FROGS: A Serial Reversible Greedy Search Algorithm

Dennis Sundman, Saikat Chatterjee, Mikael Skoglund
School of Electrical Engineering and ACCESS Linnaeus Centre
KTH - Royal Institute of Technology, SE-10044 Stockholm, Sweden
denniss@kth.se, sach@kth.se, skoglund@kth.se

***Abstract*—For compressed sensing, in the framework of greedy search reconstruction algorithms, we introduce the notion of initial support-set. The initial support-set is an estimate given to a reconstruction algorithm to improve the performance of the reconstruction. Furthermore, we classify existing greedy search algorithms as being serial or parallel. Based on this classification and the goal of robustness to errors in the initial support-sets we develop a new greedy search algorithm called FROGS. We end the paper with careful numerical experiments concluding that FROGS perform well compared to existing algorithms (both in terms of performance and execution time) and that it is robust against errors in the initial support-set.**

***Index Terms*—Compressed sensing, greedy search, greedy pursuit, initial support.**

I. Introduction

Compressed sensing (CS) [1], [2] refers to a linear undersampling problem, where the underlying sampled signal is inherently sparse. The challenge in CS is to reconstruct the sparse signal from as few measurements as possible. Since CS benefits from sparsity in signals, this technique has in recent works been considered in scenarios where correlation among multiple data-sets can provide for additional sparsity. Examples of such scenarios are power spectrum estimation in sensor networks where data is correlated both among nodes as well as over time [3], [4] and in magnetic resonance imaging [5], [6] where consecutive snapshots are correlated.

In the literature, there are three main classes of reconstruction algorithms: convex relaxation, non-convex and greedy-search. The greedy search (GS) algorithms are particularly interesting since they provide good performance while maintaining a low computational complexity. In practical scenarios GS algorithms provide equivalent performance to convex-relaxation based methods at much lower complexity. The GS algorithms use computationally simple linear algebraic tools in an iterative manner to first detect a support-set and then, based on the support-set, estimate the signal. Since the key in the GS algorithms is to detect the support-set, providing an initial support-set estimate to the solution algorithm should provide for improved performance. The motivation to using initial support-sets comes from the way multiple data often is correlated. We further discuss the background of initial support-sets in section II-A.

The main contributions of this paper are two-fold: (1) A fundamental classification of existing GS algorithms in terms of their different behavior, and (2) based on the previous categorization and the initial support-set information, we development of a new greedy search algorithm, forward-reverse orthogonal greedy search (FROGS). For the first contribution, we notice two independent categories under which the common GS algorithms can be classified: (1) serial or parallel and (2) reversible or irreversible. We notice that reversible algorithms are particularly suited for accepting an initial support-set estimate since they can correct for potential errors in the estimate. Furthermore, to the best of the authors' knowledge there is only one existing algorithm in the literature that can be classified as being both serial and reversible: cyclic matching pursuit (CMP) [7], [8]. Thus, for the second contribution, we design FROGS which is a serial and reversible algorithm carefully designed to be robust to errors in the initial support-set while still benefiting from the good parts in the initialization.

This work was funded in part by VINNOVA.

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

II. Signal Model and Problem Formulation

In the standard CS problem, we acquire (i.e., sample) a K-sparse signal $\mathbf{x} \in \mathbb{R}^N$ via the linear measurements

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

where $\mathbf{A} \in \mathbb{R}^{M \times N}$ is a random matrix representing the sampling procedure, $\mathbf{w} \in \mathbb{R}^M$ represents some measurement noise (we assume $w_i \sim \mathcal{N}(0, \sigma_w)$) and $\mathbf{y}$ is the measurement data. We refer to the set of indices corresponding to the non-zero elements in $\mathbf{x}$ as the support-set of $\mathbf{x}$, denoted $\mathcal{T} \triangleq \{\forall i : x_i \neq 0\}$ and its complement $\bar{\mathcal{T}} \triangleq \{\forall i : x_i = 0\}$. With the notion of support-set we can denote the active components of $\mathbf{x}$ by $\mathbf{x}_{\mathcal{T}}$, where $\mathbf{x}_{\mathcal{T}}$ refers to selecting only the non-zero components in $\mathbf{x}$. Consequently, we have that $\mathbf{x}_{\bar{\mathcal{T}}} = \mathbf{0}$. From the measurement data $\mathbf{y}$ and the measurement matrix $\mathbf{A}$, the standard CS problem boils down to finding $\mathbf{x}$.

