Hindawi
Computational Intelligence and Neuroscience
Volume 2018, Article ID 1723191, 10 pages
https://doi.org/10.1155/2018/1723191

Research Article

An Intelligent Grey Wolf Optimizer Algorithm for Distributed Compressed Sensing

Haiqiang Liu, Gang Hua, Hongsheng Yin, and Yonggang Xu

China University of Mining and Technology, Xuzhou 221116, China

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

Received 1 October 2017; Accepted 31 December 2017; Published 31 January 2018

Academic Editor: Raşit Köker

Distributed Compressed Sensing (DCS) is an important research area of compressed sensing (CS). This paper aims at solving the Distributed Compressed Sensing (DCS) problem based on mixed support model. In solving this problem, the previous proposed greedy pursuit algorithms easily fall into suboptimal solutions. In this paper, an intelligent grey wolf optimizer (GWO) algorithm called DCS-GWO is proposed by combining GWO and q-thresholding algorithm. In DCS-GWO, the grey wolves' positions are initialized by using the q-thresholding algorithm and updated by using the idea of GWO. Inheriting the global search ability of GWO, DCS-GWO is efficient in finding global optimum solution. The simulation results illustrate that DCS-GWO has better recovery performance than previous greedy pursuit algorithms at the expense of computational complexity.

1. Introduction

Compressed sensing (CS) [1, 2] is a new signal sampling theory which has broken through the limit of Nyquist sampling theorem. If there are no more than k nonzero entries in the signal $\mathbf{x} \in R^n$, $\mathbf{x}$ is called a sparse signal and the sparsity of $\mathbf{x}$ is k. If $\mathbf{x}$ is sparse, it can be recovered from much fewer samples. We can get the measurement signal $\mathbf{y} = \mathbf{\Phi x}$ by projecting $\mathbf{x}$ onto the measurement matrix $\mathbf{\Phi} \in R^{m \times n}$, where $m \ll n$. Because $m < n$, it is an NP-hard problem to recover $\mathbf{x}$ from $\mathbf{y}$. However, if $k < m < n$ and $\mathbf{\Phi}$ satisfies the Restrict Isometry Property (RIP) with order k, $\mathbf{x}$ can be perfectly recovered. Gaussian random matrix [1], partial Fourier matrix [3], Bernoulli random matrix [2], and so on can be used as measurement matrix. Greedy pursuit algorithms [4–7], l_1 minimization algorithms [8–10], and intelligent optimal algorithms [11–13] are proposed to recover $\mathbf{x}$ from $\mathbf{y}$.

CS theory just exploits intrasignal correlation, which makes it not efficient in dealing with multiple signals. An expanded version of CS, Distributed Compressed Sensing (DCS) [14, 15], which can exploit not only intrasignal correlation but also intersignal correlation is proposed. With proper joint recovery algorithms, the measurement number needed in DCS can be further reduced. In this paper, we call the problem of jointly recovering signals as DCS problem. Several joint sparse models (JSM) and corresponding joint recovery algorithms are proposed to solve the DCS problem. One-Step Greedy Algorithm (OSGA) [15] is proposed to solve the DCS problem based on JSM-1. Greedy pursuit algorithms, including Simultaneous Orthogonal Matching Pursuit (SOMP) [15], Simultaneous Iterative Hard Thresholding (SIHT) [16], and Simultaneous Hard Thresholding Pursuit (SHTP) [16] are proposed to solve the DCS problem based on JSM-2. In [17, 18], two intelligent optimization algorithms based on particle swarm optimization and simulated annealing are proposed to solve the DCS problem based on JSM-2. However, as our analysis in Section 2.1, JSM-1 and JSM-2 are stringent on the description of signal correlation, which makes them reflect less intersignal and intrasignal correlations. In [19], JSM-3 is proposed. As a generalization of JSM-1 and JSM-2, it can reflect more intersignal and intrasignal correlations. This paper focuses on solving the DCS problem based on JSM-3. We notice that Joint Subspace Pursuit (Joint-SP) [20], Joint Orthogonal Matching Pursuit (Joint-OMP) [20], Sparsity Adaptive Matching Pursuit for DCS (DCS-SAMP) [21], and Forward-Backward Pursuit for DCS (DCS-FBP) [22] are proposed to solve the DCS problem based on JSM-3.

However, as greedy pursuit algorithms, they easily fall into suboptimal solutions.

GWO algorithm [23] is proposed by Mirjalili et al. in 2014, which simulates the hunting behavior and leadership hierarchy of grey wolves. As top apex predators, grey wolves normally live and hunt in a pack which includes 5–12 wolves. As an optimization algorithm, GWO has probability to accept a less optimal solution, which makes it avoid being stuck in local optimal solutions. Because of this, GWO draws much attention in solving some optimization problems and NP hard problems. Kumar et al. [24] apply GWO in system reliability optimization. Mirjalili [25], Hassanin et al. [26], and Sánchez [27] adopt GWO to train Artificial Neural Network (ANN). Muangkote et al. [28] propose an improved GWO algorithm and apply it in training q-Gaussian Radial Basis Functional-link nets (qRBFLNs) neural networks. Li et al. [29] propose a modified discrete GWO algorithm to solve the image segmentation problem. Emary et al. [30] propose a binary GWO algorithm and use it to select the optimal feature for the purpose of classification. In these areas, GWO performs better or comparable to other prevailing nature-inspired optimization algorithms [31–35].

In solving the DCS problem based on JSM-3, greedy pursuit algorithms easily fall into suboptimal solutions. From the above analysis of GWO, we know that it is a recently proposed global search optimization algorithm. Its performance is superior or comparable to other prevailing algorithms in solving some optimization problems. As illustrated in Section 3, the DCS problem based on JSM-3 can be modeled as an optimization problem. These reasons motivate us to exploit GWO to solve the DCS problem based on JSM-3.

In this paper, an intelligent grey wolf optimizer (GWO) [23] algorithm called DCS-GWO is proposed to solve the DCS problem based on JSM-3. DCS-GWO is essentially a GWO algorithm, the grey wolves' positions are initialized by using the q-thresholding algorithm [36] and updated by using the strategy of GWO. Inheriting the global search ability of GWO, DCS-GWO has better recovery performance than previous greedy pursuit algorithms at the expense of computational complexity.

The remainder of this paper is organized as follows. In Section 2, we introduce related background knowledge, including DCS model, joint sparse models (JSM), grey wolf optimizer (GWO) algorithm, and q-thresholding algorithm. In Section 3, we introduce the DCS-GWO algorithm. In Section 4, we provide the simulation results. Conclusions are stated in Section 5.

