Ways to Sparse Representation: A Comparative Study

YANG Jingyu (杨敬钰), PENG Yigang (彭义刚), XU Wenli (徐文立), DAI Qionghai (戴琮海)**

Tsinghua National Laboratory for Information Science and Technology, Department of Automation, Tsinghua University, Beijing 100084, China

Abstract: Many algorithms have been proposed to achieve sparse representation over redundant dictionaries or transforms. A comprehensive understanding of these algorithms is needed when choosing and designing algorithms for particular applications. This research studies a representative algorithm for each category, matching pursuit (MP), basis pursuit (BP), and noise shaping (NS), in terms of their sparsifying capability and computational complexity. Experiments show that NS has the best performance in terms of sparsifying capability with the least computational complexity. BP has good sparsifying capability, but is computationally expensive. MP has relatively poor sparsifying capability and the computations are heavily dependent on the problem scale and signal complexity. Their performance differences are also evaluated for three typical applications of time-frequency analyses, signal denoising, and image coding. NS has good performance for time-frequency analyses and image coding with far fewer computations. However, NS does not perform well for signal denoising. This study provides guidelines for choosing an algorithm for a given problem and for designing or improving algorithms for sparse representation.

Key words: sparse representation; redundant dictionary/transform; nonlinear approximation; matching pursuit; basis pursuit; noise shaping

Introduction

Over the past decade, redundant dictionaries and transforms have played important roles in various applications such as time-frequency analysis\(^1\), signal denoising\(^{2,3}\), data regression\(^4\), image deconvolution\(^5\), picture coding\(^6,7\), and compressive sensing\(^8\). Unlike orthogonal transforms, a redundant dictionary or transform leads to non-unique representations of a given signal. Many algorithms have been developed to find sparse representations good for a variety of signal processing tasks.

In general, these algorithms can be categorized into greedy pursuit algorithms, \(\ell_p\) norm regularization based algorithms, and iterative shrinkage algorithms. In greedy pursuit algorithms, the atom in a given dictionary that best matches the residue is selected to approximate the signal. The residue is updated by subtracting the component associated with the chosen atom. The pioneer algorithm in this category is the matching pursuit (MP) algorithm\(^1\). Its well-known variants include the orthogonal matching pursuit (OMP)\(^9\), order recursive matching pursuit (ORMP)\(^10\), and tree-based pursuit (TBP)\(^11\) algorithms. Instead of using the \(\ell_0\) norm to measure sparseness, many algorithms seek to find sparse solutions by minimizing the \(\ell_p\) norm of the coefficients subject to reconstruction requirements. Since these algorithms are formulated as \(\ell_p\) norm regularized inverse problems such as deconvolution\(^5\) and compressive sensing\(^8\), these
algorithms can be referred to as $\ell_p$ norm regularized algorithms. The basis pursuit (BP)\cite{12}, focal under-determined system solver (FOCUSS)\cite{13}, and least absolute shrinkage and selection operator (LASSO)\cite{4} algorithms fall into this category. In the third category, an initial set of coefficients is obtained and then iteratively modified with shrinkage to achieve sparser representation. Early examples are the iterative projection-based noise shaping (NS)\cite{14} and local competition (LC)\cite{15} for sparse image representation. The theoretical formulation and convergence of the iterative shrinkage framework were provided by Daubechies et al.\cite{16} Yang et al.\cite{17} gave a comprehensive overview.

These algorithms were developed with various motivations, so their performance varies for different applications. Their performance differences for typical applications must be known to understand their pros and cons. However, a comprehensive comparison of all these algorithms is difficult. A good strategy is to compare their performance for each category. There have been some performance evaluations of greedy pursuit algorithms and $\ell_p$ norm regularization based algorithms\cite{12,18}. However, to our knowledge, there is no analysis taking the iterative shrinkage algorithms into account.

