AUTOMATIC THRESHOLD ESTIMATION FOR ITERATIVE SHRINKAGE ALGORITHMS USED WITH COMPRESSED SENSING

Nasser Mourad[*], *James P. Reilly*[†]

[*]Department of Electrical Eng.,
Aswan Faculty of Eng., South Valley University, Aswan, Egypt

[†] Electrical & Computer Eng.
McMaster University, 1280 Main St. w., Hamilton, Ontario, Canada L8S 4K1

email: naser.elsayed@asw-eng.svu.edu.eg, reillyj@mcmaster.ca

ABSTRACT

Recently, a new class of algorithms has been developed which iteratively build a sparse solution to an underdetermined linear system of equations. These algorithms are known in the literature as Iterative Shrinkage Algorithms (ISA). ISA algorithms depend on a thresholding parameter, which is usually provided by the user. In this paper we develop a new approach for automatically estimating this thresholding parameter. The proposed approach is general in a sense that it does not assume any distribution on the entries of the dictionary matrix, nor on the nonzero coefficients of the solution vector. In addition, the proposed approach is simple and can be adapted for use with newly evolving ISA algorithms. Moreover, the simulation results show that these proposed algorithms outperform their previous counterparts.

Indexing Terms: *Compressed Sensing, Hard Thresholding, Iterative Shrinkage algorithms.* .

I. INTRODUCTION

The determination of a *sparse* solution to an *underdetermined* linear system of equations has gained tremendous interest in recent years. Many algorithms have been developed in the literature for solving the aforementioned problem, and such algorithms have a wide range of applications such as source coding, denoising, source separation, and medical imaging [1]. Mathematically speaking, we are given $\boldsymbol{b} = \boldsymbol{A}\boldsymbol{x}$, where both $\boldsymbol{b} \in R^n$ and $\boldsymbol{A} \in R^{n\times N}$ are known, while $\boldsymbol{x} \in R^N$ is an *s-sparse* coefficient vector to be determined. Here $n < N$ and *s-sparse* means that at most $s \leq n$ entries of the solution vector $\boldsymbol{x}$ are nonzeros.

An s-sparse solution vector can be determined by solving one of the following optimization problems [2]

$$\boldsymbol{x} = \arg\min_{\boldsymbol{z}} ||\boldsymbol{b} - \boldsymbol{A}\boldsymbol{z}||^2_{\ell_2} \quad \text{s.t} \quad ||\boldsymbol{z}||_{\ell_0} \leq s \tag{1}$$

$$\boldsymbol{x} = \arg\min_{\boldsymbol{z}} ||\boldsymbol{b} - \boldsymbol{A}\boldsymbol{z}||^2_{\ell_2} + \lambda||\boldsymbol{z}||_{\ell_0} \tag{2}$$

where λ is a regularization parameter, and $||\boldsymbol{z}||_{\ell_0}$ is the ℓ_0-norm, which counts the number of nonzero entries in $\boldsymbol{z}$. Mathematically speaking, $||\boldsymbol{z}||_{\ell_0} = |\mathcal{I}_1(\boldsymbol{z})|$ is defined as the cardinality of the set $\mathcal{I}_1(\boldsymbol{z})$, where $\mathcal{I}_1(\boldsymbol{z}) := \{z_i : z_i \neq 0\}$ is the set of nonzero coefficients. Unfortunately, solving (1) or (2) is known to be NP hard and requires a combinatorial search. Many algorithms have been proposed in the literature for relaxing these two problems by proposing approximate solutions to them [3], [4], [5]. However, these methods are often found to be inefficient[1]. This is especially the case for high-dimensional problems, as often encountered in image processing.

Recently, a new iterative class of algorithms referred to as Iterative Shrinkage algorithms (ISA) [2], [6], [7], [8], [9] have been proposed. Despite their simple structure, ISA algorithms are shown to be very effective in solving (1) and (2). In Section II we summarize some of the well known ISA algorithms.

