Signal Processing 108 (2015) 628–641

Contents lists available at ScienceDirect

Signal Processing

journal homepage: www.elsevier.com/locate/sigpro

A fast tree-based algorithm for Compressed Sensing with sparse-tree prior[☆]

H.Q. Bui[*], C.N.H. La, M.N. Do

University of Illinois at Urbana-Champaign, Urbana, IL 61801, United States

ARTICLE INFO

Article history:
Received 15 May 2014
Received in revised form
15 October 2014
Accepted 18 October 2014
Available online 29 October 2014

Keywords:
Compressed Sensing
Sparse-tree prior
Tree structure
Tree-based algorithm
Greedy
Orthogonal Matching Pursuit

ABSTRACT

In Compressed Sensing, the *sparse representation property* of an unknown signal in a certain basis has been used as the only prior knowledge for signal reconstruction from a limited number of measurements. Recently, more and more research has focused on model-based recovery algorithms, in which special structures of the unknown signal are exploited in addition to the sparse prior. A popular structure is the sparse-tree structure exhibited in the wavelet transform of piecewise smooth signals and in many practical models. In this paper, a reconstruction algorithm that exploits this sparse-tree prior, the Tree-based Orthogonal Matching Pursuit (TOMP) algorithm, is proposed and studied in detail. Theoretical analyses and empirical experiments show that the proposed algorithm gives reconstruction quality comparable with more sophisticated algorithms, while being much simpler.

© 2014 Elsevier B.V. All rights reserved.

1. Introduction

In Compressed Sensing, the existence of a *sparse representation* of an unknown signal in a certain basis has been used extensively as prior knowledge for signal reconstruction from a limited number of measurements [1–6]. Recently, several recovery algorithms have incorporated special signal structures such as block sparsity or tree sparsity into the reconstruction process [7–11].

In [11,12] the Tree-based Orthogonal Matching Pursuit (TOMP) algorithm was primitively introduced to recover signals with a sparse-tree prior. In this paper, we further investigate TOMP and derive a sufficient condition for successful recovery of sparse-tree signals. Moreover, more experiments are carried out to compare TOMP with more recent tree-based recovery methods. The results show that TOMP provides competitive quality with more sophisticated methods, while being much simpler.

The main contributions of this paper are new results on TOMP, including: (1) a new theoretical analysis of the algorithm's performance, (2) more organized experiments comparing the proposed algorithm with other tree-based reconstruction algorithms and (3) a detailed analysis on computational complexity of TOMP.

The outline of the paper is as follows. First, existing sparse inverse problems and reconstruction algorithms are reviewed in Section 2. After that, the *sparse-tree model* will be presented in Section 3. The proposed TOMP algorithm will be introduced in Section 4, followed by two implementation methods in Section 5. The theoretical analysis of the proposed algorithm will be provided in Section 6. Section 7 will provide some experimental results, followed by the discussion in Section 8. Finally, the paper will be concluded with Section 9.

2. Background

2.1. Sparse inverse problem and Compressed Sensing

For an unknown signal $\boldsymbol{s}$ of length N, suppose that only a limited number of *non-adaptive linear measurements*

[☆] This work was supported in part by the US National Science Foundation under Grant Nos. CCF-0635234 and CCF-0916953.

[*] Corresponding author.

E-mail addresses: huybui1@illinois.edu (H.Q. Bui), hchinhln@gmail.com (C.N.H. La), minhdo@illinois.edu (M.N. Do).

http://dx.doi.org/10.1016/j.sigpro.2014.10.026
0165-1684/© 2014 Elsevier B.V. All rights reserved.

($M \ll N$) can be acquired due to physical constraints:

$$\boldsymbol{\Phi s} = \boldsymbol{b}, \tag{1}$$

where $\boldsymbol{\Phi}$ is a fixed $M \times N$ measurement matrix, and $\boldsymbol{b}$ is a length-M vector that contains the measured data. The inverse problem is to reconstruct $\boldsymbol{s}$ from $\boldsymbol{b}$.

Suppose that $\boldsymbol{s}$ has an expansion in some basis via a fixed $N \times N$ transform matrix $\boldsymbol{W}$ as

$$\boldsymbol{s} = \boldsymbol{Wx}. \tag{2}$$

If only K out of N entries ($K \ll N$) of $\boldsymbol{x}$ are non-zero then $\boldsymbol{x}$ is called a *K-sparse* signal. The number of non-zero coefficients in $\boldsymbol{x}$ is called the *sparsity* of $\boldsymbol{x}$:

$$S(\boldsymbol{x}) = \|\boldsymbol{x}\|_0. \tag{3}$$

Let $\boldsymbol{A} = \boldsymbol{\Phi W}$, then the inverse problem (1) becomes

$$\boldsymbol{Ax} = \boldsymbol{b}. \tag{4}$$

Since $M \ll N$, both (1) and (4) are underdetermined systems. To solve (4), most current methods use the sparse prior and search for the *sparsest solution*:

$$\min_{\boldsymbol{x}} S(\boldsymbol{x}) \quad \text{s.t.} \quad \boldsymbol{Ax} = \boldsymbol{b}. \tag{5}$$

We refer to (5) as the *sparse inverse problem*.

In practice, most signals are not exactly sparse but can be well approximated by sparse signals [13]. In such signals, only $K \ll N$ coefficients have significant values (i.e. greater than some threshold) while the remaining coefficients have very small values; and when these coefficients are sorted in descending order, the coefficient magnitude decays quickly. These signals are called compressible signals. All Compressed Sensing algorithms work with compressible signals.

2.2. Existing sparse reconstruction algorithms

Problem (5) is an NP-hard problem. One approach to solve (5) is to use *greedy search* methods, such as [14,6,15–17], to find the locations of the significant coefficients in $\boldsymbol{x}$. A popular and simple greedy method is Orthogonal Matching Pursuit (OMP) [14,6]. Originally, OMP was developed to find the optimal sparse representation of a signal in a redundant dictionary. Each OMP iteration searches the dictionary for the atom that is most correlated with the residual from the previous iteration and estimates the value of the corresponding coefficient by orthogonally projecting the data onto the whole set of selected atoms. The main limitation of OMP is that each iteration must correctly select an index. Once an index has been selected, it cannot be removed from the selected set. StOMP [17], CoSaMP [15], and Subspace Pursuit [16] improve upon OMP by selecting more than one index at each iteration and including a backtracking or pruning step to refine the selected set.

Besides the greedy approaches, there are many other approaches to solve the sparse inverse problem including *Basic Pursuit* or l_1-minimization [18,3–5], Majorization–Minimization [19–21] and Approximate Message Passing (AMP) [22].

2.3. Model-based reconstruction algorithms

Recently, more research has focused on recovery sparse signals that embed additional structures. Intuitively, taking into account these structures make the recovery process easier, in terms of less required number of measurements or faster recovery time.

In 2005, [10,11] independently proposed to exploit the sparse-tree structure in the wavelet transform of piecewise smooth signals. In [11,12] TOMP was introduced to recover signals with a sparse-tree prior. The same structure was also exploited in the Tree-based Majorization–Minimization (TMM) algorithm [23] which is an extension of Majorization–Minimization (MM) approach.

In 2010, Baraniuk et al. [7] formulated the theory of *Model-based Compressed Sensing* for the recovery of sparse and compressible signals that have sparse-tree or sparse-block structures. These theories proved that it is advantageous to use special signal structures as additional priors for signal recovery. In particular, the authors proposed two methods to recover sparse-tree and sparse-block signals based on CoSaMP [15] and Iterative Hard Thresholding (IHT)[24] algorithms.

The tree-structure of wavelet coefficients was also incorporated into Bayesian-based methods, such as [8,9]. In these papers, the tree-structure was embedded in a statistical model and different Bayesian inference methods were used for the reconstruction, such as Variational Bayesian [9] or Markov Chain Monte Carlo [8]. Although these methods can give high quality results, the main disadvantage is their expensive computation, as will be discussed later in Section 6.3. Recently, Som and Schniter [25] proposed a new message-passing-based method for compressive imaging that takes into account the Markov-tree prior.

3. The sparse-tree model of signals

3.1. The sparse-tree model

Many signals that we encounter in practice can be modeled as trees. In this paper, we consider the following sparse-tree model.

Definition 3.1 (*Sparse-tree model*). A signal $\boldsymbol{x}$ is said to conform to the sparse-tree model if it satisfies the following properties:

1. $\boldsymbol{x}$ is sparse or compressible.
2. The coefficients of $\boldsymbol{x}$ can be organized into one or several trees.
3. The non-zero or significant coefficients of $\boldsymbol{x}$ are connected together in rooted sub-trees of the trees.
4. When going from the roots to the leaves of the trees, the maximum magnitude of the coefficients at each level will be decreasing.

This sparse-tree model might seem very restrictive at the first glance. However, it is an effective model for many real world problems and signals. In the following section, we provide two examples in practice where this model can be applied.

3.2. Examples of the sparse-tree model

3.2.1. Example 1

An interesting problem in which the sparse-tree model can be applied is modeling the spreading of a disease in a population. In this problem, each person in that population is represented by a vertex of the tree and the sources of the disease are represented by the roots. Suppose that we do some random group testing on that population and we are interested in discovering people who have acquired the disease and the seriousness of the disease at each of them. In this case, $\boldsymbol{x}$ is the vector of coefficients where each coefficient represents the seriousness of the disease demonstrating at each person. Intuitively, the seriousness is decreasing with distance from the sources. A problem similar to this case is the passing of a rumor in a population where the roots are the sources of the rumor and the coefficients represent the confidence of the rumor.

