A greedy algorithm for sparse recovery using precise metrics
Balakrishnan Varadarajan, Sanjeev Khudanpur, Trac D. Tran

Abstract
We explore a new approach to reconstruct sparse signals from a set of projected measurements. Unlike older methods that rely on the near orthogonality property of the sampling matrix $\Phi$ for perfect reconstruction, our approach can be used to reconstruct signals where the columns of the sampling matrix need not be nearly orthogonal. We use the blockwise matrix inversion formula to get a closed form expression for the increase or decrease in the $L_2$ norm of the residue obtained by eliminating or adding (respectively) to the assumed support of the unknown signal $x$. We use this formula to design a computationally tractable algorithm to obtain the non-zero components of the unknown signal $x$. Compared to popular existing sparsity seeking matching pursuit algorithms, each step of the proposed algorithm is locally optimal with respect to the actual objective function. Experiments show that our algorithm is significantly better than conventional techniques when the sparse coefficients are drawn from $\mathcal{N}(0,1)$ or decays exponentially.

I. INTRODUCTION AND MOTIVATION

Let us assume that $x \in \mathbb{R}^N$ is an unknown signal with $\|x\|_0 = \text{supp}(x)$. Let $y \in \mathbb{R}^M$ be an observation of $x$ via $M$ linear measurements represented by a matrix $\Phi$. In other words

$$y = \Phi x$$

We assume that $K \leq \frac{M}{2}$. In this case, it is known that (under some mild assumptions on $\Phi$)

$$x = \arg\min_{y=\Phi w} \|w\|_0$$

The above problem is of interest when we get to observe a linear projection of a signal rather than the original signal itself. It was established by Tropp.et.al that only $O(K \log N)$ measurements are enough to reliably recover a $K$-sparse signal in $\mathbb{R}^N$. The problem of recovering, however, is NP-hard since it requires searching through all possible $N \choose K$ possible column subsets of $\Phi$. It has been shown that if the sampling matrix $\Phi$ further satisfies certain properties relating to near orthogonality, then the following $L_1$ norm optimization reconstructs $x$ exactly:

$$x = \arg\min_{y=\Phi w} \|w\|_1$$

Unfortunately, the complexity of the linear programming algorithms (also known as Basis Pursuit) is in the order of $O(N^3)$, preventing them from practical large-scale applications. Several fast convex relaxation algorithms have been proposed to solve or approximate the solution of BP. The most popular example is the gradient projection method in [5].

An alternative approach to sparse recovery is based on the idea of iterative greedy pursuit, trying to approximate the $\ell_0$ solution directly. The earliest examples include matching pursuit, orthogonal matching pursuit (OMP) [8], and various variants such as stagewise OMP (StOMP) [4] and the regularized OMP (ROMP) [7]. The reconstruction complexity of these algorithms is around $O(KMN)$, which is significantly lower than that of basis pursuit. However, they require more measurements for perfect reconstruction and they lack provable reconstruction quality. More recently, greedy algorithms with a backtracking mechanism such as subspace pursuit (SP) [3] and compressive sampling matching pursuit (CoSaMP) [6] offer comparable theoretical reconstruction quality as that of the linear programming methods and low reconstruction complexity. Our proposed algorithm belongs in this class of recovery.
The projection of \( R \) to indices \( R \) corresponding to indices \( v \). Notation 2. \( \Phi \) for any matrix \( M \times |C| \) whose columns correspond to desirable indices in the support set. Subspace pursuit has two key steps in its iteration.

1) **Expansion step**: At a \( l^{th} \) iteration, if \( T^{l−1} \) is the set of indices corresponding to the current estimate of \( \text{supp}(x) \), \( K \) largest components corresponding to the largest magnitude entries of the residue(a measure of error between the actual vector \( y \) and estimated vector \( \hat{y} \)) is added to \( T^{l−1} \), thereby making a new set \( \tilde{T}^l \).

2) **Contraction step**: Projecting the observation \( y \) onto the support set \( \tilde{T}^l \) gives a projection vector \( x_p \). The largest elements of \( x_p \) is chosen to be \( T^l \).

One obvious drawback with this approach is that in either the expansion or the contraction step, there is no quantification of the qualities of the new support sets. In other words, there is no guarantee that the residual error from the support set \( T^l \) is lower than that of \( T^{l−1} \). In this paper, we propose improvements of the Subspace Pursuit Algorithm described in [3]. We propose algorithms to address the following problems, making sure that its overall complexity (big \( O \)) is comparable to that of the basic SP algorithm.