As shown in Section II, ISA algorithms depend on a user-defined parameter that must be set manually. This parameter is either the sparsity level s or a thresholding level θ, which depends on the regularization parameter λ. Unfortunately, the performance of the ISA algorithms is greatly affected by the values of these two parameters [10]. In Section III-A we summarize previous approaches proposed in the literature for estimating these two parameters.

In this paper we propose a simple yet efficient technique for automatically and adaptively determining the thresholding parameter θ. The proposed approach is general in the sense that it does not assume any distribution on the entries of the dictionary matrix nor on the nonzero coefficients of the solution vector. The aim of the proposed technique is to automate ISA algorithms.

II. ITERATIVE SHRINKAGE ALGORITHMS

In this section we provide a summary of some ISA algorithms that use different thresholding strategies.

II-A. Iterative Hard Thresholding (IHT) algorithm

The Iterative Hard Thresholding (IHT) algorithm was first introduced in [2]. This algorithm solves the s–sparse problem (1) and the ℓ_0–regularized optimization problem (2), respectively as follows:

$$(\text{IHT}_1) \quad \boldsymbol{x}^{i+1} = \mathcal{H}^s\left(\boldsymbol{x}^i + \mu^i \boldsymbol{A}^T\left(\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}^i\right)\right)$$

$$(\text{IHT}_2) \quad \boldsymbol{x}^{i+1} = \mathcal{H}_\theta\left(\boldsymbol{x}^i + \mu^i \boldsymbol{A}^T\left(\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}^i\right)\right),$$

This work was supported in part by the Egyptian Ministry of High Education and the Natural Sciences and Engineering Research Council of Canada (NSERC).

where $\boldsymbol{x}^i$ is the solution vector at the ith iteration, $\theta = \sqrt{\lambda}$, $\mathcal{H}^s(\cdot)$ is a non-linear operator that only retains the s coefficients with the largest magnitude, and $\mathcal{H}_\theta$ is the element wise hard thresholding operator

$$\mathcal{H}_\theta(x[k]) = \left\{ \begin{array}{ll} x[k] & |x[k]| > \theta \\ 0 & |x[k]| \leq \theta \end{array} \right. . \quad (3)$$

In the basic IHT algorithm derived in [2], the parameter μ is set to unity, while in [11], the authors suggest allowing its value to depend on the iteration index, to improve the stability of the algorithm.

II-B. Stagewise OMP (StOMP) algorithm

The StOMP algorithm [6] starts with initial solution $\boldsymbol{x}^0 = \boldsymbol{0}$ with the associated initial support $\mathcal{I}_0 = \phi$ and initial residual $\boldsymbol{r}^0 = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}^0 = \boldsymbol{b}$. The algorithm executes the following steps, for $i = 1, 2, \ldots$

1) Calculate the residual correlation $\boldsymbol{c}^i = \boldsymbol{A}^T\boldsymbol{r}^{i-1}$
2) Find the support $\mathcal{J}_i = \{j : |c_i(j)| > \theta\}$
3) Update the support $\mathcal{I}_i = \mathcal{I}_{i-1} \cup \mathcal{J}_i$
4) Set $\boldsymbol{x}^i = 0$ and update $(\boldsymbol{x}^i)_{\mathcal{I}_i} = \left(\boldsymbol{A}^T_{\mathcal{I}_i}\boldsymbol{A}_{\mathcal{I}_i}\right)^{-1}\boldsymbol{A}^T_{\mathcal{I}_i}\boldsymbol{b}$, where $\boldsymbol{A}_{\mathcal{I}_i}$ denote the $n \times |\mathcal{I}_i|$ matrix with columns chosen using index set $\mathcal{I}_i$, and $(\boldsymbol{x}_i)_{\mathcal{I}_i}$ represent $\boldsymbol{x}_i$ supported in $\mathcal{I}_i$.
5) Update the residual $\boldsymbol{r}^i = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}^i$ and go to Step 1.

The thresholding parameter θ used in the second step is estimated using the procedure described in the next section.