3.2.2. Example 2

Another example of the sparse-tree model is the relationship among the wavelet coefficients of a piecewise smooth signal. Consider (2) when the transform $\boldsymbol{W}$ is an L-level 1-dimensional wavelet transform. In that case, the entries of $\boldsymbol{x}$ consist of

$$\boldsymbol{x} = \left\{ \{x_{0,p}^{(s)}\}_{1 \le p \le N/2^L}, \quad \{x_{l,p}^{(w)}\}_{1 \le l \le L,\ 1 \le p \le N/2^{L-l+1}} \right\},$$

where $x_{0,p}^{(s)}$ are the scaling coefficients and $x_{l,p}^{(w)}$ are the wavelet coefficients. In this notation, l is the scale of a wavelet coefficient, where 1 is the coarsest scale and L is the finest scale, and p is the shift of a coefficient in a level. The elements of $\boldsymbol{x}$ can be arranged into binary trees where the roots are the wavelet coefficients at the coarsest level, as shown in Fig. 1. Each wavelet coefficient $x_{l,p}^{(w)}$ has two children $x_{l+1,2p-1}^{(w)}$ and $x_{l+1,2p}^{(w)}$. The entries in $\boldsymbol{x}$ can be specified either in the vector form x_i or in the tree-based form $x_{l,p}$.

By examining the wavelet transform $\boldsymbol{x}$ of a piecewise-smooth signal, three distinguishing properties can be observed:

P1 *Vector $\boldsymbol{x}$ is sparse or compressible; i.e. only a few entries in $\boldsymbol{x}$ are non-zero or significant.*
P2 *The non-zero or significant entries of $\boldsymbol{x}$ are likely to be connected in a tree structure.*
P3 *The wavelet coefficients tend to decay across scales from coarse to fine scales [26].*

Properties **P2** and **P3** are important additional priors that have not been considered adequately in recovery algorithms. These properties hold because each discontinuity of the signal generates a set of large wavelet coefficients in a “cone of influence” [26], which is also referred to as the wavelet footprint [27]. In particular, *if a coefficient is non-zero or significant then its ancestors are more likely to be non-zero or significant*. Therefore, the significant coefficients of $\boldsymbol{x}$ form the rooted sub-trees as illustrated in Fig. 1.

3.3. Descendant and ancestor

In this section, we present several important concepts that will facilitate the development of our proposed algorithm. These concepts were introduced in the context of wavelet tree model [28] and are recalled here for completeness. Consider a signal $\boldsymbol{x}$ whose coefficients can be organized into one or several trees.

A *descendant* of a node x_i is a node x_j that can be reached from node x_i by following the children nodes. For example, in Fig. 1, all descendants of node $x_{1,2}^{(w)}$ are $\{x_{2,3}^{(w)}, x_{2,4}^{(w)}, x_{3,5}^{(w)}, x_{3,6}^{(w)}, x_{3,7}^{(w)}, x_{3,8}^{(w)}\}$.

An *ancestor* of a node x_i is a node x_j that can be reached from x_i by following the parent nodes. The *history set* of a node x_i is the set of ancestors of x_i up to its root. For

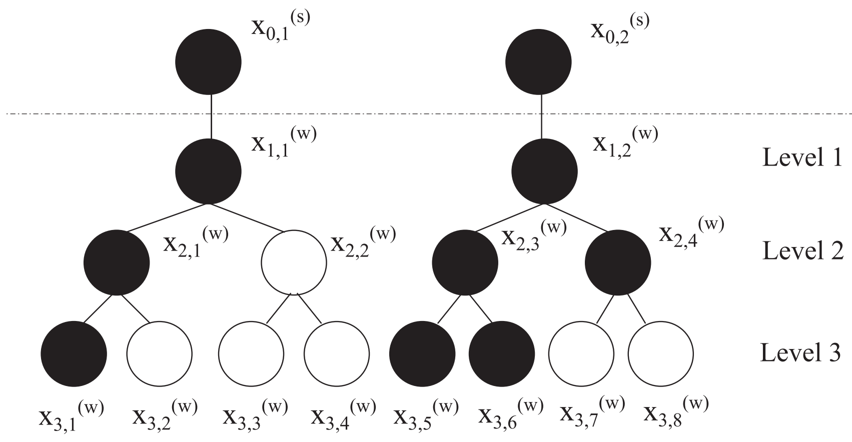


Fig. 1. The tree structure of the coefficient vector resulted from a 3-level wavelet decomposition of a length-16 signal, where black nodes represent significant coefficients and white nodes represent insignificant coefficients.

example, in Fig. 1, $x_{1,1}^{(w)}$ is an ancestor of node $x_{3,1}^{(w)}$ and the history set of node $x_{3,1}^{(w)}$ is $\{x_{2,1}^{(w)}, x_{1,1}^{(w)}\}$.

4. Tree-based orthogonal matching pursuit algorithm

This section describes in detail our proposed *Tree-based Orthogonal Matching Pursuit (TOMP)* algorithm for solving the sparse inverse problem for signals with the sparse-tree characteristic. Let $\boldsymbol{x}$ be a K-sparse signal in $\mathbb{R}^N$ which satisfies the sparse-tree model. Let $\boldsymbol{A}$ be a $M \times N$ random measurement matrix and $\boldsymbol{b} = \boldsymbol{A}\boldsymbol{x}$ be the vector of measurements. The ith column of $\boldsymbol{A}$ is denoted by $\boldsymbol{a}_i$.

Let Λ_k be the set of selected indices after the kth iteration. The initial set Λ_0 consists of indices of entries which are expected to be significant at the roots of the trees. For example, in the wavelet transform, all scaling coefficients are likely to be significant.

Algorithm 4.1 (*TOMP for signal conforming to the sparse-tree model*). INPUT

- $M \times N$ measurement matrix $\boldsymbol{A}$.
- Length-M vector $\boldsymbol{b}$ of measurements.
- Relaxation parameter $\alpha \in [0, 1]$.
- Downward extending parameter d.
- Index array P that maps each node index to the index of its parent.
- Index array Q that contain the indices of the children of each node.

OUTPUT

- Reconstructed vector $\hat{\boldsymbol{x}}$ of length N which has a sparse-tree structure.

PROCEDURE

1. Initialize the set Λ_0 with indices of the entries at the roots of the trees. Initialize the residual as

$$\boldsymbol{r}_0 = \boldsymbol{b} - P_{\text{span}\{\boldsymbol{a}_i : i \in \Lambda_0\}}\boldsymbol{b}. \tag{6}$$

Set $k=0$ and *stop=false*

2. While *stop = = false*:
   (a) $k = k+1$.
   (b) Form the *candidate set* $\boldsymbol{C}_k$ that contains the indices of the descendants of selected nodes within d levels.

$$\boldsymbol{C}_k = \bigcup_{i \in \Lambda_{k-1}} D_d(i), \tag{7}$$

   where $D_d(i)$ denotes the index set of all descendants of nodes $\boldsymbol{x}_i$ within d levels. Hence d is named the *downward extending parameter*.
   (c) Form the *finalist set* $\boldsymbol{F}_k$ as

$$\boldsymbol{F}_k = \{i \in \boldsymbol{C}_k \text{ s.t. } |\langle \boldsymbol{r}_{k-1}, \boldsymbol{a}_i \rangle| \geq \alpha \max_{j \in \boldsymbol{C}_k} |\langle \boldsymbol{r}_{k-1}, \boldsymbol{a}_j \rangle|\}, \tag{8}$$

   where α is a given relaxation parameter.
   (d) Select the index i_k from the finalist set $\boldsymbol{F}_k$ such that

$$i_k = \arg\min_{i \in \boldsymbol{F}_k} \|\boldsymbol{b} - P_{\text{span}\{\boldsymbol{a}_j : j \in \Lambda_{k-1} \cup \boldsymbol{H}(i)\}}\boldsymbol{b}\|_2, \tag{9}$$

   where $\boldsymbol{H}(i)$ denotes the index set of the ancestors of node $\boldsymbol{x}_i$ up to the root.
   (e) Augment the index set as

$$\Lambda_k = \Lambda_{k-1} \cup \boldsymbol{H}(i_k), \tag{10}$$

   where $\boldsymbol{H}(i_k)$ is the index set of the ancestors of node $\boldsymbol{x}_{i_k}$ up to the roots.
   (f) Update the residual

$$\boldsymbol{r}_k = \boldsymbol{b} - P_{\text{span}\{\boldsymbol{a}_i : i \in \Lambda_k\}}\boldsymbol{b}. \tag{11}$$

   (g) If $\|\boldsymbol{r}_k\|_2^2 \leq \varepsilon$, a selected threshold, or the number of selected columns exceed a certain limit, e.g. $M/2$, set *stop=true*
3. Non-zero coefficients of the estimated signal $\hat{\boldsymbol{x}}$, indexed by Λ_k, are the solution of

$$\boldsymbol{A}_{\Lambda_k}\hat{\boldsymbol{x}}_{\Lambda_k} = \boldsymbol{b} - \boldsymbol{r}_k, \tag{12}$$

where $\boldsymbol{A}_{\Lambda_k}$ consists of columns of $\boldsymbol{A}$ indexed by Λ_k.

