `update` to iteratively refine the support-set. When the residual-norm is no longer decreasing, the PP algorithm quits. The parameter i tells how many iterations are allowed at maximum. By controlling i, a forceable quit of the algorithm saves computation.

The general structure of both SP and CoSaMP can be constructed using PP as shown in Algorithm 1. In Algorithm 1, initial support-set $\mathcal{T}_0$ and residual $\mathbf{r}_0$ are supplied for later use by PP.

2.2 Difference between sp and cosamp

The main difference between SP and CoSaMP is that while SP chooses $K_{\text{grow}} = K$, CoSaMP chooses $K_{\text{grow}} = 2K$. In practice, these algorithms perform similarly. However, it is clear that for each data vector $\mathbf{x}$ and each measurement matrix $\mathbf{A}$ they will provide a different solution. This is the basic idea we want to exploit in creating a generalized algorithm,

Algorithm 1 : SP and CoSaMP

```
Input: $\mathbf{y}, \mathbf{A}, K, K_{\text{grow}}$
Initialization:
  1: $\mathcal{T}_0 \leftarrow \texttt{max\_indices}(\mathbf{A}^T\mathbf{y}, K)$
  2: $\mathbf{r}_0 \leftarrow \texttt{resid}(\mathbf{y}, \mathbf{A}_{\mathcal{T}_0})$
Iteration:
  1: $[\hat{\mathcal{T}}, \mathbf{r}, \hat{\mathbf{x}}] \leftarrow \text{PP}(\mathbf{y}, \mathbf{A}, \mathcal{T}_0, \mathbf{r}_0, K, K_{\text{grow}}, \infty)$
Output:
  1: $\hat{\mathbf{x}}, \hat{\mathcal{T}}$
```

which will be described in the next section.

3 Look ahead Parallel Pursuit

We call the new algorithm look ahead parallel pursuit (LAPP). The term look ahead comes from that the algorithm, in each iteration step, chooses the best value for K_{grow} based on a minimum residual-norm criterion. For SP, K_{grow} is fixed every iteration to K and for CoSaMP K_{grow} is fixed to $2K$. In LAPP, we let the algorithm choose any K_{grow} between K and $2K$ dynamically for each iteration step.

To determine which K_{grow} to use in each iteration step, LAPP evaluates the residual norm. By chosing the user-defined parameter $i = 1$, the algorithm evaluates the residual norm for each K_{grow} after one iteration step of the PP algorithm. By increasing the parameter i, the PP algorithm is allowed to perform more iterations (i.e. look "further in the future"). Thus, by choosing a bigger i, the LAPP algorithm is provided with a better residual to base its choice of K_{grow} on. We can make the algorithm look all the way to the future (usually less than $\mathcal{O}(K)$ steps) by letting $i = \infty$. By doing so we can guarantee that the fitting residual norm is smaller than that of both SP and CoSaMP. To save computational resources, one can use a lower value of i without much loss in performance. LAPP outputs the support-set estimate $\hat{\mathcal{T}}$, the signal estimate $\hat{\mathbf{x}}$ and the residual $\mathbf{r}$.

3.1 Comparison of Complexity

For $i = \infty$, the complexity of LAPP is a factor K^2 times higher than that of SP and CoSaMP. In the simulation results shown in section 4, it turns out that the use of $i = 1$ leads to a substantial better performance than SP and CoSaMP. At $i = 1$ the increased complexity is only a factor of

Algorithm 2 : LAPP

```
Input: $\mathbf{y}, \mathbf{A}, K, i$
Initialization:
 1: $\mathcal{T}_0 \leftarrow \texttt{max\_indices}(\mathbf{A}^T\mathbf{y}, K)$
 2: $\mathbf{r}_0 \leftarrow \texttt{resid}(\mathbf{y}, \mathbf{A}_{\mathcal{T}_0})$
 3: $k \leftarrow 0$
Iteration:
 1: repeat
 2:   $k \leftarrow k + 1$
 3:   $l \leftarrow 0$
 4:   repeat
 5:     $l \leftarrow l + 1$
 6:     $K_{\text{grow}} \leftarrow K + l - 1$
 7:     $[\mathcal{T}_l, \mathbf{r}_l] \leftarrow \texttt{update}(\mathbf{y}, \mathbf{A}, \mathcal{T}_{k-1}, \mathbf{r}_{k-1}, K, K_{\text{grow}})$
 8:     $[\cdot, \cdot, \mathbf{r}_l] \leftarrow \text{PP}(\mathbf{y}, \mathbf{A}, \mathcal{T}_l, \mathbf{r}_l, K, K_{\text{grow}}, i)$
 9:     $n(l) \leftarrow \|\mathbf{r}_l\|_2^2$
10:   until $(l = K + 1)$
11:   $l^*$ such that $n(l^*) \leq n(l), \forall l$
12:   $K_{\text{grow}} \leftarrow K + l^* - 1$
13:   $[\mathcal{T}_k, \mathbf{r}_k] \leftarrow \texttt{update}(\mathbf{y}, \mathbf{A}, \mathcal{T}_{k-1}, \mathbf{r}_{k-1}, K, K_{\text{grow}})$
14: until $(\|\mathbf{r}_k\|_2 \geq \|\mathbf{r}_{k-1}\|_2)$
15: $k \leftarrow k - 1$                                  (‘Previous iteration count’)
Output:
 1: $\hat{\mathcal{T}} \leftarrow \mathcal{T}_k$
 2: $\hat{\mathbf{x}} \in \mathbb{R}^N$ such that $\hat{\mathbf{x}}_{\mathcal{T}_k} = \mathbf{A}_{\mathcal{T}_k}^{\dagger}\mathbf{y}$ and $\hat{\mathbf{x}}_{\bar{\mathcal{T}}_k} = \mathbf{0}$
 3: $\mathbf{r} \leftarrow \mathbf{r}_k$
```

K. Usually K is ignored in the complexity calculation since it is much smaller compared to N and M.

For OMP, one need to do one matched filter and one orthogonal projection in each iteration. Assuming that the orthogonal projection is implemented using the recursive block-wise matrix inversion, and assuming a worst-case of K iterations, OMP complexity is $\mathcal{O}(KMN)$. SP and CoSaMP are required to perform one matched filter and two orthogonal projections in each iteration. Therefore their complexity is $\mathcal{O}(KMN)$.

In LAPP with $i = 1$ has the complexity of $\mathcal{O}(K^2MN)$, while LAPP with $i = \infty$ has the complexity of $\mathcal{O}(K^3MN)$. Since K is generally much smaller than both M and N, the increase in complexity for LAPP is not prohibitive.

4 Experiments and Results

For the experiments and results, we compare the LAPP algorithm with SP, CoSaMP and OMP. OMP is known for being very good at recovering sparse signals with Gaussian non-zero components, while at the same time it gives poor performance for binary (0/1) signals. We characterize the measurement noise as signal-to-measurement-noise-ratio as

$$\text{SMNR} = 10 \log_{10} \frac{\mathbb{E}\left\{\|\mathbf{x}\|_2^2\right\}}{\mathbb{E}\left\{\|\mathbf{w}\|_2^2\right\}}. \tag{4}$$

We show results for binary and Gaussian signals in a clean setting and in a setting with measurement noise. In the noisy setting, we show results for SMNR = 20 dB.

To compare the algorithms, we use the signal-to-reconstruction-noise-ratio (SRER) which is defined as

$$\text{SRER} = 10 \log_{10} \frac{\mathbb{E}\left\{\|\mathbf{x}\|_2^2\right\}}{\mathbb{E}\left\{\|\mathbf{x}-\hat{\mathbf{x}}\|_2^2\right\}}. \tag{5}$$

Next we describe the simulation setup. In any CS setup, all sparse signals are expected to be exactly reconstructed (in absence of noise) if the number of measurements are more than a certain threshold value. The computational complexity to test this uniform reconstruction ability is exponentially high. Instead, we can rely on empirical testing, where SRER is computed for random measurement matrix ensemble. We define the fraction of measurements

$$\alpha = \frac{M}{N}. \tag{6}$$

Using α, the testing is performed as follows:

1. Given the signal parameter N, choose an α (such that M is an integer).
2. Randomly generate an $M \times N$ sensing matrix $\mathbf{A}$ where the components are drawn independently from an i.i.d. Gaussian source (i.e. $a_{m,n} \sim \mathcal{N}\left(0, \frac{1}{M}\right)$) and then scale the columns of $\mathbf{A}$ to unit-norm.
3. Generate a support-set $\mathcal{T}$ of cardinality K. The support-set is uniformly chosen from $\{1, 2, ..., N\}$.
4. Randomly generate a sparse signal vector $\mathbf{x}$ with non-zero components determined by the support-set in step 3. The non-zero components in the vector are chosen independently from a Gaussian source for the Gaussian signals, and chosen as 1 for the binary signal.

5. Compute the measurement vector $\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{w}$, where $\mathbf{w}$ is standard iid Gaussian noise.
6. Apply the CS algorithms on the data $\mathbf{y}$, $\mathbf{A}$.

In the simulation procedure above, a number Q different sensing matrices $\mathbf{A}$ are created. For each sensing matrix, P data vectors are generated. In total, we will average over $Q \cdot P$ data to evaluate the performance.

4.1 Parameters and simulation set-up

For the plots presented in this paper, we have chosen: $N = 500$, $K = 20$. We have chosen the number of matrices $\mathbf{A}$ to $Q = 200$ and the number of data-sets $\mathbf{x}$ to 200 (i.e. $P = 200$), giving a total number of $Q \cdot P = 40000$ data for statistics.

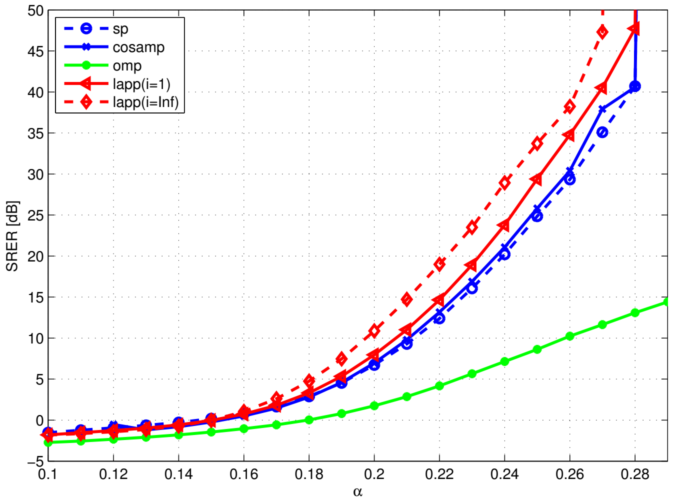
4.2 Analysis of the simulation results

We provide four plots for the simulation results. The two first set of plots are for clean and noisy binary signals, while the second set of plot are for clean and noisy Gaussian signals. The comparison with binary and Gaussian signals are of particular interest since OMP is known to outperform SP and CoSaMP for Gaussian signals.

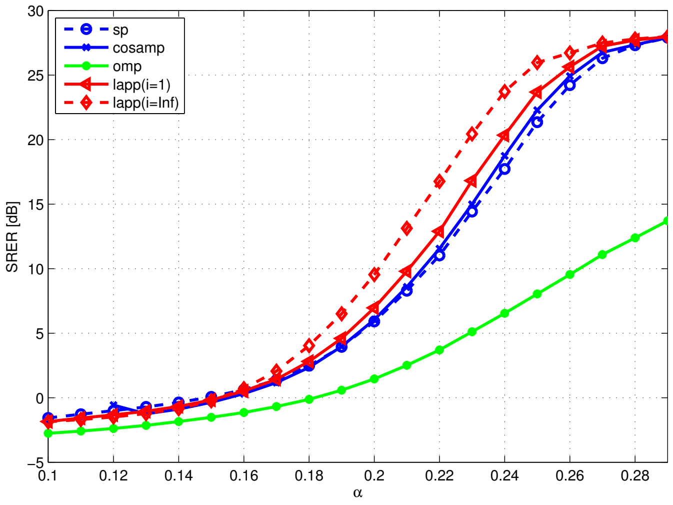
In Fig. 1a and Fig. 1b, we see the results of the different algorithms for a binary signal. For both figures at $\alpha = 0.22$, LAPP with $i = \infty$ performs about 6 dB better than SP and CoSaMP at its best point. Even choosing LAPP with a small parameter $i = 1$ still outperforms all other algorithms (except LAPP with $i = \infty$). Notice here how OMP performs much worse than all other algorithms.

In Fig. 2a and Fig. 2b, we see the results of the different CS algorithms for a Gaussian signal. In comparison to the binary signal, OMP now performs similarly to the other algorithms and in the case of noisy measurements, OMP outperforms SP and CoSaMP at most measurement points. LAPP with $i = 1$ also beats both SP and CoSaMP at all measure points for both figures, but in the noisy setting (Fig. 2b) OMP is better. In the clean setting (Fig. 2a) for very small $\alpha < 0.18$, LAPP with $i = 1$ is beaten with a small margin by OMP. However, LAPP with $i = \infty$ performs better than all other algorithms. For clean measurements in Fig. 2a, at $\alpha = 0.2$, LAPP with $i = \infty$ is some 10 dB better than SP and CoSaMP.

Reproducible results: In the spirit of reproducible results, we provide necessary downloadable MATLAB codes in the following website: *https://sites.google.com/site/saikatchatt/softwares/*.



(a) Clean measurements

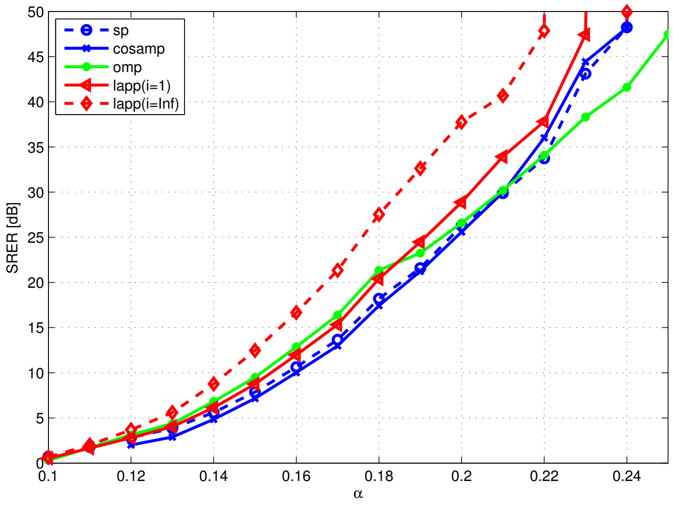


(b) Noisy measurements, where SMNR = 20 dB

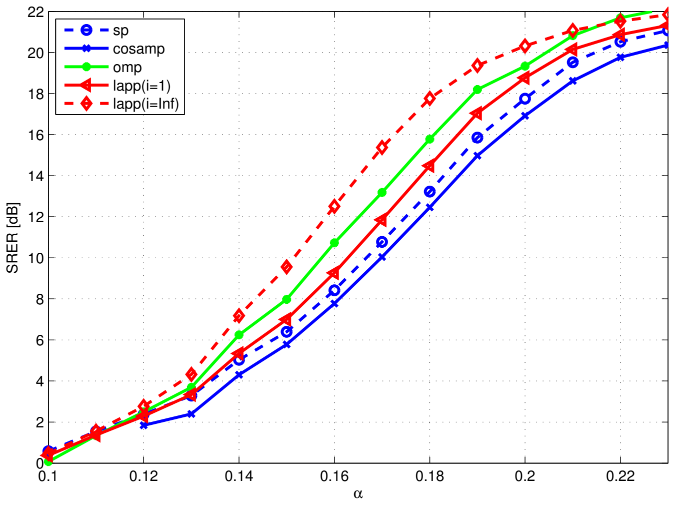
Figure 1: SRER for a binary signal, where the signal parameters $K = 20$, $N = 500$.

References

[1] D.L. Donoho, "Compressed sensing," *IEEE Trans. Inf. Theory*, vol. 52, no. 4, pp. 1289 –1306, April 2006.



(a) Clean measurements



(b) Noisy measurements, where SMNR = 20 dB

Figure 2: SRER for a Gaussian signal, where the signal parameters $K = 20$, $N = 500$.

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

[3] A.Y. Yang, M. Gastpar, R. Bajcsy, and S.S. Sastry, "Distributed sensor

perception via sparse representation," *Proc. of the IEEE*, vol. 98, no. 6, pp. 1077 –1088, June 2010.

[4] Tong Tong Wu, Yi Fang Chen, Trevor Hastie, Eric Sobel, and Kenneth Lange, "Genome-wide association analysis by lasso penalized logistic regression," *Bioinformatics*, vol. 25, no. 6, pp. 714–721, 2009.

[5] J.W. Phillips, R.M. Leahy, and J.C. Mosher, "Meg-based imaging of focal neuronal current sources," *IEEE Trans. Med. Imaging*, vol. 16, no. 3, pp. 338 –348, June 1997.