We use the following notations in this paper. Lowercase bold-face denotes a vector. Uppercase bold-face denotes a matrix. For the vector $\mathbf{x} \in R^n$, $\|\mathbf{x}\|_q$ $(q > 1)$ denotes the l_q norm of $\mathbf{x}$. If $\|\mathbf{x}\|_0 \leq k < n$, $\mathbf{x}$ is called sparse signal and the sparsity is k. $T \triangleq \{1 \leq i \leq n \mid \mathbf{x}(i) \neq 0\}$ denotes the support set of $\mathbf{x}$. For the matrix $\mathbf{\Phi} \in R^{m \times n}$, $\mathbf{\Phi}_{:,j}$ denotes the jth column of $\mathbf{\Phi}$ and $\mathbf{\Phi}_{i,:}$ denotes the ith row of $\mathbf{\Phi}$. For the set $S = \{1, 2, \ldots, n\}$, $|S|$ denotes the cardinality of S. $G \subseteq S$ denotes that G is a subset of S. $\mathbf{\Phi}_{:,G}$ denotes the matrix composed of the columns $\{\mathbf{\Phi}_{:,j}\}_{j \in G}$. $\mathbf{\Phi}_{G,:}$ denotes the matrix composed of the rows $\{\mathbf{\Phi}_{i,:}\}_{i \in G}$. $\mathbf{\Phi}^T$ denote the transpose matrix of the matrix $\mathbf{\Phi}$. $\mathbf{\Phi}^\dagger = (\mathbf{\Phi}^T\mathbf{\Phi})^{-1}\mathbf{\Phi}^T$ denotes the pseudo-inverse matrix of $\mathbf{\Phi}$.

2. Background Knowledge

In this part, we introduce related background knowledge of this paper. The DCS model is introduced in Section 2.1. Joint sparse models are introduced in Section 2.2. Grey wolf optimizer is introduced in Section 2.3. At last, the q-thresholding algorithm is introduced in Section 2.4.

2.1. DCS Model. Suppose that there are J signals $\mathbf{x}_j \in R^n$ which are sparse and have the sparsity $k^j < n$, where $j \in \{1, 2, \ldots, J\}$. They are individually measured by the measurement matrix $\mathbf{\Phi} \in R^{m \times n}$ as [14, 21]. That is,

$$\mathbf{y}_j = \mathbf{\Phi} \times \mathbf{x}_j, \quad j \in \{1, 2, \ldots, J\}. \tag{1}$$

Without the consideration of noise, the measurement process can be denoted as

$$\mathbf{Y} = \mathbf{\Phi}\mathbf{X}, \tag{2}$$

where $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_J] \in R^{n \times J}$ denotes the joint signal and $\mathbf{Y} = [\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_J] \in R^{m \times J}$ denotes the measurement signal. The DCS problem is to recover $\mathbf{X}$ from $\mathbf{Y}$ jointly by exploiting intersignal and intrasignal correlations. For the matrix $\mathbf{X} \in R^{n \times J}$, the set of indices corresponding to nonzero rows of $\mathbf{X}$ is the joint support set of $\mathbf{X}$, which can be denoted as $R(\mathbf{X}) \triangleq \{1 \leq i \leq n \mid \mathbf{X}_{i,:} \neq \mathbf{0}\}$. If there are no more than K nonzero rows in $\mathbf{X}$, $\mathbf{X}$ is called jointly sparse and the joint sparsity is K.

2.2. Joint Sparse Models. The JSM reflects the intersignal correlations and intrasignal correlations. There are mainly three JSMs.

(1) JSM-1. JSM-1 is called sparse common support and innovation model [14, 15]. JSM-1 can be written as

$$\mathbf{x}_j = \mathbf{z} + \mathbf{z}_j, \tag{3}$$

where $j \in \{1, 2, \ldots, J\}$. $\mathbf{z}$ is the common component shared by all signals. $\mathbf{z}_j$ is the innovation component for each signal. The sparsity of $\mathbf{z}$ is $K_\mathbf{z}$. The sparsity of $\mathbf{z}_j$ is K_j.

(2) JSM-2. JSM-2 is called sparse common support model [14, 15]. JSM-2 can be written as

$$\mathbf{x}_j = \mathbf{z}_j, \tag{4}$$

where $j \in \{1, 2, \ldots, J\}$. For each signal, the support set of $\mathbf{z}_j$ is the same, but the coefficients are individual. The sparsity of $\mathbf{z}_j$ is K_j.

(3) JSM-3. JSM-3 is called mixed support model [19, 20]. JSM-3 has common component $\mathbf{c}_j$ and innovation component $\mathbf{z}_j$. For each signal, the support set of $\mathbf{c}_j$ is the same, but the nonzero coefficients are individual. For each signal, the

innovation component is completely independent, not only the coefficients but also the support set. JSM-3 can be written as

$$\mathbf{x}_j = \mathbf{c}_j + \mathbf{z}_j, \tag{5}$$

where $j \in \{1, 2, \ldots, J\}$. The sparsity of $\mathbf{c}_j$ is K_c. The sparsity of $\mathbf{z}_j$ is K_j. Obviously, JSM-3 is a generalization of JSM-1 and JSM-2. If $\mathbf{z}_j = \mathbf{0}$, JSM-3 reduces to JSM-2. If both the coefficients and support set of $\mathbf{c}_j$ are the same for each signal, JSM-3 reduces to JSM-1. JSM-3 is less stringent in describing signal correlations, so, it can reflect more signal correlations. Our algorithm is proposed to solve the DCS problem based on JSM-3.

2.3. Grey Wolf Optimizer. Grey wolf optimizer (GWO) [31] algorithm is a recently proposed intelligent optimization algorithm. As apex predators, grey wolves own special leadership hierarchy and hunting mechanism. Grey wolves are divided into four categories, alpha (α), beta (β), delta (δ), and omega (ω). α is the leader which makes decision to hunt, rest, forward, or stop. β assists α make decision and reinforces α's commands. δ executes the decision and manages ω wolves which are the lowest ranking of grey wolves.

Before hunting, the grey wolves firstly encircle the prey. The distance between the wolf and prey is computed by using (6). The wolf's position is updated by using (7).

$$\mathbf{d} = \left|\mathbf{c} \cdot \mathbf{h}_p(t) - \mathbf{h}(t)\right| \tag{6}$$

$$\mathbf{h}(t+1) = \mathbf{h}_p(t) - \mathbf{a}\mathbf{d}, \tag{7}$$

where t denotes the current iteration, $\mathbf{h}_p$ denotes the prey's position vector, $\mathbf{a}$ and $\mathbf{c}$ are two coefficient vectors, and $\mathbf{h}$ denotes a grey wolf's position vector. The coefficient vectors $\mathbf{a}$ and $\mathbf{c}$ are determined as follows:

$$\begin{aligned} \mathbf{a} &= 2\mathbf{v}\boldsymbol{\gamma}_2 - \mathbf{v} \\ \mathbf{c} &= 2\boldsymbol{\gamma}_1, \end{aligned} \tag{8}$$

where $\boldsymbol{\gamma}_1$, $\boldsymbol{\gamma}_2$ are random vectors between $\mathbf{0}$ and $\mathbf{1}$ and the vector $\mathbf{v}$ decreases from $\mathbf{2}$ to $\mathbf{0}$ linearly in the iteration course.