5. Implementation details

In this section, an implementation of Algorithm 4.1 based on Gram–Schmidt orthogonalization process is provided. In this implementation, whenever a new column $\boldsymbol{a}_i$ is selected, i is stored into the selected set Λ_k. At the same time, $\boldsymbol{a}_i$ is orthonormalized with respect to all of the previously selected columns and then stored in the set of orthonomalized selected columns $\boldsymbol{U}_k$, named the *Gram–Schmidt selected set*.

After Eqs. (6)–(8), for each node $i \in \boldsymbol{F}_k$, a corresponding sub-tree containing that node and its ancestors is formed. Each column in $\{\boldsymbol{a}_{\boldsymbol{H}(i)}\}$ is orthonormalized against $\boldsymbol{U}_{k-1}$ and the remaining columns in $\{\boldsymbol{a}_{\boldsymbol{H}(i) \setminus \Lambda_{k-1}}\}$ to form the set of orthonormalized columns $\{\boldsymbol{a}^{\perp}_{\boldsymbol{H}(i)}\}$. This step is performed by using the Gram–Schmidt process.

Then, the current residual is projected onto each sub-tree and the resulting residuals are recorded and compared.

$$\boldsymbol{r}_{temp} = \boldsymbol{b} - P_{\text{span}\{\boldsymbol{a}_j : j \in \Lambda_{k-1} \cup \boldsymbol{H}(i)\}}\boldsymbol{b}$$

$$= \boldsymbol{r}_{k-1} - \sum_{j \in \boldsymbol{H}(i) \setminus \Lambda_{k-1}} \langle \boldsymbol{r}_{k-1}, \boldsymbol{a}_j^{\perp} \rangle \boldsymbol{a}_j^{\perp}.$$

The sub-tree that gives the smallest residual is selected.

$$i_k = \arg\min_{i \in \boldsymbol{F}_k} \|\boldsymbol{r}_{temp}\|. \tag{13}$$

This gives the solution of (9). The selected set Λ_k and the new residual $\boldsymbol{r}_k$ are updated through (10) and (11).

The selected set $\boldsymbol{U}_k$ is updated by adding the selected orthonormalized sub-tree:

$$\boldsymbol{U}_k = \boldsymbol{U}_{k-1} \cup \{\boldsymbol{a}^{\perp}_{\boldsymbol{H}(i_k) \setminus \Lambda_{k-1}}\}. \tag{14}$$

The algorithm terminates when the stopping rules are satisfied. By caching the set of orthonomalized selected columns, the computational cost at each iteration is significantly reduced.

6. Analysis of TOMP

In this section, we provide a theoretical analysis on the performance of the proposed algorithm. First, we derive a sufficient condition for the recovery of sparse tree signal using TOMP in Section 6.1. Next, we analyze the effect of the parameters on the algorithm in Section 6.2. Finally, Section 6.3 analyzes the computational complexity of the proposed algorithm and compares it with other methods.

6.1. Reconstruction condition

To set the context for our analysis, we recall the definition of the *cumulative coherence function* in [29].

Definition 6.1 (*Cumulative coherence function*). Given a dictionary $\mathcal{D}$ with atoms $\boldsymbol{\phi}_\lambda$. Let Ω be the index set of all atoms in $\mathcal{D}$.

The cumulative coherence function is defined as

$$\mu_1^{(\Omega)}(K) = \max_{\Lambda \subset \Omega, |\Lambda| = K} \max_{\boldsymbol{\psi} \in \Omega \setminus \Lambda} \sum_{\lambda \in \Lambda} |\langle \boldsymbol{\psi}, \boldsymbol{\phi}_\lambda \rangle|, \tag{15}$$

where Λ is a subset of Ω and $\boldsymbol{\psi}$ is an atom in the remaining set $\Omega \setminus \Lambda$.

The condition for reconstruction, as stated in Theorem 6.4, is the same as the condition for Orthogonal Matching Pursuit [29]. However, in this paper, the condition is derived for the recovery of sparse-tree signals using our proposed TOMP algorithm. For completeness, the result of [29] is restated here.

Theorem 6.2 (*Theorem 3.5 in [29]*). *Suppose that $\mu_1^{(\Omega)}$ is the cumulative coherence function of $\mathcal{D}$ and Ω is the index set of all atoms in $\mathcal{D}$. OMP will recover the sparsest K-term representation of the signal whenever*

$$\mu_1^{(\Omega)}(K-1) + \mu_1^{(\Omega)}(K) < 1. \tag{16}$$

Before stating our results, we would like to introduce some basic notations that are required for later development. Let $\boldsymbol{x}$ be the vector to be recovered and assume that $\boldsymbol{x}$ satisfies the sparse-tree model. Let Λ be the index set of non-zero coefficients of $\boldsymbol{x}$, $|\Lambda| = K$. The nodes on the tree that have indices in the set Λ are called the optimal nodes. The data vector $\boldsymbol{b}$ can be represented as

$$\boldsymbol{b} = \sum_{i \in \Lambda} x_i \boldsymbol{a}_i.$$

If after the $(k-1)$th iteration, the algorithm has correctly found the indices of some non-zero coefficients in $\boldsymbol{x}$, i.e. $\Lambda_{k-1} \subset \Lambda$, then the residual can be represented as

$$\begin{aligned}
\boldsymbol{r}_{k-1} &= \boldsymbol{b} - P_{\text{span}\{\boldsymbol{a}_i : i \in \Lambda_{k-1}\}} \boldsymbol{b} \\
&= \sum_{i \in \Lambda} x_i \boldsymbol{a}_i - \sum_{i \in \Lambda_{k-1}} \beta_i^{(k-1)} \boldsymbol{a}_i \\
&= \sum_{i \in \Lambda_{k-1}} \left(x_i - \beta_i^{(k-1)} \right) \boldsymbol{a}_i + \sum_{i \in \Lambda \setminus \Lambda_{k-1}} x_i \boldsymbol{a}_i \\
&= \sum_{i \in \Lambda} c_i^{(k-1)} \boldsymbol{a}_i,
\end{aligned}$$

where

$$c_i^{(k-1)} = \begin{cases} x_i - \beta_i^{(k-1)}, & i \in \Lambda_{k-1} \\ x_i, & i \notin \Lambda_{k-1} \end{cases} \tag{17}$$

Also, note that

$$|\langle \boldsymbol{r}_{k-1}, \boldsymbol{a}_i \rangle| = 0 \quad \forall i \in \Lambda_{k-1}$$

since $\boldsymbol{r}_{k-1}$ is the residual after orthogonally projecting $\boldsymbol{b}$ onto the space spanned by the columns $\{\boldsymbol{a}_i\}_{i \in \Lambda_{k-1}}$.

In the following analysis, the levels of the nodes on the tree are increasing from roots to leaves, with 1 being the level of the roots, as described in Fig. 1.

Lemma 6.3. *Assume that the signal $\boldsymbol{x}$ conforms to the sparse-tree model. Let $\tilde{\boldsymbol{C}}_k = \Lambda_{k-1} \cup \boldsymbol{C}_k$, where $\boldsymbol{C}_k$ contains all the nodes descending from the nodes in Λ_{k-1} up to d levels. Then*

$$\max_{j \in \Lambda} |c_j^{(k-1)}| = \max_{j \in \Lambda \cap \tilde{\boldsymbol{C}}_k} |c_j^{(k-1)}| \tag{18}$$

or equivalently,

$$\max_{j \in \Lambda \setminus \tilde{\boldsymbol{C}}_k} |c_j^{(k-1)}| \le \max_{j \in \Lambda \cap \tilde{\boldsymbol{C}}_k} |c_j^{(k-1)}| \tag{19}$$

Proof. Suppose that $\boldsymbol{x}$ is a signal which conforms to the sparse-tree model defined in Section 3. Let L_n be the set of indices at level n of the tree. Then

$$\max_{i \in L_n} |x_i| = \max_{i \in \Lambda \cap L_n} |x_i|, \tag{20}$$

since $|x_i| = 0 \ \forall i \notin \Lambda$.

At the kth iteration, the candidate set $\boldsymbol{C}_k$ contains nodes extended from Λ_{k-1}. Suppose that $\Lambda_{k-1} \subset \Lambda$, i.e. the algorithm has selected correctly nodes in Λ. If $\boldsymbol{r}_{k-1} \neq 0$, there is at least one node in $\boldsymbol{C}_k$ that is in Λ, or $\Lambda \cap \boldsymbol{C}_k \neq \varnothing$, since all non-zero coefficients in $\boldsymbol{x}$ are connected in a rooted tree.

Note that:

$$\left(\Lambda \setminus \tilde{\boldsymbol{C}}_k \right) \cap \Lambda_{k-1} = \left(\Lambda \setminus (\Lambda_{k-1} \cup \boldsymbol{C}_k) \right) \cap \Lambda_{k-1} = \varnothing$$

and

$$(\Lambda \cap \boldsymbol{C}_k) \cap \Lambda_{k-1} = \Lambda_{k-1} \cap \boldsymbol{C}_k = \varnothing,$$

since $\Lambda_{k-1} \subset \Lambda$ and $\Lambda_{k-1} \cap \boldsymbol{C}_k = \varnothing$.

From (17), $|c_j^{(k-1)}| = |x_j|$ for $j \in \Lambda \setminus \tilde{\boldsymbol{C}}_k$ and for $j \in \Lambda \cap \boldsymbol{C}_k$.