[6] D. Sundman, S. Chatterjee, and M. Skoglund, "On the use of compressive sampling for wide-band spectrum sensing," in *Proc. IEEE Int. Symp. Signal Processing and Inf. Tech. (ISSPIT)*, Dec. 2010, pp. 354 –359.

[7] J.A. Tropp and A.C. Gilbert, "Signal recovery from random measurements via orthogonal matching pursuit," *IEEE Trans. Inf. Theory*, vol. 53, no. 12, pp. 4655 –4666, Dec. 2007.

[8] Wei Dai and O. Milenkovic, "Subspace pursuit for compressive sensing signal reconstruction," *IEEE Trans. Inf. Theory*, vol. 55, no. 5, pp. 2230 –2249, May 2009.

[9] D. Needell and J. A. Tropp, "CoSaMP: Iterative signal recovery from incomplete and inaccurate samples," *Appl. and Comp. Harm. Analysis*, vol. 26, pp. 301–321, May 2009.

[10] David L. Donoho, Yaakov Tsaig, Iddo Drori, and Jean luc Starck, "Sparse solution of underdetermined linear equations by stagewise orthogonal matching pursuit," Tech. Rep., 2006.

[11] S. Chatterjee, D. Sundman, and M. Skoglund, "Look ahead orthogonal matching pursuit," in *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing (ICASSP)*, May 2011, pp. 4024 –4027.

Paper D

Greedy Pursuits for Compressed Sensing of Jointly Sparse Signals

Dennis Sundman, Saikat Chatterjee, and Mikael Skoglund

Published in
19'th European Signal Processing Conference

Issue Date:	*29 Aug. - 2 Sep. 2011*
On page(s):	*368 - 372*
Location:	*Barcelona, Spain*

©2011 IEEE
The layout has been revised

Greedy Pursuits for Compressed Sensing of Jointly Sparse Signals

Dennis Sundman, Saikat Chatterjee, and Mikael Skoglund

Abstract

For compressed sensing with jointly sparse signals, we present a new signal model and two new joint iterative-greedy-pursuit recovery algorithms. The signal model is based on the assumption of a jointly shared support-set and the joint recovery algorithms have knowledge of the size of the shared support-set. Through experimental evaluation, we show that the new joint algorithms provide significant performance improvements compared to regular algorithms which do not exploit a joint sparsity.

Keywords: Compressed sensing, distributed compressed sensing, joint compressed sensing.

1 Introduction

Compressed sensing (CS) [1, 2] is based on a sparse signal model and utilizes an under-determined system of linear equations for data acquisition and reconstruction. The CS process is in general computationally light, and straight forward, in the acquisition part and computationally heavy in the reconstruction part. For the reconstruction part, we note in the literature three classes of algorithms: convex relaxation based, non-convex and iterative-greedy-pursuits.

Due to the complexity in the reconstruction part of CS, iterative-greedy-pursuit algorithms have recently attracted much attention. Examples of such algorithms are Matching pursuit [3], Orthogonal matching pursuit (OMP) [4], Look ahead orthogonal matching pursuit [5] and Subspace pursuit (SP) [6]. From a measurement vector, the main principle of iterative-greedy-pursuit algorithms is to estimate the underlying support-set of a sparse vector followed by evaluating the associated signal values. The support-set is the set of indices corresponding to non-zero elements of a sparse vector. To estimate the support set and the associated signal values, the iterative-greedy-pursuit algorithms use linear algebraic tools, e.g. the matched filter and least squares solution.

Recently we have seen a rising interest in solving CS for multiple jointly sparse signals [7]. We refer to such a problem as the *joint CS problem* and application scenarios referenced in the literature are magnetoencephalography [8–11], spectrum analysis [12] and wireless sensor networks [13]. A joint CS problem has two main aspects: (1) system set-up, (2) signal model. Depending on these two aspects, a reconstruction algorithm needs to be developed.

First, for the system set-up, we notice that [8] and [9] use a single sensor (one sensing matrix) to sample signals. Duarte et al. [7] are a bit more general and provide a system set-up where multiple sensors are present. Next, we observe that [8,9] have a signal model defined in such a way that all sparse signals have one common support-set. We refer to the signal model of [8,9] as the common support-set model. On the other hand, the signal model in [7] is defined in such a way that the signals have common and individual signal parts. We refer to this signal model as the mixed signal model.

We now discuss existing reconstruction algorithms for the joint CS problem. Based on a single sensor system set-up and a common support-set model, several greedy-pursuit algorithms have been proposed [8,9]. These algorithms can not be applied for the more general multiple sensor system set-up of [7], as well as the mixed signal model. For the multiple sensor system set-up along with the mixed signal model, convex relaxation as well as iterative greedy-pursuit algorithms were proposed in [7].

In this paper, for the joint CS problem, we present a new signal model and develop two new iterative-greedy-pursuit algorithms. Our system set-up is the same as [7] where multiple sensors are present, but the signal model is different. For the signal model, we use common and individual support-sets to characterize jointly sparse signals. Unlike [7], there is no restriction on the associated signal values. We refer to this new signal model as the *mixed support-set* model. The mixed support-set model is shown to be a generalization over all previously proposed models. To take advantage of the mixed support-set model, we develop two new iterative-greedy-pursuit algorithms based on regular OMP and regular SP. We refer to these new algorithms as *joint OMP* and *joint SP*. The only knowledge the proposed algorithms use is the cardinalities of the common and individual support-sets. The new algorithms utilize the strategy of jointly finding the common support-set iteratively. Through experimental evaluations, we show significant improvements compared to the regular OMP and SP.

Notations: Let a matrix be denoted as $\mathbf{A} \in \mathbb{R}^{M \times N}$ and a vector as $\mathbf{x} \in \mathbb{R}^{M \times N}$. $\mathcal{I}$ is the support-set of $\mathbf{x}$, which is defined in the next section. $\mathbf{A}_{\mathcal{I}}$ is the submatrix consisting of the columns in $\mathbf{A}$ corresponding to the elements in a set $\mathcal{I}$. Similarly $\mathbf{x}_{\mathcal{I}}$ is a vector formed by the components of $\mathbf{x}$ that are indexed by $\mathcal{I}$. The pseudo inverse of $\mathbf{A}$ is denoted as $\mathbf{A}^{\dagger}$ and the matrix transpose as $\mathbf{A}^{T}$.

2 Mixed Support-set Model

Using the general multiple sensor system set-up (as in [7]), we first describe the joint CS problem and then the new mixed support-set signal model. First, for the l'th sensor, we have the sparse signal $\mathbf{x}_l$ which is observed for the joint CS problem as

$$\mathbf{y}_l = \mathbf{A}_l \mathbf{x}_l + \mathbf{w}_l, \qquad \forall l \in \{1, 2, ..., L\}, \tag{1}$$

where $\mathbf{y}_l \in \mathbb{R}^{M \times 1}$ is a measurement vector, $\mathbf{A}_l \in \mathbb{R}^{M \times N}$ a measurement matrix, $\mathbf{w}_l \in \mathbb{R}^{N \times 1}$ is the measurement error, and $M < N$. $\mathbf{A}_l$ and $\mathbf{w}_l$ are independent across l. The signal vector $\mathbf{x}_l$ has K_l non-zero components with indices $\mathcal{I}_l = \{i : x_l(i) \neq 0\}$. $\mathcal{I}_l$ is referred to as the support-set of $\mathbf{x}_l$ and the cardinality is $|\mathcal{I}_l| = \|\mathbf{x}_l\|_0 = K_l$. Using the set of $\{\mathbf{y}_l\}_{l=1}^{L}$, the joint CS reconstruction problem endeavors for finding $\{\mathbf{x}_l\}_{l=1}^{L}$ by exploiting some shared structure among the l sensors defined by the underlying signal model.

Now, we describe the new mixed support-set signal model with a shared structure where the signal vector $\mathbf{x}_l$ consists of two parts

$$\mathbf{x}_l = \mathbf{z}_l^{(c)} + \mathbf{z}_l^{(p)}, \qquad \forall l \in \{1, 2, ..., L\}. \tag{2}$$

In the new model (2) both $\mathbf{z}_l^{(c)}$ and $\mathbf{z}_l^{(p)}$ have independent non-zero components. There are $K_l^{(p)}$ non-zero values associated with $\mathbf{z}_l^{(p)}$. For simplicity we assume that the non-zero values are located uniformly at random over the support-set $\mathcal{I}_l^{(p)} \in \{1, 2, \ldots, N\}$, and $\forall l \in \{1, 2, \ldots, L\}$. For $\mathbf{z}_l^{(c)}$ there are similarly $K^{(c)}$ non-zero components with the constraint that the support-set is shared, $\mathcal{I}_l^{(c)} = \mathcal{I}^{(c)}, \forall l \in \{1, 2, \ldots, L\}$. The elements of $\mathcal{I}^{(c)}$ are the same (common) to all signals, but unknown to the reconstructor[1]. This gives a support-set $\mathcal{I}_l$ for each set of signals as

$$\mathcal{I}_l = \mathcal{I}^{(c)} \cup \mathcal{I}_l^{(p)}, \qquad \forall l \in \{1, 2, ..., L\}. \tag{3}$$

[1]For easy practical implementation, we assume that the elements are uniformly distributed over $\mathcal{I}^{(c)}$ and $\mathcal{I}_l^{(p)}$.

We define $K_{l,\max} = |\mathcal{I}^{(c)}| + |\mathcal{I}_l^{(p)}| = K^{(c)} + K_l^{(p)}$. Note that the support-sets can intersect, so $K_{l,\max} \geq K_l$.

From the set of measurement vectors $\{\mathbf{y}_l\}_{l=1}^L$, the task is to find a good estimate $\{\hat{\mathbf{x}}_l\}_{l=1}^L$ using the a-priori knowledge of the cardinality of the support-sets $\{K_l^{(p)}\}_{l=1}^L$ and $K^{(c)}$.

2.1 Generalization over existing signal models

The purpose of this section is to clarify what is new in the mixed support-set model. Let us compare our model with the mixed signal model of [7], where $\mathbf{x}_l$ is composed of common and individual parts

$$\mathbf{x}_l = \mathbf{z}^{(c)} + \mathbf{z}_l^{(p)}, \qquad \forall l \in \{1, 2, ..., L\}. \tag{4}$$

Here $\mathbf{z}^{(c)}$ represents a common sparse signal part and $\mathbf{z}_l^{(p)}$ represents the individual (private) signal part for the l'th sensor. Note that $\mathbf{z}^{(c)}$ is fixed for all the data set. Comparing (2) and (4) we can say that the new mixed support-set model is a generalization over the mixed signal model in the sense that the former allows for different values in the common signal part $\mathbf{z}_l^{(c)}$.

The common support-set model [8, 9] used in magnetoencephalography has no individual signal parts at all. Their model is

$$\mathbf{x}_l = \mathbf{z}_l^{(c)}, \qquad \forall l \in \{1, 2, ..., L\}. \tag{5}$$

Therefore our model (2) also generalizes the model (5).

The regular CS algorithms OMP and SP were not constructed to exploit the knowledge of a shared structure. Furthermore, none of the algorithms presented in [7–9] can solve the joint CS problem based on the new mixed support-set model (2). Therefore, in the following two sections, we develop two new iterative-greedy-pursuit algorithms.

3 Joint Orthogonal Matching Pursuit

In this section, based on the regular OMP algorithm, we present a new joint OMP algorithm for solving the joint CS problem (1). This algorithm works for multiple sensors and is based on the new general mixed support-set model (2). For clarity in the algorithmic notation, we define three functions as follows

$$\texttt{resid}(\mathbf{y}, \mathbf{A}) \triangleq \mathbf{y} - \mathbf{A}\mathbf{A}^{\dagger}\mathbf{y}, \tag{6}$$

$$\texttt{max\_indices}(\mathbf{x}, k) \triangleq \{\text{the set of indicies corresponding to the } k \text{ largest amplitude components of } \mathbf{x}\}, \tag{7}$$

and

$$\text{add}_1(\mathbf{s}, \mathcal{I}) \triangleq \{\text{In the } N \times 1 \text{ vector } \mathbf{s}, \text{ add 1 to every index corresponding to the elements in the support-set } \mathcal{I}\}. \tag{8}$$

3.1 Joint Orthogonal Matching Pursuit

We now describe the new joint OMP algorithm that uses the new mixed support-set model. For developing the joint OMP algorithm, we modify the regular OMP algorithm so that it can use an estimate of the common support-set as an initial support-set. Our assumption is that over the iterations, the estimate of the common support-set will improve. The necessary modifications to the regular OMP is presented in subsection 3.2.

We now show the steps of the joint OMP in algorithm 3. In the

Algorithm 3 : joint OMP

```
Input: ${\mathbf{A}_l}_{l=1}^L$, ${K_l^{(p)}}_{l=1}^L$, $K^{(c)}$ ,${\mathbf{y}_l}_{l=1}^L$
Initialization:
 1: $k \leftarrow 0$                                   ('Iteration counter')
 2: $\mathcal{I}^{(c)} \leftarrow \emptyset$          ('Initial common support-set')
Iterations:
 1: repeat
 2:   $k \leftarrow k + 1$
 3:   $\mathbf{s} \leftarrow \mathbf{0}_{N\times 1}$
 4:   for $\forall l \in \{1, 2, ..., L\}$ do
 5:     $K_{l,\max} \leftarrow K^{(c)} + K_l^{(p)} - \text{size}(\mathcal{I}^{(c)})$
 6:     $(\mathcal{I}_l, \hat{\mathbf{x}}_l, n_l) \leftarrow \text{OMP}(\mathbf{A}_l, K_{l,\max}, \mathbf{y}_l, \mathcal{I}^{(c)})$
 7:     $\mathbf{s} \leftarrow \text{add}_1(\mathbf{s}, \mathcal{I}_l)$
 8:   end for
 9:   $\mathcal{I}^{(c)} \leftarrow \texttt{max\_indices}(\mathbf{s}, k)$
10: until $(k = K^{(c)})$
Output: ${\hat{\mathbf{x}}_l}_{l=1}^L$, ${\mathcal{I}_l}_{l=1}^L$
```

k'th iteration stage, the algorithm finds a temporary $K_{l,\max}$ (step 5), which is passed on to the underlying modified OMP together with the estimated common support-set $\mathcal{I}^{(c)}$ from the previous iteration (step 6). An N-sized zero vector $\mathbf{s}$ adds for each estimated support-set $\mathcal{I}_l$ an one at the location of the predicted non-zero element (step 7). If the modified OMP algorithm finds one support-set element that is common for all data, the value of $\mathbf{s}$ at this index will have the maximum size of L. The indices corresponding to the k largest elements in $\mathbf{s}$ are then chosen as the common support-set $\mathcal{I}^{(c)}$ (step 9). Once an element is chosen, it will be fed to the modified OMP algorithm in the next iteration and always remain in $\mathcal{I}^{(c)}$. Because a chosen index will always remain in the common support-set, it is important to carefully select each index to add to the common support-set. We find that, in practice, the use of $\text{add}_1(\cdot,\cdot)$ and $\texttt{max\_indices}(\cdot,\cdot)$ allows us for a good estimate of $\mathcal{I}^{(c)}$.

3.2 Modified OMP

As mentioned in subsection 3.1, we now describe the modified OMP in algorithm 4. Instead of initializing with an empty support-set and begin iterating from the first component as in the regular OMP, we here allow the algorithm to take an initial support-set and continue building the final support-set from this. This modification boils down to the regular OMP as shown by Tropp and Gilbert [4] when the initial support-set $\mathcal{I}_{\text{ini}} = \emptyset$. The modified OMP algorithm starts with finding a residual

Algorithm 4 : OMP (including modifications)

```
Input: $\mathbf{A}$, $K_{\max}$, $\mathbf{y}$, $\mathcal{I}_{\text{ini}}$.
Initialization:
  1: $\mathcal{I}_0 \leftarrow \mathcal{I}_{\text{ini}}$
  2: $\mathbf{r}_0 \leftarrow \texttt{resid}(\mathbf{y}, \mathbf{A}_{\mathcal{I}_0})$
  3: $k \leftarrow |\mathcal{I}_0|$
Iteration:
  1: repeat
  2:     $k \leftarrow k + 1$
  3:     $i_{\max} \leftarrow \texttt{max\_indices}\left(\mathbf{A}^T \mathbf{r}_{k-1}, 1\right)$
  4:     $\mathcal{I}_k \leftarrow \mathcal{I}_{k-1} \cup i_{\max}$
  5:     $\hat{\mathbf{x}}_{\mathcal{I}_k} \leftarrow \mathbf{A}^{\dagger}_{\mathcal{I}_k} \mathbf{y}$
  6:     $\mathbf{r}_k \leftarrow \texttt{resid}(\mathbf{y}, \mathbf{A}_{\mathcal{I}_k})$
  7: until $(k = K_{\max})$
Output:
  1: $\hat{\mathcal{I}} \leftarrow \mathcal{I}_k$
  2: $\hat{\mathbf{x}}$ such that $\hat{\mathbf{x}}_{\mathcal{I}_k} = \mathbf{A}^{\dagger}_{\mathcal{I}_k} \mathbf{y}$ and $\hat{\mathbf{x}}_{\bar{\mathcal{I}}_k} = \mathbf{0}$
  3: $n_r \leftarrow \|\mathbf{r}_k\|_2$
```

(step 2 of Initialization), where $\mathcal{I}_0 = \mathcal{I}_{\text{ini}}$. If the initial support-set

$\mathcal{I}_{\text{ini}} = \emptyset$, the matrix $\mathbf{A}_{\mathcal{I}_0}$ is empty and the residual becomes $\mathbf{y}$. At the k'th iteration stage the modified OMP algorithm forms the matched filter, identifies the index corresponding to the largest amplitude (step 3) and adds this to the support-set (step 4). It proceeds with solving a least squares problem with the selected indices (step 5), subtracts the least squares fit and produces a new residual (step 6). This process is updated until $K_{\max}$ components have been picked in the support-set. Normally the regular OMP also has a stopping criterion when the residual norm is non-decreasing. We have removed this stopping criterion to force the modified OMP such that it picks $K_{\max}$ elements.

