A Fast Approximation Algorithm for Tree-Sparse Recovery

Chinmay Hegde, Piotr Indyk, Ludwig Schmidt[1]
Massachusetts Institute of Technology

***Abstract*—Sparse signals whose nonzeros obey a tree-like structure occur in a range of applications such as image modeling, genetic data analysis, and compressive sensing. An important problem encountered in recovering signals is that of *optimal tree-projection*, i.e., finding the closest tree-sparse signal for a given query signal. However, this problem can be computationally very demanding: for optimally projecting a length-n signal onto a tree with sparsity k, the best existing algorithms incur a high runtime of $O(nk)$. This can often be impractical.**

We suggest an alternative approach to tree-sparse recovery. Our approach is based on a specific *approximation* algorithm for tree-projection and provably has a near-linear runtime of $O(n \log(kr))$ and a memory cost of $O(n)$, where r is the dynamic range of the signal. We leverage this approach in a fast recovery algorithm for tree-sparse compressive sensing that scales extremely well to high-dimensional datasets. Experimental results on several test cases demonstrate the validity of our approach.

I. Introduction

Over the last decade, the concept of sparsity has attracted significant attention among researchers in statistical signal processing, information theory, and numerical optimization. Sparsity serves as the foundation of *compressive sensing* (CS), a new paradigm for digital signal and image acquisition [1, 2]. A key result in CS states that a k-sparse signal of length n can be recovered using only $m = O(k \log n/k)$ linear measurements; when $k \ll n$, this can have significant benefits both in theory and in practice.

However, several classes of real-world signals and images possess additional structure beyond mere sparsity. One example is the class of signals encountered in digital communication: these are often "bursty" and hence can be modeled as sparse signals whose nonzeros are grouped in a small number of *blocks*. A more sophisticated example is the class of natural images. Here, the dominant coefficients in the wavelet-domain representation can be modeled as a *rooted, connected tree* [3]. These (and several other) notions of structure can be concisely captured via the notion of a *structured sparsity model*. In the CS context, structured sparsity can enable signal recovery algorithms that succeed with merely $O(k)$ linear measurements [4].

Our focus in this paper is on *tree-structured sparsity*. Tree-sparse data are not only interesting from a theoretical perspective but also naturally emerge in a range of applications such as imaging and genomics [3, 5, 6]. Of particular interest to us is the following problem: given an arbitrary signal $x \in \mathbb{R}^n$, find the k-sparse tree-structured signal $\widehat{x}$ that minimizes the error $\|x - \widehat{x}\|_2$. This problem arises in several settings including signal / image compression and denoising [7, 8]

The optimal tree-projection problem has a rich history; see, for example, the papers [9–11] and references therein. The best available (theoretical) performance for this problem is achieved by the dynamic-programming (DP) approach of [12], building upon the algorithm first developed in [11]. For signal length n and target sparsity k, the algorithm has a runtime of $O(nk)$. Unfortunately, this means that the algorithm does not easily scale to real-world problem sizes. For example, even a modestly-sized natural image (say, of size $n = 512 \times 512$) can only be expected to be tree-sparse with parameter $k \approx 10^4$. In this case, nk exceeds 2.5×10^9, and hence the runtime quickly becomes impractical for megapixel-size images.

In this paper, we develop an alternative approach for tree-projection. The core of our approach is a novel approximation algorithm that provably has a *near-linear* runtime of $O(n \log(kr))$ and a memory cost of $O(n)$, where r is the dynamic range of the signal. Importantly, the memory cost is independent of the sparsity parameter k. Therefore, our approach is eminently suitable for applications involving very high-dimensional signals and images, which we demonstrate via several experiments.

Our tree-projection algorithm is *approximate*: instead of exactly minimizing the error $\|x - \widehat{x}\|_2$, we merely return an estimate $\widehat{x}$ whose error is at most a constant factor c_1 times the optimal achievable error (i.e., that achieved by [12]). Moreover, our signal estimate $\widehat{x}$ is tree-sparse with parameter $c_2 k$, therefore giving us a *bicriterion* approximation guarantee.