Let n_0 be the minimum level of nodes in $\Lambda \setminus \tilde{C}_k$, then

$$\Lambda \setminus \tilde{\boldsymbol{C}}_k = \Lambda \cap \left\{ \bigcup_{n \ge n_0} L_n \right\} = \bigcup_{n \ge n_0} \{\Lambda \cap L_n\} \tag{21}$$

First, consider the left-hand side of (19)

$$\begin{aligned}
\max_{j \in \Lambda \setminus \tilde{\boldsymbol{C}}_k} |c_j^{(k-1)}| &= \max_{j \in \Lambda \setminus \tilde{\boldsymbol{C}}_k} |x_j| \\
&= \max_{j \in \cup_{n \ge n_0} \{\Lambda \cap L_n\}} |x_j| \\
&= \max_{j \in \cup_{n \ge n_0} L_n} |x_j| \\
&= \max_{j \in L_{n_0}} |x_j|,
\end{aligned}$$

where the decaying of coefficients across levels has been taken into account.

Next, consider the right-hand side of (19)

$$\begin{aligned}\max_{j\in\Lambda\cap\tilde{\boldsymbol{C}}_k}|c_j^{(k-1)}| &\ge \max_{j\in\Lambda\cap\boldsymbol{C}_k}|c_j^{(k-1)}|\\ &= \max_{j\in\Lambda\cap\boldsymbol{C}_k}|x_j|\\ &\ge \max_{j\in L_{n_0-1}}|x_j|\\ &\ge \max_{j\in L_{n_0}}|x_j|. \qquad \square\end{aligned}$$

Theorem 6.4. *Let* $\boldsymbol{x}$ *be a vector conforming to the sparse-tree model with* K *non-zero elements. TOMP will recover* $\boldsymbol{x}$ *correctly if the sensing matrix* $\boldsymbol{A}$ *satisfies*

$$\alpha\mu_1^{(\Omega)}(K)+\mu_1^{(\Omega)}(K-1)<1, \tag{22}$$

where $\alpha\in[0,1]$ *is the relaxation parameter and* Ω *is the index set of all columns of matrix* $\boldsymbol{A}$.

Proof. Suppose that $\boldsymbol{x}$ conform to the sparse-tree model and $\boldsymbol{x}$ is exactly sparse with K non-zero elements. Consider the case $\alpha=1$, in which the finalist set $\boldsymbol{F}_k$ contains only one index.

At the kth iteration, the algorithm will find the correct indices in Λ if

$$\max_{j\in\boldsymbol{C}_k\cap\Lambda}|\langle\boldsymbol{r}_{k-1},\boldsymbol{a}_j\rangle| > \max_{j\in\boldsymbol{C}_k\setminus\Lambda}|\langle\boldsymbol{r}_{k-1}.\boldsymbol{a}_j\rangle| \tag{23}$$

Since we have $|\langle\boldsymbol{r}_{k-1},\boldsymbol{a}_i\rangle|=0\ \forall i\in\Lambda_{k-1}$, condition (23) is the same as

$$\max_{j\in\tilde{\boldsymbol{C}}_k\cap\Lambda}|\langle\boldsymbol{r}_{k-1},\boldsymbol{a}_j\rangle| > \max_{j\in\tilde{\boldsymbol{C}}_k\setminus\Lambda}|\langle\boldsymbol{r}_{k-1},\boldsymbol{a}_j\rangle|, \tag{24}$$

where $\boldsymbol{C}_k$ is expanded to $\tilde{\boldsymbol{C}}_k=\boldsymbol{C}_k\cup\Lambda_{k-1}$.

Consider the right-hand side of (24):

$$\begin{aligned}\text{RHS} &= \max_{j\in\tilde{\boldsymbol{C}}_k\setminus\Lambda}\Big|\sum_{i\in\Lambda}c_i^{(k-1)}\langle\boldsymbol{a}_i,\boldsymbol{a}_j\rangle\Big|\\ &\le \max_{i\in\Lambda}|c_i^{(k-1)}|\max_{j\in\tilde{\boldsymbol{C}}_k\setminus\Lambda}\sum_{i\in\Lambda}|\langle\boldsymbol{a}_i,\boldsymbol{a}j\rangle|\\ &\le \max_{i\in\Lambda}|c_i^{(k-1)}|\mu_1^{(\Omega)}(K).\end{aligned}$$

Next, consider the left-hand side of (23):

$$\begin{aligned}\text{LHS} &= \max_{j\in\tilde{\boldsymbol{C}}_k\cap\Lambda}\left|\sum_{i\in\Lambda}c_i^{(k-1)}\langle\boldsymbol{a}_i,\boldsymbol{a}_j\rangle\right|\\ &= \max_{j\in\tilde{\boldsymbol{C}}_k\cap\Lambda}\left|c_j^{(k-1)}+\sum_{i\in\Lambda\setminus\{j\}}c_i^{(k-1)}\langle\boldsymbol{a}_i,\boldsymbol{a}_j\rangle\right|\\ &\ge \max_{j\in\tilde{\boldsymbol{C}}_k\cap\Lambda}\left[|c_j^{(k-1)}|-\sum_{i\in\Lambda\setminus\{j\}}|c_i^{(k-1)}||\langle\boldsymbol{a}_i,\boldsymbol{a}_j\rangle|\right]\\ &\ge \max_{j\in\tilde{\boldsymbol{C}}_k\cap\Lambda}|c_j^{(k-1)}|-\max_{i\in\Lambda}|c_i^{(k-1)}|\max_{j\in\tilde{\boldsymbol{C}}_k\cap\Lambda}\sum_{i\in\Lambda\setminus\{j\}}|\langle\boldsymbol{a}_i,\boldsymbol{a}_j\rangle|\\ &\ge \max_{j\in\tilde{\boldsymbol{C}}_k\cap\Lambda}|c_j^{(k-1)}|-\max_{i\in\Lambda}|c_i^{(k-1)}|\max_{j\in\Lambda}\sum_{i\in\Lambda\setminus\{j\}}|\langle\boldsymbol{a}_i,\boldsymbol{a}_j\rangle|\\ &\ge \max_{j\in\tilde{\boldsymbol{C}}_k\cap\Lambda}|c_j^{(k-1)}|-\max_{j\in\Lambda}|c_j^{(k-1)}|\mu_1^{(\Omega)}(K-1)\\ &= \max_{j\in\Lambda}|c_j^{(k-1)}|-\max_{j\in\Lambda}|c_j^{(k-1)}|\mu_1^{(\Omega)}(K-1).\end{aligned}$$

The last step follows from Lemma 6.3. Thus, the condition such that the algorithm determines the column(s) correctly at kth iteration is

$$\mu_1^{(\Omega)}(K)+\mu_1^{(\Omega)}(K-1)<1. \tag{25}$$

In the general case, when $\alpha<1$, there will be more than one node in the finalist set $\boldsymbol{F}_k$. Following the same steps as in the above proof, the condition such that after (8) step, there is at least one correct node in $\boldsymbol{F}_k$, is

$$\alpha\mu_1^{(\Omega)}(K)+\mu_1^{(\Omega)}(K-1)<1. \tag{26}$$

Since the correct nodes must minimize the residual, after the final selection step (9), i_k must be in Λ. Given that the signal follows the sparse-tree model, the index set $\boldsymbol{H}(i_k)$ contains indices of the optimal nodes. Consequently, TOMP will correctly recover the signal. □

6.2. *Effect of parameter variations*

Since TOMP selects entries by expanding a set of selection trees, the final selected set is a set of connected sparse-trees. Moreover, only tree branches that lead to the smallest residual via orthogonal projection are selected at each iteration.

The parameters $d\in\mathbb{Z}, d\ge 1$ and $\alpha\in[0,1]$ are the tuning parameters for TOMP. Larger d leads to larger *candidate sets*, which allows us to reach further down significant coefficients of $\boldsymbol{x}$, but at the cost of more computation per iteration. On the other hand, small d helps to reduced computational cost at each iteration by limiting the number of candidates being considered but the finest levels of the tree might not be reached. If the signal strictly conforms to the proposed sparse-tree model, especially property 4 in Definition 3.1, the finest level coefficients will have small enough magnitudes to have significant effect on the reconstruction quality. Otherwise, the reconstruction quality will be affected if d is too small. Thus, by varying d, one can control the trade-off between computational cost and robustness of the algorithm.

The relaxation parameter α allows further restriction of the search space to the finalist set by a fast evaluation of the inner products in (8) instead of a costly evaluation of the residual norms in (9). Smaller values of α lead to bigger finalist sets, which means more accurate selection, but also at the cost of increased computation.

Some special cases for d are:

- $d=\infty$ means the search space contains every node.
- $d=1$ means only one new node, which is directly connected to the already selected set, is selected at each iteration. In this case the selection step (9) of TOMP can be achieved via evaluating inner products with residual $\boldsymbol{r}_{k-1}$.

Similarly, the special cases of α are:

- $\alpha=0$ leads to an exhaustive search of all possible history sets within the candidate set to determine the one leading to smallest residual.
- In general $\alpha=1$ means only one finalist is selected at each iteration. In this case, if $d=\infty$, TOMP is similar to OMP except that TOMP selects a whole set $\boldsymbol{H}(i_k)$ rather

than only a single i_k. This selection approach ensures that the recovered signal will have the tree structure. If the signal satisfies our assumption **P2**, this modification leads to the correct reconstruction, since coefficients in $\boldsymbol{H}(i)$ are significant whenever coefficient x_i is significant.

6.3. Computational complexity analysis

In this section, the computational complexity of TOMP, in terms of the number of multiplications, will be analyzed and compared with several algorithms, including OMP [14,6], ModelCS (model-based CoSaMP) [7] and TSWCS (Tree-Structure Wavelet Compressed Sensing) [8]. Since the number of iterations is different for each method, the average computational cost per iteration will be computed and compared. The results are summarized in Table 1.

