Intelligent Automation & Soft Computing
DOI: 10.32604/iasc.2020.010125
Article

Wiener Model Identification Using a Modified Brain Storm Optimization Algorithm

Tianhong Pan[1,*], Ying Song[2] and Shan Chen[2]

[1]Anhui Engineering Laboratory of Human-Robot Collabration System and Intelligent Equipment, School of Electrical Engineering and Automation, Anhui University, Hefei, 230601, China

[2]School of Electrical and Information Engineering, Jiangsu University, Zhenjiang, 212013, China

[*]Corresponding Author: Tianhong Pan. Email: thpan@ahu.edu.cn

Abstract: The Wiener model is widely used in industrial processes. It is composed of a linear dynamic block and a nonlinear static block. Estimating the Wiener model is challenging because of the diversity of static nonlinear functions and the immeasurableness of intermediate signals owing to the series structure of the Wiener model. Existing optimization algorithms cannot satisfy the requirements of accuracy and efficiency of identification and often lose into a local optimum. Herein, a modified Brain Storm Optimization (mBSO) is proposed to estimate the parameters of the Wiener model. Many different combinations of individuals from intra or extra-groups ensure the diversity of the proposed mBSO algorithm. Furthermore, the mBSO algorithm incorporates a multiplicative term. It is triggered by the current state of the population that achieves a good balance between global exploration and local exploitation. Comparative experiments are presented to demonstrate the effectiveness and efficiency of the proposed method.

Keywords: Convergence; modified Brain Storm Optimization (mBSO); parameter estimation; Wiener model

1 Introduction

The Wiener model describes the behavior of nonlinear systems adequately, and are widely applied in industrial processes, such as chemical [1], biological plant [2], and chromatographic separation processes [3]. Although the effectiveness of the Wiener model has been confirmed, estimating the structure and parameters of the Wiener model is still challenging in industrial applications [4,5].

Many studies have been reported including different techniques for identifying the Wiener model using various classical optimization methods. Wang et al. proposed the least squares iterative identification algorithm and the gradient iteration algorithm to identify Wiener nonlinear systems [6]. Furthermore, Ding et al. offered a Newton iterative identification algorithm to determine a unique Wiener system [7]. However, classical optimization methods assumed a smooth search space of the continuous derivative and optimized in the direction of the gradient. Therefore, it is easy to fall into a local extremum.

To improve the modeling accuracy and efficiency of the optimization process as well as eliminate the occurrence of optimal local trapping, stochastic evolutionary optimization algorithms are presented as an alternative and effective tool to solve nonlinear optimizations. Different stochastic algorithms can be classified according to the inspiration behind their population update mechanism [8]. One category is the evolutionary algorithm, in which the biological evolution process inspires the update process. A representative is the genetic algorithm (GA) that is used to obtain the optimal solution by mimicking the mechanism of natural selection and inheritance [9]. It is well known that classical optimization algorithms iteratively seek the optimal solution from a single initial value, while the GA processes multiple individuals simultaneously in the group. Bipin et al. adopted an adaptive GA and corresponding results were compared with those of classical GA and particle swarm optimization (PSO) algorithm [9]. The result indicated that the GA convergence accuracy was low.

The other is the swarm intelligence algorithm, in which the update process is inspired by the behaviors of some living organisms [10]. Many swarm intelligence algorithms are called foraging algorithms as they mimic the foraging behavior of animals and/or insects, such as particle swarm optimization [11] and ant colony optimization (ACO) [12]. They are inspired by bird-watching behavior and ant-searching behavior for food in nature, respectively. Compared with GAs, PSO models a society rather than the principle of survival of the fittest. Every particle in a group represents a potential solution. Wu used the Bacterial Foraging Optimization Particle Swarm Optimization (BFOPSO) algorithm to test four classical functions and modeled a Wiener simulation. The results indicated its superiority in the overall searching ability. However, the BFOPSO algorithm failed to demonstrate its efficiency in high-dimension optimization problems. Although ACO exhibits strong robustness, its searching time is longer thus affecting its convergence speed. Moreover, if excessive "elites" are selected, ACO will cause premature stagnation in the process owing to an earlier convergence to the local suboptimal solution.

Brain Storm Optimization (BSO) algorithm is a meta-heuristic optimization algorithm developed by mimicking the human brainstorming procedure [13]. Compared with other animals, the human's thinking pattern is more intelligent [14]. Therefore, optimization algorithms based on brainstorming should demonstrate better performance in accuracy and convergence than other swarm intelligence optimizations. However, the original BSO cannot obtain the global solution during successive iterations because of premature or local minima trapping [15]. Hence, a modified BSO (mBSO) algorithm is presented herein. In its update process, new ideas can be generated from a cluster center and another randomly chosen cluster. Compared with the original BSO algorithm, many combination methods are used to generate new ideas to improve the diversity of mBSO algorithms. In addition, to prevent an algorithm falling into a local optimal and facilitate in obtaining an optimal value in a population, a multiplicative term is introduced. It can intelligently change the searching domain according to the current state of an individual combined with a global-best version. Numerical and industrial cases have been presented to illustrate the performance of the mBSO algorithm.