- Given a current support set \( \mathcal{I} \), let \( y_{r,\mathcal{I}} = \text{residue}(y, \Phi_{\mathcal{I}}) \) denote the residue of \( \mathcal{I} \). For any index \( \mathcal{J} \subset \mathcal{I} \), let \( y_{r,\mathcal{J}} = \text{residue}(y, \Phi_{\mathcal{J}}) \). We compute an exact closed form expression for \( ||y_{r,\mathcal{J}}||_2 - ||y_{r,\mathcal{I}}||_2 \). Using this, we replace the contraction step of the Subspace pursuit.
- Given a current support set \( \mathcal{J} \) and an index \( i \) with \( i \notin \mathcal{I} \), we compute the exact closed form for \( ||y_{r,\mathcal{J}}||_2 - ||y_{r,\mathcal{J} \cup \{i\}}||_2 \). We then use this to replace the expansion step in the subspace pursuit algorithm.

**II. Mathematical Preliminaries**

In this section, we develop the notations necessary for understanding our main result.

**Definition 1.** Consider an arbitrary set \( \mathcal{I} \subset \{1, \ldots, N\} \). The matrix \( \Phi_{\mathcal{I}} \) consists of the columns of \( \Phi \). The span of \( \Phi_{\mathcal{I}} \) is formally defined as

\[
\text{span}(\Phi_{\mathcal{I}}) = \{ y_{\mathcal{I}} : y_{\mathcal{I}} = \Phi_{\mathcal{I}} x, x \in \mathbb{R}^{|\mathcal{I}|}\}
\]

**Notation 1.** For any matrix \( Q \), \( Q[\mathcal{R},\mathcal{C}] \) denotes the sub-matrix of \( Q \) obtained by choosing the rows corresponding to indices \( \mathcal{R} \) in \( Q \) and columns corresponding to indices \( \mathcal{C} \) in \( Q \). For any vector \( v \), \( v_{\mathcal{R}} \) denotes the sub-vector of \( v \) corresponding to indices \( \mathcal{R} \). Similarly for any matrix \( Q \) of size \( M \times N \), we denote \( Q[:,\mathcal{C}] \) as a submatrix of size \( M \times |\mathcal{C}| \) whose columns correspond to \( \mathcal{C} \).

**Notation 2.** (Correlations) The auto-correlation matrix of \( \Phi \), is denoted as \( \Psi = \Phi^T \Phi \). So for any \( \mathcal{I} \subset \{1, \ldots, n\} \) and \( \mathcal{J} \subset \{1, \ldots, n\} \), we have

\[
\Psi[\mathcal{I},\mathcal{J}] = \Phi[:,\mathcal{I}]^T \Phi[:,\mathcal{J}]
\]

Next we use \( \xi \) to denote the cross-correlation between the observation \( y \) and the projection matrix \( \Phi \).

\[
\xi = \Phi^T y
\]

**Definition 2.** The pseudo-inverse of an invertible matrix \( \Phi_{\mathcal{I}} \) is defined as

\[
\Phi_{\mathcal{I}}^+ := (\Phi_{\mathcal{I}}^T \Phi_{\mathcal{I}})^{-1} \Phi_{\mathcal{I}}^T
\]

We also define \( \Phi^\dagger \) to be

\[
\Phi^\dagger := (\Phi^T \Phi)^{-1}
\]

**Definition 3.** The projection of \( y \) onto \( \text{span}(\Phi_{\mathcal{I}}) \) is given by

\[
\text{proj}(y, \Phi_{\mathcal{I}}) = \Phi_{\mathcal{I}} \Phi_{\mathcal{I}}^+ y
\]

\(^1\)Our proposed algorithm uses \( |\mathcal{J}| = |\mathcal{I}| - 1 \) to determine the index to be thrown out.
Definition 4. The residue corresponding to the support set $I$ is defined as

$$\text{residue}(y, \Phi_I) = y - \text{proj}(y, \Phi_I)$$

Definition 5. We define the quality of prediction of a support set $I$ to be simply the $L_2$ norm of the residual error. We represent this quantity as $S_I$

$$S_I = \| \text{residue}(y, \Phi_I) \|_2^2 = y^T y - y^T \Phi_I \Phi_I^+ y$$

(4)

where $\Phi_I^+$ is the Moore-Penrose pseudo-inverse of $\Phi$

Lemma 1. For any $I \subset \{1, \ldots, N\}$, we have $S_I = \min_{x \in \mathbb{R}^{|I|}} \| y - \Phi_I x \|_2^2$