The comparison that we provide in this section is for the recovery of a signal that has the sparse-tree structure in the wavelet domain. Let N be the length of the signal, K be the number of non-zero coefficients and M be the number of measurements. Since the mappings between the nodes on the tree and the indices are one-to-one, the terms *nodes* and *indices* are used interchangeably in the following analysis.

6.3.1. Complexity analysis for TOMP

Assume that the signal $\boldsymbol{x}$ to be recovered is an L-level wavelet decomposition of a length-N signal $\boldsymbol{s}$. Let N_0 be the number of scaling coefficients of $\boldsymbol{x}$. Then $N_0 = N/2^L$.

1. Initialization step: In the initialization phase, all N_0 columns of A which correspond to the scaling coefficients are selected. Let Λ_0 be the initial set of selected indices. The measurement vector is projected onto the space spanned by these columns. In the Gram–Schmidt implementation, these columns are orthonormalized with each other, which has the complexity of $\mathcal{O}\left(MN_0^2\right)$ multiplications. Consequently, the complexity for the initialization step is $\mathcal{O}\left(MN_0^2\right)$.

Table 1
Average computational complexity per iteration of the algorithms, in terms of number of multiplications, to recover a length-N, K-sparse signal with sparse-tree structure in the wavelet domain from M random measurements. In ModelCS, $\mathcal{C}$ is the cost of the tree-approximation algorithm being used. In TOMP, L is the number of levels and d is the downward extending parameter.

Algorithm	Average number of multiplications per iteration
OMP	$\mathcal{O}(MN+MK/2)$
ModelCS	$\mathcal{O}(MN+\mathcal{C})$
TSWCS	$\mathcal{O}(MN^2)$
TOMP	$\mathcal{O}\left(\frac{MN}{2^{L-d}}\right)$

Average computational complexity per iteration of the algorithms, in terms of number of multiplications, to recover a length-N, K-sparse signal with sparse-tree structure in the wavelet domain from M random measurements. In ModelCS, $\mathcal{C}$ is the cost of the tree-approximation algorithm being used. In TOMP, L is the number of levels and d is the downward extending parameter.

2. The first iteration: In the first iteration of the algorithm, the candidate set C_1 contains the indices of the nodes in the first d-level of the wavelet trees. For N_0 binary wavelet trees, C_1 has $N_0(2^d-1)$ candidates.

   To find the finalist set F_1, $N_0(2^d-1)$ inner products between the residual and the columns of $\boldsymbol{A}$ with indices in C_1 must be computed. This step has the complexity of $\mathcal{O}\left(MN_0(2^d-1)\right) \approx \mathcal{O}\left(MN_0(2^d)\right)$ multiplications.

   For each finalist in F_1, the algorithm estimates the residual that would be formed if the node and its ancestors are selected. To do that, the corresponding columns of $\boldsymbol{A}$ are orthonormalized against the previously selected columns with indices in Λ_0 and against each other. In TOMP, the candidate set is formed by extending downward d levels from the selected nodes. Thus, each finalist has at most $d-1$ ancestors that have not been selected from the previous iterations. The maximum complexity for this orthonormalization step is $2M\sum_{j=0}^{d-1} N_0+j \approx \mathcal{O}\left(MN_0d+Md^2\right)$.

   The total complexity for the first iteration is thus

$$\mathcal{O}\left(MN_0 2^d\right)+\mathcal{O}\left(\left|\boldsymbol{F}_0\right|(MN_0d+Md^2)\right),$$

   where $|\boldsymbol{F}_0|$ denotes the cardinality of the finalist set $\boldsymbol{F}_0$. Typically, after the thresholding step in (8), there are only several finalists in the set $\boldsymbol{F}_0$. Consequently, the complexity of the first iteration is $\mathcal{O}\left(MN_0 2^d\right)$.
3. From second iteration onward: From the second iteration onward, at each iteration, at most d nodes from the candidate set are selected and their descendants are added to the candidate set. Thus, the size of the candidate set increases by a small constant at each iteration. The complexity at each following iteration can be approximated by that of the first iteration, which is $\mathcal{O}\left(MN_0 2^d\right)$.

With this estimation, the complexity of k iterations of the algorithm would be $\mathcal{O}\left(k2^dMN_0\right)$. The total complexity for TOMP is $\mathcal{O}\left(MN_0^2\right)+\mathcal{O}\left(kMN_0 2^d\right)$.

Since TOMP selects many indices at each iteration, the number of iterations k for TOMP is less than the number of non-zero coefficients K in signal x. Moreover, for most practical applications of the tree model, the number of roots N_0 is small compared to the number of nodes on the tree. As a result, the total complexity of TOMP is bounded by $\mathcal{O}\left(KMN_0 2^d\right)$. Finally, the average complexity per iteration for TOMP is approximately $\mathcal{O}\left(MN_0 2^d\right) \approx \mathcal{O}\left(MN/2^{L-d}\right)$.

6.3.2. Complexity analysis for OMP, modelCS and TSWCS

In OMP, at the nth iteration, $N-n+1$ inner products must be computed between r_{n-1} and the remaining columns. After that, the selected column is orthonormalized against all $n-1$ columns that have been selected from the previous iterations. OMP stops after K iterations and the total complexity is $\mathcal{O}\left(KMN+MK^2/2\right)$ multiplications. Thus, the average complexity per iteration for OMP is $\mathcal{O}(MN+MK/2)$ multiplications.

CoSaMP improves over OMP by selecting many columns at each iteration and refining them later, which requires a smaller number of iterations. ModelCS is developed from CoSaMP by applying a tree approximation algorithm during the pruning step to maintain the tree structure on the selected columns. As a result, ModelCS has a complexity of OMP plus the complexity of the tree-approximation step. Let $\mathcal{C}$ be the complexity of the tree-approximation algorithm, the total complexity of ModelCS for k iterations is $\mathcal{O}(kMN + Mk^2/2 + k\mathcal{C})$ multiplications. The complexity per iteration is $\mathcal{O}(MN + Mk + \mathcal{C}) \approx \mathcal{O}\,(MN + \mathcal{C})$ multiplications.

TSWCS uses Gibbs sampling and Markov Chain Monte Carlo to infer the posterior distribution of each element in the unknown signal. At each iteration, TSWCS needs to sample each element of the signal sequentially, and thus the total cost of TSWCS is significantly higher than OMP. The computational complexity of TSWCS is approximately $\mathcal{O}\left(kMN^2\right)$ multiplications, where k is the number of iterations. The average complexity per iteration for TSWCS is $\mathcal{O}\left(MN^2\right)$ multiplications.

7. Experimental results

In this section, three main experiments are presented. First, to demonstrate the correctness of the recovery of tree structure, a test signal is recovered using different methods: OMP [14,6], CoSaMP [15], ModelCS (model-based CoSaMP) [7], TSWCS (Tree-Structure Wavelet Compressed Sensing) [8], and our proposed TOMP. Second, the average reconstruction signal-to-noise ratios (SNRs) of different methods are compared at different numbers of measurements on a random piecewise smooth signal. The execution times are also compared. Finally, the *phase diagram* [17] of our proposed method is generated and presented to show the phase transition of the algorithm under different regimes.

We obtained the TSWCS and ModelCS code from the corresponding authors. For a fair comparison between different methods, we modified the ModelCS code to use MATLAB Wavelet Toolbox instead of Rice Wavelet Toolbox as in the original code. This modification allows us to run all algorithms on the same input. Moreover, ModelCS requires an estimation of the signal sparsity as input. In our experiments, we run ModelCS multiple times with varying values and report the results which have maximum SNR. We vary the sparsity from 10 percent the length of the signal to 50 percent the number of measurements.

The reconstruction quality is measured by using the reconstruction signal-to-noise ratio (SNR), defined as

$$\mathrm{SNR} = 10\,\log_{10}\left(\frac{\sigma_{\boldsymbol{x}}^2}{\mathrm{MSE}(\boldsymbol{x}, \hat{\boldsymbol{x}})}\right)\mathrm{dB}, \tag{27}$$

where $\sigma_{\boldsymbol{x}}^2$ is the variance of the elements of the ground truth vector $\boldsymbol{x}$ and $\mathrm{MSE}(\boldsymbol{x}, \hat{\boldsymbol{x}})$ is the mean squared error between the original $\boldsymbol{x}$ and the reconstruction $\hat{\boldsymbol{x}}$.

7.1. Reconstruction examples

In this experiment, different methods are used to reconstruct a perfect sparse-tree signal from a small number of measurements. The test signal is a piecewise polynomial with maximum order of 3 and one discontinuity. The signal is decomposed by a 4-level wavelet transform using Daubechies wavelets of 4 vanishing moments. With this choice of the test signal and the transform, the coefficient vector has a perfect tree structure since a wavelet with d vanishing moments is orthogonal with any polynomial of degree less than d. The measurement matrix is a random matrix of i.i.d. Gaussian entries with normalized columns. In this setup, the length of the input signal is chosen to be 64 and the number of measurements is 35. For TSWCS and ModelCS, the recommended values of the parameters are used. For our proposed TOMP, $dlev = 2$ and $\alpha = 0.9$. Fig. 2 shows the reconstructed signals. Fig. 3 displays the reconstructed trees, where black nodes are non-zero nodes. As shown on this figure, TOMP, TSWCS and ModelCS maintain the tree structure in the recovered signal very well, where as OMP and CoSaMP fail to capture it. The reason for the worse performance of OMP and CoSaMP is that these methods do not exploit the tree structure exhibited in the signal during the recovery process.