After the process of encircling the prey, the hunting is led by α, β, and δ. All wolves' positions are updated according to the positions of α, β, and δ. Firstly, the distances between a wolf and the best three wolves are computed by using (9). Then, the position of the wolf is updated by using (10) and (11).

$$\begin{aligned} \mathbf{d}_\alpha &= \left|\mathbf{c}_1\mathbf{h}_\alpha(t) - \mathbf{h}(t)\right|, \\ \mathbf{d}_\beta &= \left|\mathbf{c}_2\mathbf{h}_\beta(t) - \mathbf{h}(t)\right|, \\ \mathbf{d}_\delta &= \left|\mathbf{c}_3\mathbf{h}_\delta(t) - \mathbf{h}(t)\right| \end{aligned} \tag{9}$$

$$\begin{aligned} \mathbf{h}_1 &= \mathbf{h}_\alpha - \mathbf{a}_1\mathbf{d}_\alpha, \\ \mathbf{h}_2 &= \mathbf{h}_\beta - \mathbf{a}_2\mathbf{d}_\beta, \\ \mathbf{h}_3 &= \mathbf{h}_\delta - \mathbf{a}_3\mathbf{d}_\delta \end{aligned} \tag{10}$$

$$\mathbf{h}_p(t+1) = \frac{\mathbf{h}_1 + \mathbf{h}_2 + \mathbf{h}_3}{3} \tag{11}$$

After all wolves' positions are updated, the process of hunting the prey goes to the next iteration in which the new best three solutions are generated. The iteration repeats until the stopping criterion is satisfied.

2.4. q-Thresholding Algorithm. q-thresholding is a joint recovery algorithm proposed in [36]. If the joint sparsity level K is known, we can estimate the joint support set of joint signal by using the following:

$$\begin{aligned} &\widehat{I}\{\boldsymbol{\Phi}, \mathbf{Y}, K, q\} \\ &= \left\{\text{indices of the } K \text{ largest values in } \left\{\left\|\boldsymbol{\Phi}_{:,j}^T\mathbf{Y}\right\|_q\right\}_{j=1}^n\right\} \end{aligned} \tag{12}$$

3. DCS-GWO: Grey Wolf Optimizer Algorithm for Distributed Compressed Sensing

In this part, we firstly introduce the DCS-GWO in Section 3.1. Next, we analyze DCS-GWO's computational complexity in Section 3.2.

3.1. DCS-GWO. DCS-GWO is essentially a GWO algorithm. It has four basic elements: cost function, initial positions, generating mechanism, and stopping criterions. In this part, we firstly introduce DCS-GWO's four basic elements and then summarize it in Algorithm 1.

(1) Cost Function. Similar to DC-SAMP and DCS-FBP, we can use a two-step strategy to solve the DCS problem. Firstly, the joint support set I of $\mathbf{X}$ is estimated. Then, the joint signal can be estimated by using the least square method as follows:

$$\begin{aligned} \operatorname{rec}\mathbf{X}_{\widehat{I},:} &= \boldsymbol{\Phi}_{:,\widehat{I}}^{\dagger}\mathbf{Y}, \\ \operatorname{rec}\mathbf{X}_{S-\widehat{I},:} &= \mathbf{0}, \end{aligned} \tag{13}$$

where $\widehat{I}$ denotes the estimated joint support set and $S \triangleq \{1, 2, \ldots, n\}$.

We can find that if the joint support set is estimated accurately, it must satisfy $\|\boldsymbol{\Phi}_{:,\widehat{I}}\boldsymbol{\Phi}_{:,\widehat{I}}^{\dagger}\mathbf{Y} - \mathbf{Y}\|_F = 0$. As $\|\boldsymbol{\Phi}_{:,\widehat{I}}\boldsymbol{\Phi}_{:,\widehat{I}}^{\dagger}\mathbf{Y} - \mathbf{Y}\|_F \geq 0$, we define the cost function as

$$f(I) = \left\|\boldsymbol{\Phi}_{:,\widehat{I}}\boldsymbol{\Phi}_{:,\widehat{I}}^{\dagger}\mathbf{Y} - \mathbf{Y}\right\|_F. \tag{14}$$

We can estimate the joint support set by solving the following:

$$\min_{I \in \Theta} \quad f(I), \tag{15}$$

where Θ is the set consisting of all the K-cardinality subsets of S.

(2) Initial Positions. We assume that the wolf number set $L_{\text{wolf}} = \{1, 2, \ldots, L\}$. We use the q-thresholding algorithm to initialize the grey wolves' positions. I_l^0 denotes the initial position of the lth wolf where $l \in L_{\text{wolf}}$. It is estimated by using

$$I_l^0 \triangleq \widehat{I}\{\boldsymbol{\Phi}, \mathbf{Y}, K, q_l\}, \quad l \in L_{\text{wolf}}, \tag{16}$$

where q_l is a random number in the closed interval $[1, 2]$.

Input: The joint sparsity K; the wolf number set $L_{\text{wolf}} = \{1, 2, \ldots, L\}$; the limiting parameter C_n; the stopping criterion ε.
Initialization: Initialize the lth wolf's position I_l^0 by using the Eq. (16); Initialize the best three positions by
$I_{\text{best1}}^0 = \text{argmin}_{\{I_l^0\}_{l\in L_{\text{wolf}}}} f(I_l^0)$, $I_{\text{best2}}^0 = \text{argmin}_{\{I_l^0\}_{l\in L_{\text{wolf}}-\text{bes1}}} f(I_l^0)$, $I_{\text{best3}}^0 = \text{argmin}_{\{I_l^0\}_{l\in L_{\text{wolf}}-\text{bes1}-\text{best2}}} f(I_l^0)$;
the iteration number $t = 1$; the allowed maximum iterations number $N_{\max}$.
Judgement: if $f(I_{\text{best1}}^0) < \varepsilon$, set $\widehat{I} = I_{\text{best1}}^0$, output the joint signal by using the Eq. (13) and stop. Otherwise, go to the iteration.
Iteration:
Step 1. Update all wolves' positions ($l = 1 : L$):
Step 1.1. Define $U_1 = (I_{\text{best1}}^{t-1} \cap I_{\text{best2}}^{t-1} \cap I_{\text{best3}}^{t-1}) \cup I_l^{t-1}$.
Step 1.2. If $|U_1| > C_n$, randomly choose $C_n - (I_{\text{best1}}^{t-1} \cap I_{\text{best2}}^{t-1} \cap I_{\text{best3}}^{t-1})$ elements from I_l^{t-1} to forma set U_2 and define $U = (I_{\text{best1}}^{t-1} \cap I_{\text{best2}}^{t-1} \cap I_{\text{best3}}^{t-1}) \cup U_2$. If $|U_1| < C_n$, randomly choose $C_n - |U_1|$ elements from $S - U_1$ to form a set U_3 and define $U = U_1 \cup U_3$.
Step 1.3. Use the least square method to estimate a temporary solution $\text{rec}\,\mathbf{X}_l^t$ by using the Eq. (17).
Step 1.4. Update the lth wolf's position by using the Eq. (18).
Step 2. Update the best three wolves' positions: $I_{\text{best1}}^t = \text{argmin}_{\{I_l^t\}_{l\in L_{\text{wolf}}}} f(I_l^t)$, $I_{\text{best2}}^t = \text{argmin}_{\{I_l^t\}_{l\in L_{\text{wolf}}-\text{best1}}} f(I_l^t)$,
$I_{\text{best3}}^t = \text{argmin}_{\{I_l^t\}_{l\in L_{\text{wolf}}-\text{best1}-\text{best2}}} f(I_l^t)$.
Step 3. Check the terminate criterion: If $f(I_{\text{best1}}^t) < \varepsilon$ or $t > N_{\max}$, set the final joint support set $\widehat{I} = I_{\text{best1}}^t$ and terminate the iteration. Otherwise, set $t = t + 1$ and go to the next iteration.
Output: Estimate the joint signal by using the Eq. (13).