Proof:

$$f(x) = \| y - \Phi_I x \|_2^2 = (y - \Phi_I x)^T (y - \Phi_I x)$$

Since $f(x)$ is convex, the point of minima satisfies $f'(x) = 0$. Invoking the properties of the generalised Moore-Penrose inverse, the optimal $\hat{x}$ satisfies $\hat{x} = \Phi_I^+ y$. This gives the required expression for $S_I$. When $\Phi_I$ is full-rank, the expression boils down to

$$S_I = y^T y - y^T \Phi_I (\Phi_I^T \Phi_I)^{-1} \Phi_I^T y$$

III. ALGORITHM I

In order to motivate our key discoveries, we present the algorithm that can be visually compared with Subspace Pursuit. This algorithm is presented in Algorithm 1. The pseudo-code for GREEDY-ELIMINATE and GREEDY-ADD are presented in Algorithm 2 and 3 respectively.

```
Algorithm:
GREEDY-PURSUIT($y, \Phi, K, \Delta$)
$T^0 := \text{GREEDY-ADD}(y, \Phi, K, \emptyset)$;
$R_0 = y^T y - y^T \Phi_{T^0} \left( \Phi_{T^0}^T \Phi_{T^0} \right)^{-1} \Phi_{T^0}^T y$;
if $\Delta$ is unspecified then
  $\Delta := K$;
end
$l := 1$;
while $\Delta > 0$ do
  $\tilde{T}^l := \text{GREEDY-ADD}(y, \Phi, \Delta, T^{l-1})$;
  $T^{l+1} := \text{GREEDY-ELIMINATE}(y, \Phi, \Delta, \tilde{T}^l)$;
  $R_{l+1} = y^T y - y^T \Phi_{T^{l+1}} \left( \Phi_{T^{l+1}}^T \Phi_{T^{l+1}} \right)^{-1} \Phi_{T^{l+1}}^T y$;
  if $R_{l+1} \geq R_l$ then
    $T_{l+1} := T_l$;
    $R_{l+1} := R_l$;
    $\Delta := \Delta - 1$;
    break;
  end
  $l := l + 1$;
end
return $T_l$
```

Algorithm 1: Algorithm GREEDY-PURSUIT
Algorithm 2: GREEDY-ELIMINATE

\[
\begin{align*}
\mathcal{I} & := \{1, \ldots, \|S\|\}; \\
\xi & := \Phi^T_S y; \\
Q &= (\Phi^T_S \Phi_S)^{-1}; \\
\text{while } |\mathcal{I}| > \|S\| - K \text{ do} \\
& \quad \hat{i} = \text{argmin}_{i \in \mathcal{I}} \frac{(\xi^T \Phi^T_{S-i})^2}{Q[i, i]}; \\
& \quad \mathcal{I} := \mathcal{I} \setminus \hat{i}; \\
& \quad Q[\mathcal{I}, \mathcal{I}] := Q[\mathcal{I}, \mathcal{I}] - \frac{1}{Q[i, i]} Q[\mathcal{I}, \hat{i}] Q[\hat{i}, \mathcal{I}]; \\
\end{align*}
\]

return $S_I$

Algorithm 3: Algorithm GREEDY-ADD

\[
\begin{align*}
\xi & \leftarrow \Phi^T y; \\
\Psi & \leftarrow \Phi^T \Phi; \\
\chi[J, :) & \leftarrow (\Phi^T_J \Phi_J)^{-1} \Phi^T_J \Phi; \\
\mathcal{I} & := J; \\
\text{while } |\mathcal{I}| < |J| + K \text{ do} \\
& \quad \hat{i} = \text{argmax}_{i \in \{1, \ldots, n\} \setminus \mathcal{I}} \frac{(\xi^T \chi[J, i] - \xi)^2}{\Psi[i, i] - \Psi[J, i]^T \chi[J, i]}; \\
& \quad w := \chi[J, \hat{i}]^T \Phi_J^T; \\
& \quad v := w^T \Phi - \Psi[i, :] / \Psi[i, i] - \Psi[J, i]^T \chi[J, i]; \\
& \quad \chi[J, :) \leftarrow \chi[J, :) + \chi[J, \hat{i}] v; \\
& \quad \chi[\hat{i}, :) \leftarrow -v; \\
& \quad \mathcal{I} := \mathcal{I} \cup \hat{i}; \\
\end{align*}
\]

return $\mathcal{I}$

A. Contrast with Subspace Pursuit

At the first glance, the Greedy Pursuit algorithm looks strikingly similar to Subspace Pursuit. In contrast to Subspace Pursuit, our algorithm

- replaces the adhoc correlation based selection of $K$ indices with a sophisticated locally optimal selection,
- replaces the adhoc removal of $K$ indices with a locally optimal selection.