At a high level, our algorithm can be viewed as an extension of the complexity-penalized residual sum of squares (CPRSS) formulation proposed by Donoho in [9]. We pose the exact tree projection as a (nonconvex) sparsity-constrained optimization problem. We perform a Lagrangian relaxation of the sparsity constraint and, similar to Donoho, solve the relaxed problem using a dynamic program (DP) with runtime (as well as memory cost) of $O(n)$. We then iterate this step $O(\log(kr))$ times by conducting a binary search over the Lagrangian relaxation parameter until we arrive at the target sparsity k. A careful termination criterion for the binary search gives our desired approximation guarantee.

We combine our approximate tree-projection algorithm with the model-based CS framework proposed in [4]. This produces an extremely fast CS recovery algorithm for tree-sparse signals. We present several experiments on synthetic and real-world signals that demonstrate the benefits of our approach.

[1] Authors ordered alphabetically.

II. Background

A. Sparsity and Structure

A signal $x \in \mathbb{R}^n$ is said to be k-sparse if no more than k of its coefficients are nonzero. The support of x, denoted by $\text{supp}(x) \subseteq [n]$, indicates the locations of its nonzero entries.

Suppose that we possess some additional *a priori* information about the support of our signals of interest. One way to model this information is as follows [4]: denote the set of *allowed supports* with $\mathbb{M}_k = \{\Omega_1, \Omega_2, \ldots, \Omega_L\}$, where $\Omega_i \subseteq [n]$ and $|\Omega_i| = k$. Often it is useful to work with the closure of $\mathbb{M}_k$ under taking subsets, which we denote with $\mathbb{M}_k^+ = \{\Omega \subseteq [n] \mid \Omega \subseteq S \text{ for some } S \in \mathbb{M}_k\}$. Then, we define a *structured sparsity model*, $\mathcal{M}_k \subseteq \mathbb{R}^n$, as the set of vectors $\mathcal{M}_k = \{x \in \mathbb{R}^n \mid \text{supp}(x) \in \mathbb{M}_k^+\}$. The number of allowed supports L is called the "size" of the model $\mathcal{M}_k$; typically, $L \ll \binom{n}{k}$.

We define the *model-projection* problem for $\mathcal{M}_k$ as follows: given $x \in \mathbb{R}^n$, determine a $x^* \in \mathcal{M}_k$ such that $\|x - x^*\|_p$ is minimized for a norm parameter p. In general, this problem can be hard, since $\mathcal{M}_k$ is typically non-convex. However, several special choices of models $\mathcal{M}_k$ *do* admit polynomial-time model-projection methods; see [13] for an overview.

B. Tree-Sparsity

Our focus in this paper is the *tree-structured sparsity model* (or simply, the tree-sparsity model). We assume that the n coefficients of a signal $x \in \mathbb{R}^n$ can be arranged as the nodes of a full d-ary tree. Then, the tree-sparsity model comprises the set of k-sparse signals whose nonzero coefficients form a *rooted, connected* subtree. It can be shown that the size of this model is upper bounded by $L \leq (2e)^k/(k+1)$ [4].

For the rest of the paper, we denote the set of supports corresponding to a subtree rooted at node i with $\mathbb{T}_i$. Then $\mathbb{M}_k = \{\Omega \subseteq [n] \mid \Omega \in \mathbb{T}_1 \text{ and } |\Omega| = k\}$ is the formal definition of the tree-sparsity model (we assume node 1 to be the root of the entire signal).

The tree-sparsity model can be used for modeling a variety of signal classes. A compelling application of this model emerges while studying the wavelet decomposition of piecewise-smooth signals and images. It is known that wavelets act as *local* discontinuity detectors [5]. Since the supports of wavelets at different scales are nested, a signal discontuinity will give rise to a chain of significant coefficients along a branch of the wavelet tree (see Fig. 1).

The problem of projecting onto the tree-sparsity model has received a fair amount of attention in the literature over the last two decades. Numerous algorithms have been proposed, including the condensing sort-and-select algorithm (CSSA) [10], complexity-penalized residual sum-of-squares (CPRSS) [9], and optimal-pruning [11]. In contrast to CSSA and CPRSS, our algorithm offers worst-case approximation guarantees for arbitrary signals. While the optimal-pruning approach guarantees an *optimal* k-sparse tree-projection by using a dynamic program, it has a runtime of $O(n^2)$. The recent paper [12] gives an improved version of this approach with a runtime of $O(nk)$. However, this is still impractical for high-dimensional signal processing applications.