In addition to the sparse signal estimate $\hat{\mathbf{x}}$, we also output the estimated support-set and the final residual norm. The residual norm is here given for completion and could be used as a stopping criterion in the joint algorithm, although we do not take this approach.

4 Joint Subspace Pursuit

For solving the joint CS problem (1), we develop the joint SP algorithm based on the mixed support-set model and a modified SP algorithm. The joint SP algorithm uses the functions (6), (7) and (8), defined in section 3.

4.1 Joint Subspace Pursuit

We here describe the new joint SP algorithm that uses the new mixed support-set model. For developing the joint SP algorithm, we modify the regular SP algorithm so that it can use an estimate of the common support-set as an initial support-set. The necessary modifications to the regular SP is presented in subsection 4.2.

We now show the steps of the joint SP in algorithm 5. In contrast to the joint OMP, the joint SP algorithm does not need any iterator because it uses the sum-residual-norm as stopping criteria. Another difference to the joint OMP is that the joint SP algorithm has to keep track of its old data (step 4). The maximum $K_{l,\max}$ (which is found in the initialization) is then fed into the modified SP algorithm (step 5). Similarly to the joint OMP, the N-sized zero vector $\mathbf{s}$ is used to find the common support-set $\mathcal{I}^{(c)}$ (step 6 and 9). The sum-residual-norm is formed in (step 8). The difference to the joint OMP is that, in each iteration, the joint SP picks the indices corresponding to the $K^{(c)}$ largest components of $\mathbf{s}$ (instead of the k largest, as in joint OMP). This common support-set is always of the same size $K^{(c)}$, but refined through iterations. We stop when the value of the sum-residual-norm no longer decreases.

Algorithm 5 : joint SP

```
Input: $\{\mathbf{A}_l\}_{l=1}^L$, $\{K_l^{(p)}\}_{l=1}^L$, $K^{(c)}$ ,$\{\mathbf{y}_l\}_{l=1}^L$
Initialization:
 1: $r_n \leftarrow \infty$; $\hat{\mathbf{x}}_l \leftarrow \mathbf{0}_{N\times 1}, \forall l \in \{1,2,\dots L\}$
 2: $\mathcal{I}^{(c)} \leftarrow \emptyset$; $\mathcal{I}_l \leftarrow \emptyset, \forall l \in \{1,2,\dots L\}$
 3: $K_{l,\max} = K^{(c)} + K_l^{(p)}, \forall l \in \{1,2,\dots L\}$
Iteration:
 1: repeat
 2:     $\mathbf{s} \leftarrow \mathbf{0}_{N\times 1}$
 3:     for $\forall l \in \{1,2,...,L\}$ do
 4:         $(r_n^{\text{old}}, \mathcal{I}_l^{\text{old}}, \hat{\mathbf{x}}_l^{\text{old}}) \leftarrow (r_n, \mathcal{I}_l, \hat{\mathbf{x}}_l)$
 5:         $(\mathcal{I}_l, \hat{\mathbf{x}}_l, n_l) \leftarrow \text{SP}(\mathbf{A}_l, K_{l,\max}, \mathbf{y}_l, \mathcal{I}^{(c)})$
 6:         $\mathbf{s} \leftarrow \text{add}_1(\mathbf{s}, \mathcal{I}_l)$
 7:     end for
 8:     $r_n \leftarrow \sum_{l=1}^L n_l$
 9:     $\mathcal{I}^{(c)} \leftarrow \texttt{max\_indices}(\mathbf{s}, K^{(c)})$
10: until $(r_n \geq r_n^{\text{old}})$
Output: $\{\hat{\mathbf{x}}_l^{\text{old}}\}_{l=1}^L$, $\{\mathcal{I}_l^{\text{old}}\}_{l=1}^L$
```

Since the joint SP algorithm stops on the criterion of sum-residual-norm it turns out to converge using less iterations, why it is also less computationally intensive than the joint OMP.

4.2 Modified SP

As mentioned in subsection 4.1, we here describe the modified SP in algorithm 6. The initialization phase has, compared to the regular SP, been modified in a similar way as the OMP algorithm so that it can use an initial support-set. This algorithm boils down to the regular SP as defined by Dai and Milenkovic [6] when $\mathcal{I}_{\text{ini}} = \emptyset$. At k'th iteration stage, the modified SP algorithm forms the matched filter $\mathbf{A}^T\mathbf{r}_{k-1}$, identifies the indices corresponding to the $K_{\max}$ largest amplitudes followed by joining with the old support-set (step 3 of Iteration). This support-set $\mathcal{I}'$ is likely to be bigger than $K_{\max}$. The algorithm solves a least squares problem with the selected indices of $\mathcal{I}'$ and identifies the new indices corresponding to the $K_{\max}$ largest amplitudes (step 4 and 5 of Iteration) followed by finding the residual (step 6). This process is repeated until the residual norm no longer decreases.

In addition with the sparse signal estimate $\hat{\mathbf{x}}$, we also output the estimated support-set $\hat{\mathcal{I}}$ and the final residual norm.

Algorithm 6 : SP (including modifications)

```
Input: $\mathbf{A}$, $K_{\max}$, $\mathbf{y}$, $\mathcal{I}_{\text{ini}}$
Initialization:
 1: $\mathcal{I}_0 \leftarrow \texttt{max\_indices}\left(\mathbf{A}^T\mathbf{y}, K_{\max}\right) \cup \mathcal{I}_{\text{ini}}$
 2: $\mathbf{r}_0 \leftarrow \texttt{resid}(\mathbf{y}, \mathbf{A}_{\mathcal{I}_0})$
 3: $k \leftarrow 0$
Iteration:
 1: repeat
 2:     $k \leftarrow k+1$
 3:     $\mathcal{I}' \leftarrow \mathcal{I}_{k-1} \cup \texttt{max\_indices}\left(\mathbf{A}^T\mathbf{r}_{k-1}, K_{\max}\right)$
 4:     $\hat{\mathbf{x}}$ such that $\hat{\mathbf{x}}_{\mathcal{I}'} = \mathbf{A}_{\mathcal{I}'}^{\dagger}\mathbf{y}$ and $\hat{\mathbf{x}}_{\bar{\mathcal{I}}'} = \mathbf{0}$
 5:     $\mathcal{I}_k \leftarrow \texttt{max\_indices}(\hat{\mathbf{x}}, K_{\max})$
 6:     $\mathbf{r}_k \leftarrow \texttt{resid}(\mathbf{y}, \mathbf{A}_{\mathcal{I}_k})$
 7: until $(\|\mathbf{r}_k\|_2 \geq \|\mathbf{r}_{k-1}\|_2)$
 8: $k \leftarrow k-1$                                   ('Previous iteration count')
Output:
 1: $\hat{\mathcal{I}} \leftarrow \mathcal{I}_k$
 2: $\hat{\mathbf{x}}$ such that $\hat{\mathbf{x}}_{\mathcal{I}_k} = \mathbf{A}_{\mathcal{I}_k}^{\dagger}\mathbf{y}$ and $\hat{\mathbf{x}}_{\bar{\mathcal{I}}_k} = \mathbf{0}$
 3: $n_r \leftarrow \|\mathbf{r}_k\|_2$
```

5 Simulation Results

In the simulations we are interested in finding how much performance can be gained by exploiting the mixed support-set model with the new algorithms (joint OMP and joint SP) over the regular algorithms (regular OMP and regular SP). We report the results for clean and noisy measurement cases. In the noisy case we have chosen signal-to-measurement-noise-ratio (SMNR) 20 dB, i.e., $10\log_{10}\frac{\mathbb{E}\{\|\mathbf{x}\|_2^2\}}{\mathbb{E}\{\|\mathbf{w}\|_2^2\}} = 20$. Note that we drop the subscript l because we are averaging over all sensors l.

To compare the algorithms, we use two performance measurements. The first performance measure is the signal-to-reconstruction-noise-ratio (SRNR) which is defined as

$$\text{SRNR} = 10\log\mathbb{E}\left\{\frac{\|\mathbf{x}\|_2^2}{\|\mathbf{x}-\hat{\mathbf{x}}\|_2^2}\right\}. \tag{9}$$

The second performance measure is the the distortion $d(\mathcal{I},\hat{\mathcal{I}}) = 1 - (|\mathcal{I}\cap\hat{\mathcal{I}}|/|\mathcal{I}|)$ which measures the performance of the support-set estimation [14]. The distortion of the support-set is a valuable performance measurement since the algorithms we compare endeavor to estimate the underlying support-set. Considering a large number of realizations (data vectors), we can compute the average of $d(\mathcal{I}, \hat{\mathcal{I}})$. We define the average support-cardinality error (ASCE) as follows

$$\text{ASCE} = \mathbb{E}\left\{ d(\mathcal{I}, \hat{\mathcal{I}}) \right\} = 1 - \mathbb{E}\left\{ \frac{|\mathcal{I} \cap \hat{\mathcal{I}}|}{|\mathcal{I}|} \right\}. \tag{10}$$

Next we describe the simulation setup. In any CS setup, all sparse signals are expected to be exactly reconstructed if the number of measurements are more than a certain threshold value. The computational complexity to test this uniform reconstruction ability is exponentially high. Instead, we can rely on empirical testing, where SRNR and ASCE is computed for random measurement matrix ensemble. We define the fraction of measurements

$$\alpha = \frac{M}{N}. \tag{11}$$

Using α, the testing is performed as follows:

1. Given the signal parameter N, choose an α (such that M is an integer).
2. Randomly generate a set of $M \times N$ sensing matrices $\{\mathbf{A}_l\}_{l=1}^{L}$ where the components are drawn independently from an i.i.d. Gaussian source (i.e. $a_{m,n} \sim \mathcal{N}\left(0, \frac{1}{M}\right)$) and then scale the columns of $\mathbf{A}_l$ to unit-norm.
3. Generate support-sets $\mathcal{I}^{(c)}$ and $\{\mathcal{I}_l^{(p)}\}_{l=1}^{L}$ of cardinality $K^{(c)}$ and $\{K_l^{(p)}\}_{l=1}^{L}$, respectively. The support-sets are uniformly chosen from $\{1, 2, ..., N\}$.
4. Randomly generate a set of signal vectors $\{\mathbf{x}_l\}_{l=1}^{L}$ following (2), where $\{\mathbf{z}_l^{(c)}\}_{l=1}^{L}$ and $\{\mathbf{z}_l^{(p)}\}_{l=1}^{L}$ corresponding to the non-zero components (support-sets determined in step 3). The non-zero components in the vectors are chosen independently from a Gaussian source.
5. Compute the measurements $\mathbf{y}_l = \mathbf{A}_l \mathbf{x}_l + \mathbf{w}_l, \forall l \in \{1, 2, ..., L\}$. Here $\mathbf{w}_l \sim \mathcal{N}(\mathbf{0}, \sigma_l^2 \mathbf{I}_M)$.
6. Apply the CS algorithms on the data $\{\mathbf{y}_l\}_{l=1}^{L}$.

In the simulation procedure above, for each $l \in \{1, 2, \ldots, L\}$, Q sets of sensing matrices are created. For each data set and each sensing matrix, P sets of data vectors are created. In total, we will average over $L \cdot Q \cdot P$ data to evaluate the performance.

5.1 Parameters and simulation set-up

For the plots presented in this paper, we have chosen: $N = 500$, $K^{(c)} = 10$, and $\forall l, K_l^{(p)} = K^{(p)} = 10$. We have chosen $L = 10$ for which we have chosen number of $\mathbf{A}_l$'s to 50 (i.e. $Q = 50$) and the number of data-sets $\mathbf{x}$ to 50 (i.e. $P = 50$), giving a total number of $10 \cdot 50 \cdot 50 = 25000$ data for statistics.

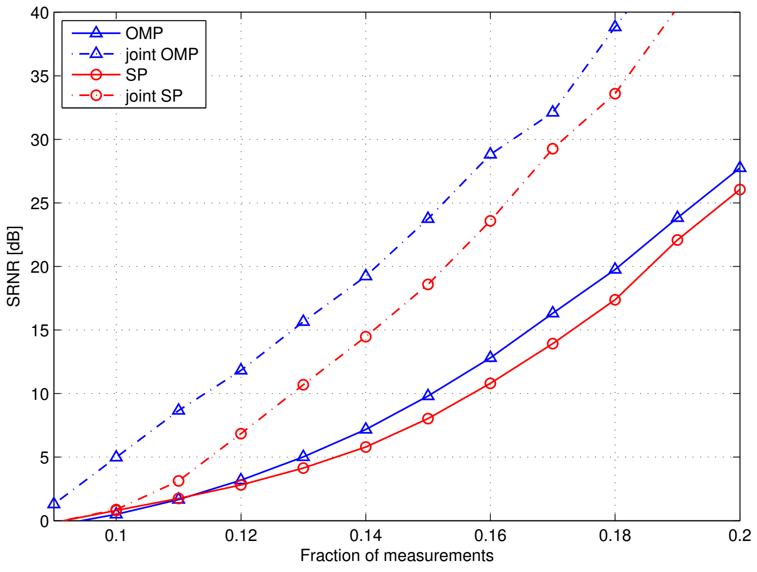


Figure 1: SRNR for clean measurements with $K^{(c)} = 10$, $K^{(p)} = 10$.

5.2 Analysis of the simulation results

We provide four figures showing the results of the numerical simulations.

Figure 1 and Figure 2 shows the SRNR of the reconstructed signal for a clean and a noisy (SMNR = 20 dB) measurement case, respectively. In the clean case, we notice that the joint OMP performs almost 15 dB better than the regular OMP at $\alpha = 0.15$. In the noisy case this number is slightly lower with about 10 dB improved performance (still $\alpha = 0.15$). Corresponding numbers for the SP-based algorithm is about 10 dB improvement for the clean measurements and 8 dB improvement for the noisy case, both at $\alpha = 0.15$.

In Figure 3 and Figure 4 we similarly show the ASCE of the recon-

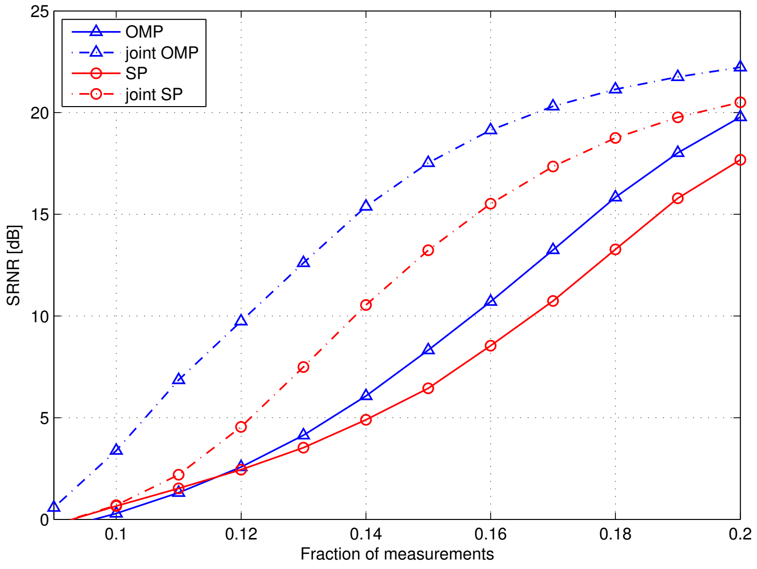


Figure 2: SRNR for noisy measurements with SMNR = 20 dB, $K^{(c)} = 10$, $K^{(p)} = 10$.

structed signal for a clean and a noisy (SMNR = 20 dB) case. In the noisy case, we notice that none of the algorithms can perfectly find the support-set, as expected. We also notice that the joint OMP very quickly converges to its best performance. Also, the joint OMP is significantly better than all the other algorithms at low α's.