In the next section we will introduce the notion of initial support-set $\mathcal{T}_{\text{ini}}$, used as initial knowledge given to a reconstruction algorithm to improve the solution.

A. Initial Support-set

In a local cognitive sensor network where the sensors are estimating the power spectrum, several parts of the underlying

signal will be highly correlated due to strong transmitters in the vicinity. In particular, the location in the frequency band of several peaks in the power spectrum will be shared among the sensors [9]. Thus by sharing support-set information, parts of a prior support-set can be estimated and used as initialization to an algorithm to improve the local solution [10]. Similarly, for tracking a single signal which is slowly varying in time [3], feeding a support-set estimate from a previous state as an initial knowledge in the current state may improve the performance significantly. Based on the above arguments we introduce the notion of initial support-set, referred to as

$$\mathcal{T}_{\text{ini}} = \mathcal{T}_{\text{part}} \cup \mathcal{T}_{\text{err}}, \tag{2}$$

where $\mathcal{T}_{\text{part}}$ is a correctly estimated part and $\mathcal{T}_{\text{err}}$ is erroneously estimated.

B. Classifications of GS Algorithms

Using the measurement vector collected from the sensor, the main principle of the GS algorithms is to estimate the underlying support-set of $\mathbf{x}$ followed by evaluating the associated signal values. To estimate the support-set and the associated signal values, the GS algorithms use linear algebraic tools in an iterative manner. Examples of tools used are the matched filter detection and least-squares estimation. A crucial point worth mentioning is that the success of the GS algorithms mainly depends on their efficiency in estimating the support-set. Once the support-set is found, the associated signal values can be obtained by a simple least-squares estimation. It thus makes sense that if a perfect initial support-set $\mathcal{T}_{\text{ini}}$ (see section II-A) is available, the performance of the GS algorithms will improve.

In the literature, we note two main algorithmic approaches for the GS algorithms: (1) the categorization of *serial* or *parallel*, and (2) *reversible* or *irreversible* construction mechanism. First let us consider the algorithmic approach of serial or parallel support-set construction strategy. If serial construction is performed then elements of the support-set are chosen one-by-one; in contrast, for parallel construction, several elements of the support-set are chosen simultaneously. Next we consider the algorithmic approach of reversible and irreversible construction. If irreversible construction is performed then an element already added to the support-set, remains there forever; in contrast, for reversible construction, an element of the support-set (chosen in the past) can be removed later (if the element is found to be unreliable). Therefore, considering serial or parallel construction, a GS algorithm can be categorized either as a serial pursuit (S-pursuit) or a parallel pursuit (P-pursuit) algorithm. On the other hand, considering reversible or irreversible, a GS algorithm can either use a reversible support-set (R-support) or an irreversible support-set (I-support) construction mechanism.

We categorize several GS algorithms in Table I where we consider existing OMP [11], SP [12], CoSaMP [13], look ahead orthogonal least-squares (LAOLS) [14], stagewise omp (StOMP) [15], backtracking OMP (BAOMP) [16], projection-based OMP (POMP) [14], look ahead parallel pursuit (LAPP) [17], regularized OMP (ROMP) [18], cyclic matching pursuit (CMP), and the new forward-reverse orthogonal greedy search (FROGS) algorithm.

