Hindawi
Computational Intelligence and Neuroscience
Volume 2018, Article ID 1365747, 9 pages
https://doi.org/10.1155/2018/1365747

Research Article

A Bat-Inspired Sparse Recovery Algorithm for Compressed Sensing

Wanning Bao, Haiqiang Liu, Dongbo Huang, Qianqian Hua, and Gang Hua

China University of Mining and Technology, Xuzhou 221116, China

Correspondence should be addressed to Gang Hua; ghua@cumt.edu.cn

Received 8 June 2018; Accepted 28 August 2018; Published 29 October 2018

Academic Editor: Paolo Del Giudice

Compressed sensing (CS) is an important research area of signal sampling and compression, and the essence of signal recovery in CS is an optimization problem of solving the underdetermined system of equations. Greedy pursuit algorithms are widely used to solve this problem. They have low computational complexity; however, their recovery performance is limited. In this paper, an intelligence recovery algorithm is proposed by combining the Bat Algorithm (BA) and the pruning technique in subspace pursuit. Experimental results illustrate that the proposed algorithm has better recovery performance than greedy pursuit algorithms. Moreover, applied to the microseismic monitoring system, the BA can recover the signal well.

1. Introduction

In recent years, the collected data quantity by the Internet of things (IoT) has increased dramatically. To facilitate storage and transmission, signal sampling based on the traditional Shannon–Nyquist sampling theory works as follows: massive data are collected at the sampling stage and most are discarded at the compression stage. This process is extremely wasteful. CS, which was proposed by D. Donoho, E. Candès, and T. Tao, brings a revolutionary breakthrough in signal sampling [1–3]. If only K nonzero elements exist in a signal, the signal is called K-sparse signal with sparsity K. CS theory indicates that sparse signals can be recovered more accurately using less measurements than the Nyquist sampling principle.

The signal recovery problem in CS is a nonconvex combinatorial optimization problem. It can be solved by three types of algorithms. The first type is greedy pursuit algorithms, including Matching Pursuit (MP), Orthogonal Matching Pursuit (OMP) [4], Generalized Orthogonal Matching Pursuit (GOMP) [5], and Subspace Pursuit (SP) [6]. They are two-stage algorithms. In the first stage, they seek support iteratively. In the second stage, the signal can be recovered by using the least square method. This kind of algorithm is effective; however, they have weak recovery performance. The second kind of algorithm is a convex optimization algorithm, which recovers signals by converting nonconvex problems to convex problems. The most common convex optimization algorithms are Basic Pursuit (BP) and Gradient Projection for Sparse Reconstruction (GPSR) [7]. Although they have high accuracy, their computational complexity is high. The third kind of recovery algorithm is Bayesian algorithms [8], which can be divided into the following two types: Maximum A Posteriori (MAP) Estimation and Hierarchical Bayesian. The former underlines a signal distribution, and the latter introduces one or two variables, which control the sparse signal. Bayesian algorithms achieve a balance of high accuracy and short recovery time [9, 10]. In recent years, some scholars have applied Swarm Intelligence Algorithms to signal recovery in CS, such as Particle Swarm Optimization (PSO) [11] and the Grey Wolf Optimizer Algorithm [12]. These methods have good global search ability. However, they have a slow convergence velocity and are easy to be trapped into local optimum.

Recently, Yang et al. simulated bat echolocation and proposed a Bat Algorithm (BA) [13] based on stochastic optimization, which has simple structure, fast convergence, strong robustness, and parallel computing. The BA combines the advantages of PSO, Harmony Search (HS),

Simulated Annealing (SA), and other algorithms. Many scholars applied it to optimization problems, such as numerical optimization, economic dispatch with random wind power [14], multirobot formation reconfiguration [15], microgrid operation management [16], and image processing [17]. In these areas, the BA shows better performance than do other intelligence optimizer algorithms.

In this paper, inspired by the BA and the greedy pursuit algorithm, an intelligence recovery algorithm for CS is proposed. OMP is used to initialize the swarm. The BA and the pruning technique are combined to update populations. The proposed algorithm outperforms the traditional greedy pursuit algorithms. Moreover, it runs faster than other swarm intelligence algorithms such as PSO.

The remainder of this paper is organized as follows: Section 2 briefly introduces the basic theory of CS and the BA. In Section 3, the proposed algorithm is introduced in detail. In Section 4, experiments are done to evaluate the algorithm performance. Finally, conclusions are drawn in Section 5.

The notations used in the rest of this paper include the following: $|\cdot|$ denotes the absolute value of a real number or the cardinality of a set. $\|\cdot\|_p (p = 0, 1, 2, \ldots)$ represents l_p – norm. For the vector $x \in R^n$, $\hat{x}$ is the estimate of x. In a matrix $A \in R^{m\times n}$, a_j is the j-th column of A. $E = \{1 \le i \le n | x_i \ne 0\}$, which denotes the positions of nonzero elements in x, and it is called support set. $S = \{1, 2, \cdots, n\}$ is the universal set of support. The matrix $A_E = \{a_j\}_{j\in E\subseteq S}$ is submatrix of A. The pseudoinverse of a matrix A is defined as $A^+ = (A^T A)^{-1} A^T$, where T denotes the transpose operator.

2. Basic Theory

2.1. Compressed Sensing. Supposing an n-dimensional signal $y = (y_1, y_2, \ldots, y_n)^T$ can be represented on a basis $\Psi \in R^{n\times n}$ as $y = \Psi x$, where $x = (x_1, x_2, \ldots, x_n)^T$, if there are only K nonzero elements in $x = (x_1, x_2, \ldots, x_n)^T$, y is called K – sparse signal with sparsity K, and the positions of nonzero elements in x are called the support set, which can be denoted as $E = \{1 \le i \le n | x_i \ne 0\}$. The signal y can be sampled by projection onto the measurement matrix $\Phi \in R^{m\times n} (m \ll n)$ as

$$b = \Phi y = \Phi\Psi x = Ax, \tag{1}$$

where $b = (b_1, b_2, \ldots, b_m)^T$ is the measurement signal, y is the original signal, x is the sparse signal, Φ is the measurement matrix, and Ψ is the sparse basis. $A \in R^{m\times n}$ is the sensing matrix.