The MP, BP, and NS algorithms are selected as representative of each class of algorithms for a comparative study. Their abilities to achieve sparse representations (referred to as the sparsifying capability) in terms of the ideal recovery and nonlinear approximation (NLA) for various scenarios. Experiments on time-frequency analysis, signal denoising, and image coding were conducted to compare the performance differences for these algorithms for practical applications.

\section{Sparse Representation Algorithms}

\subsection{Sparse representation}

Signal $f$ is a vector which lies in the Hilbert space $\mathcal{H} = \mathbb{R}^N$. A dictionary $D$ consists of $M$ unit-norm vectors which span $\mathbb{R}^N$. That is $D = \{g_i : i \in I\}$, where $I$ is a countable index set whose cardinality, $\text{card}(I)$, is $M$. Each element $g_i$ in $D$ is called an atom. The dictionary is redundant since $M > N$. Signal $f$ can be represented as the superposition of atoms in $D$ as

$$f = \sum_{i \in I} c_i g_i$$  \hspace{1cm} (1)

where $c = \{c_i : i \in I\}$ is the set of coefficients for the corresponding atoms. Since the number of atoms in $D$ is larger than the dimensionality of the space they span, there are infinite number sets of coefficients that satisfy the perfect reconstruction (PR) in Eq. (1). The sparsest set of coefficients has the fewest nonzero coefficients, i.e., the one with the minimum $\ell_0$ norm\cite{19}. Thus, the sparsest representation is determined by

$$\tilde{c}(f, D) = \arg\min_c \|c\|_0, \text{ s.t } f = \sum_{i \in I} c_i g_i$$  \hspace{1cm} (2)

Unfortunately, this optimization problem is non-deterministic polynomial-time (NP)-hard\cite{19}. Thus, many algorithms resort to finding suboptimal yet sparse enough solutions.

\subsection{Matching pursuit}

The MP algorithm introduced by Mallat and Zhang is a pure greedy algorithm\cite{11}. Initially, it computes the inner-product of the signal and atoms over the dictionary and selects the largest one, i.e., $\langle f, g_i \rangle = \sup_{i \in I} |\langle f, g_i \rangle|$. Thus, $f$ can be expressed as

$$f = \langle g_{i_0}, f \rangle g_{i_0} + r_i$$  \hspace{1cm} (3)

where $\langle g_{i_0}, f \rangle g_{i_0}$ represents the projection of the signal onto the selected atom and $r_i$ is the residue.

Then, this decomposition is successively applied on the residue. After $M$ iterations, $f$ is decomposed as follows:

$$f = \sum_{n=0}^{M-1} \langle g_{i_n}, r_n \rangle g_{i_n} + r_M$$  \hspace{1cm} (4)

where $\langle g_{i_n}, r_n \rangle = \sup_{i \in I} |\langle g_i, r_n \rangle|$ at each step. The procedure terminates when a predefined tolerant approximation error is met.

The implementation of the MP algorithm is very straightforward and is not dependent on any particular transforms or dictionaries. However, the algorithm is computationally expensive for large dimensional dictionaries. Due to the nonorthogonality of the redundant dictionaries, the successive approximation property does not hold. As a result, an atom can be re-selected in a subsequent procedure. Thus, the MP algorithm can be trapped into suboptimum solutions\cite{12,20}. Many variants\cite{8-11} of MP have been proposed. However, the computational complexity is also increased by the
introduced extra procedures such as orthogonalization.

1.3 Basis pursuit

Instead of solving the $\ell_0$ norm minimization problem, Chen et al. \cite{12} proposed solving the $\ell_1$ norm minimization problem

$$\min \| c \|_1, \quad \text{s.t. } f = \sum_{i \in I} c_i g_i$$

which is called basis pursuit (BP). The change of the objective function converts the non-convex optimization problem in Eq. (1) into the convex optimization problem in Eq. (5), which is much easier to solve. Moreover, the $\ell_1$ norm minimization problem has the same solution as the $\ell_0$ norm minimization problem with certain conditions \cite{19-21}.