The remainder of the paper is organized as follows: Section 2 describes the optimal problem. Section 3 details the original BSO and the mBSO. Section 4 presents the numerical simulation and industrial cases. Finally, discussions and conclusions are summarized in Section 5 and Section 6 respectively.

2 Problem Description

A Wiener model consists of a linear, time-invariant, dynamic subsystem followed by a static nonlinearity [16].

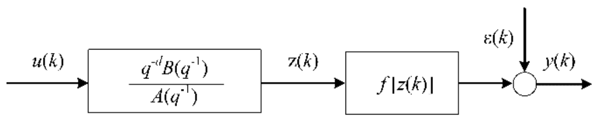


Figure 1: Structure of the Wiener model

The differential equation of the Wiener model is as follows:

$$\begin{cases} A(q^{-1})z(k) = q^{-d}B(q^{-1})u(k) \\ y(k) = f\left(z(k)\right) + \varepsilon(k) \end{cases}, \tag{1}$$

where $f(\cdot)$ is a nonlinear function, $u(k)$ and $z(k)$ are the input and output of the dynamic linear block, respectively. $y(k)$ is the output of the Wiener model, $\varepsilon(k)$ is white noise, i.e., $\varepsilon(k)\in(0,\sigma^2)$, and k is the number of sampling instants.

The polynomials $A(q^{-1})$ and $B(q^{-1})$ are defined as follows:

$$A(q^{-1}) = 1 + a_1 q^{-1} + \cdots + a_n q^{-n}$$
$$B(q^{-1}) = b_0 + b_1 q^{-1} + \cdots + b_m q^{-m}$$

The vector θ is defined as follows:

$$\theta = [-a_1, -a_2, \cdots, -a_n, b_0, b_1, \cdots, b_m]^T \tag{2}$$

Combining Eq. (1) and Eq. (2), the output of the linear block is rewritten as follows:

$$z(k) = \theta^T \varphi(k) \tag{3}$$

with

$$\begin{aligned}\varphi(k) = [&z(k-1), z(k-2), \cdots, z(k-n),\\ &u(k-d), u(k-d-1), \cdots, u(k-d-m)]^T\end{aligned}$$

Subsequently, the estimated output $\hat{y}(k)$ is:

$$\hat{y}(k) = f\left(\theta^T \varphi(k)\right) \tag{4}$$

The parameter $\hat{\theta}$ can be estimated by minimizing the objective function.

$$\begin{aligned}&J = \arg\min_{\hat{\theta}} \frac{1}{k}\sum_{l=1}^{k}\left[y(l) - f(z(l))\right]^2\\ &\quad s.t. \quad \theta^{\min} \le \hat{\theta} \le \theta^{\max}\end{aligned} \tag{5}$$

Owing to the nonlinear static block of the Wiener model, the parameter $\hat{\theta}$ cannot be solved directly. In this study, an intelligent algorithms, i.e., BSO, is used as a tool to obtain the optimal value.

3 Modified Brainstorm Optimization Algorithm

3.1 BSO Algorithm

The BSO algorithm is inspired by the brainstorming process in human-problem solving [17]. In BSO, each idea represents a potential solution to a problem and is updated in each iteration. Initially, n ideas are clustered into m clusters with k-means clustering. The best idea in each group is maintained as the cluster center. In each iteration, a new individual x^{new} is generated as follows:

• Clustering

n individuals are classified into m clusters. According to their fitness value in each cluster, the best individual will be chosen as the cluster center.

• Disruption

Randomly select an original idea x^{org} from the population, and change its value in a randomly selected cluster using Eq. (6)

$$x^{new} = x^{org} + \xi \times N(0,1), \tag{6}$$

$$\xi = rand() \times \log sig((0.5H_{\max} - H_{cur}) / s), \tag{7}$$

where $N(0,1)$ represents a Gaussian distribution with mean 0 and standard deviation 1, ξ is a dynamically updated step-size, $\log sig()$ is a logarithmic sigmoid transfer function, $H_{\max}$ is the maximum number of iterations, H_{cur} is the current iteration number, s is for changing $\log sig()$ function's slope, and $rand()$ is a random value within (0,1).

• Generation

The generation of x^{new} exhibits three characteristics: Leading, colonial and flexible.

(1) Leading

Good ideas will be clues for the next generation, i.e., assigning cluster centers with a high probability of participating in the creation of new individuals.

(2) Colonial

All individuals in the group have been thoroughly discussed according to the learning mechanism. Assume that m clusters exist; therefore, m_j is the total number of individuals in one cluster and $\sum_{j=1}^{m} m_j = n, j \in [1,m]$. In the BSO algorithm, a new individual can be created from the choice of cluster center x_j^c (see Eq. (8)) or the random individual x_j^r (see Eq. (9)) in the j cluster,

$$x^{new} = x_j^c \tag{8}$$

$$x^{new} = x_j^r \tag{9}$$

Furthermore, it can be obtained from the combination of two cluster centers (see Eq. (10)) or two random individuals by probability (see Eq. (11)).

$$x^{new} = rand() \times x_j^c + (1 - rand()) \times x_g^c, \tag{10}$$

$$x^{new} = rand() \times x_j^r + (1 - rand()) \times x_g^r, \tag{11}$$

where $g \neq j, x_g^r \neq x_j^r$, $j, g \in [1,m]$, and $rand()$ is a random value between 0 and 1. Detailed combination methods are described in Fig. 2.