IV. Optimality Analysis

Theorem 1. Consider two sets such that $\mathcal{I} \subset \{1, \ldots, N\}$ and $\mathcal{J} \subseteq \mathcal{I}$. Let $\mathcal{I} = \mathcal{I} \setminus \mathcal{J}$. Let $A = \Phi_J^T [\mathcal{J}, \mathcal{J}]$, $B = \Phi_J^T [\mathcal{J}, \mathcal{J}]$ and $D = \Phi_J^T [\mathcal{J}, \mathcal{J}]$. Then the following fact can be established

\[
S_{\mathcal{J}} - S_{\mathcal{I}} = \xi_{\mathcal{J}}^T B D^{-1} B^T \xi_{\mathcal{J}} + \xi_{\mathcal{J}}^T D \xi_{\mathcal{J}} + 2 \xi_{\mathcal{J}}^T B \xi_{\mathcal{J}} = \left(\xi_{\mathcal{J}}^T B + \xi_{\mathcal{J}}^T D\right) D^{-1} \left(\xi_{\mathcal{J}}^T B + \xi_{\mathcal{J}}^T D\right)^T.
\]

and

\[
\Phi_J^\dagger = A - BD^{-1} B^T.
\]

2 An indepth discussion of optimality is presented in Section.
Another way to compute the residue of $I$ is by using (4). This, however, involves taking pseudo-inverse of a potentially huge matrix. (5) shows how we can compute the quality of a support set($J$) from the parameters of its superset($I$) in an elegant fashion. The cost of computing the formula is analyzed in Section VII. When $|J|=|I|-1$, it is equivalent to computing the inner product of two vectors as described in the corollary below.

**Corollary 1.** If $I \subset \{1, \ldots, N\}$ and let $i \in I$. Denote $J = I \setminus \{i\}$. Then

\[ S_J - S_I = \frac{\left[ \xi_I^T \Phi^\dagger_I [\cdot, i] \right]^2}{\Phi^\dagger_I [i, i]} . \]

**Proof** We postpone the proof of the above theorem to Appendix B.

The subspace pursuit algorithm also performs a greedy elimination step by choosing the $K$ indices corresponding to the largest elements of $\Phi^\dagger_I y$. The following lemma establishes the relation of their approximation with our exact closed form described in (5).

**Lemma 2.** For any $I \subset \{1, \ldots, N\}$, the process of selecting $X \subset I$ with $|X|=K$ corresponding to the largest elements of $\Phi^\dagger_I y$ is the same as solving

\[ X = \arg \max_{J \subset I, |J|=K} \sum_{i \in J} \left( \xi_I^T \Phi^\dagger_I [\cdot, i] \right)^2. \]  

**Proof** Note that $\Phi^\dagger_I y = \Phi^\dagger_T \xi_I$. The top $K$ absolute values of this vector is same as minimizing the expression mentioned in 6.

As is apparent, the greedy elimination scheme in Subspace pursuit algorithm does not involve $\Phi^\dagger_I [i, i]$ in the denominator. Moreover, it disregards the corellation in the errors of two different indices.

**Algorithm GREEDY-ELIMINATE:**

Based on Corollary 1, we propose a locally optimal algorithm called GREEDY-ELIMINATE (described in Algorithm 2) which attempts to solve the NP-hard optimization

\[ \arg \min_{J \subset I} (S_J - S_I). \]

This is achieved by calling GREEDY-ELIMINATE($y, \Phi_I, K$) 4.

**Lemma 3.** For any $J \subset \{1, \ldots, N\}$ and $i \not\in J$, let $I = J \cup \{i\}$. Then

\[ S_J - S_I = \frac{\left( \xi_J - \xi_I^T \Phi^\dagger_J \Psi [J, i] \right)^2}{\Psi [i, i] - \Psi [J, i]^T \Phi^\dagger_J \Psi [J, i]}. \]  

**Proof** The result directly follows by inserting the equations for the blockwise inversion formulas described in (13), (14) and (15) into (8).

Once again we establish the resemblance between GREEDY-ADD and the addition step in Subspace Pursuit by going over the following lemma.