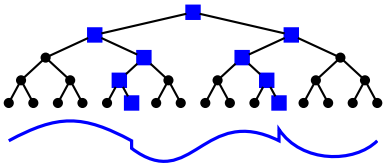

Fig. 1. A binary wavelet tree for a one-dimensional signal. The squares denote the large wavelet coefficients that arise from the discontinuities in the piecewise smooth signal drawn below. The support of the large coefficients forms a rooted, connected tree.

C. Compressive Sensing

Suppose that instead of collecting all the coefficients of a k-sparse vector $x \in \mathbb{R}^n$, we merely record $m = O(k \log n/k)$ inner products (measurements) of x with $m \ll n$ preselected vectors, i.e., we observe an m-dimensional vector $y = Ax$, where $A \in \mathbb{R}^{m \times n}$ is the measurement matrix. The central result of compressive sensing (CS) is that under certain assumptions on A, x can be *exactly* recovered from y, even though A is rank-deficient (and therefore has a nontrivial nullspace).

Numerous algorithms for signal recovery from compressive measurements have been developed. Of special interest to us are iterative support selection algorithms, e.g. CoSaMP [14]. Such algorithms can be modified to use any arbitrary structured sparsity model when given access to a model-projection oracle [4]. These modified "model-based" recovery algorithms offer considerable benefits both in theory and in practice. For the tree-sparsity model, a modified version of CoSaMP provably recovers tree-sparse signals using $m = O(k)$ measurements, thus matching the information-theoretic lower bound.

III. Tree-Sparse Approximation

Recall that the main goal in tree-projection is the following: for a given signal $x \in \mathbb{R}^n$, minimize the quantity $\|x - x_\Omega\|_p$ over $\Omega \in \mathbb{M}_k$ for a given $1 \leq p < \infty$.[2] In this paper, we are interested in the following related problem: find a $\widehat{\Omega} \in \mathbb{M}_{k'}$ with $k' \leq c_2 k$ and

$$\left\|x - x_{\widehat{\Omega}}\right\|_p \leq c_1 \min_{\Omega \in \mathbb{M}_k} \|x - x_\Omega\|_p \,. \tag{1}$$

We highlight two aspects of the modified problem (1): (i) Instead of projecting into the model $\mathcal{M}_k$, we project into the slightly larger model $\mathcal{M}_{k'}$. (ii) Instead of finding the best projection, we provide a multiplicative guarantee for the approximation error.

We propose an approximation algorithm that achieves (1). First, we approach the modified problem via a Lagrangian relaxation, i.e., we relax the sparsity constraint and keep the requirement that the support Ω forms a connected subtree:

$$\arg\min_{\Omega \in \mathbb{T}_1} \|x - x_\Omega\|_p^p + \lambda |\Omega| \,. \tag{2}$$

[2] Our approximation algorithm also solves the ℓ_∞-version of the tree-sparse approximation problem (and, in this case, even identifies the optimal projection, not only a provably good one). Since the focus in model-based compressive sensing usually lies on $p = 1$ or $p = 2$, we limit our proofs here to ℓ_p-norms with $p < \infty$.

Note that the parameter λ controls the trade-off between the approximation error and the sparsity of the recovered support. Second, we use a binary search to find a suitable value of λ, resulting in an overall recovery guarantee of the form (1).

A. Solving the Lagrangian relaxation

We first transform the problem in (2) into a slightly different form. Note that (2) is equivalent to $\arg\max_{\Omega\in\mathbb{T}_1} \|x_\Omega\|_p^p - \lambda|\Omega|$. We can now rewrite this problem as

$$\arg\max_{\Omega\in\mathbb{T}_1} \sum_{i\in\Omega} y_i \tag{3}$$

where $y_i = |x_i|^p - \lambda$. Hence the goal is to find a rooted subtree which maximizes the sum of weights y_i contained in the subtree. In contrast to the original tree approximation problem, there is no sparsity constraint, but the weights associated with nodes can now be negative.

Problem (3) is similar to the CPRSS formulation proposed by Donoho [9]. However, our technical development here is somewhat different because the underlying problems are not exactly identical. Therefore, we summarize our approach in Alg. 1 and outline its proof for completeness.