Algorithm 1: DCS-GWO.

(3) Update Strategy. The update strategy of DCS-GWO inherits from GWO. In DCS-GWO, t denotes the current iteration. For $l \in L_{\text{wolf}}$, the lth wolf's position is updated according to the previous best three wolves' positions I_{best1}^{t-1}, I_{best2}^{t-1}, and I_{best3}^{t-1} and its previous position I_l^{t-1}. A parameter C_n is used to limit the set size where $K < C_n < \text{spark}(\mathbf{\Phi})$.

The position of the lth wolf is updated as follows. In Step 1.1, a set U_1 is formed according to the previous best three positions and the lth wolf's previous position, $U_1 = (I_{\text{best1}}^{t-1} \cap I_{\text{best2}}^{t-1} \cap I_{\text{best3}}^{t-1}) \cup I_l^{t-1}$. In Step 1.2, if $|U_1| > C_n$, randomly choose $C_n - (I_{\text{best1}}^{t-1} \cap I_{\text{best2}}^{t-1} \cap I_{\text{best3}}^{t-1})$ elements from I_l^{t-1} to form the set U_2 and define the temporary position $U = (I_{\text{best1}}^{t-1} \cap I_{\text{best2}}^{t-1} \cap I_{\text{best3}}^{t-1}) \cup U_2$; if $|U_1| < C_n$, randomly choose $C_n - |U_1|$ elements from $S - U_1$ to form a set U_3 and define the temporary position $U = U_1 \cup U_3$. In Step 1.3, the temporary solution $\text{rec}\,\mathbf{X}_l^t$ is estimated by using the least square method as

$$\begin{aligned} \left(\text{rec}\,\mathbf{X}_l^t\right)_{U,:} &= \mathbf{\Phi}_{:,U}^{\dagger}\mathbf{Y}, \\ \left(\text{rec}\,\mathbf{X}_l^t\right)_{S-U,:} &= \mathbf{0}. \end{aligned} \tag{17}$$

Lastly, the position of the lth wolf is updated by

$$I_l^t \triangleq \left\{\text{indices of the } K \text{ largest values in } \left\{\left\|\left(\text{rec}\,\mathbf{X}_l^t\right)_{i,:}\right\|_q\right\}_{i=1}^{n}\right\}, \tag{18}$$

where $l \in \{1, 2, \ldots, L\}$. After all wolves' positions are updated, we can update the best three wolves' positions according to the cost function.

(4) Stopping Criterion. In order to avoid too many iterations, we set a maximum allowed iteration number $N_{\max}$ and a small positive number ε as stopping criterions. If $f(I_{\text{best1}}^t) < \varepsilon$ or the number of iterations reaches $N_{\max}$, the iteration process is terminated.

We summarize the DCS-GWO in Algorithm 1.

3.2. Computational Complexity Analysis of DCS-GWO. According to Algorithm 1, the initialization and iteration contribute the main computational complexity of DCS-GWO. The computational complexity of initialization is $O(LJmn)$. In each iteration, the main computational complexity lies in Steps 1.3 and 1.4 which, respectively, have upper limit value $O(m^3) + O(Jm^2)$ and $O(mK^2) + O(JmK)$. Because $K \leq m$, the computational complexity upper limit value of each iteration is $O(m^3) + O(Jm^2)$. Because the total number of iterations is not more than $LN_{\max}$, the computational complexity upper limit value of DCS-GWO is $O(N_{\max}Lm^3) + O(N_{\max}LJm^2) + O(LJmn)$. Obviously, as an intelligent optimizer algorithm, DCS-GWO has higher computational complexity than greedy pursuit algorithms. However, as swarm intelligence algorithm, it can run in parallel to reduce the running time.

4. Simulation Results and Analysis

4.1. Experiment Configuration. In this section, the performance of DCS-GWO is compared with other algorithms that can solve the DCS problem based on JSM-3, including Joint OMP [20], Joint SP [20], DCS-SAMP [21], and DCS-FBP [22]. The algorithms proposed in [15–18] are designed for JSM-1 or JSM-2, they are not discussed in this part. The parameters of DCS-GWO are set as $N_{\max} = \max(10 * K, 500)$, $L = 8$, $C_n = 0.8\,\text{m}$, and $\varepsilon = 1e - 5$. We use the following hypothesis in the simulation.

We use the Gaussian random matrix as measurement matrix $\mathbf{\Phi}$, the elements of which are randomly drawn from the standard i.i.d. and every column of which is normalized

to unit l_2 norm. All signals follow the JSM-3 with the common sparsity K_c and innovation sparsity K_j. We assume that the innovation sparsity K_j is the same for all signals. For each signal, the support sets of common component and innovation component are random subsets of the set $S = \{1, 2, \ldots, n\}$. The nonzero coefficients of the common component and innovation component are randomly drawn from the standard i.i.d. In each experiment, 200 independent trials are conducted. In each trial, the signals and measurement matrix $\mathbf{\Phi}$ are generated independently. Average Normalized Mean Squared Error (ANMSE), perfect recovery percentage, and average runtime are used to evaluate the algorithms. The ANMSE is defined as

$$\text{ANMSE} = \frac{1}{200}\sum_{i=1}^{200}\left[10 \times \log\left(\frac{\left\|\mathbf{X}^{(i)} - \widehat{\mathbf{X}}^{(i)}\right\|_2^2}{\left\|\mathbf{X}^{(i)}\right\|_2^2}\right)\right]. \quad (19)$$

The perfect recovery condition is $\|\mathbf{X}^{(i)} - \widehat{\mathbf{X}}^{(i)}\|_2 < 10^{-2}\|\mathbf{X}^{(i)}\|_2$, where $\mathbf{X}^{(i)}$ and $\widehat{\mathbf{X}}^{(i)}$, respectively, denote the original joint signal and the recovered joint signal in the ith trial. If N_S trials are success, the perfect recovery percentage is $N_S/200$. All the experiments are implemented by using Matlab R2014a on the computer with 2.5 GHz Intel Core I3 processor and 4.0 GB memory running window 7 system.

4.2. Experiment Results

(1) Requirement for Measurements. In the first simulation, we compare the performance of all algorithms against the measurement number M changing from 50 to 90 with Step 10. Other parameters are fixed as $N = 256$, $K_c = 15$, $K_j = 5$, and $J = 3$.

As Figure 1(a) shows that the perfect recovery percentage of DCS-GWO is always higher than other algorithms. Besides, DCS-GWO needs less measurement number to perfectly recover signals. When the measurement number reaches 70, DCS-GWO recovers signals perfectly. Other algorithms need the measurement number 90 to perfectly recover signals. As Figure 1(b) shows, DCS-GWO has lower ANMSE than other algorithms.

From Figure 1(c), at the expense of global search ability, DCS-GWO needs more running time than other algorithms. However, as a swarm intelligence algorithm, it can run in parallel to reduce the running time.