It can be seen that the data quantity is greatly reduced because $b = \Phi y$ is an underdetermined system of equations. There are infinitely many solutions to recover y from b. However, if y is sparse, it can be recovered from b by solving the following l_0 minimization problem:

$$\begin{aligned} &\min_x \quad \|x\|_0, \\ &\text{s.t.} \quad Ax = b, \end{aligned} \tag{2}$$

where $\|\cdot\|_0$ represents l_0 – norm.

2.2. Orthogonal Matching Pursuit (OMP). In the proposed algorithm, we use OMP to determine the initial solution.

The OMP algorithm can be stated in Algorithm 1 [4].

2.3. Bat Algorithm. In 2010, Yang et al. simulated the characteristics of bats' predatory behaviors and proposed BA [13, 18], which is a new Swarm Intelligence Optimization Algorithm.

The first step is randomly initializing the bat positions $x_i(0)$ and the velocities $v_i(0)$. Then, the BA updates the velocities and positions according to certain strategies. The velocities and positions of the i-th bat at $t + 1$ iteration are updated as follows:

$$f_i = f_{\min} + (f_{\max} + f_{\min})\beta, \tag{3}$$

$$v_i(t+1) = v_i(t) + (x_i(t) + x_*) \cdot f_i, \tag{4}$$

$$x_i(t+1) = x_i(t) + v_i(t+1), \tag{5}$$

where f_i is the pulse frequency of the i-th bat, $f_i \in [f_{\min}, f_{\max}]$, $\beta \in [0, 1]$ is a random number. $x_i(t)$ and $x_i(t+1)$ represent the positions of the i-th bat at the t and the $t + 1$ iteration, respectively; $v_i(t)$ and $v_i(t+1)$ are the velocities of the i-th bat at the t and the $t + 1$ iteration, respectively; and x_* denotes the global best position of the swarm.

For each bat, the algorithm generates a random number $rand1 \in [0, 1]$. If $rand1 > r_i(t)$, the position of this bat is updated according to formula (6), where $r_i(t)$ is the rate of pulse emission. The random walk is regarded as a procedure of the local search [13].

$$x_i(t+1) = x_j^* + \varepsilon L(t), \tag{6}$$

where x_j^* is a random solution in the best solution set composed of the best solution of each bat, $\varepsilon \in [-1, 1]$ is a random number, and $L(t)$ is the mean of the pulse loudness in the t-th iteration.

In addition, with the iterative number increasing, the pulse loudness L_i and the rate of pulse emission r_i need to be updated. During prey searching, the pulse loudness L_i will gradually decline, and the rate of pulse emission r_i will also gradually increase. The updating formulas of L_i and r_i are found in formulas (7) and (8), respectively:

$$L_i(t+1) = \alpha L_i(t), \tag{7}$$

$$r_i(t+1) = r_i(0)[1 - \exp(-\gamma t)], \tag{8}$$

where α and γ are constants, $0 < \alpha < 1$, and $0 < \gamma$.

The BA algorithm can be described in Algorithm 2 [13].

3. A Bat-Inspired Sparse Recovery Algorithm

3.1. Fitness Function. If the sparsity level K satisfying $K < \text{spark}(A)$ is known as a priori, problem (2) can be approximated as follows [11, 12]:

$$\begin{aligned} &\text{argmin}_x \quad \|Ax - b\|_2, \\ &\text{s.t.} \quad \|x_0\| \le K, \end{aligned} \tag{9}$$

Input: Sensing matrix A, measurement signal b, sparsity K.
Output: Estimated signal $\widehat{x}$, estimated support set $\widehat{E}$
Initialization: Residual signal $r_0 = b$, estimated index set $\widehat{E}_0 = \phi$, estimated atomic set $A_0 = \phi$, iteration number $t = 1$;
The t-th ($1 \le t < K$) iteration:
Step 1: Find $\lambda_t = arg\max_{j=1,2,\ldots,N} |\langle r_{t-1}, a_j \rangle|$
Step 2: Update the estimated support set and the estimated atomic set $\widehat{E}_t = \widehat{E}_{t-1} \cup \lambda_t, A_t = A_{t-1} \cup a_{\lambda_t}$;
Step 3: Estimate the signal $\widehat{x}_t = (A_t^T A_t)^{-1} A_t^T b$;
Step 4: Update the residual $r_t = b - A_t \widehat{x}_t$, update iteration counter $t = t + 1$;
Judgement: If $t \le K$, continue iterating; if $t > K$, stop the iteration, and output the estimated support set $\widehat{E}$ and the estimated signal $\widehat{x}$.

Algorithm 1: Orthogonal Matching Pursuit (OMP).

Fitness function: $f(x)$
Initialization:
Randomly initialize the positions x_i and velocities v_i of the bat population.
Initialize the pulse frequency f_i of each bat according to formula (3).
Initialize the rate of pulse emission $r_i(0)$ and the pulse loudness $L_i(0)$.
Calculate the fitness value of each bat and find the best solution in the population.
Set the stopping criterion σ.
Termination condition:
Iteration counter reaches $t_{\max}$ or the fitness value is smaller than σ.
Iteration:
Step 1: Update the bat velocity $v_i(t)$ and position $x_i(t)$ according to formulas (3)~(5);
Step 2: If ($rand1 > r_i(t)$), select a random solution in the best solutions set and update position of this bat by the random walk around the selected best solution as shown in formula (5);
Step 3: Calculate the fitness value of each bat's new position;
Step 4: If ($rand2 > L_i(t) and f(x_i(t)) < f(x_*)$), accept the new solution and adjust $r_i(t)$, $L_i(t)$ according to formulas (7) and (8);
Step 5: Find the new best solution x_* and update the best solutions set;
Step 6: If termination condition is satisfied, stop the iteration; otherwise, continue.