Theorem 1. *Let $x \in \mathbb{R}^n$ be the coefficients corresponding to a tree rooted at node 1, and let $p \geq 1$. Then* FINDTREE(x, λ, p) *runs in linear time and returns a support $\widehat{\Omega} \in \mathbb{T}_1$ satisfying*

$$\left\|x - x_{\widehat{\Omega}}\right\|_p^p + \lambda|\widehat{\Omega}| = \min_{\Omega\in\mathbb{T}_1} \|x - x_\Omega\|_p^p + \lambda|\Omega| \ .$$

Proof: As above, let $y_i = |x_i|^p - \lambda$. For a support $\Omega \in [n]$, let $y(\Omega) = \sum_{i\in\Omega} y_i$. Futhermore, we denote the total weight of the best subtree rooted at node i with $b_i^* = \max_{\Omega\in\mathbb{T}_i} y(\Omega)$. Note that $b_i^* = \max(0, y_i + \sum_{j\in\text{children(i)}} b_j^*)$ because we can choose nodes in the subtrees of i independently. A simple inductive argument shows that after the call to CALCULATEBEST$(1, x, \lambda, p)$, we have $b_i = b_i^*$ for $i \in [n]$.

Similarly, a proof by induction shows that after a call to FINDSUPPORT(i), we have $y(\Omega_i) = b_i^*$. So the support $\widehat{\Omega}$ returned by FINDTREE(x, λ, p) satisfies $y(\widehat{\Omega}) = \max_{\Omega\in\mathbb{T}_1} y(\Omega)$. This implies the guarantee in the theorem.

The time complexity follows from the fact that the algorithm makes a constant number of passes over the tree. ∎

B. Binary search over λ

Next, we use Alg. 1 in order to achieve the desired approximation guarantee (1). Since the Lagrangian relaxation (2) gives us only indirect control over the sparsity k', we perform a binary search over λ to find a suitable value. Alg. 2 contains the corresponding pseudo code. In addition to x, k, c, and p, the algorithm takes an additional parameter δ, which controls the maximum number of iterations of the binary search.

Theorem 2. *Let $x \in R^n$ be the coefficients corresponding to a tree rooted at node 1. Moreover, let $c > 1$, $\delta > 0$, and $p \geq 1$. Then* TREEAPPROX(x, k, c, p, δ) *returns a support $\widehat{\Omega} \in \mathbb{M}_{ck}^+$ satisfying*

$$\left\|x - x_{\widehat{\Omega}}\right\|_p \leq \left(1 + \frac{1}{c-1} + \delta\right)^{1/p} \min_{\Omega\in\mathbb{M}_k} \|x - x_\Omega\|_p \ .$$

Algorithm 1 (FINDTREE) Solving the Lagrangian relaxation

```
1:  function FindTree(x, λ, p)
2:      CalculateBest(1, x, λ, p)
3:      return Ω̂ ← FindSupport(1)
4:  function CalculateBest(i, x, λ, p)
5:      b_i ← |x_i|^p − λ
6:      for j ∈ children(i) do
7:          CalculateBest(j, x, λ, p)
8:          b_i ← b_i + b_j
9:      b_i ← max(0, b_i)
10: function FindSupport(i)
11:     if b_i = 0 then
12:         Ω_i ← {}
13:     else
14:         Ω_i ← {i}
15:         for j ∈ children(i) do
16:             Ω_i ← Ω_i ∪ FindSuppport(j)
17:     return Ω_i
```

Algorithm 2 (TREEAPPROX) Tree-sparse approximation

```
1:  function TreeApprox(x, k, c, p, δ)
2:      if there is a Ω̂ ∈ M_k with supp(x) ⊆ Ω̂ then
3:          return Ω̂
4:      x_max ← max_{i∈[n]}|x_i|,   x_min ← min_{i∈[n],x_i>0}|x_i|
5:      λ_l ← x_max^p,   λ_r ← 0,   ε ← δ x_min^p / k
6:      while λ_l − λ_r > ε do
7:          λ_m ← (λ_l + λ_r)/2
8:          Ω̂ ← FindTree(x, λ_m, p)
9:          if |Ω̂| ≥ k and |Ω̂| ≤ ck then
10:             return Ω̂
11:         else if |Ω̂| < k then
12:             λ_l ← λ_m
13:         else
14:             λ_r ← λ_m
15:     return Ω̂ ← FindTree(x, λ_l, p)
```

Moreover, the algorithm runs in time $O(n(\log \frac{k}{\delta} + p \log \frac{x_{\max}}{x_{\min}}))$, where $x_{\max} = \max_{i\in[n]}|x_i|$ and $x_{\min} = \min_{i\in[n],x_i>0}|x_i|$.