(3) Flexible

Similar to a random mutation process, the algorithm will add a random value $rand()$ to the generated individuals in Eq. (7).

• Updating

During each iteration, the existing individuals will generate a new idea. After comparison, better individuals will be remained and subsequently enter the next iteration [18]. Hence, the entire process of obtaining an optimal solution will end until the number of iterations reaches the upper limit.

The procedure of the BSO algorithm is listed in Algorithm 1.

3.2 Modified BSO Algorithm

In this section, two modified procedures to improve the performance of the BSO algorithm are present.

• The generation process

$$\alpha = \left(\sum_{p \in \left\{ N \middle| \left(F(x^{new}) - F(x^{hst,p}) \right) > 0 \right\}} \frac{F(x^{new}) - F(x^{hst,p})}{F(x^{new}) - F(x^{gb})} \times (x^{hst,p} - x^{new}) \right) / n \tag{12}$$

Algorithm 1: BSO

```
Begin $H_{\max}, n, m, P_1, P_2, P_3, P_4, iter = 1$
        randomly generate $n$ ideas;
        evaluate the $n$ ideas;                                          Initialization
        while $iter \le H_{\max}$
clustering  clusters $n$ ideas into $m$ clusters using k-means;
            rank and select the best as cluster center;
            for each problem variable $j$, $g$ do
                if $rand < P_1$ then
                    replace the selected cluster center;
generatio       if $rand < P_2$ then
                    select a cluster $x_j$ randomly;
                    if $rand < P_3$ then
                        $x^{new} = x_j^c$;
                    else
                        $x^{new} = x_j^r$;
                    end if
                else
                    select two clusters $x_j$ and $x_g$;
                    select two ideas $x_j^r$ and $x_g^r$;
                    if $rand < P_4$ then
                        $x^{new} = rand() \times x_j^c + (1 - rand()) \times x_g^c$;
                    else
                        $x^{new} = rand() \times x_j^r + (1 - rand()) \times x_g^r$;
                    end if
                end if
disruption          $\xi = rand() \times logsig((0.5H_{\max} - H_{cur}) / l)$; $x^{new} = x^{org} + \xi \times N(0,1)$;
updating        if $F(x^{org}) > F(x^{new})$ then
                        $x^{org} = x^{new}$;
                    end if
                end if
            end for
        end while
end begin
```

In the BSO algorithm, the generation of new individuals relies on the combination of intra and extra-groups [19]. One or two individuals can be chosen randomly from different clusters or linearly combining two cluster centers, as shown in Fig. 2. However, the modified BSO algorithm increased the population diversity by using proposed the strategy shown in Fig. 3.

Based on the original BSO algorithm, the generation process can be characterized by Eq. (13) (Fig. 3e) or Eq. (14) (Fig. 4g) in one cluster.

$$x^{new} = rand() \times x_j^c + (1 - rand()) \times x_j^r, \tag{13}$$

$$x^{new} = rand() \times x_j^{r_t} + (1 - rand()) \times x_j^{r_h}, \tag{14}$$

where $x_j^{r_t} \neq x_j^c$, $t \neq h$ and $t, h \in [1, m_j]$.

Additionally, a new idea generated from one cluster center and one random individual in two clusters is described as follows (Fig. 3f):

$$x^{new} = rand() \times x_j^c + (1 - rand()) \times x_g^r \tag{15}$$

• The global-best update

As shown in Eq. (16), the global-best information improved the performance of meta-heuristic algorithms significantly. The effect of the global-best idea x^{gb} in the population will be added after the new idea is generated as follows:

$$x^{new} = x^{org} + rand() \times (x^{gb} - x^{org}) \times \alpha, \tag{16}$$

where $F()$ is the fitness value of the individuals, x^{new} is the current individual, $x^{hst,p}$ is the historical best individual, x^{gb} is the global best individual in each iteration, and n is the total population number. N represents the current set of individuals whose individual fitness values are larger than the historical optimal individual fitness values. $\frac{F(x^{new}) - F(x^{hst,p})}{F(x^{new}) - F(x^{gb})}$ describes the learning value of the object being studied. $(x^{hst,p} - x^{new})$ is the distance from the current individual to the object being learned.

If the current individual does not behave well, the value of $F(x^{new})$ will be large.

The greater the fitness value of the current individual, the greater the $\frac{F(x^{new}) - F(x^{hst,p})}{F(x^{new}) - F(x^{gb})}$ is. This implies that the current idea must learn from other good ideas. In other words, the searching domain will tend to a global search. On the contrary, the smaller the fitness value of the current individual, the smaller α is. When α becomes smaller, the searching domain tends to local search. Triggered by the current state of the individuals, the multiplicative term α intelligently determines the searching area of the next iteration to achieve a balance between local search and global search in the population.

The procedure of the mBSO algorithm is listed in Algorithm 2.