It is interesting to notice that the performance gain of joint OMP over regular OMP is higher than the gain of joint SP over regular SP. The reason for this is understood by how the algorithm picks the common support-set. The joint OMP picks one common support-set component at each iteration and hence iterates 10 times. In each iteration the regular OMP is called once for each sensor. The joint SP on the other hand, starts with an estimate of the full common support-set and iteratively refines it. Thus, the joint OMP is more careful when choosing the support-set and progresses at a slower pace compared to the joint SP. It was observed during the experiments that because of this difference in implementation, the joint OMP has a significantly higher computational complexity.

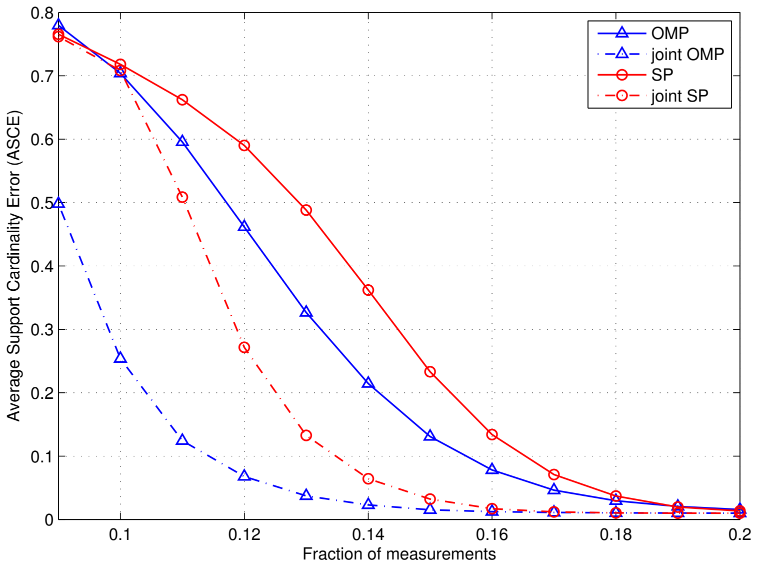


Figure 3: ASCE for clean measurements with $K^{(c)} = 10$, $K^{(p)} = 10$.

6 Conclusion

In this paper, a new mixed support-set model and two new greedy pursuit algorithms was proposed. We find that the model is a generalization of existing signal models presented in the literature earlier. To solve the joint CS problem based on this model, we developed two new algorithms, joint OMP and joint SP, based on regular OMP and SP, respectively. By experimental evaluations, we conclude that greedy pursuit algorithms can exploit joint sparsity information embedded in data.

References

[1] D.L. Donoho, "Compressed sensing," *IEEE Trans. Inf. Theory*, vol. 52, no. 4, pp. 1289 –1306, April 2006.

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

[3] S.G. Mallat and Zhifeng Zhang, "Matching pursuits with time-frequency dictionaries," in *IEEE Trans. Signal Processing*, Dec. 1993, vol. 41, pp. 3397 –3415.

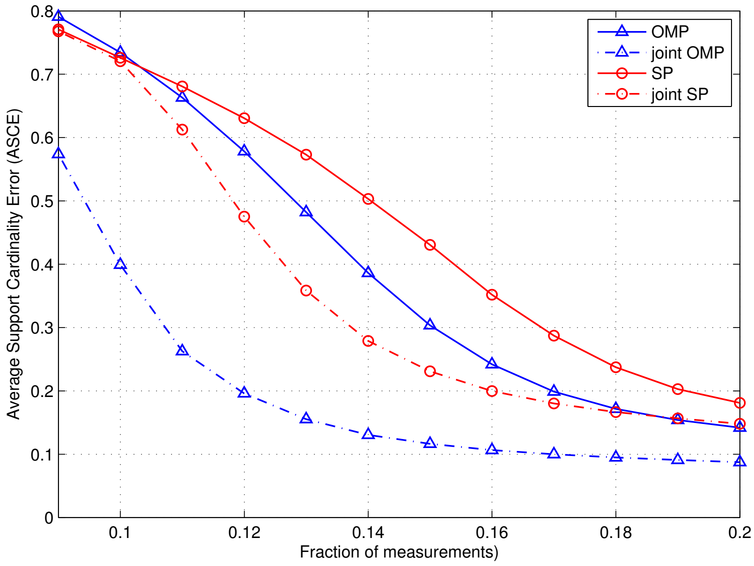


Figure 4: ASCE for noisy measurements with SMNR = 20 dB, $K^{(c)} = 10$, $K^{(p)} = 10$.

[4] J.A. Tropp and A.C. Gilbert, "Signal recovery from random measurements via orthogonal matching pursuit," *IEEE Trans. Inf. Theory*, vol. 53, no. 12, pp. 4655 –4666, Dec. 2007.

[5] S. Chatterjee, D. Sundman, and M. Skoglund, "Look ahead orthogonal matching pursuit," in *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing (ICASSP)*, May 2011.

[6] Wei Dai and O. Milenkovic, "Subspace pursuit for compressive sensing signal reconstruction," *IEEE Trans. Inf. Theory*, vol. 55, no. 5, pp. 2230 –2249, May 2009.

[7] M.F. Duarte, S. Sarvotham, D. Baron, M.B. Wakin, and R.G. Baraniuk, "Distributed compressed sensing of jointly sparse signals," *Proc. Asilomar Conf. Signals, Sys., and Comp.*, pp. 1537 – 1541, Oct. 2005.

[8] S.F. Cotter, B.D. Rao, Kjersti Engan, and K. Kreutz-Delgado, "Sparse solutions to linear inverse problems with multiple measurement vectors," *IEEE Trans. Signal Processing*, vol. 53, no. 7, pp. 2477 – 2488, Jul. 2005.

[9] R. Gribonval, H Rauhut, K. Schnass, and P Vandergheynst, "Atoms of all channels, unite! Average case analysis of multi-channel sparse recovery using greedy algorithms," *J. Fourier Anal. and Appl.*, vol. 14, no. 5, pp. 655–687, 2008.

[10] I.F. Gorodnitsky and B.D. Rao, "Sparse signal reconstruction from lim-

ited data using focuss: a re-weighted minimum norm algorithm," *IEEE Trans. Signal Processing*, vol. 45, no. 3, pp. 600 –616, Mar. 1997.

[11] J.W. Phillips, R.M. Leahy, and J.C. Mosher, "Meg-based imaging of focal neuronal current sources," *IEEE Trans. Med. Imaging*, vol. 16, no. 3, pp. 338 –348, June 1997.

[12] D. Sundman, S. Chatterjee, and M. Skoglund, "On the use of compressive sampling for wide-band spectrum sensing," in *Proc. IEEE Int. Symp. Signal Processing and Inf. Tech. (ISSPIT)*, Dec. 2010, pp. 354 –359.

[13] A.Y. Yang, M. Gastpar, R. Bajcsy, and S.S. Sastry, "Distributed sensor perception via sparse representation," *Proc. of the IEEE*, vol. 98, no. 6, pp. 1077 –1088, June 2010.

[14] G. Reeves and M. Gastpar, "A note on optimal support recovery in compressed sensing," *Proc. Asilomar Conf. Signals, Sys., and Comp.*, pp. 1576 –1580, nov. 2009.

Paper E

Greedy Pursuits for Distributed Compressed Sensing

Dennis Sundman, Saikat Chatterjee, and Mikael Skoglund

Submitted to
IEEE Transactions on Signal Processing

©2012 IEEE
The layout has been revised

Greedy Pursuits for Distributed Compressed Sensing

Dennis Sundman, Saikat Chatterjee, and Mikael Skoglund

1 Introduction

Compressed sensing (CS) [1, 2] refers to an under-sampling problem, where few samples of an inherently sparse signal are collected via a linear measurement matrix with the objective of reconstructing the full sparse signal from these few samples. Considering the fact that sparsity is ubiquitous in nature, CS has many potential applications. In the literature, the task of developing CS reconstruction algorithms has presumably been considered for a set-up where the samples are acquired by using a single sensor. While substantial endeavor has been carried out for such a single-sensor setup, not much effort has been carried out to develop algorithms for a multiple-sensors setup.

For a multiple-sensors setup, an interesting case is a distributed setup where several CS-based sensors are connected through a decentralized (distributed) network. Such a setup is useful in a wide range of applications, for example in sensor networks [3] and distributed spectrum estimation [4–6]. Considering a camera sensor network, we can envisage a scheme where a set of measurement samples (CS samples of image signals) from different angles at different positions are acquired. Instead of reconstructing the underlying signals from the corresponding samples independently, one could potentially improve the quality of the reconstructed signals by taking into account all the measurement samples. This is possible by exchanging information over the distributed, but connected network. We refer to this problem as distributed CS, where the connection between the sensors follows an arbitrary topology. Note that, a distributed CS setup becomes a centralized CS setup (or a joint CS setup) if the network is fully connected (i.e., if a sensor is connected with all other sensors) or the sensors transmit their measured samples to a common centralized point. For such a fully connected setup, we have recently developed joint greedy pursuit (GP) reconstruction algorithms in [7]. In the literature, we find that a few more attempts have been made for centralized solutions with various model assumptions [8,9]. Additionally the works based on simultaneous sparse approximation (SSA) [10,11] and multiple measurement vector (MMV) [12, 13] problems, for example simultaneous orthogonal matching pursuit (SOMP) algorithm [14], can be considered to be applied for a centralized (or joint) CS setup.

For the distributed CS setup, we notice some recent attempts to design convex relaxation algorithms [4–6, 15]. While the convex relaxation algorithms are theoretically elegant and provide good practical performance for low dimensional problems, their use for high dimensional problems are limited due to their high complexity (here, a high dimensional problem refers to the case where the dimensions of underlying signals are high). Naturally, designing computationally simple greedy pursuit (GP) algorithms is an attractive alternative. In general, a GP algorithm uses computationally simple detection and estimation techniques iteratively and hence they are computationally efficient for higher dimensional problems. While there exists several GP algorithms for the centralized setup, such as [7, 13, 14, 16], to the best of the authors' knowledge, there exists no solution for efficiently solving the distributed CS problem based on GP algorithms.

In this paper, we develop GP algorithms for a distributed CS setup. We refer to the new algorithms as distributed GP (DiGP). For the distributed CS setup, we first introduce a signal model [7] that can describe the correlations between underlying sparse signals. This new signal model is shown to be a generalization of several recently proposed signal models [9, 12, 17]. Based on this signal model, we develop three DiGP algorithms. Two of the DiGP algorithms are built upon existing GP algorithms by introducing appropriate modifications. The existing GP algorithms which we modify are orthogonal matching pursuit (OMP) [18] and subspace pursuit (SP) [19]. Our motivation for using these two GP algorithms is that they are good representatives from two main classes of existing GP algorithms. In the process of using these two GP algorithms, we realize that there is a scope of developing a new GP algorithm which has high potential for the distributed CS setup. Hence, we develop a new GP algorithm followed by constructing the third DiGP algorithm. Through simulations, we evaluate the three new DiGP algorithms and show that the algorithms provide increasingly better performance as the network connection improves. For a modestly connected network, the simulation results show that the performance is close to the fully connected (centralized) setup and much better than the completely disconnected (independent) setup.

The remainder of the paper is arranged as follows: In the next section, we describe the distributed CS setup and introduce the new signal model; we also develop a structured approach for describing the quality of connectivity in a distributed network. In Section 3, we introduce the concept of DiGP by first studying classifications of different GP algorithms and using this study we then develop two DiGP algorithms based on OMP and SP. In Section 4, a new GP algorithm is constructed with the aim of providing a DiGP algorithm with desirable properties. We end the paper with experimental evaluations in Section 5

Notations: Let a matrix be denoted as $\mathbf{A} \in \mathbb{R}^{M \times N}$ and a vector as $\mathbf{x} \in \mathbb{R}^{M \times N}$. $\mathcal{T}$ is the support-set of $\mathbf{x}$, which is defined in the next section. $\mathbf{A}_{\mathcal{T}}$ is the sub matrix consisting of the columns in $\mathbf{A}$ corresponding to the elements in the set $\mathcal{T}$. Similarly $\mathbf{x}_T$ is a vector formed by the components of $\mathbf{x}$ that are indexed by $\mathcal{T}$. We let $(.)^{\dagger}$ and $(.)^{T}$ denote pseudo-inverse and transpose of a matrix, respectively. We also use $\|.\|$ to denote the l_2 norm of a vector.

2 Distributed Compressed Sensing

Using a general multiple sensor system setup [9], we first describe the distributed CS problem and then introduce the new signal model. We have recently proposed this signal model in [7] and referred to it as the mixed support-set signal model. In the end of this section we also mention network topology and provide some algorithmic notations.

For the distributed CS problem, observing the l'th sensor, we have the sparse signal $\mathbf{x}_l \in \mathbb{R}^N$ measured as

$$\mathbf{y}_l = \mathbf{A}_l \mathbf{x}_l + \mathbf{w}_l, \qquad \forall l \in \{1, 2, ..., L\}, \tag{1}$$

where $\mathbf{y}_l \in \mathbb{R}^M$ is a measurement vector, $\mathbf{A}_l \in \mathbb{R}^{M \times N}$ is a measurement matrix, $\mathbf{w}_l \in \mathbb{R}^M$ is the measurement error. In this setup $M < N$ and hence the system is under-determined. $\mathbf{A}_l$ and $\mathbf{w}_l$ are independent across l. The signal vector $\mathbf{x}_l = [x_l(1)\ x_l(2), \ldots]$ has K_l non-zero components with a set of indices $\mathcal{T}_l = \{i : x_l(i) \neq 0\}$. $\mathcal{T}_l$ is referred to as the support-set of $\mathbf{x}_l$ with cardinality $|\mathcal{T}_l| = K_l$.

The distributed CS reconstruction problem endeavors for reconstructing $\mathbf{x}_l$ for all l by exploiting some shared structure (correlation) defined by the underlying signal model and by exchanging some information over the given network topology.

2.1 Mixed support-set model

Now, we describe the mixed support-set signal model with a shared structure where the signal vector $\mathbf{x}_l$ consists of two parts

$$\mathbf{x}_l = \mathbf{z}_l^{(c)} + \mathbf{z}_l^{(p)}, \qquad \forall l \in \{1, 2, ..., L\}. \tag{2}$$

In (2) both $\mathbf{z}_l^{(c)}$ and $\mathbf{z}_l^{(p)}$ have independent non-zero components. There are $K_l^{(p)}$ non-zero values associated with $\mathbf{z}_l^{(p)}$. The support-set of $\mathbf{z}_l^{(p)}$ is denoted by $\mathcal{T}_l^{(p)}$. For simplicity we assume that, $\forall l \in \{1, 2, \dots, L\}$, the components of $\mathcal{T}_l^{(p)}$ are drawn uniformly from the set $\{1, 2, \dots, N\}$. For $\mathbf{z}_l^{(c)}$ there are similarly $K^{(c)}$ non-zero components with the constraint that the associated support-set $\mathcal{T}_l^{(c)}$ is shared as $\mathcal{T}_l^{(c)} = \mathcal{T}^{(c)}$, $\forall l \in \{1, 2, \dots, L\}$. While the support-set $\mathcal{T}^{(c)}$ is the same (common) to all signals, it is naturally still unknown to the re-constructor[1]. Here we would like to emphasize that although $\mathcal{T}^{(c)}$ is the same for all sensors, the corresponding non-zero values of $\mathbf{z}_l^{(c)}$ are still individual and independent among the nodes. For the l'th sensor-node, this gives a support-set $\mathcal{T}_l$ for the signal $\mathbf{x}_l$ as

$$\mathcal{T}_l = \mathcal{T}^{(c)} \cup \mathcal{T}_l^{(p)}, \qquad \forall l \in \{1, 2, ..., L\}. \tag{3}$$

We define $K_{l,\max} = |\mathcal{T}^{(c)}| + |\mathcal{T}_l^{(p)}| = K^{(c)} + K_l^{(p)}$. Note that the support-sets can intersect, so $K_{l,\max} \geq K_l$.

Let us compare our new mixed support-set model with the signal models proposed recently in the literature. We first consider the mixed signal model of [9], where $\mathbf{x}_l$ is composed of common and individual parts

$$\mathbf{x}_l = \mathbf{z}^{(c)} + \mathbf{z}_l^{(p)}, \qquad \forall l \in \{1, 2, ..., L\}. \tag{4}$$

Here $\mathbf{z}^{(c)}$ represents a common sparse signal part and $\mathbf{z}_l^{(p)}$ represents the individual (private) signal part for the l'th sensor. Note that $\mathbf{z}^{(c)}$ is fixed for all the data set. Comparing (2) and (4) we can say that the new mixed support-set model is a generalization over the mixed signal model in the sense that the former allows for individual non-zero values in the common signal part $\mathbf{z}_l^{(c)}$.

The common support-set model [12,17] used in magnetoencephalography has no individual signal parts at all. Their model is

$$\mathbf{x}_l = \mathbf{z}_l^{(c)}, \qquad \forall l \in \{1, 2, ..., L\}. \tag{5}$$

[1]For easy practical implementation, we assume that the support-set components are uniformly distributed over $\mathcal{T}^{(c)}$ and $\mathcal{T}_l^{(p)}$.