7.2. Performance comparison

In this experiment, the average performance of different methods on random piecewise smooth signal with different number of measurements are computed and compared. For each number of measurements, 100 tests are run with randomly generated measurement matrices and test signals, and then the average reconstruction SNRs are computed. The input signal is a random piecewise smooth signal with 12 discontinuities. The signals has a length of 1024 samples. The wavelet coefficients are computed by a 6-level wavelet decomposition using Daubechies wavelets with 4 vanishing moments. In this case, the test signal is not exactly sparse and the coefficients are connected loosely to a tree structure. This property can be seen on the wavelet tree of a piecewise smooth signal of length 64 on Fig. 4, where the significant coefficients are displayed as black nodes. The values of the parameters of TOMP are $\alpha = 0.9$ and $dlev = 2$.

Fig. 4 compares the average performance of different algorithms in recovering a random piecewise smooth signal of length 1024, in terms of reconstruction quality and execution time. Although TSWCS gives very high SNRs, it is more computationally expensive as opposed to our algorithm, as will be discussed in Section 8. As can be seen from Fig. 4, the execution time of TOMP is closer to OMP than ModelCS and TSWCS. This experiment empirically shows that TOMP and TSWCS still performs well when the signal does not strictly conform to the sparse-tree model.

7.3. Phase diagram

Phase diagrams [17] are used to describe visually the performance of a reconstruction algorithm. A phase diagram shows the probability of successful recovery of a sparse signal under different regimes or problem suites. A *problem suite* [17] $\mathcal{S}(K, M, N)$ is defined as a collection of

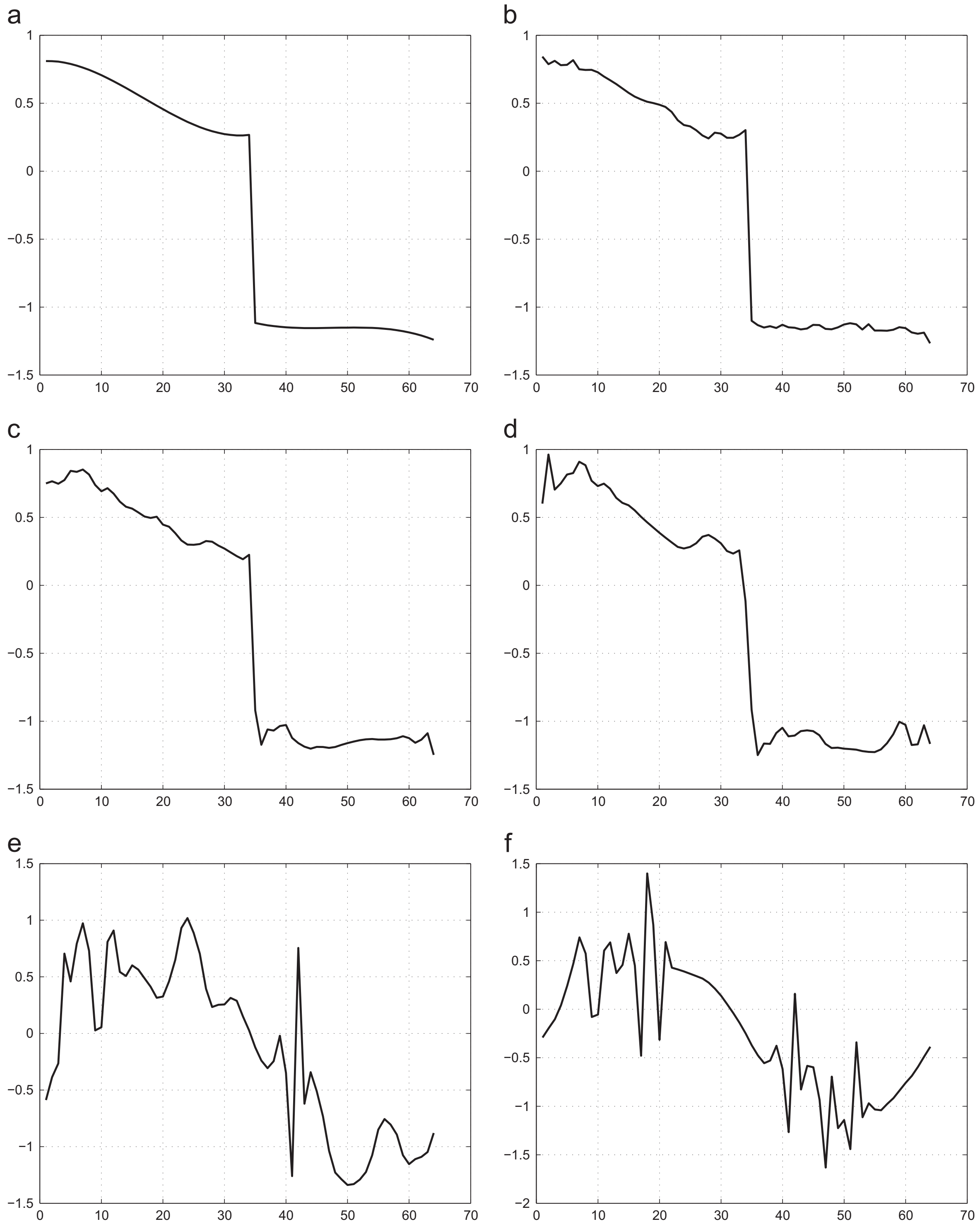


Fig. 2. An example piecewise polynomial signal of length 64 and its reconstructions from 35 linear measurements using TOMP, TSWCS, ModelCS, OMP and CoSaMP. (a) Original signal, (b) TOMP (32.3525 dB), (c) TSWCS (24.0876 dB), (d) ModelCS (19.0908 dB), (e) OMP (4.0952 dB) and (f) CoSaMP (4.3224 dB).

random matrices of size $M \times N$ and K-sparse vectors of length N. For recovery algorithms based on l_1 relaxation techniques, it has been observed that the transition from success to failure occurs sharply along a predictable line in the $\rho-\delta$ plane, where $\rho = K/M$ and $\delta = M/N$ for problem suite $\mathcal{S}(K,M,N)$. This is referred to as the *phase transition phenomenon*. The empirically observed locations of the phase transition can be predicted accurately in theory [30] for l_1-