4 Case Studies

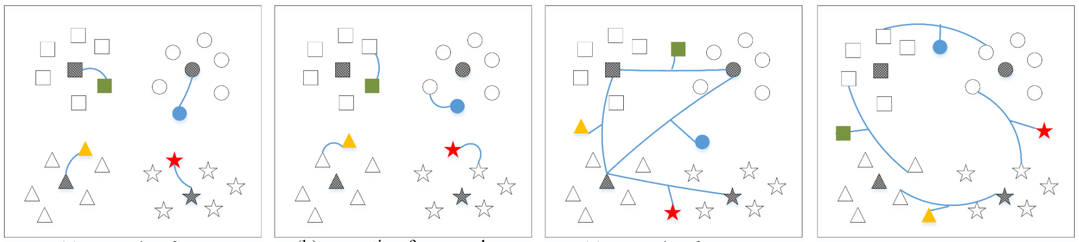
The experimental section is divided into three: The performance comparison among mBSO algorithms, original BSO algorithm, and PSO algorithm based on four benchmark functions; the comparison between the original BSO and mBSO on a second-order model, and the comparison on an actual nonlinear CE8 coupled electric drive system.

Two indexes, i.e., absolute-relative error (ARE) and root-mean-square-error ($RMSE$) are used to demonstrate the algorithm's performance.

$$ARE = \left(\left\| \hat{\theta} - \theta \right\| / \left\| \theta \right\| \right) \times 100\% \tag{17}$$

Algorithm 2: mBSO

```
Begin $H_{\max}, n, m, P_1, P_2, P_3, P_4, iter = 1$                                        initialization
        Randomly generate $n$ ideas;
        Evaluate the $n$ ideas;
clustering
    while $iter \le H_{\max}$ do
        cluster $n$ ideas into $m$ clusters using k-means;
        rank and select the best as cluster center;
        for each problem variable $j, g, t, h$ do
            if $rand < P_1$ then
                replace the selected cluster center;
generation
              if $rand < P_2$ then
                  choose one cluster;
                  if $rand < P_3$ then
                      $x^{new} = rand \times c_j^c + (1 - rand) \times c_j^r$;
                  else
                      $x^{new} = rand \times c_j^{r_t} + (1 - rand) \times c_j^{r_h}$;
                  end if
              else
                  select two clusters $x_j$ and $x_g$;
                if $rand < P_4$ && $rand < P_5$ then
                    $x^{new} = rand \times c_j^c + (1 - rand) \times c_g^c$;
                elseif $rand < P_4$ && $rand \ge P_5$ then
                    $x^{new} = rand \times c_j^c + (1 - rand) \times c_g^r$;
                else $rand \ge P_4$
                    $x^{new} = rand \times c_j^r + (1 - rand) \times c_g^r$;
                end if
              end if
        end for
disruption
        $x^{new} = x^{org} + rand \times (x^{gb} - x^{org}) \times \alpha$;
        $\alpha = eq(16)$;
        if $F(x^{org}) > F(x^{new})$ then
        if $F(x^{org}) > F(x^{new})$ then
            $x^{org} = x^{new}$;
        end if
    end while
end begin
```



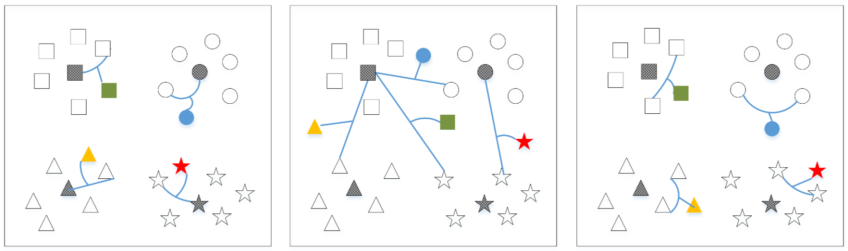
(a)generation from one cluster center

(b)generation from random individual in a cluster

(c)generation from two cluster centers

(d)generation from two random individuals in different clusters

Figure 2: Simulated searching process in the BSO Algorithm



(e)generation from the combination of one cluster center and individual in one cluster

(f)generation from the combination of one cluster center and one individual in another cluster

(g)generation from the combination of random individuals in one cluster

Figure 3: Simulated searching process in the mBSO Algorithm. Notes: Each same geometric figure represents a cluster. Unshaded shapes represent ordinary individuals in the cluster, while gray-shaded shapes represent the cluster center. Blue lines represent the generation process of new individuals, and colored shapes represent the newly-generated individuals

$$RMSE = \sqrt{\sum_{k=1}^{L}\left[\mathrm{y}(k) - \hat{y}(k)\right]^2 / L} \tag{18}$$

4.1 Benchmark Functions

Based on the Matlab software simulation platform, four selected benchmark functions (Rosenbrock, Griewank, Rastrigin and Quadric, shown in Tab. 1) are selected to test the performance of PSO, BSO and mBSO algorithms. The information of the four functions are shown in Tab. 1.

The parameters of three intelligent optimization algorithms are listed in Tab. 2. The probability value is the result obtained by multiple random experiments.