TABLE I: Classification of GS algorithms

	P-pursuit	S-pursuit
R-support	SP, CoSaMP, LAPP, BAOMP	CMP, FROGS
I-support	StOMP, ROMP	OMP, LAOLS, POMP

C. Algorithmic Notation

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

$$\texttt{resid}(\mathbf{y}, \mathbf{B}) \triangleq \mathbf{y} - \mathbf{B}\mathbf{B}^{\dagger}\mathbf{y}, \tag{3}$$

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

$$\texttt{max\_indices}(\mathbf{x}, k) \triangleq \{\textit{the set of indices corresponding to the } k \textit{ largest amplitude components of } \mathbf{x}\}. \tag{4}$$

III. FROGS Algorithm

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

For developing FROGS, we refer the serial add strategy of including a potential element in the support-set as *forward add*. This forward add strategy is directly used in standard OMP and we present it as a separate algorithm in Algorithm 1. In Algorithm 1, we supply the inputs: $\mathbf{A}$, $\mathbf{r}_k$, $\mathcal{T}_k$. Here, $\mathbf{A}$ is the sensing matrix, $\mathbf{r}_k$ is the residual vector for iteration k and $\mathcal{T}_k$ is the support-set estimate for iteration k. Then, analogous to OMP, $\texttt{max\_indices}(.,.)$ and $\texttt{resid}(.,.)$ are used and the algorithm outputs the residual $\mathbf{r}_{k+1}$ and support-set $\mathcal{T}_{k+1}$ for iteration $k+1$. Now using Algorithm 1, we define the following function.

Algorithm 1 : Forward-add

Input: $\mathbf{A}$, $\mathbf{r}_k$, $\mathcal{T}_k$

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

Output: $\mathbf{r}_{k+1}$, $\mathcal{T}_{k+1}$

Function 1. (Forward-add) For the k'th iteration, the sensing matrix $\mathbf{A}$, the current residual $\mathbf{r}_k$ and the current support-set $\mathcal{T}_k$ are given. Then, for the $(k+1)$'th iteration, the new

support-set with cardinality $(|\mathcal{T}_k|+1)$ and its corresponding residual are the outputs of the following algorithmic function

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

which exactly executes Algorithm 1.

After the forward-add strategy is performed, it is natural to include a reliability testing strategy. For this, we develop a new scheme where the k most prominent support-set elements are chosen from $(k+1)$ elements. This new selection algorithm is presented in Algorithm 2. In Algorithm 2, we supply the

Algorithm 2 : Reverse-fetch

Input: $\mathbf{A}$, $\mathcal{T}_{k+1}$, k (Note: $|\mathcal{T}_{k+1}| = k+1$)

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

Output: $\mathbf{r}'$, $\mathcal{T}'$

inputs $\mathbf{A}$, $\mathcal{T}_{k+1}$ and k. By using least squares estimation, we find an estimate of the intermediate $k+1$ non-zero elements of the sparse signal $\tilde{\mathbf{x}}$. Based on this signal estimate, the temporary support-set $\mathcal{T}'$ of cardinality k and the corresponding temporary residual $\mathbf{r}'$ are found. Using Algorithm 2 we define the following function.

Function 2. (Reverse-fetch) Let the sensing matrix $\mathbf{A}$ and a support-set $\mathcal{T}_{k+1}$ of cardinality $k+1$ be given. Then the temporary support-set $\mathcal{T}'$ with cardinality k and its corresponding residual are the outputs of the following algorithmic function

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

which exactly executes Algorithm 2.

In Fig. 1, we provide a schematic figure on how the two functions `forward_add()` and `reverse_fetch()` effect the support-set at iteration k. This figure may provide increased insight in the FROGS algorithm itself which is presented in Algorithm 3.

The inputs to FROGS are $\mathbf{y}$, $\mathbf{A}$, K and $\mathcal{T}_{\text{ini}}$. In the *initialization* phase FROGS calls a modified version of OMP (see Section III-A) procedure (step 2) to form a full support-set estimate $\mathcal{T}_0$. If there are errors in $\mathcal{T}_{\text{ini}}$, those errors will remain in $\mathcal{T}_0$ (and in $\mathbf{x}_0$). Then, in steps 3 to 6, an ordering procedure is performed which helps to arrange the corresponding residual vectors appropriately. This ordering is necessary for the reliability testing. Notice that the iteration phase starts with $k = K$. For clarity, we denote $\mathbf{r}_k$, $\mathbf{r}_{k+1}$ and $\mathbf{r}'$ as the current, intermediate and temporary residuals, respectively. In the k'th *iteration*, two main tasks are performed. First, when the algorithm performs `forward_add()`, the output is an intermediate support-set $\mathcal{T}_{k+1}$ with cardinality larger than the current support-set by one. Second, for reliability testing, the `reverse_fetch()` function is invoked to find the k elements from the intermediate support-set of cardinality $(k+1)$. These k elements form the temporary support-set