Therefore our model (2) also generalizes the model (5).

A natural question is why we use the mixed support-set model (2) for developing DiGP algorithms. While we note that the signal model is least stringent in the sense of describing a correlation structure, we also find that the signal model allows us to develop a distributed framework by exchanging limited information. In this distributed framework, we consider the estimated support-set at each CS node as the information to exchange with neighboring nodes.

2.2 Network Topology

In a distributed setup, we assume that the CS nodes are connected via a network where there is at least one connection between any two nodes; otherwise the setup is equivalent to two, or more, independent networks. An example of a simple network can be illustrated by a circular topology where each node (or each sensor) is only connected with another node through a one-way connection (see Figure 1a). We will refer to this as the worst-case network of degree 1 and denote it by a connection matrix called $\mathbf{C}_1$. By forming this circular topology of nodes and adding new connections from each node to the others in a systematic way, we can study how the overall performance of a DiGP algorithm improves as the network connectivity increases. In Figure 1b, we show a network where each node is connected to two other nodes (referred to as a degree 2 network) and we denote this network by the connection matrix $\mathbf{C}_2$. In the experimental evaluation (Section 5), we will study the performance for all intermediate networks, $\mathbf{C}_0, \mathbf{C}_1, ..., \mathbf{C}_{L-1}$ (recall that L is the total number of nodes in the network), where $\mathbf{C}_0$ denotes the use of standard GP algorithms in a disconnected setup. For the remainder of the paper, if nothing else is stated, $\mathbf{C}_2$ is assumed as the default case for all the DiGP algorithms. We will also refer to the solution of a fully connected network ($\mathbf{C}_{L-1}$) as the joint solution which is equivalent to a centralized solution.

2.3 Algorithmic notation

For clarity in the algorithmic notation, we define three functions as follows:

$$\texttt{resid}(\mathbf{y}, \mathbf{B}) \triangleq \mathbf{y} - \mathbf{B}\mathbf{B}^{\dagger}\mathbf{y}, \tag{6}$$

where $\mathbf{y}$ is a vector and $\mathbf{B}$ is a full column-rank matrix;

$$\begin{aligned}\texttt{max\_indices}(\mathbf{x}, k) \triangleq \{&\textit{the set of indices corresponding} \\ &\textit{to the } k \textit{ largest amplitude components of } \mathbf{x}\},\end{aligned} \tag{7}$$

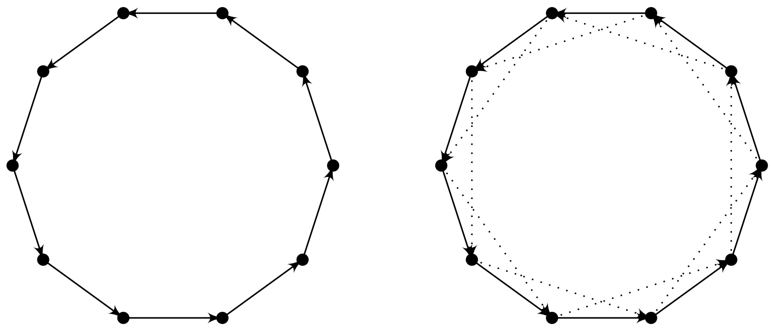

(a) Network of degree 1 (b) Network of degree 2

Figure 1: Network topologies of degree 1 ($\mathbf{C}_1$) and degree 2 ($\mathbf{C}_2$).

and

$$\text{add}_1(\mathbf{s}, \mathcal{T}) \triangleq \{\forall j \in \mathcal{T},\ \textit{perform}\ s_j = s_j + 1\}, \tag{8}$$

where $\mathbf{s} = [s_1\ s_2\ \ldots s_N]$ and $s_j \geq 0$.

For the l'th CS node, $\mathcal{L}_l$ denotes the set of indices corresponding to the neighboring connected nodes (we always consider that the l'th CS node is connected with itself and hence $\mathcal{L}_l$ has at least an element that corresponds to the l'th node itself).

3 Distributed Greedy Pursuits

In this section, we develop two different DiGP algorithms based on two existing GP algorithms. Furthermore, in Section 4 we develop a new GP algorithm on which we construct the third DiGP algorithm. The three DiGP algorithms that are developed in this paper are referred to as follows:

1. Distributed OMP (DiOMP): Where we use existing OMP as the GP algorithm after appropriate modification.

2. Distributed SP (DiSP): When we use existing SP as the GP algorithm after appropriate modification.

3. Distributed FROMP (DiFROMP): When we use new forward-reverse OMP (FROMP) as the GP algorithm. The FROMP algorithm will be developed in Section 4.1.

For these three DiGP algorithms, we find that it is possible to develop distributed algorithmic structures in two different ways. The DiOMP follows the first distributed algorithmic structure, and the DiSP and DiFROMP follows the second distributed algorithmic structure. These two distributed algorithmic structures are developed in Section 3.3 and 3.4 where they are developed as two examples of the DiGP algorithms DiOMP and DiSP respectively. However, for developing the DiGP algorithms, we first need to know preliminaries about underlying GP algorithms. This helps to bring appropriate modifications to the GP algorithms or construct new GP algorithms, so that they are better suited for the development of DiGP algorithms. The preliminaries of GP algorithms are discussed in the following section.

3.1 Preliminaries of GP Algorithms

In general, for CS reconstruction, existing GP algorithms are used with an implicit assumption of a single-sensor setup. Using the measurement vector collected from the sensor, the main principle of the GP algorithms is to estimate the underlying support-set of a sparse vector followed by evaluating the associated signal values. The support-set is the set of indices corresponding to the non-zero elements of a sparse vector. To estimate the support-set and the associated signal values, the GP algorithms use linear algebraic tools, for example the matched filter detection and least-squares estimation. A crucial point worth mentioning is that the success of the GP algorithms mainly depends on their efficiency in estimating the support-set. Once a support-set is formed, the associated signal values can be obtained by a simple least-squares estimation.

In the literature, we note two main algorithmic approaches for the GP algorithms: (1) the categorization of *serial* or *parallel*, and (2) *reversible* or *irreversible* construction mechanism. First let us consider the algorithmic approach of serial or parallel construction. If serial construction is performed then elements of the support-set are chosen one-by-one; in contrast, for parallel construction, several elements of the support-set are chosen simultaneously. Next we consider the algorithmic approach of reversible and irreversible construction. If irreversible construction is performed then an element already added to the support-set, remains there forever; in contrast, for reversible construction, an element of the support-set (chosen in the past) can be removed later (if the element is found not suitable later). Therefore, considering serial or parallel construction, a GP algorithm can be categorized either as a serial pursuit (S-pursuit) or a parallel pursuit (P-pursuit) algorithm. On the other hand, considering reversible or irreversible, a GP algorithm can either use a reversible support-set (R-support) or an irreversible support-set

Table 1: Classification of GP algorithms

	P-pursuit	S-pursuit
R-support	SP, CoSaMP, LAPP, BAOMP	FROMP
I-support	StOMP, ROMP	OMP, LAOMP, POMP

(I-support) construction mechanism.

We categorize several GP algorithms in Table 1 where we consider existing OMP [18], SP [19], CoSaMP [20], look ahead OMP (LAOMP) [21], stagewise omp (StOMP) [22], backtracking OMP (BAOMP) [23], projection-based OMP (POMP) [21], look ahead parallel pursuit (LAPP) [24], regularized OMP (ROMP) [25], and the new forward-reverse OMP (FROMP). For developing DiGP algorithms, among the existing GP algorithms we use the OMP and SP because they are generic and easy to implement. Furthermore, we note in Table 1 that there exist no GP algorithm which can be categorized as S-pursuit algorithm with the characteristics of R-support construction mechanism. Hence, we develop the new FROMP algorithm to fill the gap. The development of the FROMP algorithm and its use in constructing a DiGP algorithm are reported in Section 4. Now, for developing DiGP algorithms based on the signal model (mixed support-set model (2)) and the algorithmic architectures of GP algorithms, we find the principle strategies discussed in the following section.

3.2 Principle Strategies for DiGP Algorithms

The new iterative DiGP algorithms are developed based on two principle strategies which are invoked in each iteration of the algorithms. The two principle strategies are described below:

1. Each CS node transmits its own full support-set estimate to the neighboring connected nodes. It also receives a set of full support-set estimates from the neighboring connected nodes.

2. Using the set of all received support-set estimates and by invoking a voting mechanism, each CS node finds an estimate of the common support-set, either serially or parallelly. Then, using the common support-set estimate as the initial knowledge, each CS node finds a new estimate of the full support-set and then again exchange the full support-set information.

Using the two principles, the DiGP algorithms continue to execute until convergence. Now, considering OMP and SP as the underlying GP algorithms, we develop two DiGP algorithms in the next sections. Later, in Section 4, we develop the third DiGP algorithm based on FROMP.

3.3 Distributed OMP

For developing the distributed OMP (DiOMP) algorithm, we first modify the existing OMP algorithm and then use the modified OMP algorithm as a building block. The modified OMP is referred to as modOMP where the modification is required so that it can use an initial support-set estimate in its task of finding the full support-set information. The modOMP is presented in Appendix A.1.

Now, for developing DiOMP, we consider the l^*-th node and develop a voting method for finding an estimate of the common support-set in Algorithm 7. In Algorithm 7, we supply the inputs: $\mathcal{L}_{l^*}$, $\{\hat{\mathcal{T}}_l\}$ where

Algorithm 7 : Voting based choice of indices
Executed in l^*-th node, where $\mathcal{L}_{l^*}$ is the set of neighboring nodes (Note that $l^* \in \mathcal{L}_{l^*}$)

Input: $\mathcal{L}_{l^*}$, $\{\hat{\mathcal{T}}_l\}$ where $l \in \mathcal{L}_{l^*}$, and the desired cardinality q of common support set

```
1: $\mathbf{s}_{l^*} \leftarrow \mathbf{0}_{N\times 1}$
2: for each $l \in \mathcal{L}_{l^*}$ do
3:     $\mathbf{s}_{l^*} \leftarrow \text{add}_1(\mathbf{s}_{l^*}, \hat{\mathcal{T}}_l)$
4: end for
5: $\hat{\mathcal{T}}_{l^*}^{(c)} \leftarrow \texttt{max\_indices}(\mathbf{s}_{l^*}, q)$          (Note: $|\hat{\mathcal{T}}_{l^*}^{(c)}| = q$)
```

Output: $\hat{\mathcal{T}}_{l^*}^{(c)}$

$l \in \mathcal{L}_{l^*}$, q. Here, $\mathcal{L}_{l^*}$ denotes the connected neighboring nodes, $\{\hat{\mathcal{T}}_l\}$ is the estimated support-sets in all the connected nodes, and q is the desired cardinality of the common support-set. The output of Algorithm 7 is the common support-set estimate $\hat{\mathcal{T}}_{l^*}^{(c)}$; here we use the subscript l^* to denote the case for l^*-th node. Then, using the $\text{add}_1(.,.)$ and $\texttt{max\_indices}(.,.)$ functions, the voting algorithm finds the common support-set. In this case, we rely on the fact that the elements of the common support-set have the highest scores in terms of their occurrences. Hence, the method can be viewed as a democratic voting strategy. Using Algorithm 7, we define the following function.

Function 2 *(Voting based choice of indices) For the l^*-th node, let the set of connected neighbors $\mathcal{L}_{l^*}$, the estimated support-sets $\{\hat{\mathcal{T}}_l\}$ in the connected nodes such that $l \in \mathcal{L}_{l^*}$, and the desired cardinality q of the* *common support-set be given. Then, the estimated common support-set $\hat{\mathcal{T}}_{l^*}^{(c)}$ is the output of the following algorithmic function*

$$\hat{\mathcal{T}}_{l^*}^{(c)} \leftarrow \texttt{vote}(\mathcal{L}_{l^*}, \{\hat{\mathcal{T}}_l\}, q).$$

where the above function executes Algorithm 7.

Now we develop the DiOMP algorithm based on modOMP and the voting function. DiOMP is shown in Algorithm 8 and it is executed locally and distributively in each node of the connected network. Let us consider DiOMP for the l^*-th node. The *input* to DiOMP (Algorithm 8)

Algorithm 8 : Distributed OMP (DiOMP)
Executed in l^*-th node, where $\mathcal{L}_{l^*}$ is the set of neighboring nodes (Note that $l^* \in \mathcal{L}_{l^*}$)

```
Input: $\mathbf{A}_{l^*}$, $\mathbf{y}_{l^*}$, $K_{l^*}^{(p)}$, $K^{(c)}$
Initialization:
 1: $K_{l^*,\max} = K_{l^*}^{(p)} + K^{(c)}$
 2: $(\hat{\mathcal{T}}_{l^*}, \hat{\mathbf{x}}_{l^*}, \eta_{l^*}) \leftarrow$ modOMP$(\mathbf{A}_{l^*}, K_{l^*,\max}, \mathbf{y}_{l^*}, \emptyset)$
 3: $\hat{\mathcal{T}}_l \leftarrow \emptyset$, $\forall\, l \in \mathcal{L}_{l^*} \setminus l^*$          (i.e. except $l^*$)
 4: $k \leftarrow 0$          (iteration counter)
Iteration:
 1: repeat
 2:     $k \leftarrow k+1$
 3:     { Transmit: $\hat{\mathcal{T}}_{l^*}$ to all nodes $l \in \mathcal{L}_{l^*}$ }
 4:     { Receive: $\hat{\mathcal{T}}_l$ from all nodes $l \in \mathcal{L}_{l^*}$ }
 5:     $\hat{\mathcal{T}}_{l^*}^{(c)} \leftarrow \texttt{vote}(\mathcal{L}_{l^*}, \{\hat{\mathcal{T}}_l\}, k)$          (Note: $|\hat{\mathcal{T}}_{l^*}^{(c)}| = k$)
 6:     $(\hat{\mathcal{T}}_{l^*}, \hat{\mathbf{x}}_{l^*}, \eta_{l^*}) \leftarrow$ modOMP$(\mathbf{A}_{l^*}, K_{l^*,\max}, \mathbf{y}_{l^*}, \hat{\mathcal{T}}_{l^*}^{(c)})$
 7: until $k = K^{(c)}$
Output: $\hat{\mathbf{x}}_{l^*}$, $\hat{\mathcal{T}}_{l^*}$
```

is the l^*-th node's sensing matrix $\mathbf{A}_{l^*}$, the measurement vector $\mathbf{y}_{l^*}$, and the cardinality of the private and common support-sets $K_{l*}^{(p)}$ and $K^{(c)}$. For the *initialization* phase, before any communication has taken place, modOMP$(\cdot)$ is executed to achieve a first estimate of the l^*-th node's support-set. At this phase, all other neighboring support-set estimates (where the other neighboring nodes are identified by $\mathcal{L}_{l^*} \setminus l^*$) are initialized as empty sets $\emptyset$ and an iteration parameter k is initialized to zero. The *iteration* phase of DiOMP is characterized by three main functionalities: (1) In steps 3 and 4, the communication phase takes place, where the support-set estimates are exchanged among the nodes. Note that the l^*-th node transmits its estimated support-set $\hat{\mathcal{T}}_{l^*}$ to the all neighboring nodes indexed by $\mathcal{L}_{l^*}$ and also receives the support-set estimates $\{\hat{\mathcal{T}}_l\}$ from the neighboring nodes[2]. (2) In step 5, by using all the full support-sets estimates $\{\hat{\mathcal{T}}_l\}$ the voting strategy is invoked to achieve an estimate of the common support-set. Note that the intermediate common support-set is estimated in each iteration and its cardinality is increased one-by-one through iterations (serially). (3) Using the estimated common support-set, modOMP$(\cdot)$ is executed locally to achieve a new full support-set estimate together with a signal estimate for the l^*-th node. These three functionalities are iteratively executed until the common support-set cardinality becomes $K^{(c)}$. Therefore, the DiOMP algorithm iterates exactly $K^{(c)}$ times.

In DiOMP, it is worth mentioning the importance of the serial construction mechanism strategy for the common support-set estimation. For compressible sparse signal vectors, where the sorted amplitudes of the signal vectors quickly decays (for example, if the non-zero components of a sparse signal vector is drawn from an i.i.d. Gaussian source), it is known that the serial construction is more efficient [21]. Hence, to estimate the common support-set reliably, we use the serial construction. However, the serial construction requires more computation in practice and we endeavor for developing a parallel construction mechanism with less complexity.

3.4 Distributed SP

We now develop the second DiGP algorithm using a parallel support-set construction mechanism. The new DiGP is based on SP and hence referred to as distributed SP (DiSP). Similarly to DiOMP, we first modify the SP algorithm (we refer to the modified SP as modSP which is explained in Appendix A.2) and then use it for developing DiSP.