Proof: We analyze the three cases in which TREEAPPROX returns a support. First, note that the condition in line 2 can be checked efficiently: connect all nonzero entries in x to the root node and denote the resulting support with $\widehat{\Omega}$. If $|\widehat{\Omega}| \leq k$, we have $\widehat{\Omega} \in \mathbb{M}_k^+$ and $x \in \mathcal{M}_k$. Otherwise, $x \notin \mathcal{M}_k$ and the tail approximation error is greater than zero.

Second, if the algorithm returns in line 10, we have $|\widehat{\Omega}| \leq ck$ and $\widehat{\Omega} \in \mathbb{T}_1$ (Theorem 1). Hence $\widehat{\Omega} \in \mathbb{M}_{ck}^+$. Moreover, Theorem 1 implies

$$\left\|x - x_{\widehat{\Omega}}\right\|_p^p + \lambda_m|\widehat{\Omega}| \leq \min_{\Omega\in\mathbb{M}_k} \|x - x_\Omega\|_p^p + \lambda_m|\Omega| \ .$$

Since $|\widehat{\Omega}| \geq k = |\Omega|$ for $\Omega \in \mathbb{M}_k$, we have $\left\|x - x_{\widehat{\Omega}}\right\|_p \leq \min_{\Omega\in\mathbb{M}_k} \|x - x_\Omega\|_p$.

Finally, consider the return statement in line 15. Let Ω_l and Ω_r be the supports corresponding to λ_l and λ_r, respectively. Moreover, let $\Omega_{OPT} \in \mathbb{M}_k$ be a support with

$\|x - x_{\Omega_{OPT}}\|_p = \min_{\Omega \in \mathbb{M}_k} \|x - x_\Omega\|_p$. We use the shorthands $t_l = \|x - x_{\Omega_l}\|_p^p$, $k_l = |\Omega_l|$, and the corresponding definitions for t_r, k_r, t_{OPT}, and k_{OPT}. Note that throughout the binary search, we maintain the invariants $k_r \geq ck$ and $k_l \leq k$. The invariants also hold before the first iteration of the binary search due to our initial choices for λ_l and λ_r ($\lambda = 0$ implies $y_i \geq 0$ and $\lambda = x_{\max}^p$ implies $y_i \leq 0$, both for all $i \in [n]$).

From Theorem 1, we have $t_r + \lambda_r k_r \leq t_{OPT} + \lambda_r k_{OPT}$. This implies

$$\begin{aligned} \lambda_r(k_r - k_{OPT}) &\leq t_{OPT} - t_r \\ \lambda_r(ck - k) &\leq t_{OPT} \\ \lambda_r &\leq \frac{t_{OPT}}{k(c-1)} \, . \end{aligned}$$

At the end of the binary search we have $\lambda_l - \lambda_r \leq \varepsilon$, giving

$$\lambda_l \leq \frac{t_{OPT}}{k(c-1)} + \varepsilon \, . \tag{4}$$

Theorem 1 also implies $t_l + \lambda_l k_l \leq t_{OPT} + \lambda_l k_{OPT}$. Together with (4), we get

$$\begin{aligned} t_l &\leq t_{OPT} + \lambda_l k \\ &\leq t_{OPT} + \frac{t_{OPT}}{c-1} + \varepsilon k \\ &\leq t_{OPT} \left(1 + \frac{1}{c-1}\right) + \delta x_{\min}^p \\ &\leq \left(1 + \frac{1}{c-1} + \delta\right) t_{OPT} \, . \end{aligned}$$

The last line follows from the fact that $x \notin \mathcal{M}_k$ and hence $t_{OPT} \geq x_{\min}^p$. Taking the p-th root on both sides gives the guarantee in the theorem.