Algorithm 2: Bat Algorithm (BA).

where b is the measurement signal, and $A \in R^{m \times n}$ is the sensing matrix.

From greedy pursuit algorithms, we can see that the signal can be recovered by using a two-step strategy. First, estimate the support set and then estimate the signal using the least square method.

$$\begin{aligned} x_{\widehat{E}} &= A_{\widehat{E}}^{+} b, \\ x_{S-\widehat{E}} &= 0, \end{aligned} \tag{10}$$

where $\widehat{E}$ denotes the estimated support set and $S = \{1, 2, \cdots, n\}$. The vector x_E consists of the entries of $x \in R^n$ indexed by $i \in E \subseteq S$. The matrix $A_E = \{a_j\}_{j \in E \subseteq S}$ is the submatrix of A, and a_j is the j-th column of A. $A^+ = (A^T A)^{-1} A^T$ is the pseudoinverse of a matrix A, where T denotes the transpose operator.

Once the support set is estimated accurately, the following equation must be satisfied.

$$\left\| A_{\widehat{E}} A_{\widehat{E}}^{+} b - b \right\|_2 = 0, \tag{11}$$

where $\| \cdot \|_2$ represents $l_2 -$ norm.

Because $\| A_{\widehat{E}} A_{\widehat{E}}^{+} b - b \|_2 \ge 0$, we define the fitness function as

$$f(\widehat{E}) = \left\| A_{\widehat{E}} A_{\widehat{E}}^{+} b - b \right\|_2. \tag{12}$$

3.2. Initialization. Suppose there are I bats, the position of the first bat is initialized as the estimated support set of the OMP. For the i-th ($2 \le i \le I$) bat, its position is initialized as follows.

Randomly choose $q = 0.8 \times m$ elements from the set $S = \{1, 2, \ldots, n\}$ and form the set $V_i(0)$; here, $V_i(t)$ is regarded as the velocity of the i-th bat at the t iteration. Then, the position of the i-th ($2 \le i \le I$) bat is a set formed by K indices corresponding to the maximum absolute values in $A_{V_i(0)}^{+} \cdot b$. Other related parameters are initialized as follows: the pulsing frequency f_i of each bat is initialized according to formula (3), where $f_{\min} = 0.8 and f_{\max} = 1$. The initial emission rate $r_i(0) \in [0, 1]$. The pulse loudness $L_i(0) \in [0, 1]$, β is a random number between 0 and 1, and ε is a random number between -1 and 1, $0 < \alpha < 1$, and $0 < \gamma$.

3.3. Update Strategy. The main innovation of our algorithm is the update strategy. The iterative process combines the

thought of BA and greedy pursuit algorithm. The update strategy can be divided into three parts.

The first part is the update of velocity and position: a set $Q_i(t)$ is formed with $\beta \cdot f_i \cdot K$ elements selected randomly from the best solution E_*; we define $U_i(t) = E_i(t) \cup Q_i(t)$. If $|U_i(t)| \le q$, a set $G_i(t)$ consists of $q - |U_i(t)|$ elements selected randomly from the set $S - U_i(t)$, update $V_i(t+1) = U_i(t) \cup G_i(t)$. If $|U_i(t)| > q$, the set $G_i(t)$ consists of $q - K$ elements selected randomly from the set $Q_i(t) - E_i(t)$, update $V_i(t+1) = E_i(t) \cup G_i(t)$. The position of the i-th bat $E_i(t+1)$ is a set composed of the indices corresponding to the K maximum absolute values in $A^+_{V_i(t+1)} \cdot b$.

The second part is the local search: generate a random number $rand1 \in [0, 1]$; if $rand1 > r_i(t)$, select a random solution E_j^* from the best solutions set and update the position of this bat by randomly replacing $\varepsilon \cdot L_i(t) \cdot K$ elements of this solution with other elements in $S - E_j^*$. After updating all positions, calculate the new fitness $f(E_i(t+1))$ of each bat. Specifically, in order to accelerate the convergence speed of this algorithm, all the new solutions will be accepted in the algorithm.

The third part is the adjustment of the emission rate $r_i(t)$ and the pulse loudness $L_i(t)$ as formulas (7) and (8). If $f(E_i(t+1)) < f(E_i^*)$, update the best solutions set $E_i^* = E_i(t+1)$. If $f(E_i(t+1)) < f(E_*)$, update the global best solution $E_* = E_i(t+1)$.

3.4. Stopping Criterion. If $f(\widehat{E}) < \sigma$, the iteration is terminated; here, σ is the termination threshold. Moreover, in order to avoid too many iterations, set a maximum allowed iteration number N_max. If the iteration number is larger than N_max, the iteration is also terminated.

The algorithmic process in Algorithm 3.

3.5. Efficiency Analysis. The computational complexity of the algorithm is mainly dependent on the initialization and iteration.

The algorithm initializes solutions using OMP, which has the complexity $O(Kmn)$ [19].

In each iteration, the complexity is mainly dependent on three operations. The first is the multiplication of matrix, which has the computational complexity $O(mn)$. The second is sorting of an n-dimensional vector, which has the computational complexity $O(nlogn)$. The third is the least square, which has the computational complexity $O(m^3)$. In summary, the computational complexity in the iteration phase is $O(Im^3 + Imn + Inlogn)$, where I is the population size.