DiSP is shown in Algorithm 9, where we use the voting function of Algorithm 7 and modSP of Algorithm 14. The principle strategy in Algorithm 9 is the same as that of DiOMP; the strategy is to improve the common support-set estimation by exchanging full support-set estimates over iterations. In each iteration of DiSP, the common support-set estimate $\hat{\mathcal{T}}_{l^*}^{(c)}$ is passed to the modSP algorithm which in turn finds the full support-set estimate $\hat{\mathcal{T}}_{l^*}$. Using the voting mechanism, we here find the $\hat{\mathcal{T}}_{l^*}^{(c)}$ with full cardinality ($|\hat{\mathcal{T}}_{l^*}^{(c)}| = K^{(c)}$) in each iteration. This kind of parallel common support-set construction may allow for a faster conver-

[2]For ease of explanation, $\mathcal{L}_{l^*}$ is the same for both transmitting and receiving, but in reality and in the simulations they may be different. In the simulations, the connections are rather described by the connection matrix $\mathbf{C}$.

gence than the serial common support-set construction used in DiOMP (which is fixed). Here, we mention that the parallel common support-set construction for DiSP is realizable with high reliability because we use modSP, which has an reversible construction mechanism (i.e., modSP may remove bad elements of support-set in a later iteration). We now take a closer look on DiSP in Algorithm 9.

Algorithm 9 : Distributed SP (DiSP)
Executed in the l^*-th node, where $\mathcal{L}_{l^*}$ is the set of neighboring nodes (Note that $l^* \in \mathcal{L}_{l^*}$)

```
Input: $\mathbf{A}_{l^*}$, $\mathbf{y}_{l^*}$, $K_{l^*}^{(p)}$, $K^{(c)}$
Initialization:
 1: $K_{l^*,\max} = K_{l^*}^{(p)} + K^{(c)}$
 2: $(\hat{\mathcal{T}}_{l^*}, \hat{\mathbf{x}}_{l^*}, \eta_{l^*}) \leftarrow \text{modSP}(\mathbf{A}_{l^*}, K_{l^*,\max}, \mathbf{y}_{l^*}, \emptyset)$
 3: $\eta_{l^*}^{\text{old}} \leftarrow \eta_{l^*}$
 4: $\hat{\mathcal{T}}_l \leftarrow \emptyset,\ \forall\ l \in \mathcal{L}_{l^*} \setminus l^*$                    (i.e. except $l^*$)
Iteration:
 1: repeat
 2:    if $\eta_{l^*} > \eta_{l^*}^{\text{old}}$ then
 3:       $(\hat{\mathcal{T}}_{l^*}, \hat{\mathbf{x}}_{l^*}, \eta_{l^*}) \leftarrow (\hat{\mathcal{T}}_{l^*}^{\text{old}}, \hat{\mathbf{x}}_{l^*}^{\text{old}}, \eta_{l^*}^{\text{old}})$
 4:    end if
 5:    $(\hat{\mathcal{T}}_{l^*}^{\text{old}}, \hat{\mathbf{x}}_{l^*}^{\text{old}}, \eta_{l^*}^{\text{old}}) \leftarrow (\hat{\mathcal{T}}_{l^*}, \hat{\mathbf{x}}_{l^*}, \eta_{l^*})$
 6:    $\hat{\mathcal{T}}_l^{\text{old}} \leftarrow \hat{\mathcal{T}}_l,\ \forall\ l \in \mathcal{L}_{l^*} \setminus l^*$
 7:    { Transmit: $\hat{\mathcal{T}}_{l^*}$ to all nodes $l \in \mathcal{L}_{l^*}$ }
 8:    { Receive: $\hat{\mathcal{T}}_l$ from all nodes $l \in \mathcal{L}_{l^*}$ }
 9:    $\hat{\mathcal{T}}_{l^*}^{(c)} \leftarrow \texttt{vote}(\mathcal{L}_{l^*}, \{\hat{\mathcal{T}}_l\}, K^{(c)})$          (Note: $|\hat{\mathcal{T}}_{l^*}^{(c)}| = K^{(c)}$)
10:    $(\hat{\mathcal{T}}_{l^*}, \hat{\mathbf{x}}_{l^*}, \eta_{l^*}) \leftarrow \text{modSP}(\mathbf{A}_{l^*}, K_{l^*,\max}, \mathbf{y}_{l^*}, \hat{\mathcal{T}}_{l^*}^{(c)})$
11: until $((\eta_{l^*} \geq \eta_{l^*}^{\text{old}})$ and $(\hat{\mathcal{T}}_l = \hat{\mathcal{T}}_l^{\text{old}}\ \forall\ l \in \mathcal{L}_{l^*}))$
Output: $\hat{\mathbf{x}}_{l^*}^{\text{old}}$, $\hat{\mathcal{T}}_{l^*}^{\text{old}}$
```

Input to Algorithm 9 is the l^*-th node's sensing matrix $\mathbf{A}_{l^*}$, the measurement vector $\mathbf{y}_{l^*}$, and the size of the private and common support set $K_{l^*}^{(p)}$ and $K^{(c)}$. In the *initialization* phase of the algorithm, before any communication has taken place, modSP$(\cdot)$ is executed to achieve a first estimate of the l^*-th node's support-set. The residual norm η_{l^*} is stored to use as the performance measure and the support-set estimates of the neighboring nodes are initialized as the empty set $\emptyset$. In the *iteration* phase of DiSP, there are four main functionalities: (1) Steps 2 to 4 prevent the result from deviating away from a better solution, which empirically was observed to happen if the intermediate estimated support-set in step 10 was worse than the estimated support-set in the previous iteration (denoted by the use of 'old'). (2) Steps 7 to 8 constitute the communication phase, where the locally estimated support-sets are exchanged among the connected nodes. (3) Using the voting function (of Algorithm 7) in step 9, an estimate of the common support-set with full cardinality $K^{(c)}$ is achieved. (4) Using the estimated common support-set, modSP$(\cdot)$ is executed locally to estimate a new full support-set, to again be distributed over the network. These four functionalities are iteratively performed until convergence is achieved. For convergence, we have a stopping criterion based on two conditions to be fulfilled together: (a) the residual norm in the l^*-th node does not decrease, and (b) no new support-set estimates from connected nodes arrive. The second condition is used due to the fact that if the l^*-th node receives new improved support-set estimates from its neighbors then its own support-set estimate may improve in a later iteration.

3.5 Further Scope of Improvement

Different strategies are used for developing DiOMP and DiSP algorithms. For DiOMP, we build the common support-set estimate $\hat{\mathcal{T}}^{(c)}$ serially, that is its components are detected serially one-by-one. The DiOMP is built on the use of modOMP which is categorized as an S-pursuit algorithm with the characteristics of I-support construction mechanism (see Table 1). In the modOMP, the use of serial approach (S-pursuit) allows for a high reliability to detect the correct element in the current iteration, but (for modOMP) the irreversible support-set construction mechanism (I-support) also has a disadvantage. The disadvantage is that if a wrong element is found to be reliable and added to the support-set in a previous iteration then the element remains in the support-set forever. In contrast, for DiSP, we use the parallel approach where the common support-set estimate (with full cardinality) is refined iteratively. The DiSP is built on the use of modSP and the modSP is categorized as a P-pursuit algorithm with characteristics of the R-support construction mechanism. The R-support construction mechanism has the capability to remove a wrong element in a future iteration even though the element was found to be reliable and added to the support-set in a past iteration. This support-set construction mechanism is reversible in nature (thus the notation R-support).

Considering the advantages of the serial approach (S-pursuit) and the reversible construction mechanism (R-support), we develop a new GP algorithm in the next section that has both the characteristics. To the best of the authors' knowledge, such algorithm is not yet proposed in the literature (see Table 1). Then, we use the new GP algorithm as a building block to develop a third DiGP scheme.

4 Distributed Forward-Reverse OMP

In this section, we first develop the new GP algorithm called forward-reverse OMP (FROMP) which is an S-pursuit algorithm with R-support support-set construction mechanism. Based on FROMP, we then develop a new DiGP algorithm called distributed FROMP (DiFROMP).

4.1 Forward-Reverse OMP

The development of FROMP is based on OMP. For OMP, a careful study reveals that the use of highest-amplitude based element-selection strategy leads to a natural selection scheme in which the elements are chosen in an ordered manner. Ideally OMP serially detects the elements according to their decreasing strength of amplitudes. The success of this ordered selection strategy depends on the level of system uncertainty. For a highly under-sampled system, the highest amplitude based selection strategy may fail to detect a correct element and erroneously include a wrong element in the support-set. To improve this strategy further, a reliability testing procedure after the selection may be helpful for eliminating the errors.

For developing the FROMP, we refer the serial add strategy of including a potential element in the support-set as *forward add.* This forward add strategy is directly used in standard OMP and we present it as a separate algorithm in Algorithm 10. In Algorithm 10, we supply the in-

Algorithm 10 : Forward-add

```
Input: $\mathbf{A}$, $\mathbf{r}_k$, $\mathcal{T}_k$
1: $\tau_{\max} \leftarrow \texttt{max\_indices}(\mathbf{A}^T\mathbf{r}_k, 1)$
2: $\mathcal{T}_{k+1} \leftarrow \mathcal{T}_k \cup \tau_{\max}$
3: $\mathbf{r}_{k+1} \leftarrow \texttt{resid}(\mathbf{y}, \mathbf{A}_{\mathcal{T}_{k+1}})$
Output: $\mathbf{r}_{k+1}$, $\mathcal{T}_{k+1}$
```

puts: $\mathbf{A}$, $\mathbf{r}_k$, $\mathcal{T}_k$. Here, $\mathbf{A}$ is the sensing matrix, $\mathbf{r}_k$ is the residual vector for iteration k and $\mathcal{T}_k$ is the support-set estimate for iteration k. Then, analogous to OMP, `max_indices(.,.)` and `resid(.,.)` are used and the algorithm outputs the residual $\mathbf{r}_{k+1}$ and support-set $\mathcal{T}_{k+1}$ for iteration $k+1$. Now using Algorithm 10, we define the following function.

Function 3 *(Forward-add) For the k'th iteration, the sensing matrix* $\mathbf{A}$*, the current residual* $\mathbf{r}_k$ *and the current support-set* $\mathcal{T}_k$ *are given. Then the new support-set with cardinality* $(|\mathcal{T}_k| + 1)$ *and its corresponding residual are the outputs of the following algorithmic function*

$$(\mathbf{r}_{k+1}, \mathcal{T}_{k+1}) \leftarrow \texttt{\textit{forward\_add}}(\mathbf{A}, \mathbf{r}_k, \mathcal{T}_k),$$

which exactly executes Algorithm 10.

After the forward-add strategy is performed, it is natural to include a reliability testing strategy. For this, we develop a new scheme where the k most prominent support-set elements are chosen from $(k + 1)$ elements. This new selection algorithm is presented in Algorithm 11. In

```
Algorithm 11 : Reverse-fetch
Input: $\mathbf{A}$, $\mathcal{T}_{k+1}$, $k$                    (Note: $|\mathcal{T}_{k+1}| = k + 1$)
 1: $\tilde{\mathbf{x}}$  such that  $\tilde{\mathbf{x}}_{\mathcal{T}_{k+1}} = \mathbf{A}^{\dagger}_{\mathcal{T}_{k+1}}\mathbf{y}$ and $\tilde{\mathbf{x}}_{\bar{\mathcal{T}}_{k+1}} = \mathbf{0}$
 2: $\mathcal{T}' \leftarrow \texttt{max\_indices}(\tilde{\mathbf{x}}, k)$                (Note: $|\mathcal{T}'| = k$)
 3: $\mathbf{r}' \leftarrow \texttt{resid}(\mathbf{y}, \mathbf{A}_{\mathcal{T}'})$
Output: $\mathbf{r}'$, $\mathcal{T}'$
```

Algorithm 11, we supply the inputs $\mathbf{A}$, $\mathcal{T}_{k+1}$ and k. By using least squares estimation, we find an estimate of the intermediate $k + 1$ non-zero elements of the sparse signal $\tilde{\mathbf{x}}$. Based on this signal estimate, the temporary support-set $\mathcal{T}'$ of cardinality k and the corresponding temporary residual $\mathbf{r}'$ are found. Using Algorithm 11 we define the following function.

Function 4 *(Reverse-fetch) Let the sensing matrix* $\mathbf{A}$ *and a support-set* $\mathcal{T}_{k+1}$ *of cardinality of* $k + 1$ *be given. Then the temporary support-set* $\mathcal{T}'$ *with cardinality* k *and its corresponding residual are the outputs of the following algorithmic function*

$$(\mathbf{r}', \mathcal{T}') \leftarrow \texttt{\textit{reverse\_fetch}}(\mathbf{A}, \mathcal{T}_{k+1}, k),$$

which exactly executes Algorithm 11.

Based on the `forward_add()` and `reverse_fetch()` functions, we now develop the FROMP in Algorithm 12. Similarly to the modSP and modOMP algorithms, the inputs to FROMP are $\mathbf{A}$, $K_{\max}$, $\mathbf{y}$ and $\mathcal{T}_{\text{ini}}$. In the *initialization* phase FROMP calls the modOMP (see Appendix A.1) procedure (step 2) to form a full support-set estimate $\mathcal{T}_0$. If there are errors in $\mathcal{T}_{\text{ini}}$, those errors will remain in $\mathcal{T}_0$ (and in $\mathbf{x}_0$). Then, in steps 3

Algorithm 12 : Forward-Reverse OMP (FROMP)

```
Input: $\mathbf{A}$, $K_{\max}$, $\mathbf{y}$, $\mathcal{T}_{\text{ini}}$
Initialization:
 1: $\mathbf{R} = [\mathbf{r}_1 \; \mathbf{r}_2 \; \ldots, \mathbf{r}_{K_{\max}}]$        (For storing residuals)
 2: $(\mathcal{T}_0, \mathbf{x}_0) \leftarrow \text{mod}\textsc{omp}(\mathbf{y}, \mathbf{A}, K_{\max}, \mathcal{T}_{\text{ini}})$
 3: for $l = 1 : K_{\max}$ do
 4:   $\mathcal{T}' \leftarrow \texttt{max\_indices}(\mathbf{x}_0, l)$        (Note: $|\mathcal{T}'| = l$)
 5:   $\mathbf{r}_l \leftarrow \texttt{resid}(\mathbf{y}, \mathbf{A}_{\mathcal{T}'})$
 6: end for
 7: $k \leftarrow K_{\max}$, $\mathcal{T}_k \leftarrow \mathcal{T}'$
Iteration:
 1: repeat
 2:   $(\mathbf{r}_{k+1}, \mathcal{T}_{k+1}) \leftarrow \texttt{forward\_add}(\mathbf{A}, \mathbf{r}_k, \mathcal{T}_k)$
 3:   repeat
 4:     $(\mathbf{r}', \mathcal{T}') \leftarrow \texttt{reverse\_fetch}(\mathbf{A}, \mathcal{T}_{k+1}, k)$
 5:     if $(\|\mathbf{r}'\| < \|\mathbf{r}_k\|)$ then
 6:       $\mathcal{T}_k \leftarrow \mathcal{T}'$, $\mathbf{r}_k \leftarrow \mathbf{r}'$
 7:       $k \leftarrow k - 1$
 8:     else
 9:       break
10:     end if
11:   until $(k = 0)$
12:   $k \leftarrow k + 1$
13: until $k = K_{\max} + 1$
Output:
 1: $\hat{\mathcal{T}} = \mathcal{T}_{k-1}$
 2: $\hat{\mathbf{x}}$ such that $\hat{\mathbf{x}}_{\hat{\mathcal{T}}} = \mathbf{A}^{\dagger}_{\hat{\mathcal{T}}}\mathbf{y}$ and $\hat{\mathbf{x}}_{\bar{\hat{\mathcal{T}}}} = \mathbf{0}$
 3: $\eta = \|\mathbf{r}_{k-1}\|$
Functional form: $(\hat{\mathcal{T}}, \hat{\mathbf{x}}, \eta) \leftarrow \textsc{fromp}(\mathbf{A}, K, \mathbf{y}, \mathcal{T}_{\text{ini}})$
```

to 6, an ordering procedure is performed which helps to arrange the corresponding residual vectors appropriately. This ordering is necessary for the reliability testing. Notice that the iteration phase starts with $k = K_{\max}$. For clarity, we denote $\mathbf{r}_k$, $\mathbf{r}_{k+1}$ and $\mathbf{r}'$ as the current, intermediate and temporary residuals, respectively. In the k'th *iteration*, two main tasks are performed. First, when the algorithm performs `forward_add()`, the output is an intermediate support-set $\mathcal{T}_{k+1}$ with cardinality larger than the current support-set by one. Second, for reliability testing, the `reverse_fetch()` function is invoked to find the k elements from the intermediate support-set of cardinality $(k+1)$. These

k elements form the temporary support-set $\mathcal{T}'$. Then, considering the residual norm as the model fit measure, a comparison between residual norms is performed. For the comparison, if the temporary residual norm $\|\mathbf{r}'\|$ is smaller than the current residual norm $\|\mathbf{r}_k\|$, then the temporary support-set $\mathcal{T}'$ replaces the current support-set $\mathcal{T}_k$ and $\mathcal{T}_k$ acts as the new current support-set. Similarly if $\|\mathbf{r}'\|$ is smaller than $\|\mathbf{r}_k\|$, $\mathbf{r}'$ replaces $\mathbf{r}_k$. Now, the algorithm decreases the iteration counter with one and continue the reverse operation of refining the support-set. Note that the reverse operation is a serial operation, similar to the forward-add operation. In the case when $\|\mathbf{r}'\|$ is not smaller than $\|\mathbf{r}_k\|$, the reverse operation is not performed; we assume that the current support-set is reliable and `forward_add`() is performed for the inclusion of a new element (serially). We conclude that both the operation of increasing the support-set and the operation of correcting it are serial operations, thus this algorithm is indeed categorized as an S-pursuit algorithms with R-support construction mechanism.