Each iteration of the binary search runs in linear time (Theorem 1). The difference $\lambda_l - \lambda_r$ initially is $x_{\max}^p$ and is then halved in every iteration until it reaches ε. Hence the total number of iterations is at most

$$\log \frac{x_{\max}^p}{\varepsilon} \;=\; \log \frac{x_{\max}^p k}{\delta x_{\min}^p} \;=\; \log \frac{k}{\delta} + p \log \frac{x_{\max}}{x_{\min}} \, . \qquad \blacksquare$$

In practical applications, δ, p, $x_{\max}$, and $x_{\min}$ can be considered constants, which gives a running time of $O(n \log k)$. In follow-up work, we remove the dependence on $\frac{x_{\max}}{x_{\min}}$ and give an algorithm running in $O(n \log n)$ time while achieving the same approximation guarantee [15].

C. Tree-Structured CS Recovery

Our tree-projection algorithm TREEAPPROX (Alg. 2) is useful in a number of contexts. Here, we leverage the algorithm in model-based compressive sensing.

Using the framework of [4], we combine TREEAPPROX with CoSaMP. Alg. 3 summarizes our proposed CS recovery method. The algorithm closely resembles the tree-CoSaMP algorithm developed in [4], except that the model-projections (in lines 5 and 7) are implemented using the approximate projection method TREEAPPROX. In our experiments below, we will use TREEAPPROX with approximation parameter $c = 1.1$ and norm parameter $p = 2$. In [15], we give a full recovery scheme for the tree-sparsity model using approximate projection algorithms.

Algorithm 3 (TREE-COSAMP) Signal recovery

1: **function** TREE-COSAMP(y, k, A, t)
2: $\quad \widehat{x}_0 \leftarrow 0$
3: $\quad$ **for** $i \leftarrow 1, \ldots, t$ **do**
4: $\quad\quad v \leftarrow A^T(y - A\widehat{x}_{i-1})$
5: $\quad\quad \Gamma \leftarrow \text{supp}(\widehat{x}_{i-1}) \cup \text{TREEAPPROX}(v, 2k, c, 2, \delta)$
6: $\quad\quad z_\Gamma \leftarrow A_\Gamma^\dagger y, \quad z_{\Gamma^C} \leftarrow 0$
7: $\quad\quad \Omega \leftarrow \text{TREEAPPROX}(z, k, c, 2, \delta)$
8: $\quad\quad \widehat{x}_i \leftarrow z_\Omega$
9: $\quad$ **return** $\widehat{x} \leftarrow \widehat{x}_i$

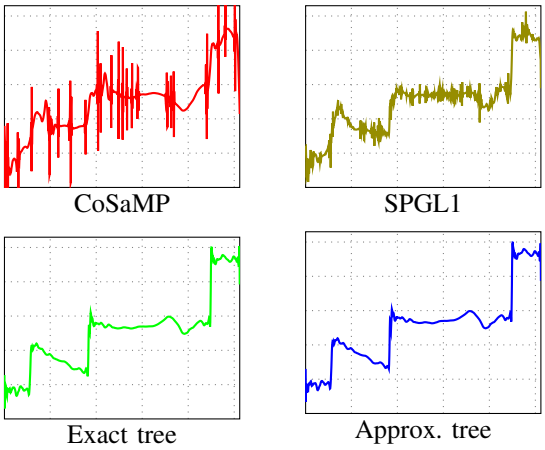

Fig. 2. CS recovery of a 1D signal using various algorithms (signal parameters: $n = 1024$, $k = \lceil 0.04n \rceil = 41$, $m = \lceil 3.5k \rceil = 144$). Both tree-based algorithms accurately recover the ground truth signal.

IV. NUMERICAL RESULTS

We now demonstrate the considerable benefits gained by our approximate tree-projection algorithm in the context of compressive sensing (CS). All experiments were conducted on a laptop computer equipped with an Intel Core i7 processor (2.66 GHz) and 8GB of RAM. Corresponding code is available on http://people.csail.mit.edu/ludwigs/code.html.

First, we run model-CoSaMP with the exact tree projection approach in [12], as well as Alg. 3, on a piecewise polynomial signal. Such signals are well-modeled as tree-sparse in the wavelet domain [3]. The signal is of length $n = 1024$ and is *exactly* tree-sparse with sparsity parameter $k = \lceil 0.04n \rceil = 41$. We record $m = \lceil 3.5k \rceil = 144$ random Gaussian measurements and perform CS recovery. For comparison, we also include recovery results using CoSaMP and ℓ_1-minimization.