The results are shown in Fig. 4. Regardless of whether the dimension of the problem is 30 or 50, it is clear that the performance of the mBSO algorithm is the best among the three algorithms.

4.2 Numerical Simulation

A classical Wiener model is used as an example herein:

$$\begin{cases} z(k) = 1.5z(k-1) - 0.7z(k-2) \\ \qquad + u(k-1) + 0.5u(k-2) \\ y(k) = f\left(z(k)\right) + \varepsilon(k) \\ f\left(z(k)\right) = \begin{cases} \sqrt{z(k)/2}, z(k) \ge 0 \\ -\sqrt{-z(k)/2}, z(k) < 0 \end{cases} \end{cases}, \tag{19}$$

where $\theta = [a_1, a_2, b_0, b_1]^T = [-1.5, 0.7, 1.0, 0.5]^T$ and $\varepsilon(k) \in (0, 0.1)$.

Table 1: Information of four benchmark functions

Name	**Function Expression**	**Search Space**
Rosenbrock	$f_1(x) = \sum_{i=1}^{d-1} (100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2)$	$[-2.408, 2.408]^d$
Griewank	$f_2(x) = \sum_{i=1}^{d} \frac{x_i^2}{4000} - \prod_{i=1}^{d} \cos(\frac{x_i}{\sqrt{i}}) + 1$	$[-600, 600]^d$
Rastrigin	$f_3(x) = \sum_{i=1}^{d} (x_i^2 - 10\cos(2\pi x_i) + 10)$	$[-5.12, 5.12]^d$
Quadric	$f_4(x) = \sum_{i=1}^{d} (\sum_{j=1}^{i} x_j)^2$	$[-100, 100]^d$

Table 2: Parameters of three intelligent optimization algorithms

Number	Algorithm	Parameter setting
1	PSO	$n = 100, d = \{30, 50\}, m = 10, w = 0.4 - 0.9, c_1 = c_2 = 1.49445$
2	BSO	$n = 100, d = \{30, 50\}, m = 10, P_1 = 0.2, P_2 = 0.8, P_3 = 0.4, P_4 = 0.5$
3	mBSO	$n = 100, d = \{30, 50\}, m = 10, P_1 = 0.05, P_2 = 0.6, P_3 = 0.6, P_4 = 0.2, P_5 = 0.6$

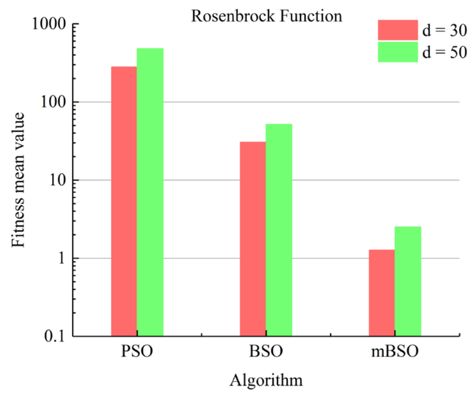
The identified results are shown in Tabs. 3–5.

Table 3: Results of the original BSO Algorithm

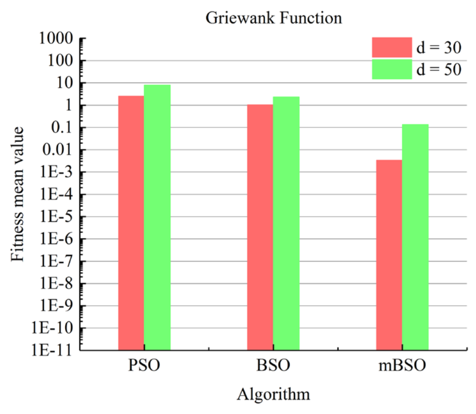
Iterations	a_1	a_2	b_0	b_1	*ARE*
20	-1.4702	0.6976	0.5833	0.7624	20.2433%
60	-1.4819	0.6785	0.8651	0.6250	8.2032%
100	-1.4894	0.6925	0.9369	0.5143	2.6272%
150	-1.4964	0.6977	0.9868	0.5207	1.0747%
200	-1.5003	0.7002	1.0073	0.4928	0.4038%
True value	-1.5	0.7	1	0.5	--

Table 4: Results of the mBSO Algorithm

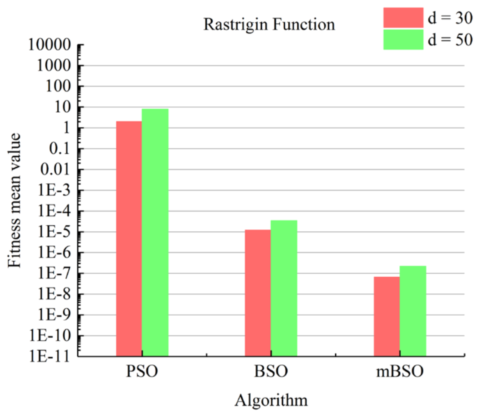
iterations	a_1	a_2	b_0	b_1	*ARE*
10	–1.4229	0.6548	1.7431	0.3658	23.8429%
15	–1.4399	0.6573	0.8923	0.8171	13.8588%
20	–1.4912	0.6910	1.0258	0.5420	2.2852%
40	–1.5103	0.7012	0.9834	0.4987	0.9820%
50	–1.5097	0.7089	1.0034	0.4999	0.6806%
True value	–1.5	0.7	1	0.5	--