4.2 Distributed FROMP

Since FROMP is an R-support algorithm, the same approach can be taken for DiFROMP as was taken for DiSP. In fact, it turns out that by just replacing modSP$(\cdot)$ with FROMP$(\cdot)$ in Algorithm 9, we can develop the DiFROMP algorithm.

Remark 1.3 Following the development of DiOMP and DiSP based on modification of OMP and SP, respectively, and then developing DiFROMP based on new FROMP, we can safely claim that many existing GP algorithms can be modified and new GP algorithms can be developed for building new DiGP algorithms. For example, we could easily modify StOMP or CoSaMP for the purpose of developing new DiGP algorithms.

5 Simulation Results

Using typical setups, we performed computer simulations in order to observe the performance of three DiGP algorithms: DiOMP, DiSP and DiFROMP. We compare their performance with two extreme cases: (1) with a centralized solution (i.e., a fully connected network where each node is connected with all other nodes and hence the connection matrix is $\mathbf{C}_{L-1}$) where we refer the algorithms as joint OMP (JOMP) [7], joint SP (JSP) [7] and joint FROMP (JFROMP); and (2) to a fully disconnected setup (the connections matrix is $\mathbf{C}_0$) where OMP, SP and FROMP are executed independently. In this paper we focus on the development of a GP framework for distributed CS and therefore limit ourselves in the gamut of GP algorithms. We first discuss the reconstruction performance measures and experimental setups, and then report the performance results of all the algorithms for clean and noisy measurement cases.

5.1 Performance measures and experimental setups

We use two performance measures. For the first performance measure, we use signal-to-reconstruction-error ratio (SRER) defined as

$$\text{SRER} = \frac{\mathcal{E}\{\|\mathbf{x}\|_2^2\}}{\mathcal{E}\{\|\mathbf{x} - \hat{\mathbf{x}}\|_2^2\}}, \tag{9}$$

where $\hat{\mathbf{x}}$ is the reconstructed signal vector and our objective is to achieve a higher SRER. Note that we drop the subscript l because we are averaging over all sensors l.

Next we define another performance measure which provides a direct measure of estimating the underlying support set. This is a distortion measure defined by $d(\mathcal{T}, \hat{\mathcal{T}}) = 1 - \left(|\mathcal{T} \cap \hat{\mathcal{T}}|/|\mathcal{T}|\right)$ [26] and we have recently used it in [21]. Here, $\mathcal{T}$ is the local support-set, that is $\mathcal{T} = \mathcal{T}^{(c)} \cup \mathcal{T}^{(p)}$. Considering a large number of realizations (signal vectors), we can compute the average of $d(\mathcal{T}, \hat{\mathcal{T}})$. We define the average support-set cardinality error (ASCE) as follows

$$\text{ASCE} = \mathbb{E}\left\{d(\mathcal{T}, \hat{\mathcal{T}})\right\} = 1 - \mathbb{E}\left\{\frac{|\mathcal{T} \cap \hat{\mathcal{T}}|}{|\mathcal{T}|}\right\}. \tag{10}$$

Note that the ASCE has the range $[0, 1]$ and our objective is to achieve a lower ASCE. Along-with SRER, the ASCE is used as the second performance evaluation measure because the principle objective of most GP algorithms is to estimate the underlying support set.

Next we describe simulation setups. In any CS setup, all sparse signals are expected to be exactly reconstructed if the number of measurements are more than a certain threshold value. The computational complexity to test this uniform reconstruction ability is exponentially high. Instead, we can rely on empirical testing, where SRER and ASCE are computed for random measurement matrix ensemble. To measure the level of undersampling, let us define the fraction of measurements

$$\alpha = \frac{M}{N}. \tag{11}$$

For a given network topology $\mathbf{C}_i$, $i \in [0, L-1]$, the steps of testing strategy are listed as follows:

1. Given the parameters N, $K^{(c)}$ and $\{K_l^{(p)}\}_{l=1}^{L}$ choose an α (such that M is an integer). We use same $K_l^{(p)}$, $\forall l$.
2. Randomly generate a set of $M \times N$ sensing matrices $\{\mathbf{A}_l\}_{l=1}^{L}$ where the components are drawn independently from an i.i.d. Gaussian source (i.e. $a_{m,n} \sim \mathcal{N}\left(0, \frac{1}{M}\right)$) and then scale the columns of $\mathbf{A}_l$ to unit-norm.
3. Randomly generate a set of signal vectors $\{\mathbf{x}_l\}_{l=1}^{L}$ following Section 2.1. The common and private support-sets are chosen uniformly over the set $\{1, 2, \ldots, N\}$. The non-zero components of $\mathbf{x}$ are independently drawn by either of the following two methods.
   (a) The non-zero components are drawn independently from a standard Gaussian source. This type of signal is referred to as Gaussian sparse signal.
   (b) The non-zero components are set to ones. This type of signal is referred to as binary sparse signal.

   Note that the Gaussian sparse signal is compressible in nature. That means, in the descending order, the sorted amplitudes of a Gaussian sparse signal vector's components decay fast with respect to the sorted indices. This decaying trend corroborates with several natural signals (for example, wavelet coefficients of an image). On the other hand, a binary sparse signal is not compressible in nature, but of special interest for comparative study, since it represents a particularly challenging case for OMP-type of reconstruction strategies [18], [19].
4. Compute the measurements $\mathbf{y}_l = \mathbf{A}_l \mathbf{x}_l + \mathbf{w}_l, \forall l \in \{1, 2, ..., L\}$. Here $\mathbf{w}_l \sim \mathcal{N}(\mathbf{0}, \sigma_l^2 \mathbf{I}_M)$.
5. Apply the CS algorithms on the data $\{\mathbf{y}_l\}_{l=1}^{L}$ independently.

In the above simulation procedure, for each node $l \in \{1, 2, \ldots, L\}$, Q sets of sensing matrices are created. Then for each sensing node, P sets of data vectors are created. In total, we will average over $L \cdot Q \cdot P$ data to evaluate the performance.

Dropping the subscript l and considering the measurement noise $\mathbf{w} \sim \mathcal{N}\left(\mathbf{0}, \sigma^2 \mathbf{I}_M\right)$, we define the signal-to-measurement-noise-ratio (SMNR) as

$$\text{SMNR} = \frac{\mathcal{E}\{\|\mathbf{x}\|_2^2\}}{\mathcal{E}\{\|\mathbf{w}\|_2^2\}}, \tag{12}$$

where $\mathcal{E}\{\|\mathbf{w}\|_2^2\} = \sigma_w^2 M$. For noisy measurement case, we report the experimental results at SMNR 20 dB.

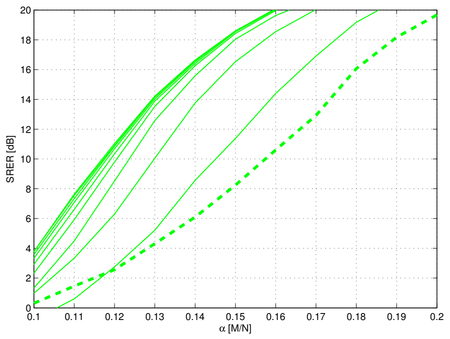
In the presence of a measurement noise, it is impossible to achieve perfect CS recovery. On the other hand, for the clean measurement case, perfect CS recovery of a sparse signal is possible if α exceeds a certain threshold. In the spirit of using CS for practical applications with a less number of measurements at clean and noisy conditions, we are mainly interested in a lower range of α where performances of the contesting algorithms can be fairly compared.

5.2 Experimental Results

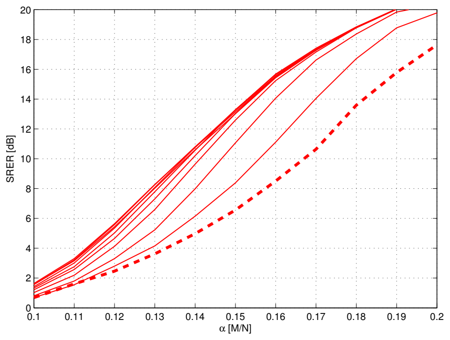
Using $N = 500$, $K^{(c)} = 10$, $K^{(p)} = 10$, $Q = 100$, $P = 100$ and $L = 10$, we performed experiments. That means, we used 500-dimensional sparse signal vectors with sparsity level less that or equal to $K^{(c)} + K^{(p)} = 20$. Such a 4% sparsity level is chosen in accordance with real life scenarios, for example most of the energy of an image signal in the wavelet domain is concentrated within $2 - 4\%$ coefficients [27]. There are $L = 10$ nodes and for each node, we create $Q = 100$ signal vectors and $P = 100$ sensing matrices. Thus, for a chosen α, we evaluate performance by averaging $100 \times 100 \times 10 = 100000$ realizations in each data point. We increment α from a lower limit to a higher limit in a small step-size (with the constraint that corresponding M is an integer for a value of α) and report the results.

Impact of network connectivity

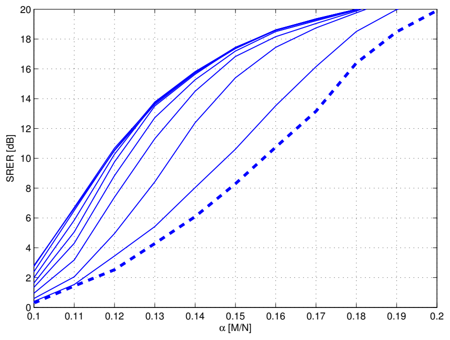
Let us first observe the effect of increasing network connectivity on the performance of all the three DiGP algorithms in Figure 2. In this case we show the results in the range of α from 0.1 to 0.2. We use Gaussian sparse signal and SMNR = 20 dB. The two extreme results are the performances for $\mathbf{C}_0$ and $\mathbf{C}_9$. Here we mention that $\mathbf{C}_0$ denotes the case of using standard GP algorithms. We show the SRER results for the three DiGP algorithms and observe that the performance improves with the increase of network connectivity. We note that the use of a degree-2 network (connection matrix is $\mathbf{C}_2$) leads to much better performance than the standard $\mathbf{C}_0$ case. For DiOMP, at $\alpha = 0.14$, we achieve 6 dB improvement in SRER compared to the $\mathbf{C}_0$ case (i.e., the OMP performance). The performance shows a saturation trend as the connectivity increases. Considering a trade-off between network connectivity (i.e., communication resource) and performance, we use $\mathbf{C}_2$ as the default network for further results. We also comment that $\mathbf{C}_2$ is quite restrictive network in the sense that it is not particularly well connected, but still its use leads to a significant gain in performance.



(a) DiOMP



(b) DiSP



(c) DiFROMP

Figure 2: Schematic figures showing the performance in SRER versus fraction of measurements for noisy measurements SMNR = 20 dB of the different distributed GP algorithms using network connectivity ranging from the local disconnected algorithms ($\mathbf{C}_0$), through $\mathbf{C}_1$, $\mathbf{C}_2$, …, $\mathbf{C}_L$. The lower-most fat, dashed, curve corresponds to a standard (disconnected) algorithm and the top most a joint algorithm (fully connected network).

Comparison between algorithms

Here we provide a comparative study between the three DiGP algorithms, the three fully connected joint GP algorithms and the three completely disconnected, standard, GP algorithms. So, we compare DiOMP, JOMP, OMP, and DiSP, JSP, SP, and DiFROMP, JFROMP, FROMP algorithms. We use the degree-2 network (connection matrix $\mathbf{C}_2$) for all the DiGP algorithms.

Figure 3 shows SRER and ASCE results for the case of Gaussian sparse signal at clean measurement conditions. In Figure 3a, the three bottom-most SRER curves correspond to the disconnected algorithms. It is important to notice that the new FROMP and DiFROMP perform better than OMP and DiOMP, respectively. We also note that SP and DiSP perform poorer corresponding to relevant competing algorithms. At $\alpha = 0.15$, we note that DiOMP provides nearly 15 dB performance improvement compared to the disconnected, standard, OMP. Thus, we can comment that our DiGP algorithms provide a significant improvement. Similar trends in performance are observed in Figure 4 for the noisy measurement condition with SMNR = 20 dB.

Next we provide the performance results for the binary sparse signal case. Figure 5 shows the results at clean measurement conditions. In this case, the most interesting observation is that the SP and its allied algorithms (DiSP and JSP) provide significant performance improvements compared to the other relevant competing algorithms. Again we note that DiGP algorithms using degree-2 network provide better results than the disconnected stand-alone GP algorithms. Similar trends in performance are observed in Figure 6 for the noisy measurement condition with SMNR = 20 dB.

Comparing all the results for two different signals at varying measurement conditions and number of measurements, we note that the new DiGP algorithms have a promise to provide a good performance. They are capable to provide a good trade-off between network connectivity (i.e., network resources) and performance.

Running-Time Comparison

At last, we endeavor to provide a running time comparison between the algorithms. This comparison provides a rough idea about computational resources. The running time results are shown in Table 2 for varying network connectivity. In this case, we performed simulations for the Gaussian sparse signal case at 20 dB SMNR and $\alpha = 0.16$. An interesting point to notice is that even though FROMP is more complex than the OMP, DiFROMP requires less computational resource than DiOMP. The reason

Table 2: Running time comparison between GP and DiGP algorithms at varying network connectivity

Network Connectivity	GP algorithms			DiGP algorithms		
	OMP	SP	FROMP	DiOMP	DiSP	DiFROMP
$\mathbf{C}_0$	0.46495	0.39855	1.0298	×	×	×
$\mathbf{C}_2$	×	×	×	4.2853	1.6673	3.0909
$\mathbf{C}_3$	×	×	×	4.2969	1.7521	3.2325
$\mathbf{C}_9$	×	×	×	4.3336	1.5053	3.3566

is that FROMP is characterized by R-support construction mechanism and hence its use for designing the DiFROMP algorithm leads to faster convergence.

Reproducible results: In the spirit of reproducible results, we provide a package with all necessary Matlab codes in the following website: https://sites.google.com/site/saikatchatt/softwares/. In this package consult the README.TXT file to obtain instructions on how to reproduce the data and figures presented in this paper.

6 Conclusion

For a distributed CS setup, we have developed a framework for constructing distributed greedy pursuit (DiGP) algorithms. Using this framework, we have shown how the two well known greedy algorithms OMP and SP can be used for developing two new DiGP algorithms. Furthermore, we have created a new GP algorithm called FROMP using insights gained from a categorization of existing GP algorithms. Then, based on FROMP we have created a third DiGP algorithm. In particular we notice that within the new framework, many other GP algorithms could be used with small modifications as a base in designing new DiGP algorithms. Through experimental evaluations we conclude that the new DiGP algorithms provide a significant improvement in performance compared to standard, disconnected, GP algorithms. We also note that the algorithms are capable of providing a trade-off between performance and network connectivity (or a trade-off between performance and communication resource).