3This corresponds to the contraction step of the SP algorithm (Step 3 of iteration, Dai and Milenkovic 3).
4This algorithm can potentially replace the contraction step of the SP algorithm.
Lemma 4. For any $\mathcal{J} \subset \{1, \ldots, N\}$, the process of choosing $\mathcal{X} \subseteq \{1, \ldots, N\} \setminus \mathcal{J}$ with $|\mathcal{X}| = K$ corresponding to the largest magnitude entries of the vector $\Phi^T \text{residue}(y, \Phi_\mathcal{J})$ is equivalent to

$$\mathcal{X} = \arg\max_{\mathcal{Y} \subseteq \{1, \ldots, N\} \setminus \mathcal{J}, |\mathcal{Y}| = K} \sum_{i \in \mathcal{Y}} \left( \xi_i - \xi^T_\mathcal{J} \Phi^+_{\mathcal{J}} \Psi[\mathcal{J}, i] \right)^2. \quad (8)$$

Proof First note that $\Phi^T \text{residue}(y, \Phi_\mathcal{J}) = \Phi^T y - \Phi^T \Phi_{\mathcal{J}} \Phi^+_{\mathcal{J}} \Phi^T y$. Recall that $\Phi^T \Phi_{\mathcal{J}} = \Psi[; \mathcal{J}]$ and $\Phi^T y = \xi$. Hence $\Phi^T \text{residue}(y, \Phi_\mathcal{J}) = \xi - \xi \Phi^+_{\mathcal{J}} \Psi[; \mathcal{J}]$. Choosing the top $K$ components of this vector is the same as maximizing $\xi$.

Yet again the selection architecture in Subspace pursuit does not involve the denominator term. It is of interest to note that this term is none other than the Schur complement. The Schur complement captures the correlation I between the $i^{th}$ column representing the quantity of interest). The following lemma enables us to iteratively compute $\chi_I$ from $\chi_{\mathcal{J}}$ in an elegant fashion.

Lemma 5. For any $S \subseteq \{1, \ldots, N\}$, let $\chi_S = \Phi^+_{\mathcal{S}} \Phi^T_{\mathcal{S}} \Phi$. Consider a set $\mathcal{J} \subseteq \{1, \ldots, N\}$ and let $i \notin \mathcal{J}$. Denote $\mathcal{I} = \mathcal{J} \cup \{i\}$. Without loss of generality, assume $\mathcal{I} = \left[ \begin{array}{c} \mathcal{J} \\ i \end{array} \right]$. It follows that

$$\chi_\mathcal{I} = \left[ \begin{array}{c} \chi_\mathcal{J} + \chi_\mathcal{J} |; ; i | \Psi[J, i] \\ -\Psi[i, i] \end{array} \right]. \quad (9)$$

where

$$v = \frac{\chi_\mathcal{J} |; ; i | \Phi_\mathcal{J}^T \Phi - \Psi[i, ;]}{\Psi[i, i] - \Psi[\mathcal{J}, i]^T \chi_\mathcal{J} |; ; i |}.$$

Proof: We request the reader to refer Appendix C for the proof.

Algorithm GREEDY-ADD:

Based on Lemma 3 and Lemma 5, we propose an approximate algorithm called GREEDY-ADD (described in Algorithm 3) that attempts to solve the NP-hard optimization

$$\mathcal{I} = \arg\max_{\mathcal{X} \subseteq \{1, \ldots, N\} \setminus \mathcal{J}, |\mathcal{X}| = K} \left( S_{\mathcal{J}} - S_{\mathcal{J} \cup \mathcal{X}} \right).$$

Algorithm GREEDY-PURSUIT

Based on these tools, we propose a greedy residual minimizing algorithm to discover the non-zero components of an unknown signal $x$. Assuming that we know the number of non-zero coefficients, the procedure GREEDY-PURSUIT initially calls GREEDY-ADD($y, \Phi, K, \emptyset$) to get the first estimate of the support$^5$. Given a current estimate of the support $T^{-i}$ at the $i^{th}$ iteration, we call procedure GREEDY-ADD to increase the size of the set to $K + \Delta$. Next, we use GREEDY-ELIMINATE to reduce this to a set of size $K$. This now becomes our new estimate $T^{i+1}$. It is easy to see that when $\Delta = 1$, $S_{T^{-i}} - S_{T^{i+1}} \geq 0$ since each of the addition and removal

$^5$This corresponds to the expansion step of the SP algorithm(Step 1 of iteration, Dai and Milenkovic).  
$^6$This assumption makes the form of (9) easier to write. Note that if the eliminated column $i$ is not the last column in $\mathcal{I}$, then a simple row-column shuffling of $\Phi_x$ will make it so.  
$^7$It is of interest to note that this estimate itself is of high quality.
steps are optimal. When $\Delta > 1$, there is no such guarantee. However, as we will see in the simulations, larger values of $\Delta$ have higher expected reduction in residue. The analogue of $\Delta$ in the subspace pursuit algorithm is set to $K$, which we use for our experiments as well. Another useful observation is that the algorithm is faster for lower values of $\Delta$. Therefore lower values of $\Delta$ is preferable when running time needs to be small.