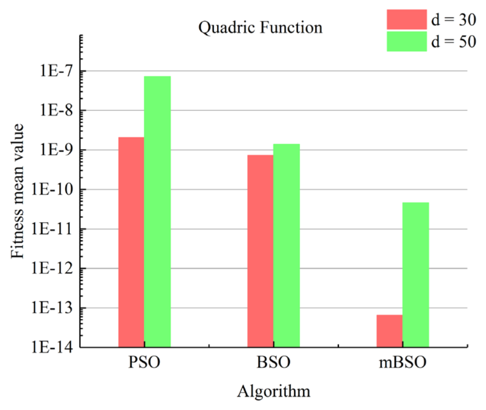
(a) f_1



(b) f_2



(c) f_3



(d) f_4

Figure 4: Comparisons of the algorithms in benchmark functions

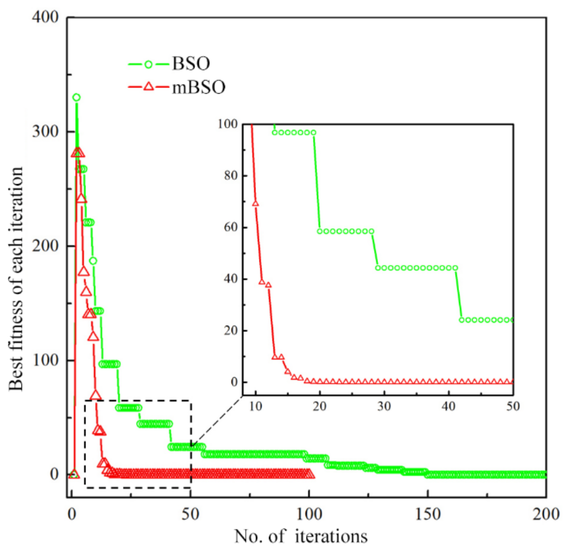


Figure 5: Optimal fitness values *vs.* iterative steps

As shown in Tab. 3 and Tab. 4, the convergent rate of the mBSO algorithm is faster than that of the original BSO algorithm. Additionally, the accuracy of the mBSO had been modified significantly. Meanwhile, the ARE was reduced to 0.6806% only after 50 iterations.

The iterative process is shown in Fig. 5; the optimal fitness value is gradually reduced, and the estimated parameters are closer to the real value. As observed, the red curve exhibits a steeper slope during the interval (5–50) and the mBSO achieved an excellent convergence.

4.3 Industrial Case

The CE8 couples two current-controlled electric motor drive systems through a pulley. The pulley is suspended in a fixed spring at one end to form a dynamic light damping mode, as shown in Fig. 6.

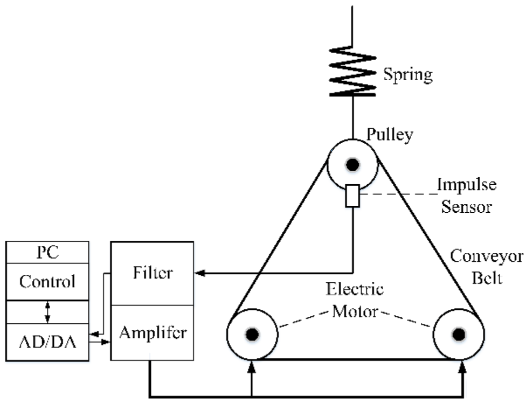


Figure 6: CE8 coupled electric drive system

The pulse sensor used in the system could not detect the positive and negative angular velocities of the pulley. Apparently, the absolute value function could describe the irreversible nonlinear process. The system model was a dynamic third-order linear model, which was described as follows [20]:

$$\begin{cases} z(k) = \dfrac{b_1 q^{-1} + b_2 q^{-2} + b_3 q^{-3}}{1 + a_1 q^{-1} + a_2 q^{-2} + a_3 q^{-3}} u(k) + w(k) \\ y(k) = |z(k)| + \varepsilon(k) \end{cases}, \tag{20}$$

where $u(k)$ the input voltage is signal; $y(k)$ is the output angular velocity; $z(k)$ is an intermediate signal, $w(k)$ and $\varepsilon(k)$ are interference signals.

To confirm the performance of the mBSO, both PSO and BSO algorithms were used to compare with the mBSO. The parameters used in the PSO, BSO and mBSO algorithms are shown in Tab. 5.

The difference between the actual and estimated outputs of the mBSO algorithm is shown in Fig. 7, and the corresponding iterative process is shown in Fig. 8.