Algorithm 3 : Forward-reverse orthogonal greedy search (FROGS)

Input: $\mathbf{y}$, $\mathbf{A}$, K, $\mathcal{T}_{\text{ini}}$

Initialization:

1: $\mathbf{R} = [\mathbf{r}_1\ \mathbf{r}_2\ \ldots, \mathbf{r}_K]$ (For storing residuals)
2: $(\mathcal{T}_0, \mathbf{x}_0) \leftarrow \text{OMP}(\mathbf{y}, \mathbf{A}, K, \mathcal{T}_{\text{ini}})$
3: **for** $l = 1 : K$ **do**
4: $\quad\mathcal{T}' \leftarrow \texttt{max\_indices}(\mathbf{x}_0, l)$ (Note: $|\mathcal{T}'| = l$)
5: $\quad\mathbf{r}_l \leftarrow \texttt{resid}(\mathbf{y}, \mathbf{A}_{\mathcal{T}'})$
6: **end for**
7: $k \leftarrow K$, $\mathcal{T}_k \leftarrow \mathcal{T}'$

Iteration:

1: **repeat**
2: $\quad(\mathbf{r}_{k+1}, \mathcal{T}_{k+1}) \leftarrow \texttt{forward\_add}(\mathbf{A}, \mathbf{r}_k, \mathcal{T}_k)$
3: $\quad$**repeat**
4: $\quad\quad(\mathbf{r}', \mathcal{T}') \leftarrow \texttt{reverse\_fetch}(\mathbf{A}, \mathcal{T}_{k+1}, k)$
5: $\quad\quad$**if** $(\|\mathbf{r}'\| < \|\mathbf{r}_k\|)$ **then**
6: $\quad\quad\quad\mathcal{T}_k \leftarrow \mathcal{T}'$, $\mathbf{r}_k \leftarrow \mathbf{r}'$
7: $\quad\quad\quad k \leftarrow k-1$
8: $\quad\quad$**else**
9: $\quad\quad\quad$break
10: $\quad\quad$**end if**
11: $\quad$**until** $(k = 0)$
12: $\quad k \leftarrow k+1$
13: **until** $k = K+1$

Output:

1: $\hat{\mathcal{T}} = \mathcal{T}_{k-1}$
2: $\hat{\mathbf{x}}$ such that $\hat{\mathbf{x}}_{\hat{\mathcal{T}}} = \mathbf{A}^{\dagger}_{\hat{\mathcal{T}}}\mathbf{y}$ and $\hat{\mathbf{x}}_{\bar{\hat{\mathcal{T}}}} = \mathbf{0}$
3: $\gamma = \|\mathbf{r}_{k-1}\|$

Functional form: $(\hat{\mathcal{T}}, \hat{\mathbf{x}}, \gamma) \leftarrow \text{FROGS}(\mathbf{A}, K, \mathbf{y}, \mathcal{T}_{\text{ini}})$

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

By design, FROGS will accept an initial support-set but in order to let other algorithms accept the same prior knowledge, we need to modify them. This modification is presented next.