In general, the computational complexity of the proposed algorithm is similar to OMP when the signal sparsity is small. The complexity depends on the number of iterations when the sparsity is larger or when the measurement number is small. Moreover, this algorithm uses the idea of the BA, which combines the advantages of PSO, HS, SA, and other intelligence algorithms to optimize the convergence speed and search accuracy. It usually runs faster than other intelligence algorithms. OMP can generate an initial solution closed to the target, which accelerates the convergence of the BA. In addition, as swarm intelligence algorithm, it can run in parallel to reduce the running time [12, 20].

4. Experiments

In this section, we compare this algorithm with OMP, GOMP, SP, Fast Laplace, and PSO, where Fast Laplace is a Bayesian-based algorithm, and PSO is a swarm intelligence algorithm for CS. Then, we apply this algorithm to recover the real microseismic signals.

Candes and Tao proved that the Gaussian random matrix [2, 21, 22] with independent identically distribution can be a universal measurement matrix. Therefore, the measurement matrices in these experiments are the Gaussian random matrix with the size of $m \times n$. The number of iterations of the OMP, GOMP, and SP algorithms is K. The maximum number of iterations of PSO and the proposed algorithm are $N_{\max} = 100$. The algorithm will stop when $f(E^*) < \sigma$ or the number of iterations reaches N_max.

The software and hardware environment of the experiment is as follows:

Processor: Intel (R) Pentium (R) CPU G3220 @ 3.00 GHz 3.0

Memory: 4.00 G

Operating system: 64-bit Windows7

Simulation software: MATLAB R2014a

4.1. Comparison with Other Algorithms. In the first experiment, we compare the recovery performance of different algorithms against the change of sparsity. We use a 128×256 random Gaussian matrix to sample a 256-length sparse signal. The sparsity level is set as $K = 35, 40, 45, 50, 55, 60, 64, 70$. The parameters are set as $n = 256$, $m = 128$, $\sigma = 1e-8$, $f_{\min} = 0.8$, and $f_{\max} = 1$; the rate of pulse emission $r_i(0)$ and the pulse loudness $L_i(0)$ are set between 0 and 1; $\beta = 0.2$, $\varepsilon = 0.2, \alpha = 0.9$, and $\gamma = 0.9$. Here, we compare the algorithm performance with the following two metrics: recovery error and exact recovery rate.

Recovery error is a metric to evaluate the error between the original sparse signal and the recovered sparse signal. The recovery error is defined as follows (13):

$$\text{Recovery error} = \frac{1}{\text{CNT}} \sum_{j=1}^{\text{CNT}} \left(\frac{\left\| x^{(j)} - \widehat{x}^{(j)} \right\|_2^2}{\left\| x^{(j)} \right\|_2^2} \right), \tag{13}$$

where x is the original sparse signal, $\widehat{x}$ is the recovered sparse signal, and CNT = 100 is the cycle number to obtain the average performance. The simulation result of recovery error is shown in Figure 1(a). The exact recovery rate is obtained by calculating the number of times that the estimated values are exact. If $x - \widehat{x}_2 < 10^{-6}$, this experiment is considered successful. In 100 experiments, if there are s successful experiments, the exact recovery rate is $(s/100)$. The exact recovery rates of different algorithms are shown in Figure 1(b).

It can be seen that when the sparsity is large, the recovery errors of three commonly used greedy pursuit algorithms

Input: Sensing matrix A, measurement vector b, sparsity level K, termination threshold σ, maximum allowed iterative number N_max.
Output: The estimated signal $\widehat{x}$
Fitness function: $f(\widehat{E}) = \|A_{\widehat{E}} A_{\widehat{E}}^{+} b - b\|_2$
Initialization: Initialize the positions and velocities of bats (Section 3.2).
Termination condition: If $f(E^*) < \sigma$ or $t \geq N\_max$, terminated the iteration.
Iteration:

Step 1: A set $Q_i(t)$ is formed with $\beta \cdot f_i \cdot K$ elements selected randomly from the best solution E_*, define $U_i(t) = E_i(t) \cup Q_i(t)$.

Step 2: If $|U_i(t)| \leq q$, a set $G_i(t)$ consists of $q - |U_i(t)|$ elements selected randomly from the set $S - U_i(t)$, update $V_i(t+1) = U_i(t) \cup G_i(t)$. If $|U_i(t)| > q$, the set $G_i(t)$ consists of $q - K$ elements selected randomly from the set $Q_i(t) - E_i(t)$, update $V_i(t+1) = E_i(t) \cup G_i(t)$.

Step 3: The position of the i-th bat $E_i(t+1)$ is a set composed of the indices corresponding to the K maximum absolute values in $A^{+}_{V_i(t+1)} \cdot b$.

Step 4: If $rand1 > r_i(t)$, select an optimal solution from the best solutions set and then replace $\varepsilon \cdot L_i(t) \cdot K$ elements of this solution with other elements in $S - E_j^*$.

Step 5: Calculate the new fitness $f(E_i(t+1))$.

Step 6: If $rand2 > L_i(t)$, adjust $r_i(t)$ and $L_i(t)$.

Step 7: If $f(E_i(t+1)) < f(E_i^*)$, update the best solutions set $E_i^* = E_i(t+1)$. If $f(E_i(t+1)) < f(E_*)$, update the best solution $E_* = E_i(t+1)$.

Step 8: If termination condition is satisfied, stop iteration; otherwise, continue.

Algorithm 3: A Bat-Inspired Sparse Recovery Algorithm for CS.