Table 5: Parameter identification results of the stochastic algorithm

Method	a_1	a_2	a_3	b_1	b_2	b_3	*RMSE*
PSO	-2.5427	2.2956	-0.7290	0.0068	-0.0025	0.0626	0.3954
BSO	-2.5510	2.3043	-0.7320	0.0070	-0.0097	0.0721	0.3177
mBSO	-2.5639	2.3394	-0.7516	0.0068	-0.0129	0.0734	**0.2726**

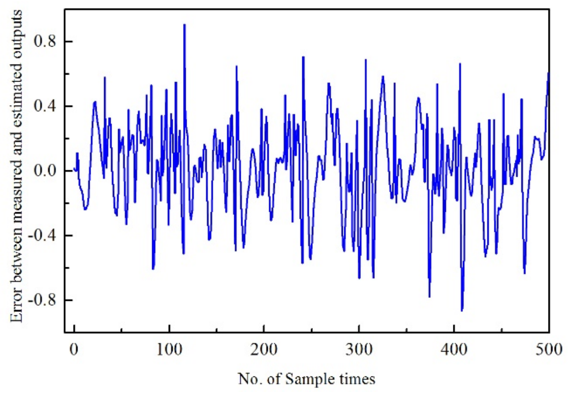


Figure 7: The deviation curve between the actual and predicted value

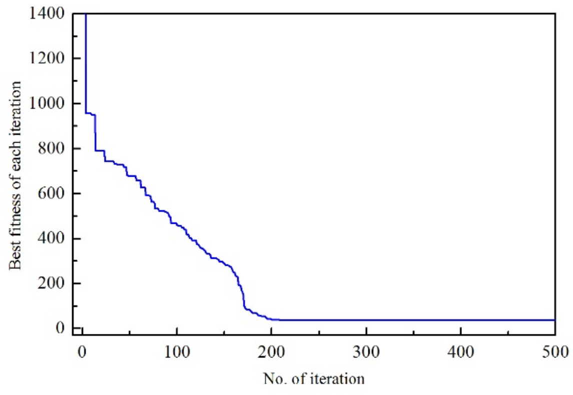


Figure 8: The optimal fitness values vs. the iterative steps

The iterative process of each stochastic algorithm is shown in Figs. 9–11.

Three algorithms are implemented in a computer with Intel(R) Core (TM) i5-3317U CPU, 4-GB memory and Windows 7 operating system. The PSO algorithm achieved convergence in 400 iterations (consuming 214.2940s), the BSO algorithm converged in 327 iterations (consuming 180.3679s), and the mBSO algorithm converged in 173 iterations (consuming 92.2496s). It can be concluded that the convergence speed of the mBSO algorithm is faster than those of the BSO and PSO algorithms.

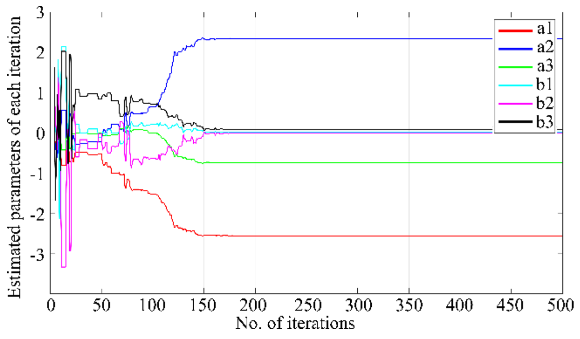


Figure 9: Parameter estimation of the mBSO

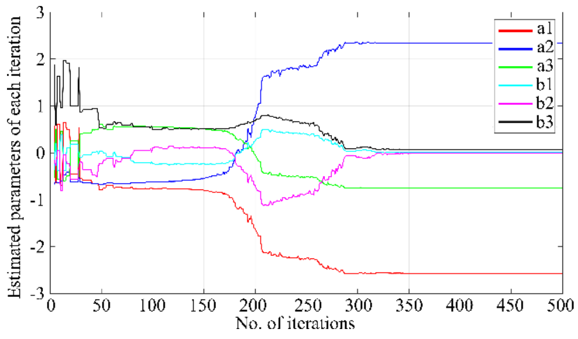


Figure 10: Parameter estimation of the BSO

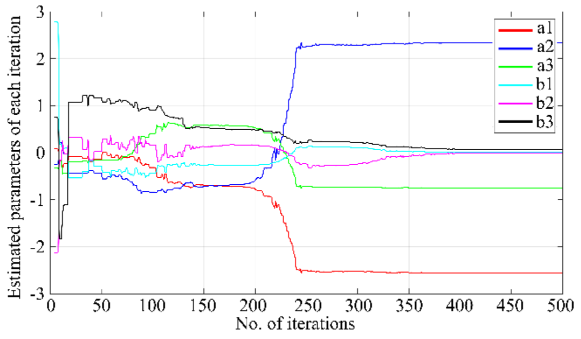


Figure 11: Parameter estimation of the PSO

5 Discussion

With optimization problems becoming increasingly complex, classical numerical optimization algorithms can no longer cater to demands, and many evolutionary optimization algorithms have been presented that achieved great success in many true-value and portfolio optimization problems. However, most stochastic optimization algorithms still suffer from "dimensional disasters". Herein, the dimensional sensitivity of the mBSO algorithm is further considered and tested using four benchmark functions. The result indicates that the increase in magnitude in the fitness mean value of the mBSO algorithm is 19.83% (shown in Fig. 4). The accuracy of the mBSO algorithm is relatively insensitive to the dimension, thus indicating that the mBSO algorithm can be used to solve large-scale optimization problems.