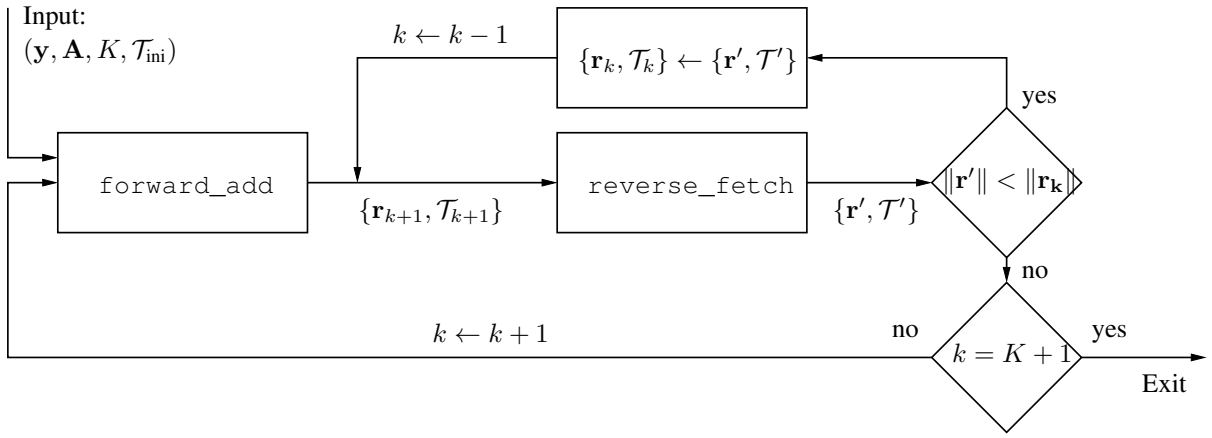

Fig. 1: Flow chart diagram of FROGS

A. Modifications to Existing Algorithms

In this paper, we are interested in algorithms that can benefit from provided information in terms of parts of an estimated initial support-set. Some very slight changes in the initialization phase of competing algorithms has to be made in order for them to gain from this support-set. These changes correspond to replacing

$$\begin{array}{lll} \hat{\mathcal{T}} \leftarrow \emptyset & & \hat{\mathcal{T}} \leftarrow \mathcal{T}_{\text{ini}} \\ \mathbf{r} \leftarrow \mathbf{y} & \text{with} & \mathbf{r} \leftarrow \texttt{resid}(\mathbf{y}, \mathbf{A}_{\hat{\mathcal{T}}}) \\ \hat{\mathbf{x}} \leftarrow \mathbf{0} & & \hat{\mathbf{x}} \quad \text{such that} \quad \hat{\mathbf{x}}_{\hat{\mathcal{T}}} = \mathbf{A}_{\hat{\mathcal{T}}}^{\dagger}\mathbf{y} \text{ and } \hat{\mathbf{x}}_{\bar{\hat{\mathcal{T}}}} = \mathbf{0} \end{array},$$

in the initialization phase and this is done to all competing algorithms. Observe that if the provided initial support-set is erroneous, the I-support algorithms will have a clear disadvantage compared to the R-support algorithms.

Furthermore, the output from StOMP contains a solution based on a support-set that is not limited by K. Therefore, we modify the result from StOMP by picking $\mathcal{T}' \leftarrow \texttt{max\_indices}(\hat{\mathbf{x}}_{\text{StOMP}}, K)$ as the support-set. Based on this support-set we then form the new estimate $\hat{\mathbf{x}}$ such that $\hat{\mathbf{x}}_{\hat{\mathcal{T}}} = \mathbf{A}_{\hat{\mathcal{T}}}^{\dagger}\mathbf{y}$ and $\hat{\mathbf{x}}_{\bar{\hat{\mathcal{T}}}} = \mathbf{0}$, as the solution for StOMP. This last modification is only done to StOMP to provide for a fair comparison.

IV. Experimental Evaluation

All experimental results presented in this paper are performed on a regular laptop running a single-threaded Matlab. For the experiments and results, we compare FROGS with SP, OMP, CMP and StOMP. We have chosen well-known algorithms as benchmark schemes, each representative of one of the four classes of algorithms in Table I. We characterize the measurement noise as signal-to-measurement-noise-ratio as

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

We show results for Gaussian signals in a setting with measurement noise, where the noise is SMNR $= 20$ dB. To compare the algorithms, we use the signal-to-reconstruction-noise-ratio (SRER) which is defined as

$$\text{SRER} = 10\log_{10}\frac{\mathbb{E}\left\{\|\mathbf{x}\|_2^2\right\}}{\mathbb{E}\left\{\|\mathbf{x}-\hat{\mathbf{x}}\|_2^2\right\}}. \tag{6}$$