As predicted by our theoretical results, Fig. 2 demonstrates that Alg. 3 achieves accurate signal recovery, while standard CS approaches offer worse recovery quality. In fact, the performance of Alg. 3 is comparable to that of model-CoSaMP with exact projections. Figure 3 demonstrates a similar improvement in the case of a much larger signal. For this experiment, the input signal is a tree-sparse image of size $n = 512 \times 512$ with sparsity parameter $k = 0.04n \approx 10{,}000$, and we use $m = 3.3k \approx 35{,}000$ random Fourier measurements.

Figure 4 plots the results of a Monte Carlo experiment that quantifies the performance of the different recovery algorithms in terms of the number of measurements m. The input test

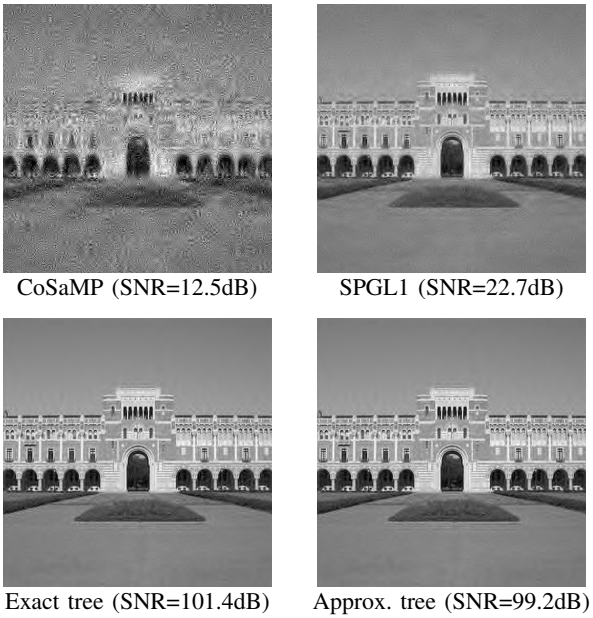

Fig. 3. CS Recovery of a 2D image using various algorithms (signal parameters: $n = 512 \times 512$, $k = 0.04n \approx 10{,}000$, $m = 3.3k \approx 35{,}000$). Both tree-based algorithms accurately recover the ground truth image.

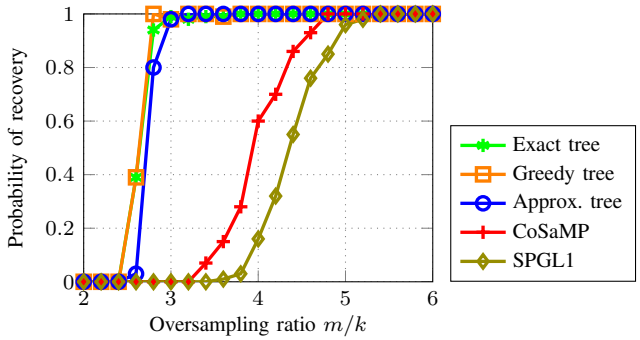

Fig. 4. Comparison of CS recovery algorithms. The probability of recovery is with respect to the measurement matrix and generated using 100 trial runs.

signal is a piecewise polynomial signal, similar to the one in Fig. 2. Each data point in this plot was generated by averaging over 100 sample trials using different measurement matrices. For this plot, "successful recovery" was defined as the event when the ℓ_2-error of the final signal estimate was within 5% of the ℓ_2-norm of the original signal. The success probability of Alg. 3 almost matches the performance of model-CoSaMP with the exact tree-projection.

Table I demonstrates the computational efficiency of our tree-projection algorithm. Using the wavelet coefficients from Fig. 3 as input, our tree-projection algorithm is more than two orders of magnitude faster than the exact tree-projection algorithm ($400\times$ speedup). Moreover, the tree-projection step is not a bottleneck in the overall recovery algorithm since the time spent on multiplying with A and A^T (at least one FFT each) dominates the runtime of our algorithm.