6 Conclusion

To accurately and quickly estimate the parameters of the Wiener model, the mBSO algorithm was presented herein. Many combination strategies of ideas in the update process and a real-time variable parameter combined with the global best design were introduced to improve the performance of the mBSO. The mBSO-based optimization technique can be extended to other block-oriented models such as the Hammerstein and the cascaded combination of Hammerstein and Wiener models.

Funding Statement: The work was supported in part by the National Nature Science Foundation of China under Grant 61873113 and the Key R&D Program of Jiangsu under Grant BE2018370.

Conflicts of Interest: No potential conflict of interest was reported by the authors.

References

[1] H. Salhi, S. Kamoun, N. Essounbouli and A. Hamzaoui, "Adaptive discrete-time sliding-mode control of nonlinear systems described by Wiener models," *International Journal of Control*, vol. 89, no. 3, pp. 611–622, 2016.

[2] H. Xia and C. Sheng, "Modeling of complex-valued Wiener systems using B-spline neural network," *Neural Networks*, vol. 22, no. 5, pp. 818–825, 2011.

[3] V. Arto, P. Hannu and A. Halme, "Modeling of chromatographic separation process with Wiener-MLP representation," *Journal of Process Control*, vol. 11, no. 5, pp. 443–458, 2001.

[4] M. Grzegorz and W. Paweł, "Kernel-based identification of Wiener–Hammerstein system," *Automatica*, vol. 83, pp. 275–281, 2017.

[5] R. R. Liu, T. H. Pan, S. Chen and Z. M. Li, "Identification of non-uniformly sampled Wiener systems with dead-zone non-linearities," *Mathematical and Computer Modelling of Dynamical Systems*, vol. 23, no. 6, pp. 595–612, 2017.

[6] D. Wang and F. Ding, "Least squares based and gradient based iterative identification for Wiener nonlinear systems," Signal *Processing*, vol. 91, no. 5, pp. 1182–1189, 2011.

[7] F. Ding, J. Ma and Y. Xiao, "Newton iterative identification for a class of output nonlinear systems with moving average noises," *Nonlinear Dynamics*, vol. 74, no. 1–2, pp. 21–30, 2013.

[8] E. Mohammed, "Global-best brain storm optimization algorithm," *Swarm and Evolutionary Computation*, vol. 37, pp. 27–44, 2017.

[9] P. R. Bipin, P. V. Rao and S. Aruna, "A new pre-distorter for linearizing power amplifiers using adaptive genetic algorithm," in *Int. Proc. on Advances in Soft Computing*, vol. 628, pp. 403–413, 2018.

[10] S. Z. Zhang, C. K. M. Lee, K. M. Yu and H. C. W. Lau, "Design and development of a unified framework towards swarm intelligence," *Artificial Intelligence Review*, vol. 47, no. 2, pp. 253–277, 2017.

[11] J. L. Wu and B. G. Xu, "Improved particle swarm optimization algorithm and its application in Wiener model identification," *Application Research of Computers*, vol. 31, no. 11, pp. 3337–3343, 2014.

[12] J. P. Liu, S. H. Xu, F. H. Zhang and L. Wang, "A hybrid genetic-ant colony optimization algorithm for the optimal path selection," *Intelligent Automation & Soft Computing*, vol. 23, no. 2, pp. 235–242, 2016.

[13] S. Cheng, Y.H. Shi, Q. Qin, Q. Zhang and R. Bai, "Population diversity maintenance in brain storm optimization algorithm," *Journal of Artificial Intelligence Soft Computing Research*, vol. 4, no. 2, pp. 83–97, 2014.

[14] Z. Jia, H. Duan and Y. H. Shi, "Hybrid brain storm optimization and simulated annealing algorithm for continuous optimization problems," *International Journal of Bio-Inspired Computation*, vol. 8, no. 2, pp. 109–121, 2016.

[15] S. Cheng, Q. Qin, J. Chen and Y. H. Shi, "Brain storm optimization algorithm: A review," *Artificial Intelligence Review*, vol. 46, pp. 445–458, 2016.

[16] M. Kaushik, S. Johan and D. C. Alexander, "Information matrix and D-optimal design with Gaussian inputs for Wiener model identification," *Automatica*, vol. 69, pp. 65–77, 2016.

[17] Y. Yang, Y. H. Shi and S. Xia, "Advanced discussion mechanism-based brain storm optimization algorithm," *Soft Computing*, vol. 19, no. 10, pp. 2997–3007, 2015.

[18] Y. H. Shi, "An optimization algorithm based on brainstorming process," *International Journal of Swarm Intelligence Research*, vol. 2, no. 4, pp. 35–62, 2014.

[19] Z. Cao, X. Hei, L. Wang, Y. H. Shi and X. Rong, "An improved brain storm optimization with differential evolution strategy for applications of ANNs," *Mathematical Problems in Engineering*, vol. 10, pp. 1–18, 2015.

[20] Y. V. R. K. Prasad, K. P. Rao and S. Sasidhara, "Hot working guide: A compendium of processing maps," 2nd ed., ASM International, Materials Park, USA, pp. 1–625, 2015.