We are interested in the scenario where an initial support-set $\mathcal{T}_{\text{ini}}$ is fed to the algorithms. Since this support-set may be corrupted with errors, we provide results where the algorithms are provided an initial support with parts correct support-set estimates $\mathcal{T}_{\text{part}}$ and parts erroneous support-set estimates $\mathcal{T}_{\text{err}}$ according to (2), for different sizes of $\mathcal{T}_{\text{part}}$ and $\mathcal{T}_{\text{err}}$. To study how the performance of different algorithms are affected by different sizes in the initial support-set, we introduce two fractions:

$$\eta = \frac{|\mathcal{T}_{\text{ini}}|}{K}, \qquad \text{and} \qquad \kappa = \frac{|\mathcal{T}_{\text{part}}|}{|\mathcal{T}_{\text{ini}}|}, \tag{7}$$

where η is the fraction of initial support and κ is the fraction of correct support.

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

$$\alpha = \frac{M}{N}. \tag{8}$$

Using α, η and κ, the testing is performed as follows:

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

The non-zero components in the vector are chosen independently from a Gaussian source.

5) Compute the measurement vector $\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{w}$, where $\mathbf{w}$ is standard iid Gaussian noise.
6) First choose η such that $|\mathcal{T}_{\text{ini}}|$ is a constant, then choose κ such that $|\mathcal{T}_{\text{part}}|$ is also a constant. Generate $\mathcal{T}_{\text{ini}} = \mathcal{T}_{\text{err}} \cup \mathcal{T}_{\text{part}}$, where $\mathcal{T}_{\text{part}}$ is randomly chosen from $\mathcal{T}$ and $\mathcal{T}_{\text{err}}$ is randomly chosen from $\{1, 2, ..., N\}$, or set to the empty set $\emptyset$.
7) Apply the CS algorithms on the data $\mathbf{y}$, $\mathbf{A}$, K and $\mathcal{T}_{\text{ini}}$.

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

A. Experimental Results

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

The CMP algorithm[1] has quite a few parameters that can be modified to tune the performance according to different set-ups. In particular we tell the algorithm to stop according to support-set size criterion and we allow the refinements in the augmentation step to be done 40 times, instead of the recommended 2 times [8]. For the StOMP algorithm we used the default parameters as provided by Sparselab[2].

1) Execution time comparison: In this paper, our main concern is not the complexity. However, it is interesting to briefly discuss the complexity difference between CMP and FROGS, since it directly follows from the construction of the algorithms. FROGS searches for the least contributing component in the support-set $\hat{\mathcal{T}}$ by selecting the component corresponding to the smallest-in-magnitude in $\hat{\mathbf{x}}$ and then tries to replace this component, while CMP instead exchanges each component in $\hat{\mathcal{T}}$ one-by-one. Because of this, CMP needs more iterations, compared to FROGS, before it finds the best component to replace. This shows in the complexity Table II.

2) Performance comparison: In Fig. 2, we show the reconstruction performance of all algorithms without any initial support-set knowledge. It is interesting to notice that FROGS performs better than both OMP and SP, although not as good as CMP.

In Fig. 3, we fix $\alpha = 0.17$ and instead let the initial support-set grow from 0 elements up to K (i.e., $\eta : 0 \to 1$). Here, we expect to see improved performance as η increases. However, since $\kappa = 0.8$ the number of errors sent to the algorithm will also increase. We see that this number of increased errors is something OMP suffers from. Also, both FROGS and CMP have a decay in performance as η becomes very big. For FROGS this decay in performance is because the initial support-set is nearly full, meaning it has troubles finding new, good components to replace with the old. Provided there are several errors in the initial support-set, we notice that FROGS works best when $\eta \leq 0.6$.