(2) Robustness against Common Sparsity. In the second simulation, we evaluate the performance of the algorithms against the common sparsity K_c changing from 1 to 15 with the step size 2. Other parameters are fixed as $N = 256$, $M = 40$, $J = 3$, and $K_j = 3$.

As Figure 2(a) shows, DCS-GWO performs far better than other algorithms in the perfect recovery percentage. For all algorithms, as the common signal sparsity K_c increases, the perfect recovery percentage decreases. We are more interested in at which sparsity the perfect recovery percentages drop below 1. The perfect recovery percentage of DCS-GWO starts to fall below 1 when $K_c > 7$; however, other algorithms already fall below 1 when $K_c > 3$. From Figure 2(b), DCS-GWO has lower ANMSE than other algorithms. As for average runtime, we can get the similarly results as Figure 1(c).

(3) Robustness against Innovation Sparsity. In the third simulation, we compare the performance of DCS-GWO with other algorithms against the innovation sparsity K_j changing from 0 to 7 with Step 1. Other parameters are set as $N = 256$, $M = 40$, $J = 3$, and $K_c = 5$.

As Figure 3(a) shows, the DCS-GWO performs extremely better than other algorithms in perfect recovery percentage. As the increase of the innovation sparsity K_j, the perfect recovery percentage is declining, that is, because the increase of joint sparsity level influences the performance of all algorithms. The perfect recovery percentages of other algorithms start to fall below 1 when $K_j > 2$; our algorithm starts to fall below 1 until $K_j > 5$. As in Figure 3(b), DCS-GWO has lower ANMSE than other algorithms. As for average runtime, we can get the similar results as Figure 1(c).

(4) Robustness against the Number of Signals. We compare our algorithm with other algorithms against the number of signals J changing from 2 to 6 with Step 1. Other parameters are set as $N = 256$, $M = 40$, $K_c = 4$, and $K_j = 4$.

The perfect recovery percentages of Joint SP and Joint OMP are not influenced obviously by the increase of signals number, because both of them recover the signals one by one rather than jointly. They have better performance than our algorithms when $J > 5$; however, our algorithm performs better than them when $J \leq 5$.

As the number of signals J increases, the perfect recovery percentages of DCSFBP, DCS-GWO, and DCSSAMP decrease. As Figure 4(a) shows, our algorithm has higher perfect recovery percentages than DCSFBP and DCSSAMP. When $J > 2$, the perfect recovery percentages of DCS-FBP and DCSSAMP already fall below 1; however, our algorithm starts to fall below 1 until $J > 5$. As Figure 4(b), DCS-GWO has lower ANMSE than other algorithms. As for average runtime, we can get the similarly results as Figure 1(c).

From the above simulations, we can see that DCS-GWO has higher perfect recovery percentage and lower ANMSE than greedy pursuit algorithms. Next, we analyze the reason of DCS-GWO's better performance according to its structure.

The main reason for DCS-GWO's better performance is its update mechanism. In Step 1.1, the common information of the best three grey wolves' positions is utilized to update all the grey wolves' positions. By this way, the previous obtained best information is preserved in the new grey wolves' positions. In Step 1.2, random perturbations are introduced into the grey wolves' positions. Benefitting from this, DCS-GWO can skip the local optimum position and guide the search in promising directions. In Steps 1.3 and 1.4, $C_n - K$ indices corresponding to the rows of the temporal joint signal which have smallest row norm values are removed from the temporal support set. By this way, the previous wrong selected indices can be removed from the temporal support set.

In contrast to DCS-GWO, greedy pursuit algorithms search the support set according to the gradient of (15), which is a local optimal search mechanism. Therefore, due to the

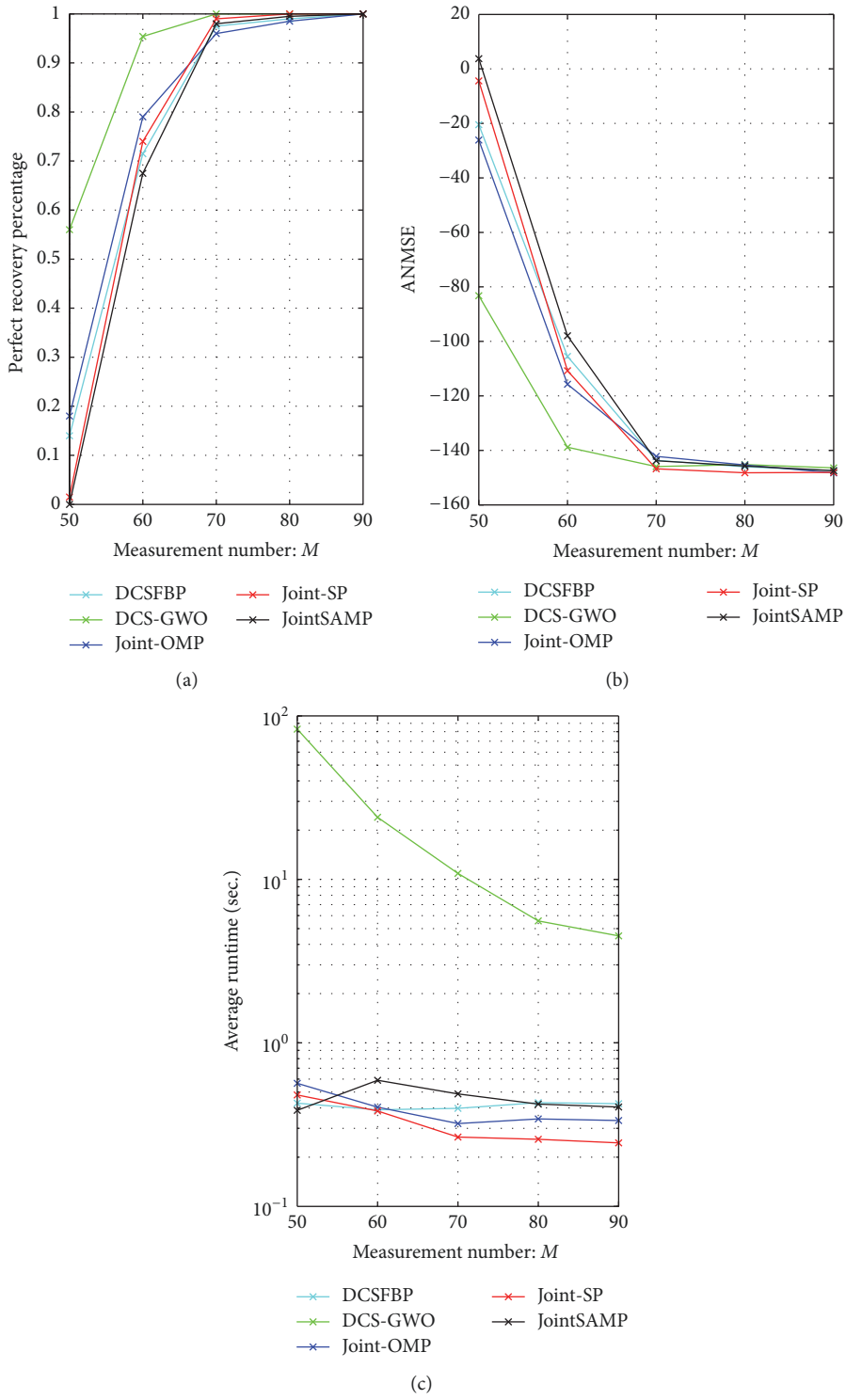

FIGURE 1: Recovery performance of different algorithms against M with $N = 256$, $K_c = 15$, $K_j = 5$, and $J = 3$. (a) Perfect recovery percentage. (b) ANMSE. (c) Average runtime.