The conversion of Eq. (5) into a standard linear programming problem is described using matrix formulations. Let $R$ be the synthesis matrix where each column corresponds to an atom of $D$, i.e., $R = [g_1, g_2, \ldots, g_M]$. Then, Eq. (5) can be rewritten as

$$\min \| c \|_1, \quad \text{s.t. } f = Rc$$

where $c$ contains the coefficients. Assume $\Phi = (R, -R)$ and $x = (u^T, v^T)^T$, where $u \geq 0$, $v \geq 0$, and $c = u - v$. Then, $R(u - v) = f$, i.e., $(R, -R)(u^T, v^T)^T = f$, or $\Phi x = f$, where $x \geq 0$. Thus, minimizing $\| c \|_1$ is equivalent to minimizing $1^T x$, where 1 is a column vector whose elements are all one. By doing so, the optimization problem in Eq. (5) amounts to the following standard linear programming problem:

$$\min 1^T x, \quad \text{s.t. } \Phi x = f, x \geq 0$$

As a variant, basis pursuit denoising (BPDN) is introduced for signal denoising where the PR cannot be satisfied \cite{12}. This is also a convex optimization problem which has the form

$$\min_\varepsilon \left( \frac{1}{2} \| f - Rc \|_2^2 + \lambda \| \varepsilon \|_1 \right)$$

where $\lambda$ is a Lagrange multiplier introduced to control how much of the residue is lost when adding new coefficients. This can also be solved with linear programming methods. The two main categories of methods for solving linear programming problems are simplex methods and interior-point methods. Interior-point methods are more appealing since they greatly improve the computational efficiency.

1.4 Noise shaping

The NS algorithm was initially introduced to get a sparse representation for the 2-D dual-tree discrete wavelet transform (DDWT) \cite{22}. The 2-D DDWT is a complex transform whose redundancy is 4 : 1 (2 : 1 for the real part) with great potential for image representation since the basis functions are specially designed to capture directional features, such as edges and contours, in images. However, the coefficients produced by the 2-D DDWT forward transform might not be sparse enough initially. The NS algorithm tends to eliminate the small coefficients and attempts iteratively to modify large coefficients to compensate for the loss of small coefficients.

Since the 2-D DDWT is a linear transform, it can be formulated with matrix notation. Let $A$ denote the analysis matrix of 2-D DDWT while $R$ is the synthesis matrix. Then $c = Af$ represents the forward transform while $f = Rc$ represents the inverse transform. Let $S$ denote the range space of $A$, i.e., $S = \{ c = Af, f \in \mathbb{R}^N \}$. Then $P_S = AR$ is the projection operator onto $S$. Represent the orthogonal complementary space of $S$ with $S^\bot$. The projection operator onto $S^\bot$ is, therefore, $P_{S^\bot} = I - AR$. For any $x \in \mathbb{R}^N$, $RP_{S^\bot}x = 0$, where the vector $c^\bot = P_{S^\bot}x$ lies in $S^\bot$. Then a solution $c^\bot \in S^\bot$ pulse any $c^\bot$ will give the same reconstruction as $c^\bot$ itself, i.e., $R(c^\bot + c^\bot) = Rc^\bot = f$. Naturally, one would like to seek a $c^\bot$ in $S^\bot$ which makes $c = c^\bot + c^\bot$ as sparse as possible. This is the main idea of how iterative shrinkage algorithms achieve sparser representation. For initialization, let $c_0 = Af$. Then, $c_0$ is modified via hard or soft thresholding to discard the small coefficients

$$\hat{c}_0 = c_0 + n_0$$

where the discarded small coefficients are modeled as noise $n_0$. Then a distorted reconstruction is obtained by inverse transform with the modified coefficients $\hat{c}_0$

$$\hat{f}_0 = R\hat{c}_0$$

The reconstruction error is transformed back to the 2-D DDWT domain and added back to the modified coefficients, generating the new coefficients for the next iteration