[1]http://imi.aau.dk/~bst/software/index.html
[2]http://sparselab.stanford.edu/

TABLE II: Execution time of some GS algorithms at $\alpha = 0.17$, $\kappa = 0.8$, $\eta = 0.5$ and SMNR $= 20$dB

	SP	OMP	StOMP	CMP	FROGS
Time	0.48	0.32	3.63	8.65	1.60

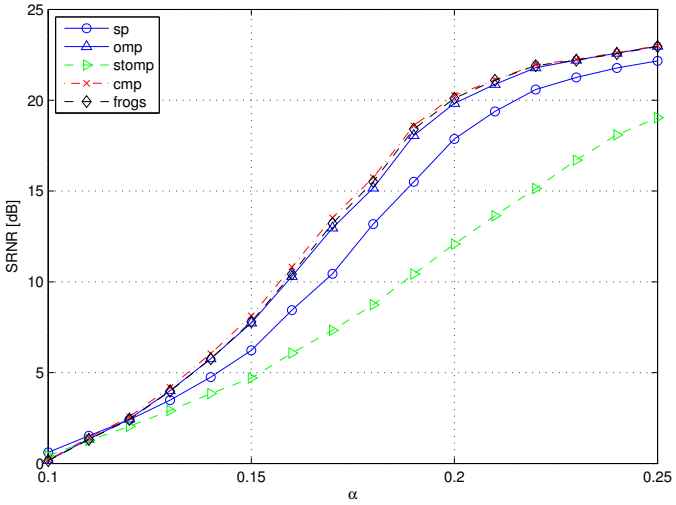

Fig. 2: SMNR $= 20$ dB and $\eta = 0$

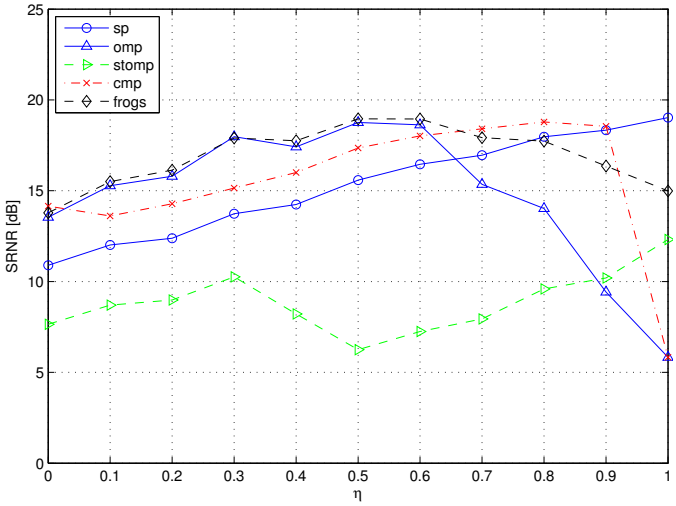

Fig. 3: SMNR $= 20$ dB, $\alpha = 0.17$, $\kappa = 0.8$

In Fig. 4, we have again fixed SMNR $= 20$ dB, $\alpha = 0.17$ and $\eta = 0.5$ and instead we vary κ. As κ becomes bigger, the performance should improve if the algorithms can efficiently benefit from the initial support-set. We here see that FROGS is the algorithm that best benefits from this knowledge. OMP works good in high κ, but fails in the lower region.

We finish our performance comparison with Fig. 5. In this figure we again vary α for SMNR $= 20$ dB, $\eta = 0.6$, $\kappa = 0.7$ to try to emulate what could happen in a real scenario similar