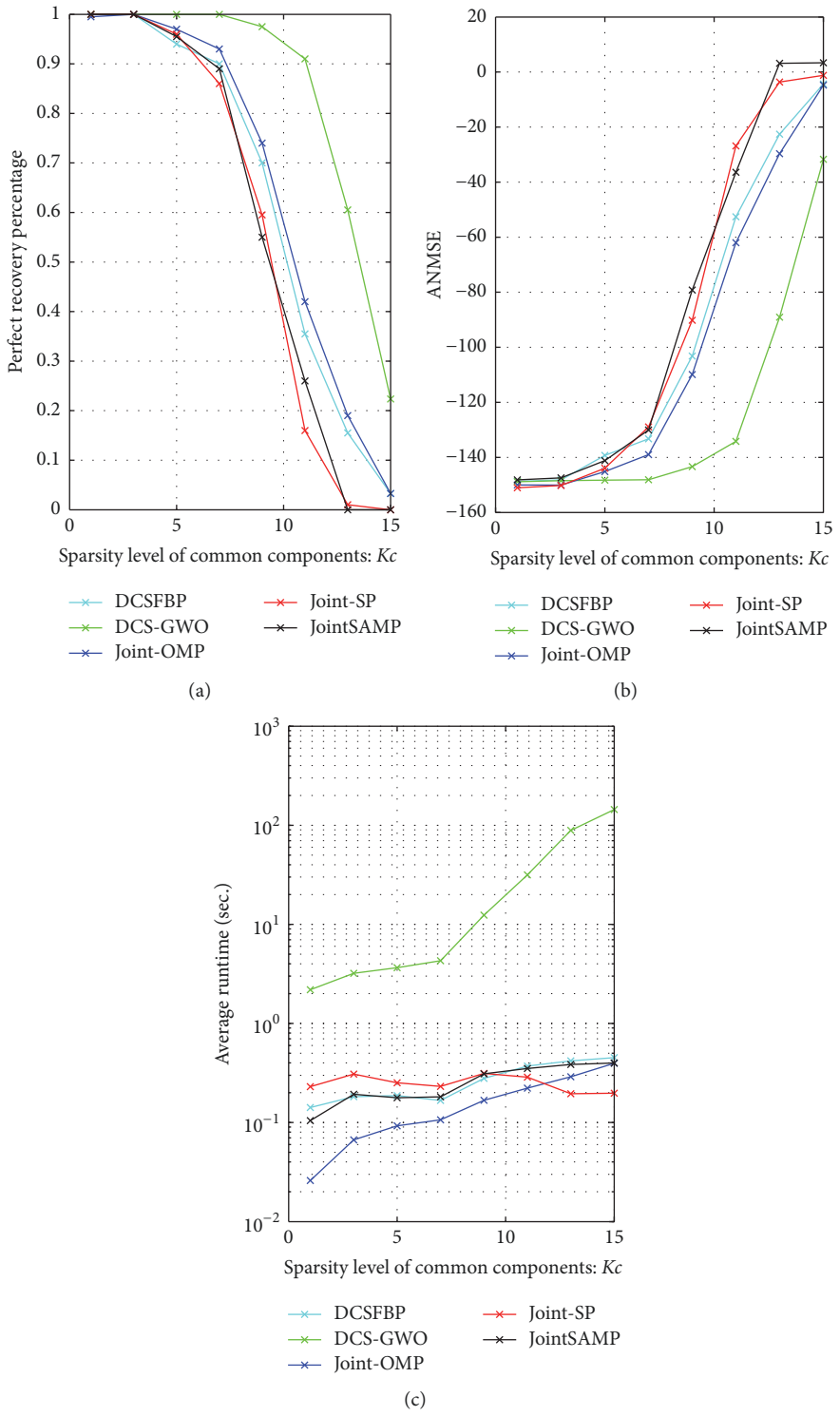

FIGURE 2: Recovery performance of different algorithms against K_c with $N = 256$, $M = 40$, $K_j = 3$, and $J = 3$. (a) Perfect recovery percentage. (b) ANMSE. (c) Average Runtime.