$$c_1 = \hat{c}_0 + A(f - \hat{f}_0)$$
Substituting Eqs. (9) and (10) into Eq. (11) gives
\[ c_i = c_0 + (I - AR)n_0 = c_0 + P^i n_0 \] (12)
The procedure is iteratively applied to the new coefficients. The threshold decreases with a uniform step size from an initial value to a target value for each iteration. Up to the \( K \)-th iteration, it can be proved that \( c_k \) can be expressed as
\[ c_k = c_0 + \sum_{i=0}^{K-1} P^i n_i \] (13)
From Eq. (13), noise introduced via thresholding in each iteration is projected onto \( S^+ \) and then added to \( c_0 \) to achieve sparser representations while preserving the PR, i.e., \( R c_k = R c_0 = f \). Figure 1 shows the block diagram of the NS algorithm, where \( w_i = k A e_i \). To compensate for the energy loss due to thresholding, the error signal is amplified by a gain factor \( k^{[14]} \). In addition, the NS algorithm can be applied to other redundant transforms in the same way. Also, the NS algorithm can be straightforwardly extended to redundant dictionaries where the synthesis matrix \( R \) is naturally available and the analysis matrix \( A \) can be a pseudo-inverse of \( R \).

2 Performance Comparison

The sparsifying abilities of the MP, BP, and NS algorithms are compared in terms of the ideal recovery and nonlinear approximation (NLA) capability. The effectiveness of these algorithms is evaluated for three typical signal processing applications.

2.1 Experiment setup

For the ideal recovery experiments, both the dictionaries and signals are randomly generated to avoid the bias of using a particular type of signal or dictionary. These experiments used two 1-D signals, Bumps, and Mallat. As shown in Fig. 2, Bumps consists mainly of peaks with smooth slopes while Mallat has piecewise smooth curves separated by steps or impulses. Two 512 × 512 pixels test images, Lena and Barbara, were taken from the image database of the Signal and Image Processing Institute at the University of South California (http://sipi.usc.edu/database/) for the 2-D cases. The time-frequency dictionary for the 1-D signals was the symlet wavelet packets (SWP) with 10 : 1 redundancy and the 2-D DDWT of 2 : 1 redundancy is adopted for the images. The test images are decomposed to six levels with the CDF 9/7 filters for the first level 2-D DDWT decomposition and the Qshift filters\[^{[22]}\] for the remaining levels. Two toolboxes, Wavelab\[^{[23]}\] and Atomizer\[^{[24]}\], are employed for the MP

![Fig. 1 Block diagram of the noise shaping algorithm\[^{[14]}\]

![Fig. 2 Test signals](a) Bumps (b) Mallat
and BP algorithm. The linear programming method uses the primal-dual log-barrier algorithm. The experiments are carried out on a desktop with Pentium(R) IV 3.0-GHz CPU and 512-MB RAM.

2.2  Sparsifying capability

2.2.1  Ideal recovery

To evaluate the sparsifying capability, one would like to know how well an algorithm can recover the sparse representation if the signal is known to be sparse under the given dictionary. Given a dictionary, a sparse signal is produced by weighting only a few atoms with specified coefficients. An algorithm has ideal recovery capability if it can recover the pre-specified atoms and coefficients given the synthetic sparse signal and the dictionary. Theoretical results\[16,25\] for the ideal recovery capability are difficult to apply in practical evaluations because the ideal recovery conditions are quite specific for both the dictionary and the signal, and cannot be satisfied for general signals and dictionaries. Therefore, the dictionaries and signals similarly were randomly generated\[18\].