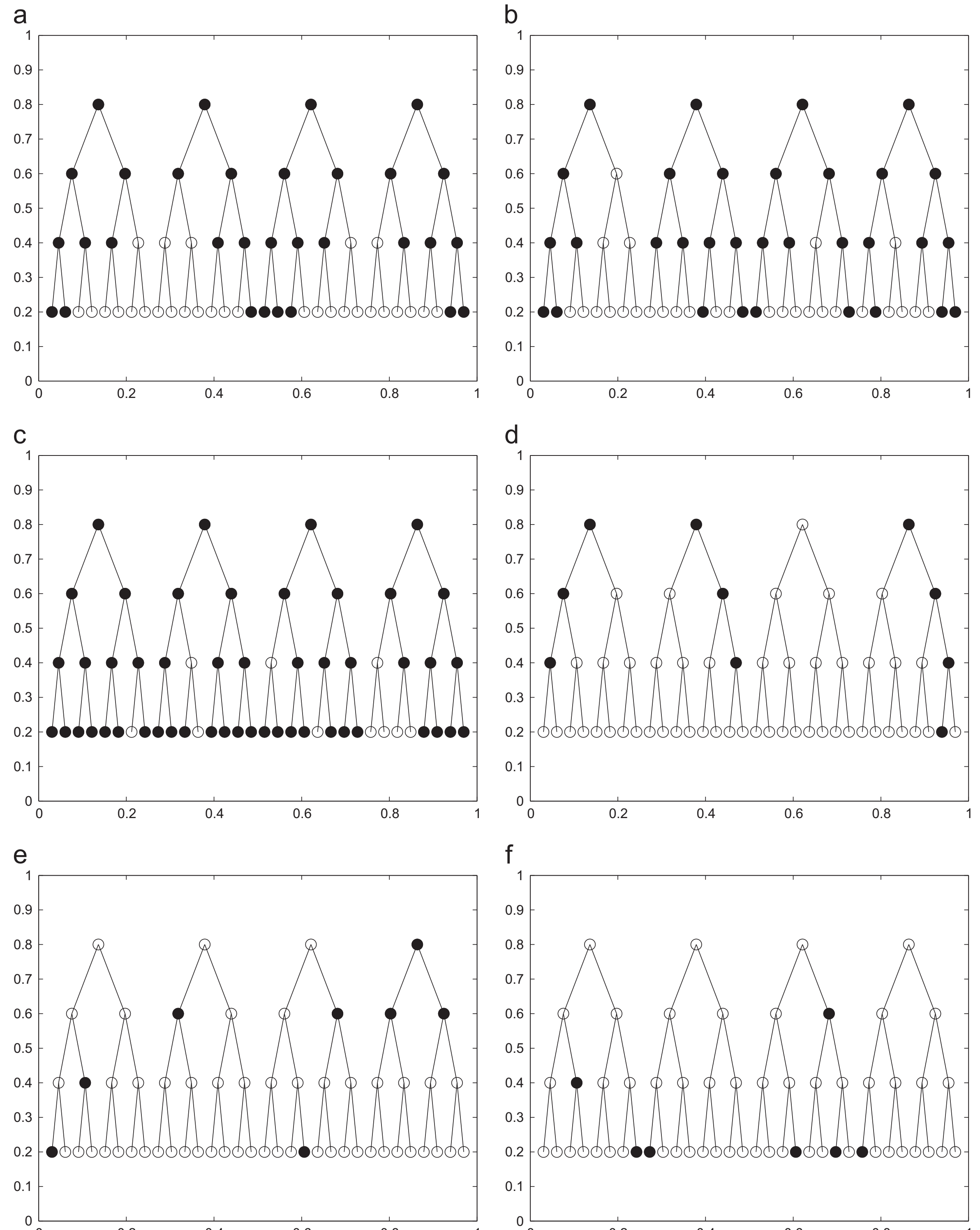
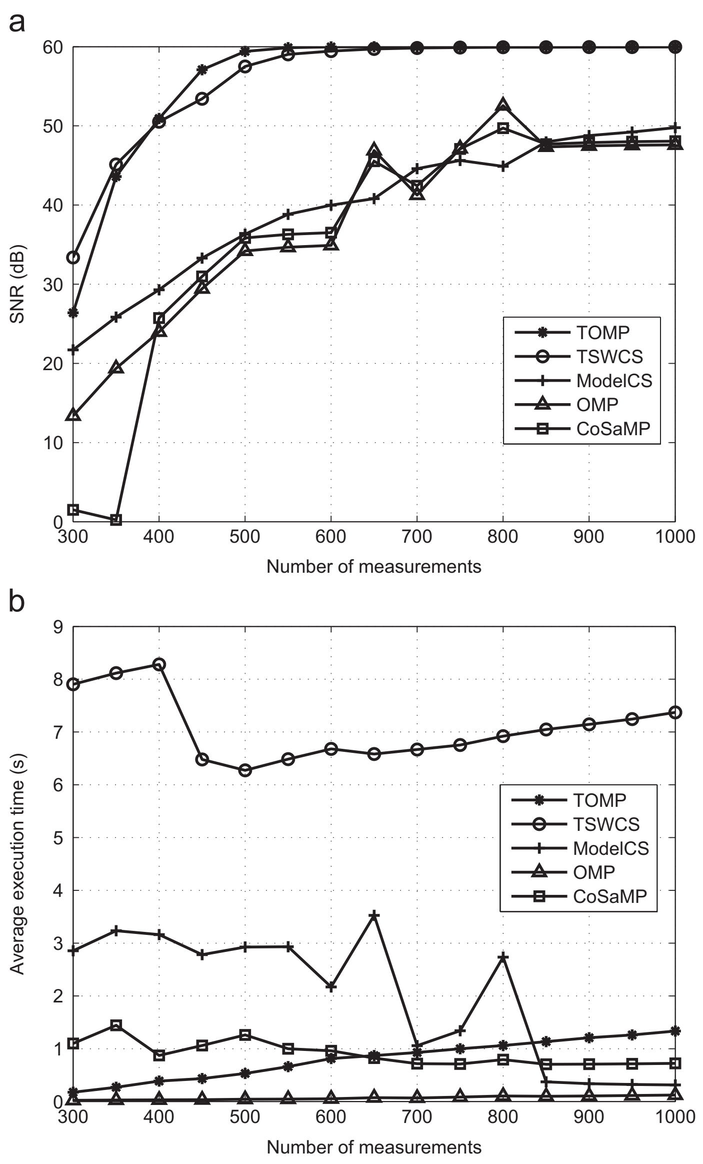


Fig. 3. The wavelet trees of the original signal and reconstructed signals using TOMP, TSWCS, ModelCS, OMP and CoSaMP. Black nodes are non-zero nodes. (a) Original, (b) TOMP, (c) TSWCS, (d) ModelCS, (e) OMP and (f) CoSaMP.

based methods. However, for greedy methods, there has been no theory for the prediction of the phase transition line, although the phenomenon can be still observed empirically in many cases.

In this experiment, the phase diagrams of the above methods are computed and compared for sparse-tree signals. For each set (K, M, N), a random sparse-tree signal with sparsity of K is generated, together with a random



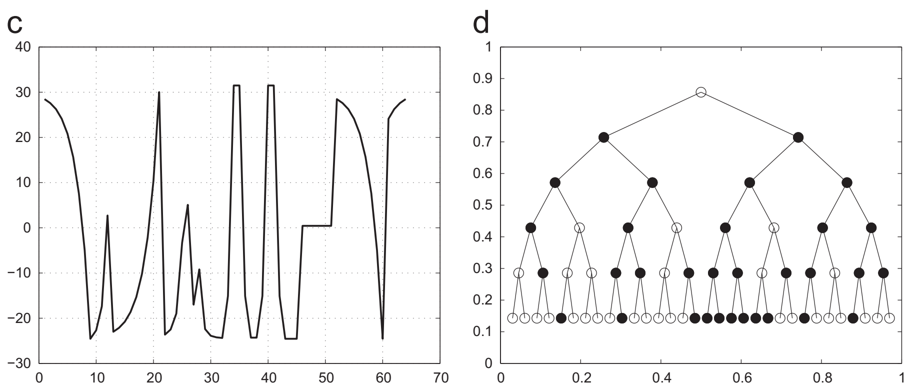


Fig. 4. Performance comparison between different algorithms for the reconstruction of random piecewise smooth signals of length 1024 from different numbers of measurements. (a) Average reconstruction SNR, (b) average execution time, (c) a random piecewise smooth test signal with length 64 and (d) the corresponding wavelet tree of the above test signal.

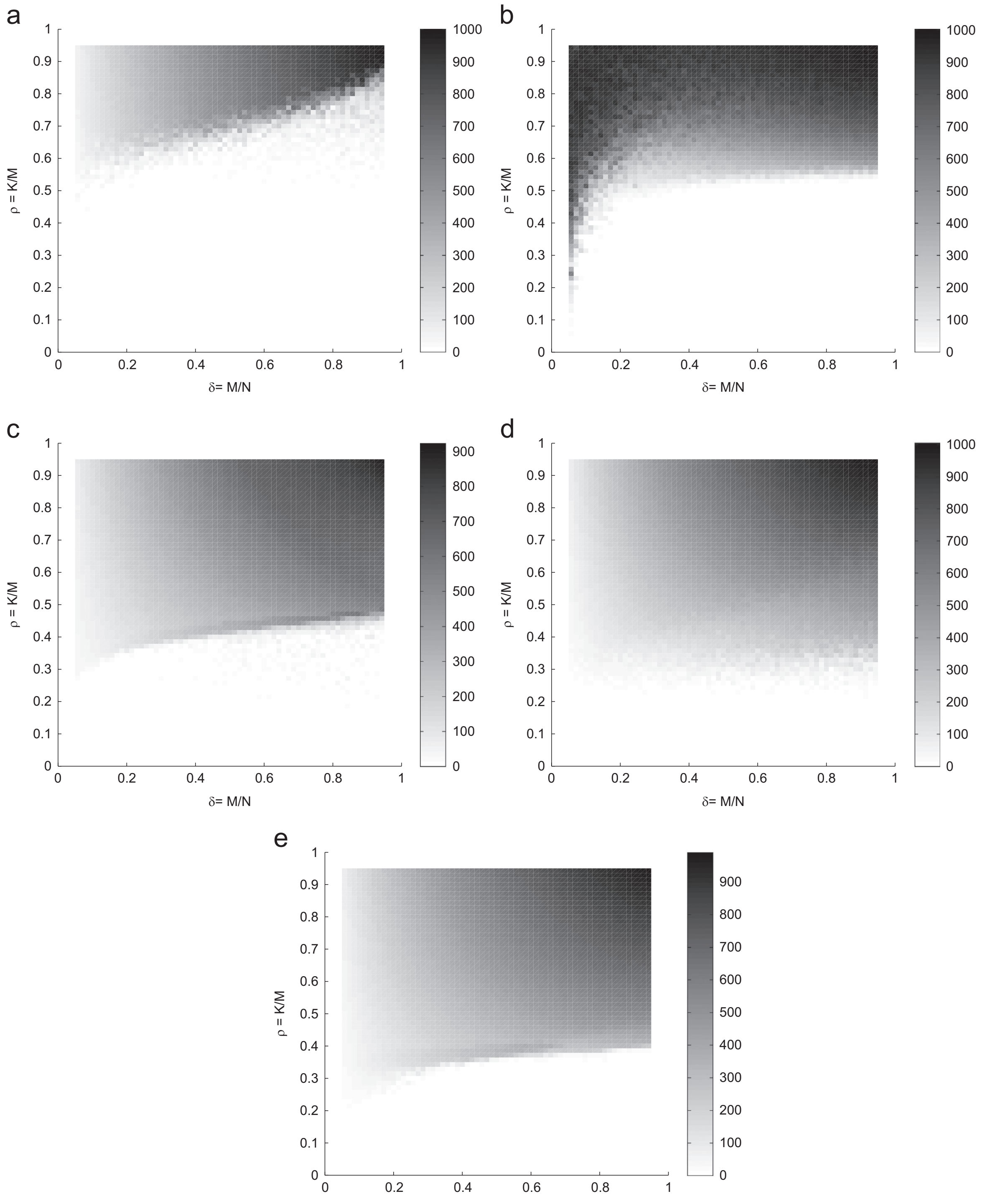


Fig. 5. Phase diagram of different methods. The brighter shade shows higher success rate. (a) TOMP, (b) TSWCS, (c) ModelCS, (d) OMP and (e) CoSaMP.

measurement matrix of dimensions $M \times N$. Fig. 5 shows the phase diagram comparison between TOMP and other methods when recover random sparse-tree signals. All algorithms exhibit the phase transition phenomenon, with sharper transition at higher number of measurements. Of these methods, TOMP provides the highest rate of successful recovery, demonstrated on the diagram by a larger area of bright shade.