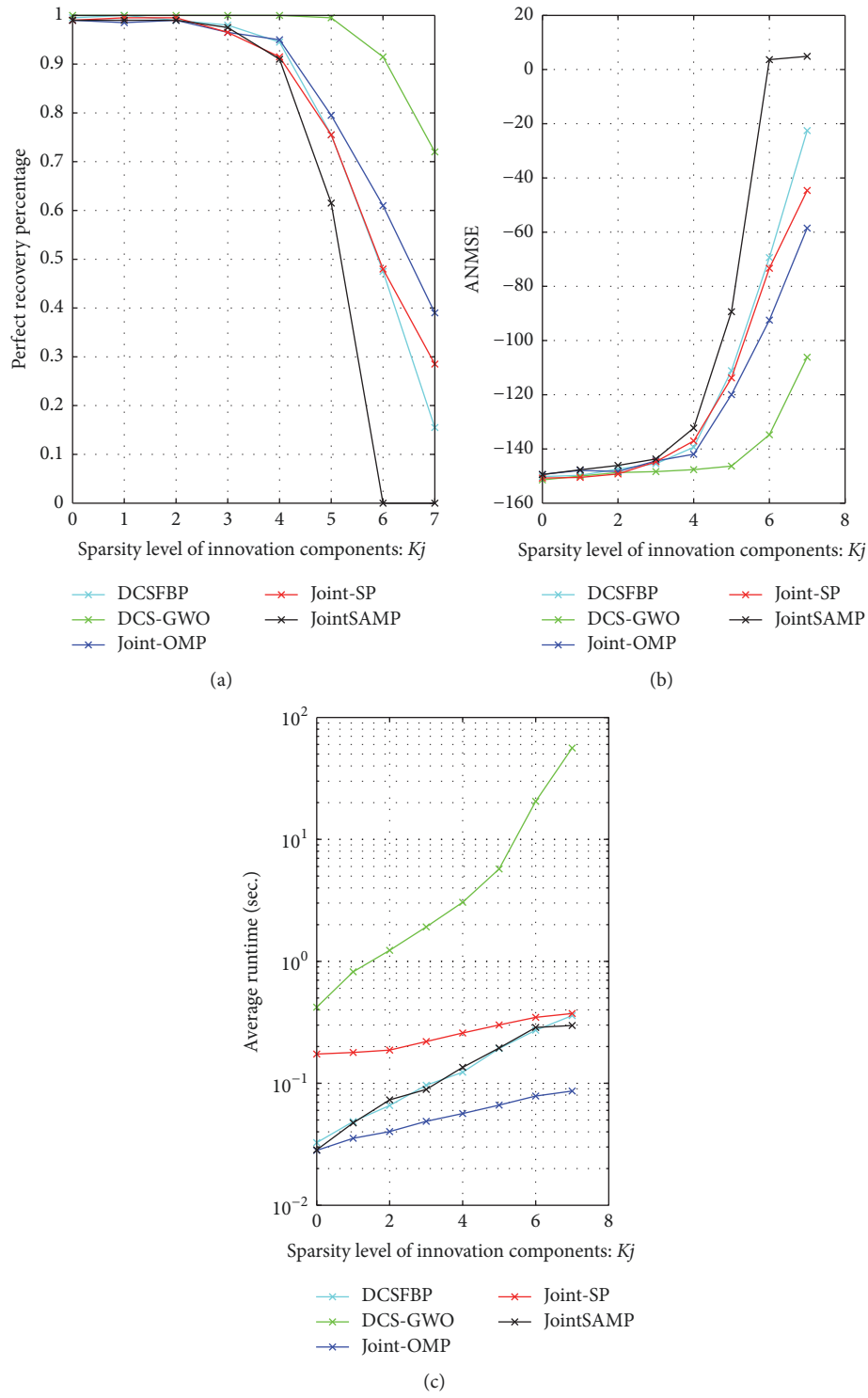

FIGURE 3: Recovery performance of different algorithms against K_j with $N = 256$, $M = 40$, $K_c = 5$, and $J = 3$. (a) Perfect recovery percentage. (b) ANMSE. (c) Average runtime.

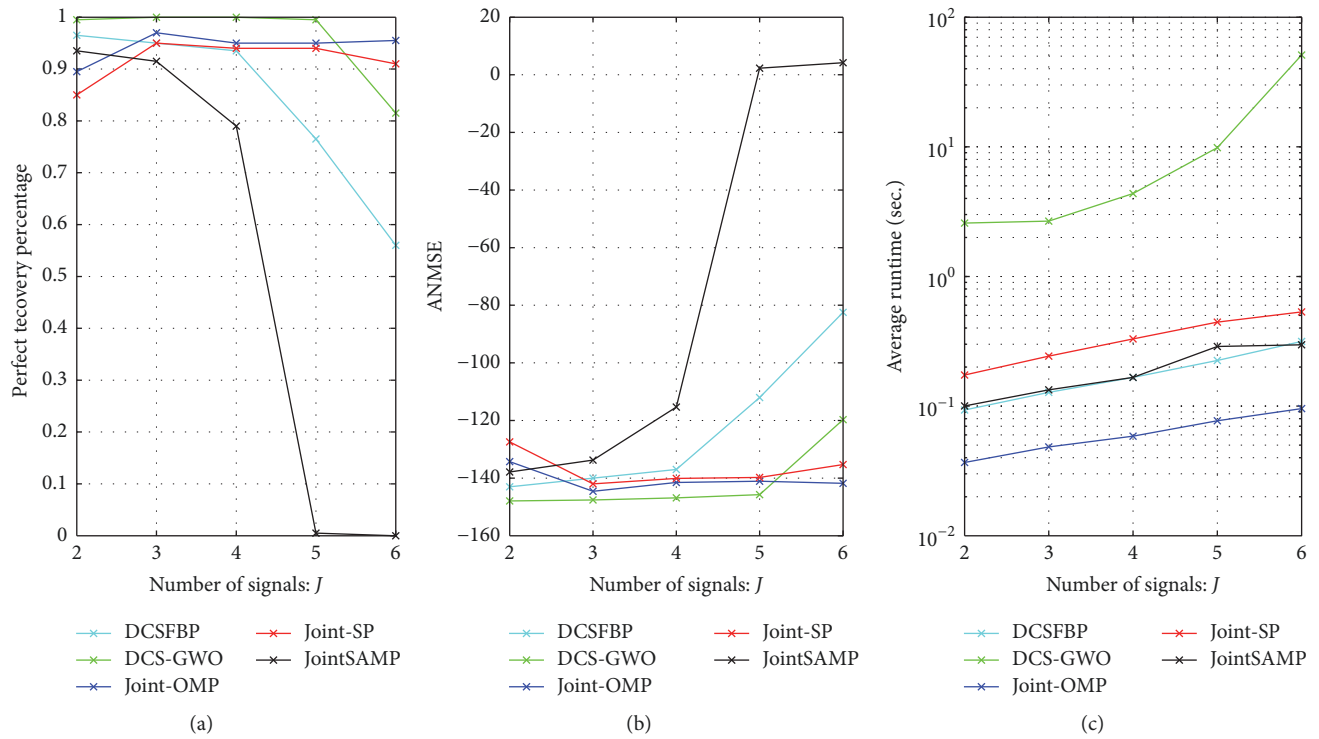

FIGURE 4: Recovery performance of different algorithms against J with $N = 256$, $M = 40$, $K_c = 4$, and $K_j = 4$. (a) Perfect recovery percentage. (b) ANMSE. (c) Average runtime.

efficiency of DCS-GWO's update mechanism, DCS-GWO can find the support set more accurately than greedy pursuit algorithms. Then, DCS-GWO can recover the signals more accurately than greedy pursuit algorithms.

5. Conclusion

In this paper, an intelligent grey wolf optimizer algorithm is proposed to solve the DCS problem based on JSM-3. The positions of grey wolves are initialized by using the q-thresholding algorithm and updated by using the idea of GWO. Inheriting the global search ability of GWO, DCS-GWO overcomes greedy pursuit algorithms' shortcoming of easily falling into suboptimal solutions. The simulation results illustrate that DCS-GWO has higher perfect recovery percentage and lower ANMSE than other algorithms. DCS-GWO has higher computational complexity than other algorithms. However, as a swarm intelligence algorithm, it can compute in parallel to reducing the running time. In the future work, we will focus on developing effective update strategies to reduce the running time. Moreover, there are many more recent nature-inspired algorithms, such as Ant Lion Optimizer (ALO) [37], Moth-Flame Optimization (MFO) algorithm [38], Whale Optimization Algorithm (WOA) [39], and Multiverse Optimizer (MVO) [40]. We will exploit them to solve the DCS problem based on JSM-3.

Conflicts of Interest

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

Acknowledgments

This work is financially supported by National Science Foundation of China (no. 51574232).

References

[1] D. L. Donoho, "Compressed sensing," *Institute of Electrical and Electronics Engineers Transactions on Information Theory*, vol. 52, no. 4, pp. 1289–1306, 2006.

[2] E. J. Candes, J. Romberg, and T. Tao, "Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information," *Institute of Electrical and Electronics Engineers 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?" *Institute of Electrical and Electronics Engineers Transactions on Information Theory*, vol. 52, no. 12, pp. 5406–5425, 2006.

[4] N. B. Karahanoglu and H. Erdogan, "Compressed sensing signal recovery via forward-backward pursuit," *Digital Signal Processing*, vol. 23, no. 5, pp. 1539–1548, 2013.

[5] J. A. Tropp and A. C. Gilbert, "Signal recovery from random measurements via orthogonal matching pursuit," *Institute of*

Electrical and Electronics Engineers Transactions on Information Theory, vol. 53, no. 12, pp. 4655–4666, 2007.