The dictionary is created as a random $20 \times 40$ matrix $\mathbf{R}$ with elements uniformly distributed in $[-1, 1]$. Then each column of $\mathbf{R}$ is normalized to form an atom. A sparse signal under the random dictionary is produced by setting four coefficients of $\mathbf{c}$ in random locations to random nonzero values whose magnitudes are larger than ten for easy manipulation of the very small coefficients. The signal is synthesized as $\mathbf{f} = \mathbf{Rc}$. Given the artificial signal and dictionary, the MP, BP, and NS algorithms are then used to recover the pre-specified coefficients $\mathbf{c}$. The experiments are run 1000 times to stabilize the observation. The tolerant approximation error is set to 1% of the signal energy for the MP algorithm. The default settings in Atomizer are used in the BP algorithm. For the NS algorithm, the threshold is uniformly decreased from the maximum of the initial coefficients to zero in 100 iterations. The BP and NS algorithms simultaneously modify all the coefficients. They do not obtain exact zero coefficients in general. The coefficients smaller than 1.0 are omitted for the BP and NS algorithms.

The MP algorithm ideally recovers the pre-specified coefficients 70.9% of the time. The ideal recovery rate is 98.7% for the BP algorithm and 98.3% for the NS algorithm. This suggests that NS and BP both have stronger sparsifying capability than MP. Instead of just examining whether they can exactly recover the four predetermined coefficients, the number of recovered nonzero coefficients is also counted. The resulting histograms are plotted in Fig. 3 where the $y$-axis represents the percentage of corresponding occurrences over 1000 experiments. An algorithm with the number of nonzero coefficients concentrated around four indicates good sparsifying capability. As can be observed in Fig. 3, the performances of the BP and NS algorithms are comparative and both are much better than that of the MP algorithm. Although the signal $\mathbf{f}$ is generated by $\mathbf{Rc}$, the sparsest representation of signal $\mathbf{f}$ under the dictionary is not necessarily $\mathbf{c}$ since both $\mathbf{R}$ and $\mathbf{c}$ are randomly produced. Therefore, the number of nonzero coefficients of the recovered

![Fig. 3  Ideal recovery performance for the three algorithms](image-url)
coefficients can be less than 4 as in Fig. 3. In this case, the MP, BP, and NS algorithms all obtain a sparser solution than $c$, which demonstrates their promising sparsifying capability.

### 2.2.2 Nonlinear approximation

In general applications, the sparsest representation for a given signal is not known under a dictionary. Hence, evaluation of the sparsifying capability by ideal recovery is not suitable for general signals and dictionaries. Nonlinear approximation (NLA) approximates the original signal by selecting the $M$ largest coefficients. Then, metrics for the reconstruction quality, such as the mean squared error (MSE) or the peak signal-to-noise ratio (PSNR), can be calculated for various $M$. The sparse representation with the higher reconstruction quality for the same $M$ has higher NLA power.

NLA curves of the MP, BP, and NS algorithms are presented in Fig. 4 for 1-D signals. The NS and MP algorithms consistently provide sparser representation than the BP algorithm for both Bumps and Mallat. For 2-D signals, the MP, BP, and NS algorithms are applied with 2-D DDWT for the two test images. The NLA curves plotted in Fig. 5 show that the coefficients generated by the three algorithms are much sparser than those given by the DDWT forward transform. The MP and BP algorithms have comparable performance, but the NS algorithm has the strongest sparsifying capability of the three algorithms. NS outperforms MP and BP by up to 2 dB for Barbara as shown in Fig. 5b.

![Fig. 4 Nonlinear approximation of coefficients obtained by the MP, BP, and NS algorithms for (a) Bumps and (b) Mallat under the redundant SWP dictionary.](image)

![Fig. 5 Nonlinear approximation curves for the forward transform, MP, BP, and NS algorithms using the 2-D DDWT dictionary.](image)