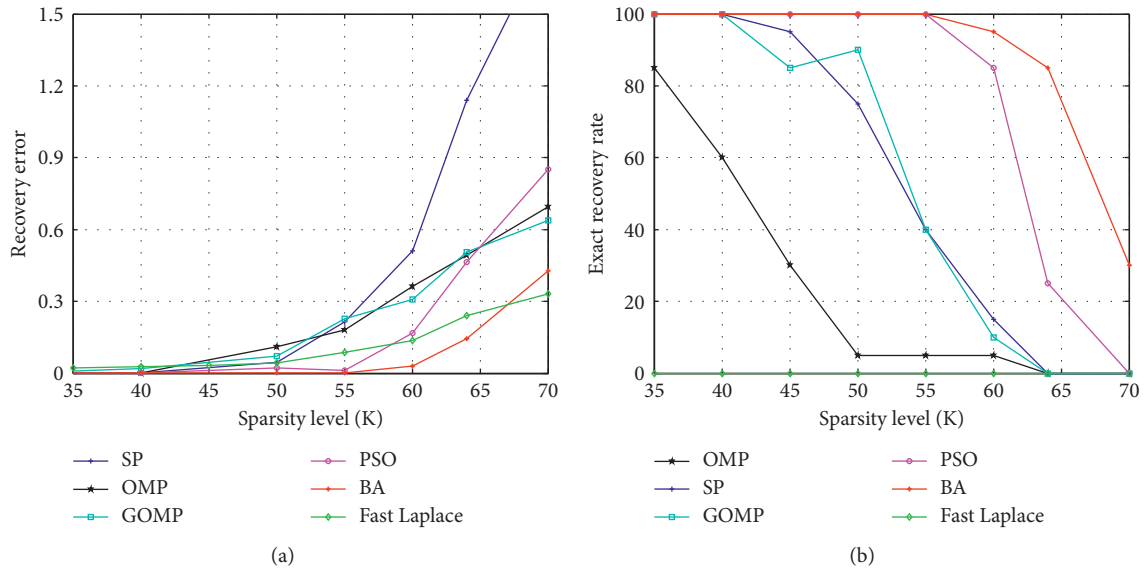

Figure 1: Comparison of the algorithm recovery performance against the change of sparsity with $n = 256$, $m = 128$, and the sparsity $K = 35, 40, 45, 50, 55, 60, 64, 70$. (a) Recovery error. (b) Exact recovery rate.

such as OMP, GOMP, and SP increase rapidly, and the exact recovery rate decreases quickly. The recovery error of the Fast Laplace algorithm is lower than that of traditional greedy pursuit algorithms. However, experiments demonstrate that the exact recovery rates of the Bayesian algorithm are 0 because there exists no experiment satisfying $x - \widehat{x}_2 < 10^{-6}$. The swarm intelligence algorithms also perform well. The recovery error of the proposed algorithm is lower than 0.4, and the exact recovery rate is higher than other five algorithms when the sparsity is very large.

In the second experiment, we compare the recovery performances against the change of measurement number. We use a $m \times 256$ Gaussian matrix to sample a 256-length signal with sparsity 30. The measurement number is set as $m = 53, 58, 63, 68, 78, 88, 98, 108, 118$, and other parameters are the same to the first experiment.

Figures 2(a) and 2(b) illustrate the recovery error and the exact recovery rate, respectively, of the six algorithms under different measurement numbers. As shown in the figure, the recovery of the proposed algorithm performs better than those commonly used greedy pursuit algorithms such as OMP, GOMP, and SP, when the number of measurement vectors is small. It also has more advantages compared with the PSO algorithm and the Fast Laplace algorithm.

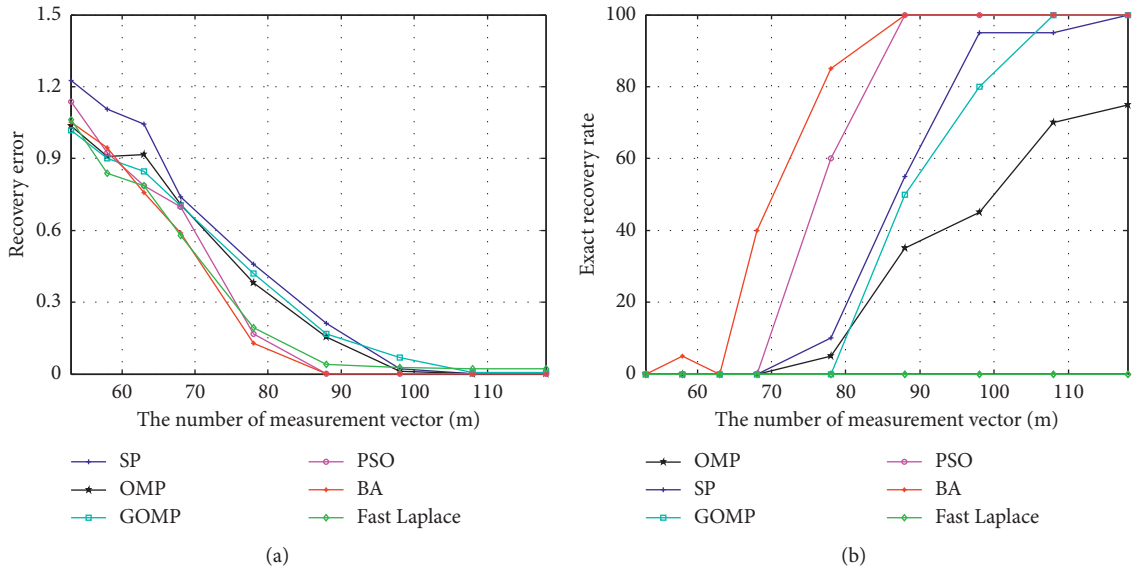

Figure 2: Comparison of the signal recovery performance against the change of measurement number with $n = 256$, $m = 53, 58, 63, 68, 78, 88, 98, 108, 118$ and sparsity $K = 30$. (a) Recovery error. (b) Exact recovery rate.