A. Numerical Comparisons of Optimality

Before we move on to our main experiments, we do some empirical analysis of the various sub-parts of our algorithm. The motive for this analysis is to quantify the differential gain of our algorithms at various stages.

1) Greedy Add: As already stated, Algorithm $\text{Algorithm}$ can be used to replace the expansion step of the SP algorithm. In this section, we make a simple simulation that enables this comparison empirically. The sequence of steps in our experiments is described below.

- Fix a $K$ such that $1 \leq K \leq M$. We repeat the following steps several times.
  - Generate a $M \times N$ matrix $\Phi$ where each entry is generated from $N(0, 1)$. Normalize each of the $N$ columns.
  - Randomly pick a vector $y \in \mathbb{R}^M$ on the unit $M$ dimensional sphere $S_M$. This can be done by choosing each element of $y$ randomly from $N(0, 1)$ and then normalizing $y$.
  - Let $I = \text{GREEDY-ADD}(y, \Phi, K, \emptyset)$ and let $\tilde{I}$ be obtained by choosing the top $K$ largest magnitude components of $\Phi^T y$. Compute the values of $\log \left[ \text{residue}(y, \Phi_I) \right]$ and $\log \left[ \text{residue}(y, \Phi_{\tilde{I}}) \right]$.
  - Compute the empirical average values of $\log \left[ \text{residue}(y, \Phi_I) \right]$ and $\log \left[ \text{residue}(y, \Phi_{\tilde{I}}) \right]$ from the above simulations.

We compare the empirical average values of the log residues of the GREEDY-ADD algorithm with that used by the SP algorithm for different values of $K$, fixing $M = 64$ and $N = 128$. The improvement is illustrated in Figure 1(a). The rate at which the log-residue decays using the GREEDY-ADD strategy significantly outperforms that of the SP throwing step.

2) Greedy Eliminate: We have already shown how we can replace the contraction step of the SP algorithm with our proposed greedy eliminate algorithm (Algorithm $\text{Algorithm}$). In this subsection, we provide quantitative comparisons between our algorithm with the contraction step of the SP algorithm. Recall that the contraction step simply projects the observed vector $y$ onto the given support $I$. It then selects the components with the highest magnitude. We perform the following sequence of steps for our experiment:

- Fix a $K$ such that $1 \leq K < M-4$. We repeat the following steps several times:
  - Generate a $M \times (M-4)$ matrix $\Phi$ with each entry drawn from $N(0, 1)$. Normalize each of the $K$ columns.
  - Randomly pick a vector $y \in \mathbb{R}^M$ on the unit $m$ dimensional sphere $S_M$.
  - Let $J = \text{GREEDY-ELIMINATE}(y, \Phi, K, \{1, \ldots, M-4\})$ and let $\tilde{J}$ be obtained by simply choosing the top $M - 4 - K$ magnitude components of $\Phi^T_{\{1,\ldots,M-4\}} y$. Compute the values of $\log \left[ \text{residue}(y, \Phi_J) \right]$ and $\log \left[ \text{residue}(y, \Phi_{\tilde{J}}) \right]$.
  - Compute the empirical average values of $\log \left[ \text{residue}(y, \Phi_J) \right]$ and $\log \left[ \text{residue}(y, \Phi_{\tilde{J}}) \right]$ from the above simulations.

We compare the empirical average values of the log residues of the GREEDY-ELIMINATE algorithm with that employed in the SP algorithm for different values of $K$, fixing $M = 64$ and $N = 128$. The accuracy gain is illustrated in figure 1(b). Again, we have shown that our elimination strategy is much superior to that of the naive projection used by the SP algorithm. $\text{Algorithm}$

---

8. We, therefore, decrement the value of $\Delta$ every time we find that the current value of $\Delta$ does not increase the quality of prediction.