### 2.3 Practical applications

#### 2.3.1 Time-frequency analysis

Identification of time-frequency components in signals is an important signal processing task. The MP algorithm is originally developed for time-frequency analyses\[^1\]. The BP algorithm recovers finer time-frequency structures than the MP algorithm\[^12\]. No published work has reported results of time-frequency analyses using the NS algorithm. This study compares the time-frequency analysis performance of the MP, BP, and NS algorithms for Mallat (Fig. 2b) with the SWP dictionary. The obtained time-frequency planes in Fig. 6 show that the time-frequency analyses of the NS and BP algorithms are very close to the ideal result while the MP algorithm spreads energy over more
Fig. 6 Time-frequency analysis of Mallat signal with the NS, MP, and BP algorithms (A darker grayscale value associates with a stronger time-frequency component and vice versa.)

time-frequency locations. Other signals yield similar results. This results show the potential of the NS algorithm for time-frequency analyses.

### 2.3.2 Denoising

Recently, denoising with sparse representation has received considerable attention\[2,3\]. Here, the denoising performances of the three algorithms are compared for noisy signals generated by adding white Gaussian noise of different levels to the original signals. For the MP and NS algorithms, sparse representations are first obtained for noisy signals, while the hard thresholding then applied with a universal threshold \( T \). For BP, BPDN formulated in Eq. (8) is used for denoising. Both the Lagrange multiplier \( \lambda \) in the BPDN and \( T \) are set to \( \sigma \sqrt{2 \ln N} \), where \( \sigma \) is the standard deviation of the noise and \( N \) is the number of coefficients\[12\]. The signals are scaled to \([0 \ 255]\) and the denoising performance is evaluated based on the PSNR calculated from the original signal and the denoised reconstruction. As shown in Table 1, the MP algorithm gives the best denoising performance. NS performs the worst for most cases, which is not consistent with the strongest sparsifying capability shown in previous experiments. The reason may be that the NS algorithm fails to recover the sparse representation in the presence of noise since the noise is spread over the new coefficients at the feedback step, Eq. (11). Therefore, the NS algorithm should not be directly used for signal denoising.

### 2.3.3 Image coding

One natural application of sparse representation is image coding since less significant coefficients are to be coded. However, better NLA performance does not directly lead to higher coding efficiency since the entropy coding stage is quite important to the overall coding performance. This study investigates whether the high NLA performance of these algorithms can be translated to the high coding performance. The resulting DDWT coefficients in the NLA experiments are coded with SPIHT\[26\], a well-known wavelet-based coding scheme. Since the 2-D DDWT has 2 : 1 redundancy, the 2-D DDWT has double pyramidal structures. Therefore, each pyramid is separately coded with SPIHT. The coding results for Lena and Barbara at typical bit rates in bit per pixel (bpp) are listed in Tables 2 and 3. Of the three algorithms, the NS algorithm outperforms the BP algorithm by about 1 dB and the MP algorithm by 2 dB on average. In spite of the introduced redundancy, the NS-based coding scheme still shows better coding efficiency than the state-of-the-art JPEG2000 at low bit rates. The NS-based scheme also provides comparable performance at higher bit...
rates without special optimization applied to the subband coding. Effective use of the characteristics of the DDWT coefficients in the subband coding will further improve coding efficiency. These preliminary results verify the effectiveness of the NS algorithm in sparsifying redundant transform coefficients and show its potential for image coding. Further details about DDWT-based image coding were given by Yang et al.[6]

<table>
<thead>
<tr>
<th>Bit rate (bpp)</th>
<th>JPEG2000</th>
<th>DDWT-MP</th>
<th>DDWT-BP</th>
<th>DDWT-NS</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>29.94</td>
<td>29.39</td>
<td>28.83</td>
<td>30.52</td>
</tr>
<tr>
<td>0.2</td>
<td>32.95</td>
<td>32.30</td>
<td>31.80</td>
<td>33.40</td>
</tr>
<tr>
<td>0.3</td>
<td>34.86</td>
<td>34.01</td>
<td>33.69</td>
<td>34.94</td>
</tr>
<tr>
<td>0.4</td>
<td>36.12</td>
<td>35.07</td>
<td>34.84</td>
<td>36.24</td>
</tr>
<tr>
<td>0.5</td>
<td>37.24</td>
<td>35.96</td>
<td>36.01</td>
<td>37.12</td>
</tr>
<tr>
<td>0.8</td>
<td>39.22</td>
<td>37.44</td>
<td>37.87</td>
<td>39.09</td>
</tr>
<tr>
<td>1.0</td>
<td>40.35</td>
<td>38.28</td>
<td>38.91</td>
<td>39.98</td>
</tr>
</tbody>
</table>