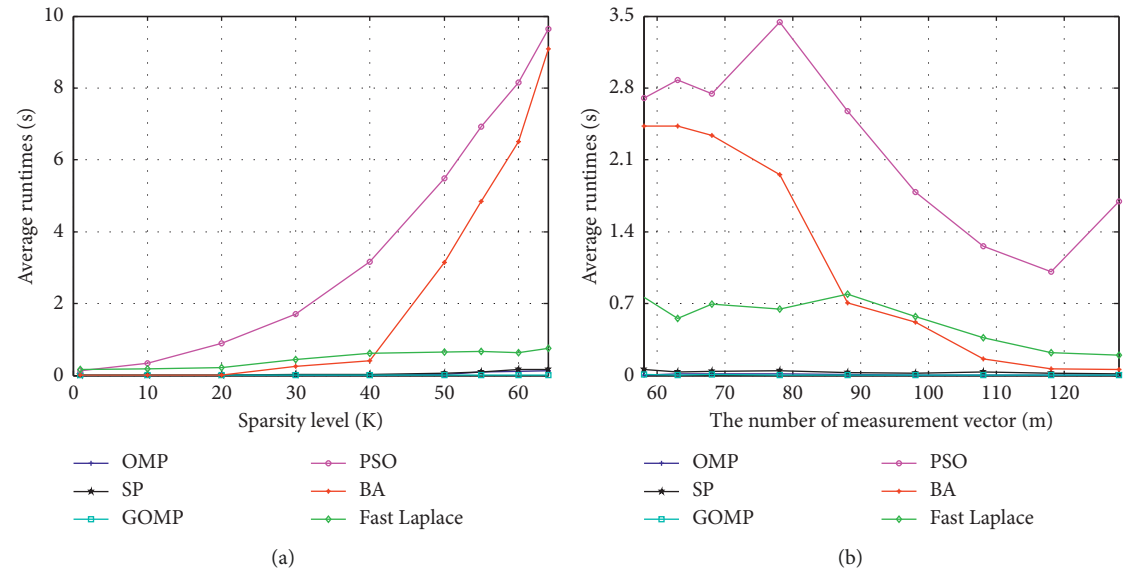

Figure 3: Comparison of average running times. (a) Running time of the six algorithms against the change of sparsity with $n = 256$, $m = 128$, and $K = 1, 10, 20, 30, 40, 50, 55, 60, 64$. (b) Running time of the six algorithms against the change of measurement number with $n = 256$, $K = 30$, and $m = 53, 58, 63, 68, 78, 88, 98, 108, 118$.

In the third experiment, we compare the recovery efficiency of different algorithms. Figure 3(a) is the running time of the six algorithms with different sparsity. The sparsity level is set as $K = 1, 10, 20, 30, 40, 50, 55, 60, 64$. Figure 3(b) is the running time of the six algorithms with different measurement number. The measurement number is taken as $m = 58, 63, 68, 78, 88, 98, 108, 118, 128$. Other parameters are the same as those of the first experiment, and the average time of each algorithm in recovering the signal is counted. As shown in Figure 3, when the sparsity is small or the measurement number is large, the recovery efficiency of this algorithm is similar to the commonly used greedy pursuit algorithms such as OMP. Although the running time of the algorithm increases when the sparsity is large or the measurement number is small, the algorithm is more efficient than other swarm intelligence algorithms such as PSO. However, as a swarm intelligence algorithm, it can run in parallel to reduce the running time [12, 20].

4.2. Application in Microseismic Signals. Microseismic monitoring is an effective technology for accident prediction in coal mines. Under the background of coal mine IoT, the collected data quantity by numerous microseismic nodes in

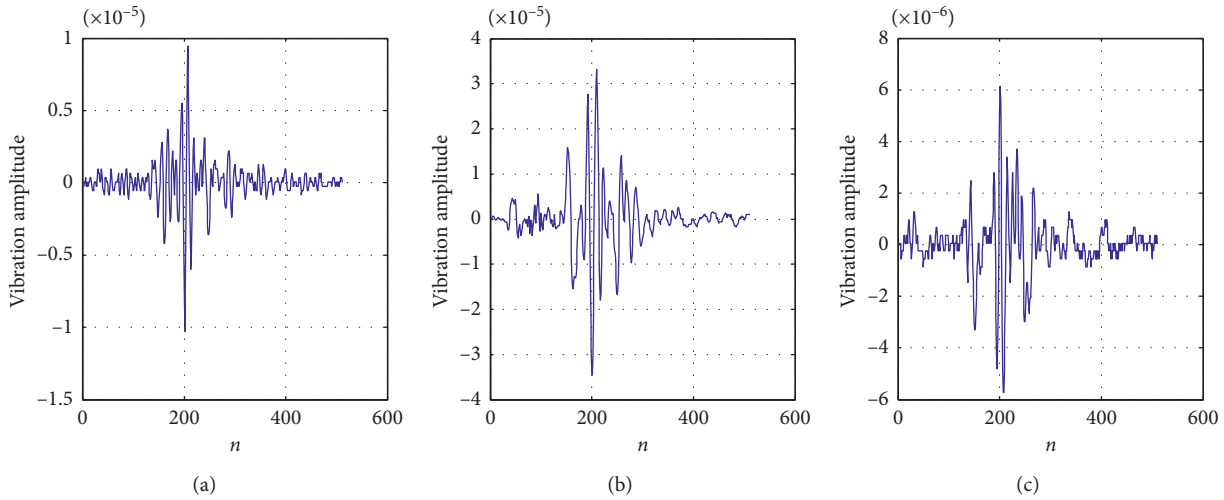

Figure 4: Original waveforms of the three microseismic signals: (a) signal 1, (b) signal 2, and (c) signal 3.