9. We choose $K$ to be less than $M$ only owing to precision issues.

10. The initial residues are the same since both algorithms have the same starting condition. The final residue would be close to $\|y\|$ as a result of which both curves converge to zero.
3) Overall comparison of reconstruction accuracy: In the previous two sections, we have demonstrated the local optimality of our algorithm. In this section, we investigate the procedure we presented in Algorithm 1 on synthetic data. We first evaluate it on a setup similar to that mentioned in [3]:

- Randomly generate a gaussian ensemble $\Phi$ of size $m \times N$. In our experiments, we choose $m = 128$ and $N = 256$ respectively.
- The input vector $x$ is generated using either of the following criteria:
  1) We randomly choose $K$ indices of the vector $x$. Each of the $K$ indices are set to a non-zero value each drawn from $\mathcal{N}(0, 1)$.
  2) We randomly choose $K$ indices of the vector $x$. Each of the $K$ indices are set to a non-zero value such that the values decay exponentially (when ordered) from 1 to 0.1.
- We repeat the process 1000 times for each $K$ varying from 1 to $\frac{M}{2}$. The frequency of exact reconstruction is then noted.

As depicted in Figure 1(c), we significantly outperform the Subspace Pursuit and Basis Pursuit. Figure 1(d) show our performance relative to the standard baselines when the non-zero entries of the sparse signal are drawn according to $\mathcal{N}(0, 1)$. 

(c) Magnitude of $\text{supp}(x)$ decay exponentially

(d) Non-zero entries drawn from $\mathcal{N}(0, 1)$. 
A. Computational complexity of Greedy Eliminate

Computing $Q$ (refer to line 3 of Algorithm 2) requires inverting a $|S| \times |S|$ matrix which takes up to $O(|S|^3)$. Computing the minimum over $i \in I$ (refer to line 6) requires computing $I$ dot-products. Hence has a complexity of $O(|I|^2)$. Updating $Q$ also requires $O(|I|^2)$ time. Hence the entire complexity of the while loop would be $O(|S|^3)$. The complexity of GREEDY-ELIMINATE is thus $O(|S|^3)$.

1) Generalised Greedy Eliminate: In line 6 of Algorithm 2 we compute the increase in residual error upon eliminating a single index $i \in I$. However theorem 11 gives a general expression for the increase in the residual error upon eliminating a set of indices $J \subseteq I$. The generalised greedy eliminate algorithm will take in an additional parameter $\delta$. At each step in the iteration, we could use 5 to compute the increase in squared error for every subset $J \subseteq I$ with $|J| = \delta$. The elimination strategy would perform $\lceil \frac{n}{\delta} \rceil$ steps. In each step the quantity in 5 requires computing the vector $\xi^T J B + \xi^T J D$ and the matrix $D^{-1}$. This requires $O(|J| + \delta^3)$ time. There are $O(|I|^\delta)$ possible subsets of size $\delta$. Thus the overall complexity for generalised greedy eliminate would be $O\left(\frac{K^{\frac{\delta+1}{\delta}}}{\delta} + \delta^2 K^{\delta+1}\right)$.

B. Computational complexity of Greedy Add

Since $\Psi$ can pre-computed, initializations (upto line 3 of Algorithm 3) require $O(|J|^3)$. $\chi$ is initially a $|J| \times N$ matrix and $\xi$ is a $M \times 1$ vector. Therefore $\xi^T J \chi[I,i] - \xi_i$ is a scalar that requires $O(|I|)$ time to be computed. Similarly computing $\Psi[i,i] - \Psi[I,i]^T \chi[I,i]$ requires $O(|I|)$ time. Hence maximizing over all $i \in \{1, \ldots, N\} \setminus I$ requires $O(N|I|)$ time. Computing $\chi[I,i]^T \Phi_J^T$ requires $O(N|I|)$ time. However the bottleneck is in computing the vector $v$ which requires computing $w\Phi$. This computation takes $O(MN)$ time. Hence the complexity of each while loop is $O(MN)$. The net complexity of the algorithm is therefore $O(|J|^3) + O(MNK)$. Since $M = O(K)$, $|J| < m$ and $n >> m$, we have this complexity to be $O(MNK)$. One could also formulate a generalised greedy add algorithm by deriving a more general version of the formula 7. We would then use this to replace line 5 of the algorithm where we would minimize the increase in squared error over all $K \subseteq \{1, \ldots, N\} \setminus I$ with $|K| = \delta$.

VI. CONCLUSION AND APPLICATIONS

In this article, we presented a novel greedy algorithm which is proven to be locally optimal. Although the original problem is NP-hard, our technique follows an accurate gradient ascent approach to compute the next optimal solution. Hence the final solution is an optimal solution. One could use various initializations to obtain several different optimal solutions to the problem and finally pick the best one. We have shown that our algorithm is particularly good for exponential and gaussian sparse signals as illustrated in Figures 1(c) and 1(d). This closely simulates the real data scenario and one could directly apply this technique to problems such as image compression.