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

[7] T. T. Do, L. Gan, N. Nguyen, and T. D. Tran, "Sparsity adaptive matching pursuit algorithm for practical compressed sensing," in *Proceedings of the 42nd Asilomar Conference on Signals, Systems and Computers (ASILOMAR '08)*, pp. 581–587, Pacific Grove, Calif, USA, October 2008.

[8] E. J. Candes and T. Tao, "Decoding by linear programming," *Institute of Electrical and Electronics Engineers Transactions on Information Theory*, vol. 51, no. 12, pp. 4203–4215, 2005.

[9] S. S. Chen, D. L. Donoho, and M. A. Saunders, "Atomic decomposition by basis pursuit," *SIAM Review*, vol. 43, no. 1, pp. 129–159, 2001.

[10] D. L. Donoho and M. Elad, "Optimally sparse representation in general (nonorthogonal) dictionaries via l^1 minimization," *Proceedings of the National Acadamy of Sciences of the United States of America*, vol. 100, no. 5, pp. 2197–2202, 2003.

[11] X. Du, L. Cheng, and D. Chen, "A simulated annealing algorithm for sparse recovery by l0 minimization," *Neurocomputing*, vol. 131, pp. 98–104, 2014.

[12] X. Fengmin and W. Shanhe, "A hybrid simulated annealing thresholding algorithm for compressed sensing," *Signal Processing*, vol. 93, no. 6, pp. 1577–1585, 2013.

[13] D. Li, Q. Wang, and Y. Shen, "Intelligent greedy pursuit model for sparse reconstruction based on l0 minimization," *Signal Processing*, vol. 122, pp. 138–151, 2016.

[14] M. F. Duarte, S. Sarvotham, M. B. Wakin, D. Baron, and R. G. Baraniuk, "Joint sparsity models for distributed compressed sensing," *Preprint*, vol. 22, pp. 2729–2732, 2008.

[15] M. F. Duarte, S. Sarvotham, D. Baron, M. B. Wakin, and R. G. Baraniuk, "Distributed compressed sensing of jointly sparse signals," in *Proceedings of the 39th Asilomar Conference on Signals, Systems and Computers*, pp. 1537–1541, 2005.

[16] J. D. Blanchard, M. Cermak, D. Hanle, and Y. Jing, "Greedy algorithms for joint sparse recovery," *IEEE Transactions on Signal Processing*, vol. 62, no. 7, pp. 1694–1704, 2014.

[17] 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.

[18] X. Du, L. Cheng, and G. Cheng, "A heuristic search algorithm for the multiple measurement vectors problem," *Signal Processing*, vol. 100, pp. 1–8, 2014.

[19] D. Sundman, S. Chatterjee, and M. Skoglund, "Distributed greedy pursuit algorithms," *Signal Processing*, vol. 105, pp. 298–315, 2014.

[20] D. Sundman, S. Chatterjee, and M. Skoglund, "Greedy pursuits for compressed sensing of jointly sparse signals," in *Proceedings of the 19th European Signal Processing Conference, EUSIPCO 2011*, vol. 2, pp. 368–372, Barcelona, Spain, September 2011.

[21] Q. Wang and Z. Liu, "A robust and efficient algorithm for distributed compressed sensing," *Computers Electrical Engineering*, vol. 37, no. 6, pp. 916–926, November 2011.

[22] Y. Zhang, R. Qi, and Y. Zeng, "Forward-backward pursuit method for distributed compressed sensing," *Multimedia Tools and Applications*, vol. 76, no. 20, pp. 20587–20608, 2017.

[23] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey wolf optimizer," *Advances in Engineering Software*, vol. 69, pp. 46–61, 2014.

[24] A. Kumar, S. Pant, and M. Ram, "System Reliability Optimization Using Gray Wolf Optimizer Algorithm," *Quality and Reliability Engineering International*, 2016.

[25] S. Mirjalili, "How effective is the Grey Wolf optimizer in training multi-layer perceptrons," *Applied Intelligence*, vol. 43, no. 1, pp. 150–161, 2015.

[26] M. F. Hassanin, A. M. Shoeb, and A. E. Hassanien, "Grey wolf optimizer-based back-propagation neural network algorithm," in *Proceedings of the 2016 12th International Computer Engineering Conference (ICENCO)*, pp. 213–218, Cairo, Egypt, December 2016.

[27] D. Sánchez, P. Melin, and O. Castillo, "A grey Wolf optimizer for modular granular neural networks for human recognition," *Computational Intelligence and Neuroscience*, vol. 2017, Article ID 4180510, 2017.

[28] N. Muangkote, K. Sunat, and S. Chiewchanwattana, "An improved grey wolf optimizer for training q-Gaussian Radial Basis Functional-link nets," in *Proceedings of the International Computer Science and Engineering Conference (ICSEC '14)*, pp. 209–214, IEEE, Khon Kaen, Thailand, August 2014.

[29] L. Li, L. Sun, J. Guo, J. Qi, B. Xu, and S. Li, "Modified Discrete Grey Wolf Optimizer Algorithm for Multilevel Image Thresholding," *Computational Intelligence and Neuroscience*, vol. 2017, Article ID 3295769, 2017.

[30] E. Emary, H. M. Zawbaa, and A. E. Hassanien, "Binary grey wolf optimization approaches for feature selection," *Neurocomputing*, vol. 172, pp. 371–381, 2016.

[31] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks (ICNN '95)*, vol. 4, pp. 1942–1948, Perth, Western Australia, 1995.

[32] M. Dorigo and G. Di Caro, "Ant colony optimization: a new meta-heuristic," in *Proceedings of the Congress on Evolutionary Computation*, pp. 1470–1477, July 1999.

[33] K. A. D. Jong, *Analysis of the behavior of a class of genetic adaptive systems [Ph.D. thesis]*, University of Michigan, Mich, USA, 1975.

[34] E. Rashedi, H. Nezamabadi-pour, and S. Saryazdi, "GSA: a gravitational search algorithm," *Information Sciences*, vol. 213, pp. 267–289, 2010.

[35] X. Yao, Y. Liu, and G. Lin, "Evolutionary programming made faster," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 2, pp. 82–102, 1999.

[36] R. Gribonval, H. Rauhut, K. Schnass, and P. Vandergheynst, "Atoms of all channels, unite! Average case analysis of multi-channel sparse recovery using greedy algorithms," *Journal of Fourier Analysis and Applications*, vol. 14, no. 5-6, pp. 655–687, 2008.

[37] S. Mirjalili, "The ant lion optimizer," *Advances in Engineering Software*, vol. 83, pp. 80–98, 2015.

[38] S. Mirjalili, "Moth-flame optimization algorithm: a novel nature-inspired heuristic paradigm," *Knowledge-Based Systems*, vol. 89, pp. 228–249, 2015.

[39] S. Mirjalili and A. Lewis, "The whale optimization algorithm," *Advances in Engineering Software*, vol. 95, pp. 51–67, 2016.

[40] S. Mirjalili, S. M. Mirjalili, and A. Hatamlou, "Multi-verse optimizer: a nature-inspired algorithm for global optimization," *Neural Computing and Applications*, vol. 27, pp. 495–513, 2015.