Table 3  Coding performance for Barbara

<table>
<thead>
<tr>
<th>Bit rate (bpp)</th>
<th>JPEG2000</th>
<th>DDWT-MP</th>
<th>DDWT-BP</th>
<th>DDWT-NS</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>24.64</td>
<td>24.10</td>
<td>24.21</td>
<td>24.83</td>
</tr>
<tr>
<td>0.2</td>
<td>27.27</td>
<td>26.52</td>
<td>26.36</td>
<td>27.52</td>
</tr>
<tr>
<td>0.3</td>
<td>29.18</td>
<td>28.50</td>
<td>28.43</td>
<td>29.30</td>
</tr>
<tr>
<td>0.4</td>
<td>30.82</td>
<td>29.69</td>
<td>29.74</td>
<td>31.08</td>
</tr>
<tr>
<td>0.5</td>
<td>32.26</td>
<td>30.85</td>
<td>31.26</td>
<td>32.21</td>
</tr>
<tr>
<td>0.8</td>
<td>35.28</td>
<td>32.83</td>
<td>33.86</td>
<td>35.30</td>
</tr>
<tr>
<td>1.0</td>
<td>37.15</td>
<td>33.70</td>
<td>35.38</td>
<td>36.68</td>
</tr>
</tbody>
</table>

2.4 Complexity analysis

The computational complexity is quite important for successful application of an algorithm. Unfortunately, it is difficult to theoretically compare the complexities of the MP, BP, and NS algorithms. One can analyze the required computations in each step of these algorithms; however, the total number of iterations is dependent on the complexity of the signal and dictionary. Therefore, their complexities are compared based on numerical experiments.

Two types of experiments are conducted, i.e., Type I experiments using random redundant dictionary for 1-D signals and Type II experiments using 2-D DDWT. The experiment settings are the same as those in Section 2.2. Different problem scales are formed by varying the signal length as shown in Table 4. For the Type I experiments, 10% of the coefficients are randomly set to nonzero values to generate the signals 1000 times for each signal length. For the Type II experiments, the original 512×512 pixels Lena is down-sampled to the desired size when necessary. The signals are simple in the Type I experiments and more complicated in the Type II experiments. The Type I experiments are small scale applications while Type II experiments are large scale applications.

Table 4  CPU run times for the experiments of the three algorithms

<table>
<thead>
<tr>
<th>Type</th>
<th>Signal length</th>
<th>Run time (min)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>MP</td>
</tr>
<tr>
<td>I</td>
<td>20</td>
<td>0.09</td>
</tr>
<tr>
<td>(1000 runs)</td>
<td>500</td>
<td>13.83</td>
</tr>
<tr>
<td></td>
<td>1000</td>
<td>54.68</td>
</tr>
<tr>
<td>II</td>
<td>128×128</td>
<td>2.18</td>
</tr>
<tr>
<td></td>
<td>256×256</td>
<td>38.33</td>
</tr>
<tr>
<td></td>
<td>512×512</td>
<td>420.38</td>
</tr>
</tbody>
</table>

The CPU run times for these experiments are listed in Table 4. For the Type I experiments, the CPU run times for the MP and NS algorithms are very close while the BP algorithm requires more than 10 times the computational time. For the Type II experiments, the NS algorithm has the shortest run times. As in the Type I experiments, the BP algorithm is still rather time demanding. However, for the Type II experiments, the computational times for the MP algorithm increase dramatically. These results are consistent with the intrinsic complexities of the algorithms as analyzed below.