We also compare our algorithm with the greedy tree approximation algorithm described in [3]. While the greedy algorithm offers good recovery and runtime performance in the noiseless setting (see Fig. 4 and Table I), it is susceptible

Algorithm	Exact tree	Approx. tree	Greedy tree	FFT
Runtime (sec)	4.4175	0.0109	0.0092	0.0075

TABLE I
RUNTIMES OF VARIOUS ALGORITHMS ON INPUT DATA FROM FIG. 3 (A SINGLE RUN OF THE PROJECTION ALGORITHM WITHOUT COSAMP). THE TIMES ARE AVERAGED OVER 10 TRIALS. $n \approx 260{,}000$, $k \approx 35{,}000$.

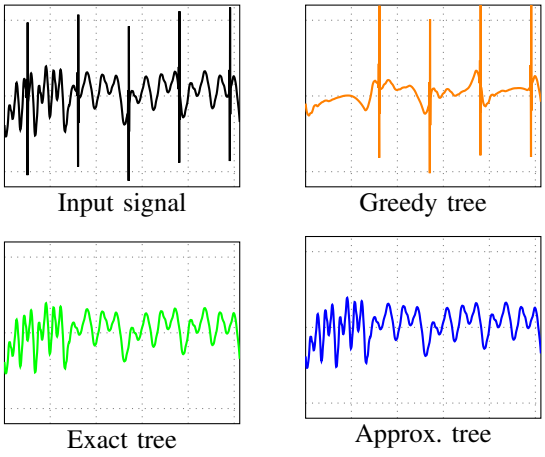

Fig. 5. Tree projection for an input signal with shot noise ($n = 1024$). In order to simplify the example, we show the results of a single tree-projection without compressive sensing recovery. The greedy algorithm fails to find a good support while our approximation algorithm matches the output of the exact tree-projection algorithm.

to shot noise (Fig. 5). In contrast, our algorithm has rigorous approximation guarantees and demonstrates robust behavior similar to the exact tree-projection algorithm.

ACKNOWLEDGEMENTS

This work was supported by grants from the MITEI-Shell program, the MADALGO center, and the Packard Foundation.

REFERENCES

[1] D. Donoho, "Compressed sensing," *IEEE Trans. Inform. Theory*, 2006.
[2] E. Candès, J. Romberg, and T. Tao, "Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information," *IEEE Trans. Inform. Theory*, 2006.
[3] R. Baraniuk, "Optimal tree approximation with wavelets," in *Proc. SPIE Ann. Meeting: Wavelet Apps. Signal Imag. Proc.*, 1999.
[4] R. Baraniuk, V. Cevher, M. Duarte, and C. Hegde, "Model-based compressive sensing," *IEEE Trans. Inform. Theory*, 2010.
[5] S. Mallat, *A Wavelet Tour of Signal Processing*. Academic Press, 1999.
[6] S. Kim and E. Xing, "Tree-guided group LASSO for multi-task regression with structured sparsity," in *Proc. Intl. Conf. Mach. Learning*, 2010.
[7] J. M. Shapiro, "Embedded image coding using zerotrees of wavelet coefficients," *IEEE Trans. Sig. Proc.*, 1993.
[8] M. Crouse, R. Nowak, and R. Baraniuk, "Wavelet-based statistical signal processing using Hidden Markov Models," *IEEE Trans. Sig. Proc.*, 1998.
[9] D. Donoho, "CART and best-ortho-basis: a connection," *Annals of Stat.*, 1997.
[10] R. Baraniuk and D. Jones, "A signal-dependent time-frequency representation: Fast algorithm for optimal kernel design," *IEEE Trans. Sig. Proc.*, 1994.
[11] M. Bohanec and I. Bratko, "Trading accuracy for simplicity in decision trees," *Mach. Learning*, 1994.
[12] C. Carter and A. Thompson, "An exact tree projection algorithm for wavelets," *IEEE Sig. Proc. Let.*, 2013.
[13] M. Duarte and Y. Eldar, "Structured compressed sensing: From theory to applications," *IEEE Trans. Sig. Proc.*, 2011.
[14] D. Needell and J. Tropp, "CoSaMP: Iterative signal recovery from incomplete and inaccurate samples," *Appl. Comput. Harmon. Anal.*, 2009.
[15] C. Hegde, P. Indyk, and L. Schmidt, "Nearly linear-time model-based compressive sensing," in *ICALP*, 2014, to appear.