II-C. Hard Thresholding Pursuit (HTP)

The HTP algorithm [9] starts with initial guess of $\boldsymbol{x} = \boldsymbol{0}$ and iterates in the following steps:

1) Calculate $\boldsymbol{y}^i = \boldsymbol{x}^i + \mu^i\boldsymbol{A}^T\left(\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}^i\right)$
2) Find the support $\mathcal{I}_i$ which contains the indices of the s largest magnitudes of $\boldsymbol{y}^i$
3) Set $\boldsymbol{x}^{i+1} = 0$ and update $(\boldsymbol{x}^{i+1})_{\mathcal{I}_i} = \left(\boldsymbol{A}^T_{\mathcal{I}_i}\boldsymbol{A}_{\mathcal{I}_i}\right)^{-1}\boldsymbol{A}^T_{\mathcal{I}_i}\boldsymbol{b}$

II-D. Two Stage Thresholding (TST) algorithms

The last of the ISA algorithms that we consider are the Subspace Pursuit (SP) algorithm [7] and the Compressive Sampling Matching Pursuit (CoSaMP) algorithm [8]. These two algorithms apply thresholding at two different stages in each iteration. We therefore refer to them as Two Stage Thresholding (TST) algorithms. The TST algorithms are described in the following way. For the SP algorithm, the value t below is set to s, whereas for CoSaMP, $t = 2s$.

Start with an s-sparse $\boldsymbol{x}_0 \in R^N$, typically $\boldsymbol{x}_0 = \boldsymbol{0}$, and iterate in the following steps:

1) Calculate the residual correlation $\boldsymbol{c}^i = \boldsymbol{A}^T\boldsymbol{r}^{i-1}$
2) Find the support $\mathcal{I}_i$ which contains the indices of the s largest entries of $\boldsymbol{x}^i$
3) Find the support $\mathcal{J}_i$ which contains the indices of the t largest entries of $\boldsymbol{c}^i$
4) Merge the two supports $\mathcal{T}_i = \mathcal{I}_i \cup \mathcal{J}_i$
5) Calculate the vector $\boldsymbol{y}^i = \left(\boldsymbol{A}^T_{\mathcal{T}_i}\boldsymbol{A}_{\mathcal{T}_i}\right)^{-1}\boldsymbol{A}^T_{\mathcal{T}_i}\boldsymbol{b}$
6) Calculate $\boldsymbol{x}^{i+1} = \mathcal{H}^s(\boldsymbol{y}^i)$

It was shown in [12] that by varying the value of t between s and $2s$, the best performance was obtained for the case $t = s$, which corresponds to the SP algorithm. Therefore, in this paper we will use the acronym TST to refer to the SP algorithm.

III. PROPOSED THRESHOLDING TECHNIQUE

As shown in Section II, all the algorithms except StOMP require a threshold parameter (s or θ) to be provided by the user. Selecting a threshold θ is equivalent to selecting a sparsity level s which equals the number of entries in the vector $(\boldsymbol{x}^i + \mu\boldsymbol{A}^T\boldsymbol{r}^i)$ that have magnitude values greater than θ. Therefore we focus in this section on methods for estimating θ.

III-A. Previous Approaches for Selecting θ

The simplest approach is to use a fixed value for θ [2], [10]. However, as demonstrated in [10], the performance of the IHT algorithm, and hence other IST algorithms, depends greatly on which value is selected. To overcome this difficulty, researchers suggest varying θ in each iteration [13], [6]. The approach suggested in [13] is to start with a large initial threshold θ_0 and then decreasing its value linearly according to the relation $\theta_i = \theta_0(1 - i/L_{itr})$, where i is the iteration number, and L_{itr} is the total number of iterations. The motivation behind this approach is to enforce the sparsity of the solution vector at the first steps by selecting large value for θ, then as the threshold becomes smaller, the error $||\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}^i||_{\ell_2}$ vanishes. Unfortunately, it was shown in [13] that the performance of this procedure depends greatly on the initial threshold θ_0 and the number of iterations L_{itr}.