The complexity of each step for both MP and NS is \( O(n) \). Therefore, the computational times are mainly determined by the number of required steps for convergence. For simple signals such as in the Type I experiments, MP converges rapidly, thus consuming less run times. The MP algorithm requires about 300 steps for convergence for the signal length of 1000. For complicated signals such as Lena in the Type II experiments, the MP converges more slowly. For example, the MP requires 50,000 steps to achieve a reconstruction quality of 38.5 dB for the 512×512 pixels.
image size, which is relatively intense. Therefore, the MP algorithm has totally different computational complexities in the Type I and Type II experiments. An interesting property of the NS algorithm is that the number of steps required for convergence is weakly related to the signal complexity and problem scale. For example, NS requires 100 steps in the Type I experiments and only 128 steps in the Type II experiments. Therefore, the computation complexity of the NS algorithm is relatively low compared with the MP and BP algorithms for both simple and complicated signals, and both large and small scale problems. All the matrix-vector multiplications for each algorithm in the Type II experiments are carried out implicitly by the fast algorithms of the 2-D DDWT forward and inverse transform, which greatly reduces run times.

For the BP algorithm, the primal-dual log-barrier algorithm has to solve a large set of linear equations. The complexity is $O(n^3)$ for standard elimination methods. The implementation of Atomizer uses an iterative algorithm instead. However, the overall computation complexity is still considerably high[12]. Therefore, the BP algorithm consistently requires the most run time in both the Type I and Type II experiments.

### 3 Discussion

Various experiments have demonstrated that these algorithms have different performances in different scenarios. First, the NS algorithm shows the best performance in terms of sparsifying capability with the lowest computational complexity. Second, the computational complexity of the MP algorithm is strongly dependent on the signal complexity and increases dramatically with the problem scale for general complex signals such as natural images. Third, the BP algorithm is more computationally demanding than the NS and MP algorithms.

These observations based on the representative algorithms can be directly generalized to entire categories because algorithms within each category share the same principle and, thus, similar behavior. Specifically, greedy pursuit algorithms sequentially select atoms over a given dictionary, which raises their complexity for complicated signals and large scale problems. Algorithms based on $\ell_p$ norm regularization are generally more time consuming even when solved with the most advanced optimization tools. With the same iterative framework, the iterative shrinkage algorithms are computationally economical and have strong sparsifying capability. Therefore, the iterative shrinkage algorithms deserve the first consideration for typical applications.

However, one has to choose the most appropriate algorithm for a given application. For example, the NS algorithm performs quite well for time-frequency analyses. For image coding, the coding results using the coefficients sparsified by the NS algorithm are even better than those of JPEG2000 at low bit rates. However, the NS algorithm should not be directly employed for noisy signals since the closed-loop feedback from the noisy signals undermines its sparsifying capability. The MP and BP algorithms are free of this problem. Some recently developed algorithms are fusing the advantages of the previous algorithms from different categories. For example, the greedy basis pursuit borrows the greedy idea of the MP algorithm to reduce the computational complexity of the BP algorithm[27]. Iterative shrinkage can be incorporated into OMP to speed up the algorithm[15]. Moreover, a crucial stage for sparse representation is to select or design an appropriate redundant dictionary or transform for a specific application. However, most dictionaries, transforms, and algorithms are optimized separately. Better performance could be obtained by jointly designing these components.

### 4 Conclusions

This paper presents a comparative study of three representative algorithms for sparse representation in terms of their sparsifying capability and computational complexity. The performance differences between these algorithms are evaluated for three typical applications. The NS algorithm has stronger sparsifying capability with lower computational complexity than the well-known MP and BP algorithms and shows great potential for practical applications. Therefore, the NS algorithm is recommended for most applications. This comparative study provides guidelines for algorithm selection and for designing or improving algorithms for sparse representations.
References