APPENDIX A

PROOF OF THEOREM 11

Proof  Recall that

$$S_I = y^T y - \left(\Phi_I^T y\right)^T \left(\Phi_I^T \Phi_I\right)^{-1} \Phi_I^T y \quad (10)$$
$$S_J = y^T y - \left(\Phi_J^T y\right)^T \left(\Phi_J^T \Phi_J\right)^{-1} \Phi_J^T y \quad (11)$$

Without loss of generality assume $I = \begin{bmatrix} J & \tilde{J} \end{bmatrix}$

This allows us to write

$$\Phi_I^T \Phi_I = \begin{bmatrix} \Phi_J^T \Phi_J & \Phi_J^T \Phi_{\tilde{J}} \\ \Phi_{\tilde{J}}^T \Phi_J & \Phi_{\tilde{J}}^T \Phi_{\tilde{J}} \end{bmatrix}$$
We note that
\[ \Phi^T_I = \begin{bmatrix} \Phi^T_T \xi \\ \Phi^T_T \chi \end{bmatrix} \]

Set \( A = \Phi^T_J \Phi_J, \ B = \Phi^T_J \Phi_{\overline{J}}, \ C = \Phi^T_J \Phi_{\overline{J}} \) and \( D = \Phi^T_J \Phi_J \).

So we have
\[ \Phi^T_I = (\Phi^T_I \Phi_I)^{-1} = \begin{bmatrix} A & B \\ B^T & D \end{bmatrix}^{-1} = \begin{bmatrix} A & B \\ B^T & D \end{bmatrix} \] (12)

The components \( A, B, D \) can be computed by the blockwise inversion formula as
\[
\begin{align*}
A &= A^{-1} + A^{-1}B(D - B^T A^{-1}B)^{-1}B^T A^{-1} \\
B &= -A^{-1}B(D - B^T A^{-1}B)^{-1} \\
D &= (D - B^T A^{-1}B)^{-1}
\end{align*}
\] (13) (14) (15)

Using these and noting that \( A, D, A, D \) are symmetric matrices, we can write
\[ (\Phi^T_J \Phi_J)^{-1} = A^{-1} = A - BD^{-1}B^T \] (16)

Using (12), we can expand (10) as
\[ S_I = y^T y - (\Phi^T_J y)^T A (\Phi^T_J y) - (\Phi^T_J y)^T D (\Phi^T_J y) - 2 (\Phi^T_J y)^T B (\Phi^T_J y) \] (17)

Using (16), we can write (11) as
\[ S_J = y^T y - (\Phi^T_J y)^T A (\Phi^T_J y) + (\Phi^T_J y)^T B D^{-1}B^T (\Phi^T_J y) \] (18)

Hence the increase in the squared error can be written as
\[ S_J - S_I = (\Phi^T_J y)^T B D^{-1}B^T (\Phi^T_J y) + (\Phi^T_J y)^T D (\Phi^T_J y) + 2 (\Phi^T_J y)^T B (\Phi^T_J y) \]

Noting that \( \Phi^T_J y = \xi \), we have the result of the proposed theorem.

APPENDIX B

PROOF OF COROLLARY 1

Proof We note that \( \xi^T_J B \) and \( D \) are scalars. Using this, we can rewrite (5) as
\[ S_J - S_I = \frac{[\xi^T_J B]^2}{D} + \xi^2_J D + 2 \begin{bmatrix} \xi^T_J B \end{bmatrix} \xi_J = \frac{1}{D} (\xi^T_J B + \xi_J D)^2 \] (19)

Finally note that \( \Phi^T_I [\cdot, i] = \begin{bmatrix} B \\ D \end{bmatrix} \) and \( \xi_I = \begin{bmatrix} \xi_J \\ \xi_J \end{bmatrix} \). Hence (19) becomes
\[ S_J - S_I = \frac{[\xi^T_I \Phi^T_I [\cdot, i]]^2}{\Phi^T_I [i, i]} \]

APPENDIX C

PROOF OF LEMMA 2

Note that
\[ \chi_I = \begin{bmatrix} A & B \\ B^T & D \end{bmatrix} \begin{bmatrix} \Phi_J^T & \Phi_{\overline{J}}^T \end{bmatrix} \Phi = \begin{bmatrix} A \Phi_J^T \Phi + B \Phi_{\overline{J}}^T \Phi \\ B^T \Phi_J + D \Phi_{\overline{J}} \end{bmatrix} \] (20)

Plugging the expressions of \( A, B, D \) from (13), (14) and (15) yields the proposed result.
REFERENCES