Another approach for selecting the value of θ was suggested in [6]. This approach assumes that the entries of the dictionary matrix $\boldsymbol{A}$ are random, and hence, according to the central limit theorem, the marginal distribution of the entries of $\boldsymbol{A}^T\boldsymbol{r}^i$ at the coordinates of $\mathcal{I}_0$ is Gaussian, where $\mathcal{I}_0$ refers to the indices of the zero entries of the solution vector $\boldsymbol{x}^{i-1}$ at iteration $(i-1)$. The Gaussianity of $\boldsymbol{A}^T\boldsymbol{r}^i$ is then utilized for selecting the value of θ. See [6] for more details about this approach. Therefore, this method is not suitable for estimating θ from $(\boldsymbol{x}^i + \mu^i\boldsymbol{A}^T(\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}^i))$, as in IHT and HTP, or when $\boldsymbol{A}$ is not an explicit matrix but instead a linear operator (e.g. partial Fourier and Hadamard transforms) for which $\boldsymbol{A}\boldsymbol{x}$ and $\boldsymbol{A}^T\boldsymbol{r}$ can be computed without storing $\boldsymbol{A}$.

Recently, an extensive computational experiment was conducted in [12] for selecting the optimum thresholding values for the IHT and the TST algorithms. However, there is no direct approach to predict optimum threshold values for newly evolving ISA algorithms from the suggested thresholds.

In the next subsection we propose a simple and general technique for iteratively estimating the threshold parameter θ. The proposed technique can be used with all the algorithms described in Section II, and it can be easily adapted to be used with any new ISA algorithm.

III-B. Proposed Thresholding Technique

The proposed approach relies on the same motivation as that used in [13]. That is, we first select a large threshold value to enforce the sparsity of the solution vector, that is gradually decreased to reduce the approximation error. However, instead of *blindly* decreasing the threshold, we suggest a data-driven strategy. To explain this point, assume that the solution vector produced by an ISA algorithm at the ith iteration has the structure $\boldsymbol{x}^i = [\boldsymbol{x}_1\ \boldsymbol{x}^i_0]$, where $\boldsymbol{x}_1 \in R^s$ contains the nonzero coefficients of the *true* solution vector $\boldsymbol{x}$, and $\boldsymbol{x}^i_0 \in R^{N-s}$

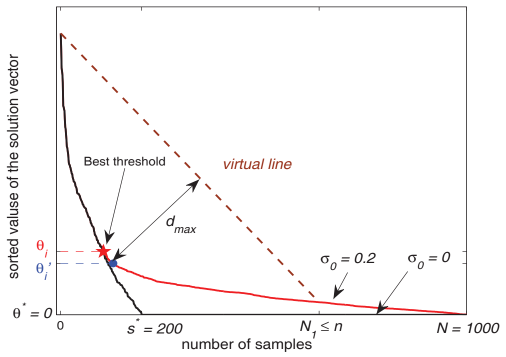

Fig. 1. Demonstration of the proposed approach for estimating the threshold parameter θ_i.

is a vector with small entries that correspond to the residual coefficients at the ith iteration. The indices of $\boldsymbol{x}_0^i$ are the indices of the zero entries of $\boldsymbol{x}$. Here we do not assume any distribution model for the entries of $\boldsymbol{x}_1$ or $\boldsymbol{x}_0^i$. Rather, we assume that both have zero mean and their respective standard deviations satisfy the relation $\sigma_1 > \sigma_0^i$. Moreover, if the underlying ISA algorithm is converging, the standard deviation of the residual will satisfy the relation $\sigma_0^i \le \sigma_0^{i-1}$. A demonstrating example is presented in Fig. 1.

In Fig. 1 the entries of $\boldsymbol{x}_1 \in R^{200}$ and $\boldsymbol{x}_0^i \in R^{800}$ are sampled from zero mean Gaussian distributions with standard deviations $\sigma_1 = 1$ and $\sigma_0 = 0.2$, respectively. The red curve in Fig. 1 represents $\boldsymbol{x}^i = [\boldsymbol{x}_1 \ \boldsymbol{x}_0^i]$ with its entries sorted in decreasing magnitude, while the black curve represents the true solution vector, which correspond to the case $\sigma_0 = 0$. This curve will be used as a *reference* for estimating the threshold. The two curves overlap for all amplitude values greater than the red star shown in Fig. 1.