8. Discussion

The above experiments show the competitive performance of TOMP for 1-dimensional sparse-tree signals. Compared with OMP, TOMP and other tree-based methods give higher reconstruction quality, especially at a low number of measurements. The TSWCS method gives the highest SNR in most cases. A disadvantage of TSWCS is its higher computational complexity, since at each iteration, the algorithm needs to sample each element of the signal to be recovered using Gibbs sampling. In contrast, at each iteration, TOMP searches for the locations of only significant elements. Compared with ModelCS, TOMP also has less computational complexity since at each iteration of ModelCS, the algorithm has to compute the inner products of the residual with all columns of the measurement matrix to "sense" the locations of the significant coefficients. On the contrary TOMP only needs to compute the inner products between the residual and the candidate columns. The highest computational cost of TOMP is the projection onto the history set for each finalist, which can be reduced by increasing the relaxation parameter α (which in effect reduces the number of finalist sets).

As discussed in Section 6.2, there is a trade-off between computational cost and robustness of the algorithm to signals that do not conform to the sparse-tree model. This trade-off is controlled by parameter d. If d is small then the computational cost at each iteration will be low but the finest levels of the tree might not be investigated. On the other hand, a large value of d ensures that the algorithm will find significant coefficients at deeper levels but the computational cost will be higher. For signals that strictly conforms to the sparse-tree model, the finest level coefficients will have small magnitudes and do not contribute much to the reconstructed signal. Thus, d can be set to a small value for these signals to reduce the computational cost.

Although TOMP is specific designed for sparse-tree signals, the algorithm still works with small deviations from the model. This is empirically proved by the results with piecewise smooth signal in Fig. 4.

For natural images, a separable wavelet transform does not effectively exploit the fact that discontinuities are formed along geometrically smooth curves. With a typical image of size 256×256, a 2-dimensional wavelet transform gives shallow trees with a maximum height of 8. TOMP does not work effectively in this situation. The recovery of images will be investigated in an upcoming paper with a more effective geometric 2-dimensional decomposition, such as curvelets [31] and contourlets [32], where significant coefficients are successively localized in a tree structure in both location and direction.

9. Concluding remarks

Most existing Compressed Sensing recovery algorithms still only exploit the sparse prior of signals, regardless of any signal structure that may present. Despite an increasing interest in model-based Compressed Sensing in the past few years, the application of the proposed methods has still been limited in practice. In this paper, we present a simple yet effective algorithm that exploits the sparse-tree structure of the signal for signal reconstruction. The *sparse-tree inverse problem* has been formulated and its usefulness has been justified. Based on that, the Tree-based Orthogonal Matching Pursuit (TOMP) algorithm is proposed and analyzed in detail, both theoretically and empirically. The experimental results confirm the state-of-the-art performance of TOMP with significantly lower computational cost.

Acknowledgments

We thank the authors of [8] and [7] for making the code of TSWCS and ModelCS algorithms, respectively, publicly available. We also thank Ha Q. Nguyen for helping with the proof of Theorem 6.4 and Benjamin Chidester for reviewing this paper. Finally, we thank the associate editor and the reviewers for the constructive comments and discussion.

References

[1] Y. Bresler, M. Gastpar, R. Venkataramani, Image compression on-the-fly by universal sampling in Fourier imaging systems, in: Proceedings of 1999 IEEE Information Theory Workshop on Detection, Estimation, Classification, and Imaging, 1999, pp. 48.
[2] M. Vetterli, P. Marziliano, T. Blu, Sampling signals with finite rate of innovation, IEEE Trans. Signal Process. 50 (6) (2002) 1417–1428.
[3] E. Candès, J. Romberg, T. Tao, Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information, IEEE Trans. Inf. Theory 52 (2) (2006) 489–509.
[4] D. Donoho, Compressed sensing, IEEE Trans. Inf. Theory 52 (4) (2006) 1289–1306.
[5] E. Candes, T. Tao, Near-optimal signal recovery from random projections: universal encoding strategies? IEEE Trans. Inf. Theory 52 (12) (2006) 5406–5425.
[6] J. Tropp, A. Gilbert, Signal recovery from random measurements via Orthogonal Matching Pursuit, IEEE Trans. Inf. Theory 53 (12) (2007) 4655–4666.
[7] R. Baraniuk, V. Cevher, M. Duarte, C. Hegde, Model-based compressive sensing, IEEE Trans. Inf. Theory 56 (4) (2010) 1982–2001.
[8] L. He, L. Carin, Exploiting structure in wavelet-based Bayesian compressive sensing, IEEE Trans. Signal Process. 57 (9) (2009) 3488–3497.
[9] L. He, H. Chen, L. Carin, Tree-structured compressive sensing with variational Bayesian analysis, IEEE Signal Process. Lett. 17 (3) (2010) 233–236.
[10] M. Duarte, M. Wakin, R. Baraniuk, Fast reconstruction of piecewise smooth signals from incoherent projections, SPARS 05.
[11] C. La, M.N. Do, Signal reconstruction using sparse tree representation, in: Proceedings of Wavelets XI at SPIE Optics and Photonics, San Diego.
[12] C. La, M. Do, Tree-based Orthogonal Matching Pursuit algorithm for signal reconstruction, in: IEEE International Conference on Image Processing, IEEE, 2006, pp. 1277–1280.
[13] Y.C. Eldar, G. Kutyniok, Compressed Sensing: Theory and Applications, Cambridge University Press, New York, NY 10013, USA, 2012.
[14] G. Davis, S. Mallat, M. Avellaneda, Greedy adaptive approximation, J. Constr. Approx. 13 (1997) 57–98.
[15] D. Needell, J. Tropp, CoSaMP: iterative signal recovery from incomplete and inaccurate samples, Appl. Comput. Harmon. Anal. 26 (3) (2009) 301–321.
[16] W. Dai, O. Milenkovic, Subspace pursuit for compressive sensing signal reconstruction, IEEE Trans. Inf. Theory 55 (5) (2009) 2230–2249.
[17] D.L. Donoho, Y. Tsaig, I. Drori, J. luc Starck, Sparse solution of underdetermined systems of linear equations by Stagewise Orthogonal Matching Pursuit, IEEE Trans. Inf. Theory 58 (2) (2012) 1094–1121.
[18] S. Chen, D. Donoho, M. Saunders, Atomic decomposition by basis pursuit, SIAM J. Sci. Comput. 20 (1) (1998) 33–61.

[19] I. Daubechies, M. Defrise, C. de Mol, An iterative thresholding algorithm for linear inverse problems with a sparsity constraint, Commun. Pure Appl. Math. 57 (11) (2004) 1413–1457.
[20] M. Figueiredo, J. Bioucas-Dias, R. Nowak, Majorization–Minimization algorithms for wavelet-based image restoration, IEEE Trans. Image Process. 16 (12) (2007) 2980–2991.
[21] M. Elad, B. Matalon, M. Zibulevsky, Coordinate and subspace optimization methods for linear least squares with non-quadratic regularization, Appl. Comput. Harmon. Anal. 23 (3) (2007) 346–367.
[22] D.L. Donoho, A. Maleki, A. Montanari, Message-passing algorithms for compressed sensing, Proc. Natl. Acad. Sci. 106 (45) (2009) 18914–18919.
[23] M. Do, C. La, Tree-based Majorize–Maximize algorithm for compressed sensing with sparse-tree prior, in: 2nd International Workshop on Computational Advances in Multi-Sensor Adaptive Processing, IEEE, 2007, pp. 129–132.
[24] T. Blumensath, M. Davies, Iterative hard thresholding for compressed sensing, Appl. Comput. Harmon. Anal. 27 (3) (2009) 265–274.
[25] S. Som, P. Schniter, Compressive imaging using approximate message passing and a Markov-tree prior, IEEE Trans. Signal Process. 60 (7) (2012) 3439–3448.
[26] S. Mallat, A Wavelet Tour of Signal Processing: The Sparse Way, Academic Press, Burlington, MA 01803, USA, 2009.
[27] P. Dragotti, M. Vetterli, Wavelet footprints: theory, algorithms, and applications, IEEE Trans. Signal Process. 51 (5) (2003).
[28] M.S. Crouse, R.D. Nowak, R.G. Baraniuk, Wavelet-based statistical signal processing using Hidden Markov models, IEEE Trans. Signal Process. 46 (4) (1998) 886–902.
[29] J. Tropp, Greed is good: algorithmic results for sparse approximation, IEEE Trans. Inf. Theory 50 (10) (2004) 2231–2242.
[30] D.L. Donoho, High-dimensional centrally symmetric polytopes with neighborliness proportional to dimension, Discret. Comput. Geom. 35 (4) (2006) 617–652.
[31] E.J. Candès, D.L. Donoho, New tight frames of curvelets and optimal representations of objects with piecewise C^2 singularities, Commun. Pure Appl. Math. 20 (2004) 219–266.
[32] M.N. Do, M. Vetterli, The contourlet transform: an efficient directional multiresolution image representation, IEEE Trans. Image Process. 14 (12) (2005) 2091–2106.