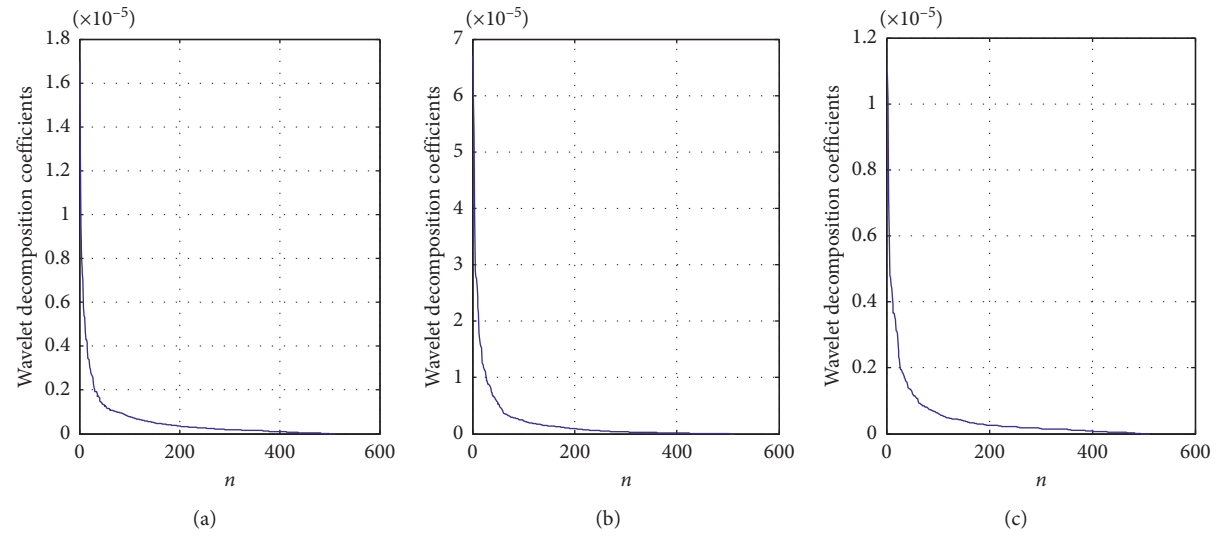

Figure 5: Wavelet decomposition coefficients of three microseismic signals: (a) signal 1, (b) signal 2, and (c) signal 3.

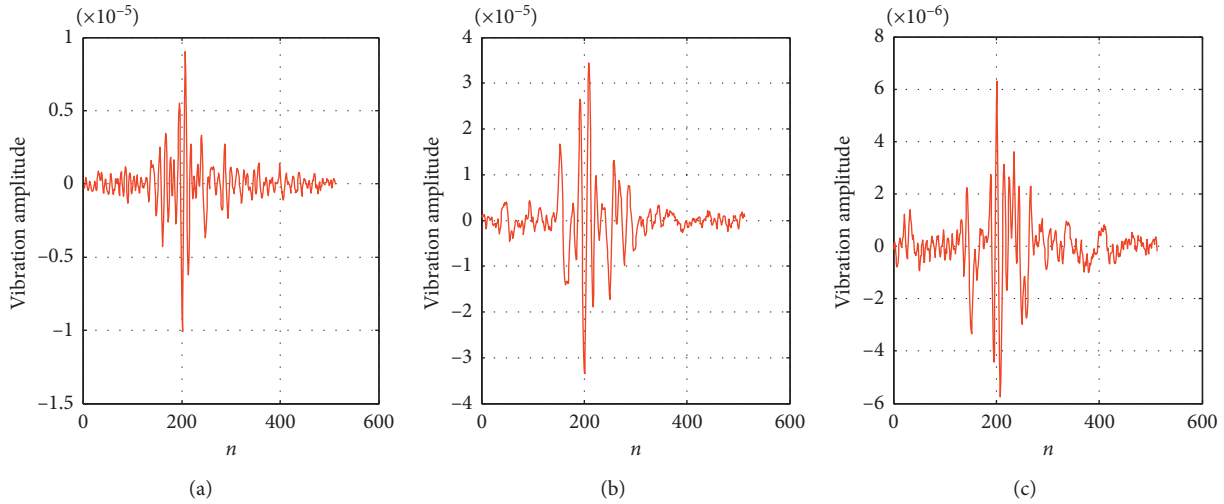

Figure 6: Recovered waveforms of the three microseismic signals: (a) signal 1, (b) signal 2, and (c) signal 3.

the monitoring area are massive [23]. The effective compression of microseismic data can reduce the storage space and energy consumption. It plays an important role in improving the data transmission rate. Participants of the project collected three microseismic signals from a coal mine in Anhui Province, China. In this section, we use these signals as the original signal and apply the proposed algorithm to recover signal.

We perform experiments on three different microseismic signals. The length of the signal is $n = 512$, as shown in Figure 4. It can be seen that the microseismic signals are one-dimensional time-frequency signals. Microseismic signals are not sparse in the time domain; however, their representations under a certain basis are sparse or compressible.

In this experiment, the 2-layer db2 wavelet transform [24] is used as sparse basis. Wavelet transform uses a cluster of functions to approximate a signal which can be seen as a decomposition of the signal. It divides the signal into a low-frequency signal and multiple high-frequency signals. The low-frequency signal is the stationary part of the signal. The high-frequency signal indicates the details of the signal. Figure 5 shows the wavelet decomposition coefficients of three groups of microseismic signals.

The measurement matrix used in this experiment is a 256×512 random Gaussian matrix. The sparsity level is $K = 128$ and $\sigma = 1e - 5$. Figure 6 shows the three groups of recovered signals. The recovery errors in the three groups of experiments are 0.1862, 0.1250, and 0.1512.