A reasonable choice of the value of θ_i is therefore the point at which the two curves coincide, i.e., at the height of the red star in Fig. 1. By selecting this threshold, all samples of $\boldsymbol{x}^i$ with magnitude values greater than θ^i most probably belong to $\boldsymbol{x}_1$, and therefore should be retained, while the other samples with magnitude values less than θ_i most probably belong to $\boldsymbol{x}_0^i$, and hence should be set to zero. Following this strategy, we can optimally select the threshold value at each iteration. Unfortunately, this procedure can not be followed in practice because the reference (black) curve is, of course, not available. To overcome this difficulty, we suggest estimating a *suboptimal* thresholding parameter $\theta_i^{'}$, and then applying a correcting parameter c_i to get $\hat{\theta}_i = c_i \theta_i^{'}$.

Estimating $\theta_i^{'}$***:*** Referring to Fig. 1, we observe a "corner point", which we indicate by the solid blue circle. Later, we relate this point to the coincidence point of the two curves. We suggest estimating a suboptimal threshold parameter $\theta_i^{'}$ as the height of this corner point. Since in practice we have $s < n$, we can estimate this corner point from the largest N_1 points in $\boldsymbol{x}^i$, where $N_1 \le n$ is a parameter to be determined. Let $\boldsymbol{w}^i \in R^{N_1}$ be the set of N_1 points of $\boldsymbol{x}^i$ that have largest magnitude, sorted in descending order. We then identify the corner point as the farthest point from the virtual line connecting the two extreme points of $\boldsymbol{w}^i$, depicted as the dashed line in Fig. 1.

Let the virtual line be represented by the equation $y = mx + h$, where the slope $m = (w^i[N_1] - w^i[1])/(N_1 - 1)$ and the intercept $h = w^i[1] - m$. Then it is not difficult to show that the *shortest* (normal) distance from the point $w^i[k]$ (the kth entry in $\boldsymbol{w}^i$) to this virtual line is given by

$$d[k] = \frac{\left|w^i[k] - mk - h\right|}{\sqrt{m^2+1}}, \quad k = 1, \dots, N_1 \qquad (4)$$

After calculating the vector of distances $\boldsymbol{d}$, we calculate $\theta_i^{'}$ as the value of $w^i[k]$ at which $d[k]$ is maximum.

Selecting the correcting parameter c_i***:*** Since $\boldsymbol{x}^i = [\boldsymbol{x}_1 \ \boldsymbol{x}_0^i]$ with $\sigma_1 > \sigma_0^i$, the probability that the height $\theta_i^{'}$ of the corner point of $\boldsymbol{x}^i$ (marked by the solid blue circle in Fig.1) is less than or equal the height θ_i of the coincidence point, (marked by the red star) is relatively high. Therefore, $\theta_i^{'} < \theta_i$ most of the time, so we assume $c_i \ge 1$. Moreover, due to our previous assumption $\sigma_0^i \le \sigma_0^{i-1}$, the difference between θ_i and $\theta_i^{'}$ decreases as the algorithm converges to the correct solution vector, at which the difference vanishes. Therefore, we suggest starting with a large correcting parameter $c_1 > 1$, and then linearly decreasing it to the value of one in N_c iterations. Then we fix $c_i = 1, \forall i > N_c$. The value of c_1 must be selected such that $c_1\theta_1^{'} < w[1]$ to retain at least one sample point after the first thresholding operation. In the simulations we used $c_1 = 2$ and $N_c = 5$. Increasing the value of N_c improves the convergence of the ISA algorithms, at the cost of slower execution.