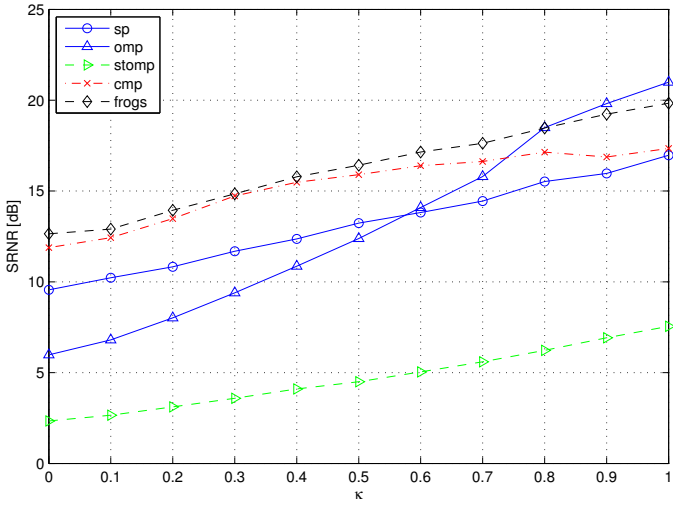

Fig. 4: SMNR $= 20$ dB, $\alpha = 0.17$ and $\eta = 0.5$

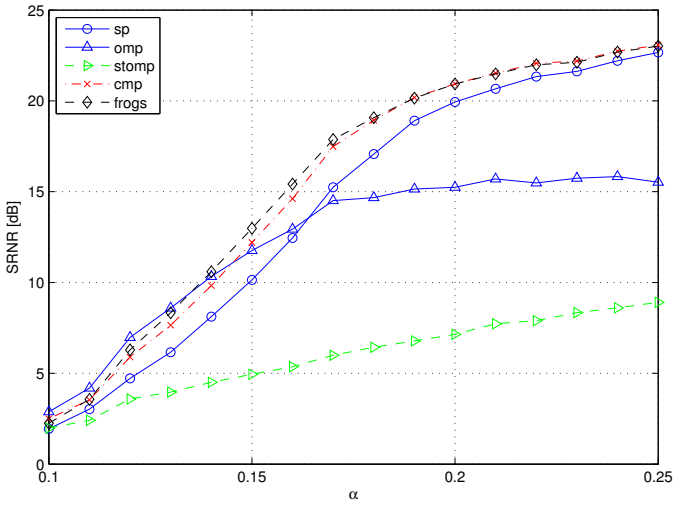

Fig. 5: SMNR $= 20$ dB, $\eta = 0.6$, $\kappa = 0.7$

to that in [10]. Although OMP is a strong contender in low α, at α above 0.15, FROGS is again the best algorithm.

An interesting point to notice in all the above figures is that FROGS is very robust to errors in the initial support-set. Also it is the one algorithm that performs best when provided good initial support-set information. We also notice that StOMP is not performing well in any figure. Its performance may be improved by tuning the parameters to the algorithm.

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

V. Conclusions

Based on new trends of multiple data set problems in the CS community, we have introduced a notion of prior information in terms of an initial support-set. We have seen how this prior information can be utilized by GS algorithms by a simple modification in the initialization phase of the algorithm. Based on how support-sets are detected, we have seen that several GS algorithms can be categorized into different groups. Based on this categorization and the initial support-set prior, we have developed a new algorithm which we call FROGS. By experimental evaluation we notice that this new algorithm performs well in its task of estimating a signal with the additional knowledge of an initial support-set.

References

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

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

[3] D. Zachariah, S. Chatterjee, and M. Jansson, "Dynamic iterative pursuit," *IEEE Trans. Signal Processing*, vol. 60, no. 9, pp. 4967 –4972, sept. 2012.

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

[5] Michael Lustig, David Donoho, and John M. Pauly, "Sparse mri: The application of compressed sensing for rapid mr imaging," *Magnetic Resonance in Medicine*, vol. 58, no. 6, pp. 1182–1195, 2007.

[6] N. Vaswani, "Ls-cs-residual (ls-cs): Compressive sensing on least squares residual," *IEEE Trans. Signal Processing*, vol. 58, no. 8, pp. 4108 –4120, aug. 2010.

[7] M.G. Christensen and S.H. Jensen, "The cyclic matching pursuit and its application to audio modeling and coding," in *Proc. Asilomar Conf. Signals, Sys., and Comp.*, nov. 2007, pp. 550–554.

[8] B.L. Sturm, M.G. Christensen, and R. Gribonval, "Cyclic pure greedy algorithms for recovering compressively sampled sparse signals," in *Proc. Asilomar Conf. Signals, Sys., and Comp.*, nov. 2011, pp. 1143 –1147.

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

[10] D. Sundman, S. Chatterjee, and M. Skoglund, "A greedy pursuit algorithm for distributed compressed sensing," in *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing (ICASSP)*, Kyoto, Japan, Mar. 2012.

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

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

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

[14] S. Chatterjee, D. Sundman, M. Vehkaperä, and M. Skoglund, "Projection-based and look ahead strategies for atom selection," *IEEE Trans. Signal Processing*, vol. 60, no. 2, pp. 634 –647, feb 2012.

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

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

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

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