5. Conclusions

In this paper, an Intelligence Sparse Recovery Algorithm is proposed for CS inspired by the BA and the pruning technique of SP. The algorithm inherits the global search ability and the local search ability of the BA, and we use OMP to obtain a better initial solution to accelerate the convergence. Compared with the traditional pursuit algorithm, the algorithm in this paper has better recovery performance under large sparsity and small measurement. At the same time, it runs faster than the intelligent optimization algorithm such as PSO. The proposed algorithm can be applied to the microseismic monitoring system, which can recover the signal well. In future research, performance and efficiency of the algorithm will be more rigorously analyzed.

Data Availability

The data of microseismic signals used to support the finding of this study have not been made available for the time being because the data are obtained through cooperation between us and the Coal Mine Group, and it must be approved by both parties before it can be disclosed.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was financially supported by the National Science Foundation of China (grant no. 51574232).

References

[1] Y. Tsaig and D. L. Donoho, "Extensions of compressed sensing," *Signal Processing*, vol. 86, no. 3, pp. 549–571, 2006.

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

[3] E. J. Candes and T. Tao, "Near-optimal signal recovery from random projections: universal encoding strategies?," *IEEE Transactions on Information Theory*, vol. 52, no. 12, pp. 5406–5425, 2006.

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

[5] W. Jian, S. Kwon, and B. Shim, "Generalized orthogonal matching pursuit," *IEEE Transactions on Signal Processing*, vol. 60, no. 12, pp. 6202–6216, 2011.

[6] W. Dai and O. Milenkovic, "Subspace pursuit for compressive sensing signal reconstruction," *IEEE Transactions on Information Theory*, vol. 55, no. 5, pp. 2230–2249, 2009.

[7] M. A. T. Figueiredo, R. D. Nowak, and S. J. Wright, "Gradient projection for sparse reconstruction: application to compressed sensing and other inverse problems," *IEEE Journal of Selected Topics in Signal Processing*, vol. 1, no. 4, pp. 586–597, 2008.

[8] S. D. Babacan, R. Molina, and A. K. Katsaggelos, "Bayesian compressive sensing using laplace priors," *IEEE Transactions on Image Processing*, vol. 19, no. 1, pp. 2873–2876, 2009.

[9] M. M. Abo-Zahhad, A. I. Hussein, and A. M. Mohamed, "Compressive sensing algorithms for signal processing applications: a survey," *International Journal of Communications Network and System Sciences*, vol. 8, no. 5, pp. 197–216, 2015.

[10] Y. Arjoune, N. Kaabouch, H. E. Ghazi, and A. Tamtaoui, "Compressive sensing: performance comparison of sparse recovery algorithms," in *Proceedings of IEEE Annual Computing and Communication Workshop and Conference*, pp. 1–7, November 2017.

[11] X. P. Du, L. Z. Cheng, and L. F. Liu, "A swarm intelligence algorithm for joint sparse recovery," *IEEE Signal Processing Letters*, vol. 20, no. 6, pp. 611–614, 2013.

[12] H. Liu, G. Hua, H. Yin, and Y. Xu, "An intelligent Grey Wolf optimizer algorithm for distributed compressed sensing," *Computational Intelligence and Neuroscience*, vol. 2018, Article ID 1723191, 10 pages, 2018.

[13] X. S. Yang, "A new metaheuristic bat-inspired algorithm," *Computational Intelligence*, vol. 284, pp. 65–74, 2010.

[14] H. Liang, Y. Liu, Y. Shen, F. Li, and Y. Man, "A hybrid bat algorithm for economic dispatch with random wind power," *IEEE Transactions on Power Systems*, vol. 33, no. 5, pp. 5052–5061, 2018.

[15] G. Li, H. Xu, and Y. Lin, "Application of bat algorithm based time optimal control in multi-robots formation reconfiguration," *Journal of Bionic Engineering*, vol. 15, no. 1, pp. 126–138, 2018.

[16] B. Bahmani-Firouzi and R. Azizipanah-Abarghooee, "Optimal sizing of battery energy storage for micro-grid operation management using a new improved bat algorithm,"

International Journal of Electrical Power and Energy Systems, vol. 56, no. 3, pp. 42–54, 2014.

[17] S. C. Satapathy, N. S. M. Raja, V. Rajinikanth, A. S. Ashour, and N. Dey, "Multi-level image thresholding using Otsu and chaotic bat algorithm," *Neural Computing and Applications*, vol. 29, no. 12, pp. 1–23, 2016.

[18] X. S. Yang and A. H. Gandomi, "Bat algorithm: a novel approach for global engineering optimization," *Engineering Computations*, vol. 29, no. 5, pp. 464–483, 2012.

[19] J. A. Tropp and S. J. Wright, "Computational methods for sparse solution of linear inverse problems," *Proceedings of the IEEE*, vol. 98, no. 6, pp. 948–958, 2010.

[20] B. Li and K. Wada, "Parallelizing particle swarm optimization," in *Proceedings of 2005 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PacRim)*, pp. 288–291, Victoria, B.C., Canada, August 2005.

[21] E. J. Candès, J. K. Romberg, and T. Tao, "Stable signal recovery from incomplete and inaccurate measurements," *Communications on Pure and Applied Mathematics*, vol. 59, no. 8, pp. 1207–1223, 2006.

[22] Y. Arjoune, N. Kaabouch, H. E. Ghazi, and A. Tamtaoui, "A performance comparison of measurement matrices in compressive sensing," *International Journal of Communication Systems*, vol. 31, no. 10, article e3576, 2018.

[23] X. Zhao, S. Liu, X. Shen, and Y. Deng, "Micro-seismic data compression and reconstruction based on distributed compressed sensing," *Journal of China University of Mining and Technology*, vol. 47, no. 1, pp. 128–138, 2018.

[24] S. G. Mallat, *A Wavelet Tour of Signal Processing: The Sparse Way*, Academic Press, Cambridge, MA, USA, 2009.