Selecting the value of N_1***:*** Since the length N of the solution vector is usually much greater than the number of nonzero entries s, which in turn is less than the length n of the observed signal, we suggest selecting $N_1 \le n$. Including the remaining $(N - N_1)$ points in the computation of the corner point does not degrade performance, but increases the overall computation cost of the algorithm unnecessarily. Moreover, the value of N_1 could be fixed or variable, depending on the ISA algorithm. For instance the IHT, HTP, and TST algorithms estimate the *overall* support of the solution vector in each iteration. Since the size s of this support is unknown, but satisfies $s < n$, it is recommended to set $N_1 = n$ with these algorithms. We followed this strategy in the simulations. On the other hand, the support of the solution vector estimated by the StOMP algorithm is *accumulated* from the indices of the largest entries in the residual correlation vector $\boldsymbol{c}^i = \boldsymbol{A}^T\boldsymbol{r}^{i-1}$. See Step (3) in Section II-B. Since the number of the largest entries of $\boldsymbol{c}^i$ decreases as the StOMP algorithm converges, it is reasonable to decrease the value of N_1 in turn to decrease the possibility of picking a false corner point. In the simulations we selected the value of N_1 used with StOMP as follows. First we selected an initial value of $N_1 = 25$ then we decreased this value by one in each iteration until $N_1 = 10$. This value is then kept constant to the end of the algorithm.

IV. SIMULATION RESULTS

In this section we investigate the impact of the proposed approach on the performance of the four algorithms (IHT, StOMP, HTP, and TST) described in Section II. In the implementation of IHT, HTP, and TST we use the exact sparsity level s, while the thresholding parameter used with StOMP is estimated using the approach proposed in [6]. For running the StOMP algorithm we used

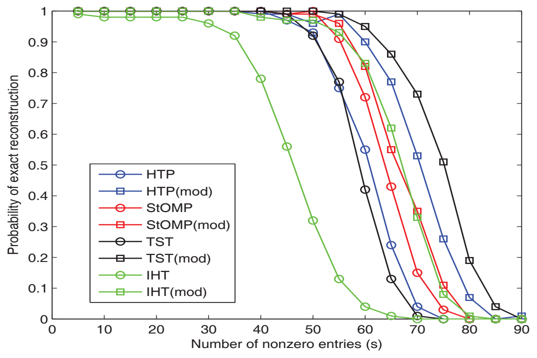

Fig. 2. Comparison between the IHT, StOMP, HTP, and TST algorithms and their counterpart modified versions IHT(mod), StOMP(mod), HTP(mod), and TST(mod) as a function of s .

the function *"SolveStOMP.m"* incorporated in *sparselab*[1]. When the proposed approach is used for estimating the thresholding parameter, we refer to the resulting algorithms as IHT(mod), StOMP(mod), HTP(mod), and TST(mod), respectively.

In this example we compare the performance of the original algorithms with their modified versions. As a measure of performance we calculate the probability of exact reconstruction (PER) of the solution vector $\boldsymbol{x}$ as a function of the number of the nonzero entries s. The PER is defined as the ratio between the number of runs at which the algorithm successfully estimates the sparse solution vector, to the total number of runs, which equals 100 in this example. The solution vector is considered estimated correctly if $\left|\left|\frac{\hat{\boldsymbol{x}}}{||\hat{\boldsymbol{x}}||} - \frac{\boldsymbol{x}}{||\boldsymbol{x}||}\right|\right|_{\ell_2} \leq 0.01$, where $\boldsymbol{x}$ is the exact solution vector and $\hat{\boldsymbol{x}}$ is its estimate.

In this example we fix $n = 200$ and $N = 1000$, while s takes the values $s = 5, 10, \ldots, 90$. For each value of s, a new random dictionary matrix $\boldsymbol{A} \in R^{n \times N}$ and a new s–sparse vector $\boldsymbol{x} \in R^N$ are randomly generated, and the measured vector is constructed as $\boldsymbol{b} = \boldsymbol{Ax}$. Different distributions for the entries of $\boldsymbol{A}$ and the nonzero entries of $\boldsymbol{x}$ have been evaluated; however, since the results are similar for different distributions, we consider only the case where the entries of $\boldsymbol{A}$ and the nonzero entries of $\boldsymbol{x}$ are sampled from a zero mean Gaussian distribution with unit variance. The results are shown in Fig. 2.