Appendix A

A.1 Modified OMP

In this section, we describe the modified OMP (modOMP) algorithm. Algorithm 13 shows the modOMP. Instead of initializing with an empty support-set and begin iterating from the first component as in the standard OMP, we allow the modOMP to use an initial support-set as input and continue building the final support-set. This modification reduces to the standard OMP (as shown in [18]) when the initial support-set $\mathcal{T}_{\text{ini}} = \emptyset$. In step 2 of the *initialization*, modOMP finds a residual, where

Algorithm 13 : modOMP

```
Input: $\mathbf{A}$, $K_{\max}$, $\mathbf{y}$, $\mathcal{T}_{\text{ini}}$.
Initialization:
  1: $\mathcal{T}_0 \leftarrow \mathcal{T}_{\text{ini}}$
  2: $\mathbf{r}_0 \leftarrow \texttt{resid}(\mathbf{y}, \mathbf{A}_{\mathcal{T}_0})$
  3: $k \leftarrow |\mathcal{T}_0|$
Iteration:
  1: repeat
  2:     $k \leftarrow k + 1$
  3:     $\tau_{\max} \leftarrow \texttt{max\_indices}\left(\mathbf{A}^T \mathbf{r}_{k-1}, 1\right)$
  4:     $\mathcal{T}_k \leftarrow \mathcal{T}_{k-1} \cup \tau_{\max}$
  5:     $\mathbf{r}_k \leftarrow \texttt{resid}(\mathbf{y}, \mathbf{A}_{\mathcal{T}_k})$
  6: until $(k = K_{\max})$
Output:
  1: $\hat{\mathcal{T}} \leftarrow \mathcal{T}_k$
  2: $\hat{\mathbf{x}}$ such that $\hat{\mathbf{x}}_{\mathcal{T}_k} = \mathbf{A}^{\dagger}_{\mathcal{T}_k} \mathbf{y}$ and $\hat{\mathbf{x}}_{\overline{\mathcal{T}}_k} = \mathbf{0}$
  3: $\eta \leftarrow \|\mathbf{r}_k\|$
Functional form: $(\hat{\mathcal{T}}, \hat{\mathbf{x}}, \eta) \leftarrow \text{modOMP}(\mathbf{A}, K_{\max}, \mathbf{y}, \mathcal{T}_{\text{ini}})$
```

$\mathcal{T}_0 = \mathcal{T}_{\text{ini}}$. If the initial support-set $\mathcal{T}_{\text{ini}} = \emptyset$, the matrix $\mathbf{A}_{\mathcal{T}_0}$ is empty and the residual becomes $\mathbf{y}$. At the k'th *iteration* stage modOMP algorithm forms the matched filter, identifies the index corresponding to the largest amplitude (step 3) and adds this to the support-set (step 4). It proceeds with solving a least squares problem with the selected indices (step 5), subtracts the least squares fit and produces a new residual (step 6). This process is updated until $K_{\max}$ components have been picked in the support-set. In addition to the support-set estimate $\hat{\mathcal{T}}$, we also output the sparse signal estimate $\hat{\mathbf{x}}$ and the final residual norm η.

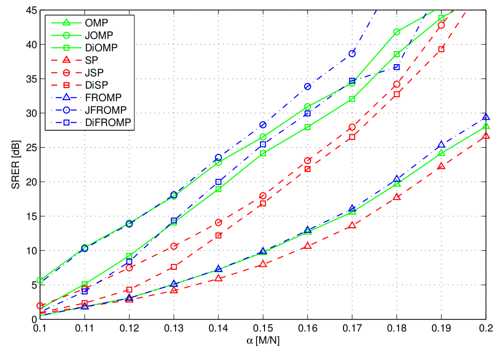
A.2 Modified SP

In this section, we describe the modified SP (modSP) in Algorithm 14. Similarly to modOMP, we provide an initial support-set $\mathcal{T}_{\text{ini}}$ to the modSP. Then, modOMP will continue to improve this support-set building the final support-set. When $\mathcal{T}_{\text{ini}} = \emptyset$, the modSP reduces to the standard SP (as shown in [19]). At k'th *iteration* stage, the modified SP algorithm

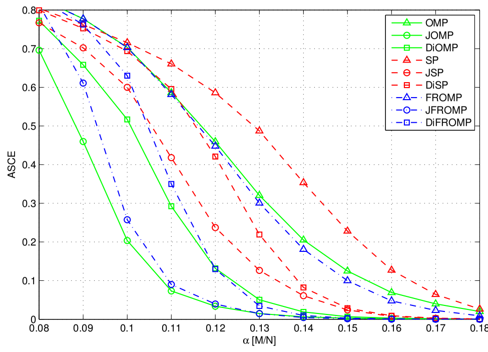
Algorithm 14 : modSP

```
Input: $\mathbf{A}$, $K_{\max}$, $\mathbf{y}$, $\mathcal{T}_{\text{ini}}$
Initialization:
  1: $\mathcal{T}' \leftarrow \texttt{max\_indices}\left(\mathbf{A}^T\mathbf{y}, K_{\max}\right) \cup \mathcal{T}_{\text{ini}}$
  2: $\hat{\mathbf{x}}$ such that $\hat{\mathbf{x}}_{\mathcal{T}'} = \mathbf{A}^{\dagger}_{\mathcal{T}'}\mathbf{y}$ and $\hat{\mathbf{x}}_{\bar{\mathcal{T}}'} = \mathbf{0}$
  3: $\mathcal{T}_0 \leftarrow \texttt{max\_indices}(\hat{\mathbf{x}}, K_{\max})$
  4: $\mathbf{r}_0 \leftarrow \texttt{resid}(\mathbf{y}, \mathbf{A}_{\mathcal{T}_0})$
  5: $k \leftarrow 0$
Iteration:
  1: repeat
  2:     $k \leftarrow k+1$
  3:     $\mathcal{T}' \leftarrow \texttt{max\_indices}\left(\mathbf{A}^T\mathbf{r}_{k-1}, K_{\max}\right) \cup \mathcal{T}_{k-1}$
  4:     $\hat{\mathbf{x}}$ such that $\hat{\mathbf{x}}_{\mathcal{T}'} = \mathbf{A}^{\dagger}_{\mathcal{T}'}\mathbf{y}$ and $\hat{\mathbf{x}}_{\bar{\mathcal{T}}'} = \mathbf{0}$
  5:     $\mathcal{T}_k \leftarrow \texttt{max\_indices}(\hat{\mathbf{x}}, K_{\max})$
  6:     $\mathbf{r}_k \leftarrow \texttt{resid}(\mathbf{y}, \mathbf{A}_{\mathcal{T}_k})$
  7: until $(\|\mathbf{r}_k\| \geq \|\mathbf{r}_{k-1}\|)$
  8: $k \leftarrow k-1$                                  (‘Previous iteration count’)
Output:
  1: $\hat{\mathcal{T}} \leftarrow \mathcal{T}_k$
  2: $\hat{\mathbf{x}}$ such that $\hat{\mathbf{x}}_{\mathcal{T}_k} = \mathbf{A}^{\dagger}_{\mathcal{T}_k}\mathbf{y}$ and $\hat{\mathbf{x}}_{\bar{\mathcal{T}}_k} = \mathbf{0}$
  3: $\eta \leftarrow \|\mathbf{r}_k\|$
Functional form: $(\hat{\mathcal{T}}, \hat{\mathbf{x}}, \eta) \leftarrow \text{modSP}(\mathbf{A}, K_{\max}, \mathbf{y}, \mathcal{T}_{\text{ini}})$
```

forms the matched filter $\mathbf{A}^T\mathbf{r}_{k-1}$, identifies the $K_{\max}$ most prominent indices and merges them with the old support-set (step 3). This support-set $\mathcal{T}'$ is likely to have a cardinality larger than $K_{\max}$ (usually $K_{\max} \leq |\mathcal{T}'| \leq 2K_{\max}$). The algorithm then forms a least squares estimate with the selected indices of $\mathcal{T}'$ and identifies the indices corresponding to the $K_{\max}$ largest amplitude (step 4 and 5). The modSP then finds the residual (step 6) and repeats the iteration process until the residual norm does not increase. In addition to the support-set estimate $\hat{\mathcal{T}}$, we also output the sparse signal estimate $\hat{\mathbf{x}}$ and the final residual norm η.

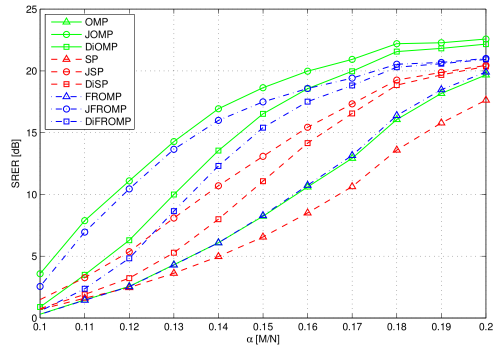


(a) SRER (in dB) vs α

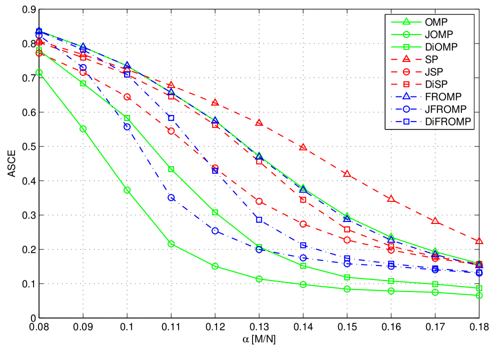


(b) ASCE vs α

Figure 3: GP, DiGP and joint GP algorithms for **Gaussian** sparse signal at **clean measurement** condition.

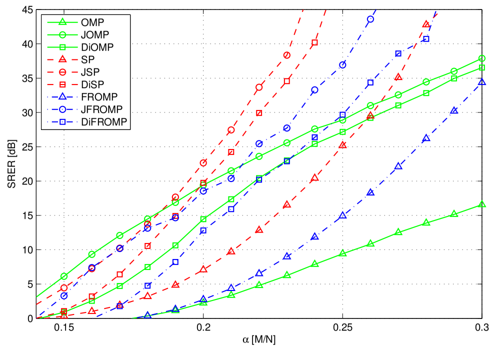


(a) SRER (in dB) vs α

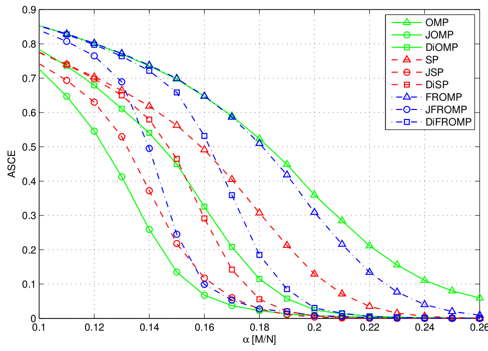


(b) ASCE vs α

Figure 4: GP, DiGP and joint GP algorithms for **Gaussian** sparse signal at **noisy measurements**, where SMNR = 20 dB.

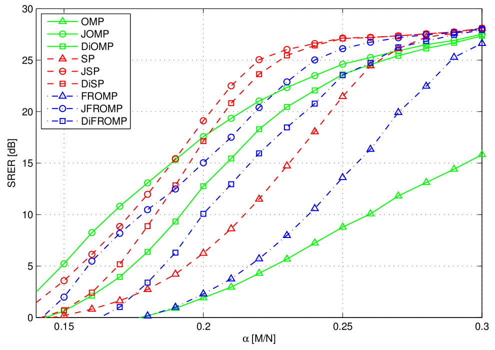


(a) SRER (in dB) vs α

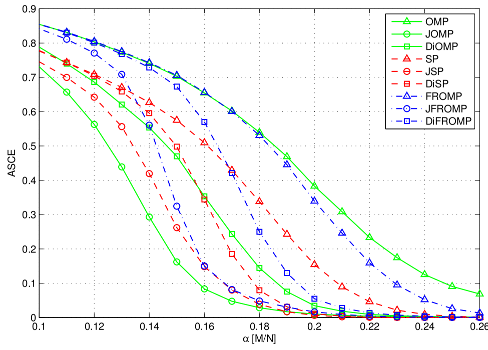


(b) ASCE vs α

Figure 5: GP, DiGP and joint GP algorithms for **binary** sparse signal at **clean measurement** condition.



(a) SRER (in dB) vs α



(b) ASCE vs α

Figure 6: GP, DiGP and joint GP algorithms for **binary** sparse signal at **noisy measurements**, where SMNR = 20 dB.

References

[1] D.L. Donoho, "Compressed sensing," *IEEE Trans. Inf. Theory*, vol. 52, no. 4, pp. 1289 –1306, April 2006.

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

[3] A.Y. Yang, M. Gastpar, R. Bajcsy, and S.S. Sastry, "Distributed sensor perception via sparse representation," *Proc. of the IEEE*, vol. 98, no. 6, pp. 1077 –1088, June 2010.

[4] F. Zeng, C. Li, and Z. Tian, "Distributed compressive spectrum sensing in cooperative multihop cognitive networks," *IEEE Journal Selected Topics in Signal Processing*, vol. PP, no. 99, pp. 1 –1, 2010.

[5] J.A. Bazerque and G.B. Giannakis, "Distributed spectrum sensing for cognitive radio networks by exploiting sparsity," *IEEE Trans. Signal Processing*, vol. 58, no. 3, pp. 1847 –1862, Mar. 2010.

[6] Q. Ling and Z. Tian, "Decentralized support detection of multiple measurement vectors with joint sparsity," in *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing (ICASSP)*, may 2011, pp. 2996 –2999.

[7] D. Sundman, S. Chatterjee, and M. Skoglund, "Greedy pursuits for compressed sensing of jointly sparse signals," in *Proc. Eur. Sig. Proc. Conf.*, Aug 2011.

[8] Dror Baron, Marco F. Duarte, Michael B. Wakin, Shriram Sarvotham, and Richard G. Baraniuk, "Distributed compressive sensing," http://arxiv.org/abs/0901.3403, 2009.

[9] M.F. Duarte, S. Sarvotham, D. Baron, M.B. Wakin, and R.G. Baraniuk, "Distributed compressed sensing of jointly sparse signals," *Proc. Asilomar Conf. Signals, Sys., and Comp.*, pp. 1537 – 1541, Oct. 2005.

[10] G. Davis, S. Mallat, and M. Avellaneda, "Greedy adaptive approximation," *J. Constr. Approx.*, vol. 13, pp. 57–98, 1997.

[11] D. Leviatan and V. Temlyakov, "Simultaneous approximation by greedy algorithms," *Advances in Computational Mathematics*, vol. 25, no. 1-3, pp. 73–90, July 2006.

[12] S.F. Cotter, B.D. Rao, Kjersti Engan, and K. Kreutz-Delgado, "Sparse solutions to linear inverse problems with multiple measurement vectors," *IEEE Trans. Signal Processing*, vol. 53, no. 7, pp. 2477 – 2488, Jul. 2005.

[13] J. Chen and X. Huo, "Theoretical results on sparse representations of multiple-measurement vectors," *IEEE Trans. Signal Processing*, vol. 54, no. 12, pp. 4634 –4643, dec. 2006.

[14] J.A. Tropp, A.C. Gilbert, and M.J. Strauss, "Simultaneous sparse approximation via greedy pursuit," in *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing (ICASSP)*, mar. 2005, vol. 5, pp. v/721 – v/724 Vol. 5.

[15] J. Mota, J. Xavier, P. Aguiar, and M. Puschel, "Distributed basis pursuit," *IEEE Trans. Signal Processing*, vol. PP, no. 99, pp. 1, 2011.

[16] J. Tropp, A. Gilbert, and M. Strauss, "Algorithms for simultaneous sparse approximation. Part I: Greedy pursuit," *Signal Processing*, vol. 86, no. 3, pp. 572–588, Mar. 2006.

[17] R. Gribonval, H Rauhut, K. Schnass, and P Vandergheynst, "Atoms of all channels, unite! Average case analysis of multi-channel sparse recovery using greedy algorithms," *J. Fourier Anal. and Appl.*, vol. 14, no. 5, pp. 655–687, 2008.

[18] J.A. Tropp and A.C. Gilbert, "Signal recovery from random measurements via orthogonal matching pursuit," *IEEE Trans. Inf. Theory*, vol. 53, no. 12, pp. 4655 –4666, Dec. 2007.

[19] Wei Dai and O. Milenkovic, "Subspace pursuit for compressive sensing signal reconstruction," *IEEE Trans. Inf. Theory*, vol. 55, no. 5, pp. 2230 –2249, May 2009.

[20] D. Needell and J. A. Tropp, "CoSaMP: Iterative signal recovery from incomplete and inaccurate samples," *Appl. and Comp. Harm. Analysis*, vol. 26, pp. 301–321, May 2009.

[21] S. Chatterjee, D. Sundman, M. Vehkaperä, and M. Skoglund, "Projection-based and look ahead strategies for atom selection," *IEEE Trans. Signal Processing*, vol. PP, no. 99, pp. 1, 2011.

[22] David L. Donoho, Yaakov Tsaig, Iddo Drori, and Jean luc Starck, "Sparse solution of underdetermined linear equations by stagewise orthogonal matching pursuit," Tech. Rep., 2006.

[23] H. Huang and A. Makur, "Backtracking-based matching pursuit method for sparse signal reconstruction," *IEEE Signal Processing Letters*, vol. 18, no. 7, pp. 391 –394, july 2011.

[24] D. Sundman, S. Chatterjee, and M. Skoglund, "Look ahead parallel pursuit," in *2011 IEEE Swedish Communication Technologies Workshop (Swe-CTW)*, oct. 2011, pp. 114 –117.

[25] D. Needell and R. Vershynin, "Signal recovery from incomplete and inaccurate measurements via regularized orthogonal matching pursuit," *IEEE Journal of Selected Topics in Signal Processing*, vol. 4, no. 2, pp. 310–316, Apr. 2010.

[26] G. Reeves and M. Gastpar, "A note on optimal support recovery in compressed sensing," *Proc. Asilomar Conf. Signals, Sys., and Comp.*, pp. 1576 –1580, nov. 2009.

[27] E.J. Candes and M.B. Wakin, "An introduction to compressive sampling," *IEEE Signal Processing Magazine*, vol. 25, no. 2, pp. 21 –30, Mar. 2008.