In Fig. 2, the curves representing the original algorithms are marked by circles, while those representing the modified algorithms are marked by squares. Moreover, each algorithm and its modified version are represented by the same color. For the original algorithms we find that IHT has the worst performance while StOMP has the best performance. The performances of HTP and TST are almost identical. However, by incorporating the proposed technique with these algorithms we find that the modified versions outperform their counterpart original ones. For instance, IHT(mod) outperforms all the original algorithms, while TST(mod) has the best performance among all algorithms. The HTP(mod) and StOMP(mod) algorithms outperform their counterpart original algorithms. It is worth mentioning that the proposed approach slightly increases the running time of the modified ISA algorithms due to the computation of the vector of distances $\boldsymbol{d}$ in each iteration. For instance, the average conversion times of IHT, HTP, StOMP, and TST for the case $s = 40$ are 0.68, 0.14, 0.14, and 0.034 *sec.*, respectively, while the computation time of their counterpart modified algorithms are 0.75, 0.21, 0.094, and 0.07 *sec*, respectively.

[1]This is a free software available at http://sparselab.stanford.edu/

V. CONCLUSION

In this paper we have proposed a new approach for estimating a thresholding parameter for use with different ISA algorithms. The proposed approach is general in a sense that it does not assume any distribution on the entries of the dictionary matrix or on the nonzero coefficients of the solution vector. Moreover, the proposed approach is simple and can be adapted to be used with newly evolving ISA algorithms. Incorporating the proposed approach with some well known ISA algorithms has resulted in considerable performance improvement.

VI. REFERENCES

[1] Michael Elad, "Sparse and Redundent Representations From Theory to Applications in Signal and Image Processing," Springer, 2010.

[2] T. Blumensath and M.E. Davies, "Iterative Thresholding for Sparse Approximations," J. Fourier Anal. Appl., vol. 14 pp.629654, 2008.

[3] Y. C. Pati, R. Rezaiifar, and P. S. Krishnaprasad, " Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition,"In 27th Annu. Asilomar Conf. Signals, Systems, and Computers, volume 1, pages 4044, 1993

[4] S. S. Chen, D. L. Donoho, and M. A. Saunders, "Atomic decomposition by basis pursuit," SIAMJournal of Scientific Computing, vol. 20, no. 1, pp. 3361, 1998

[5] B. D. Rao and K. Kreutz-Delgado, "An affine scaling methodology for best basis selection," IEEE Trans. Signal Process., vol. 47, no. 1, Jan. 1999.

[6] D.L. Donoho, Y. Tsaig, I. Drori, and J.-L. Starck, Sparse solution of underdetermined linear equations by stagewise orthogonal matching pursuit, Techni cal Report, Stanford University, 2006.

[7] W. Dai and O. Milenkvic, "Subspace pursuit for compressive sensing signal reconstruction," IEEE Trans. Inform. Theory, 55 (2009), pp.22302249

[8] D. Needell and J.A. Tropp, "CoSaMP: Iterative signal recovery from incomplete and inaccurate samples," Appl. Comput. Harmon. Anal., 26 (2009), pp.30132.

[9] S. Foucart, "Hard thresholding pursuit: an algorithm for compressive sensing," 2010. preprint

[10] T. Blumensath, M. Yaghoobi and Michael E. Davies, "Iterative Hard Thresholding and L_0 Regularisation," *ICASSP* 2007, pp. III-877 - III-880.

[11] T. Blumensath and M.E. Davies, "Normalized iterative hard thresholding: guaranteed stability and performance," IEEE Journal of selected topics in signal processing, vol. 4, pp.298309, 2010

[12] A. Maleki and D. Donoho. Optimally tuned iterative reconstruction algorithms for compressed sensing. *IEEE* Journal of Selected Topics in Signal Processing, 4(2):330341, 2010

[13] S. Lesage, R. Gribonval, F. Bimbot, and L. Benaroya, "Learning unions of orthonormal bases with thresholded singular value decomposition," *ICASSP*, 2005