International Journal of Innovative Computing, Information and Control
Volume **8**, Number **4**, April **2012**

ICIC International ©2012 ISSN 1349-4198
pp. 2665–2679

PERFORMANCE ENHANCEMENT OF THE DIFFERENTIAL EVOLUTION ALGORITHM USING LOCAL SEARCH AND A SELF-ADAPTIVE SCALING FACTOR

Ching-Hung Lee[1,2], Che-Ting Kuo[1] and Hao-Han Chang[1]

[1]Department of Electrical Engineering
Yuan Ze University
135 Yuan-Tung Road, Chung-Li, Taoyuan 32003, Taiwan
chlee@saturn.yzu.edu.tw

[2]Department of Mechanical Engineering
National Chung Hsin University
No. 250 Kuo Kuang Rd., Taichung 402, Taiwan

Received May 2010; revised September 2010

Abstract. *This paper presents a novel differential evolution (DE) algorithm using a dynamic strategy, local search, and a self-adaptive scaling factor ($DELS_{BP}$) to enhance the performance of the traditional DE algorithm. The $DELS_{BP}$ consists of a dynamic strategy, which updates the optimal vector instantaneously, and back-propagation-based local search, which enhances its searching capability from the corresponding neighborhood. In addition, a self-adaptive scaling factor strategy, developed using a fuzzy logic system, is introduced to accelerate the convergence velocity. The inputs of fuzzy systems incorporate the change in fitness values and generation number to calculate a change in the scaling factor. The performance of the $DELS_{BP}$ algorithm has been demonstrated using experimental results from a set of standard test functions and through the estimation of coefficients for an IIR filter with noise. These results demonstrate the performance and efficacy of the $DELS_{BP}$ algorithm.*

Keywords: Differential evolution, Optimization, Fuzzy system, Evolutionary computation, Estimation

1. Introduction.

Over the last few decades, evolutionary algorithms (EAs) have received attention for their potential as global optimization techniques [1-8]. As a method of solving optimization problems, a powerful stochastic global optimization technique – the differential evolution (DE) algorithm – was proposed by Storn and Price [9,10]. DE utilizes mutation and recombination operations as searching mechanisms and selection operators to determine the most promising regions of the search space. It creates new candidate solutions by combining the parent individual and several other individuals by randomly choosing from the same population. A candidate replaces the parent only if it has a better fitness value. This is a rather greedy selection scheme that often outperforms traditional EAs [1,2,4-8]. Recently, the DE algorithm has received increasing attention and has been applied to many real-world applications, e.g., data mining, pattern recognition, digital filter design, neural network training, and fuzzy system design [11-31].

Although the DE algorithm benefits from the aforementioned advantages, the DE convergence velocity is slow for high-dimensional optimization problems. In addition, DE is sensitive to algorithm parameters, such as low precision, and it cannot easily determine the global optimal solution if these parameters are not properly met. Therefore, the selection of the scaling factor F is very important and difficult. In our experience, large values of F can converge quickly, but they usually converge at a local minimum.

Inversely, small values of F converge slowly. In the literature [23], F was selected by experienced human operators. In the literature [17,19,20], several self-adaptive strategies are proposed that adjust the parameter of the traditional DE algorithm. In this paper, we developed a novel DE algorithm based on the advantages of a dynamic strategy, back-propagation local search (or the gradient-descent method), and a self-adaptive scaling factor using a fuzzy system to improve the efficiency of the traditional DE algorithm. First, the dynamic strategy updates the optimal vector instantaneously, which provides the current best information for other individuals for updating the corresponding vector. The proposed DELS$_{\text{BP}}$ not only has a local search ability but also updates the best vector instantaneously and uses a fuzzy logic system to adjust the scaling factor F. The generation number and change of fitness value are used to generate changes in F using a fuzzy approach. These improvements could increase the precision and convergence velocity of DE. The experimental results of a set of standard test functions are adopted to demonstrate the effectiveness of the technique.

For digital signal processing, digital filter design is a basic but important topic. The digital filter can preserve some desired frequencies and remove others when an external input signal is passed through the filter. In the field of time domain filter estimation, system identification is the subject of identifying coefficients given measurements of the input and output signal [33]. In this paper, time domain coefficient identification of infinite impulse response (IIR) filters was considered to demonstrate the performance of our proposed DELS$_{\text{BP}}$ algorithm.

The rest of this paper is organized as follows. Section 2 gives an introduction of traditional DE and the procedure of the DELS$_{\text{BP}}$ algorithm. Numerical results from a set of standard functions are introduced in Section 3. Section 4 describes the IIR filter and coefficient estimation scheme using the DELS$_{\text{BP}}$ algorithm. The simulation results are also reported in Section 4. Finally, Section 5 concludes this paper.

2. Traditional Differential Evolution Algorithm.

The differential evolution (DE) algorithm proposed by Storn and Price is a population-based algorithm of evolutionary computation designed to solve nonlinear global optimization problems [4-8]. DE is effective, efficient, and robust, and it utilizes mutation and recombination operations as searching mechanisms and selection operations to determine the most promising regions of the search space. New candidate solutions are created by combining the parent individual and several other individuals by randomly choosing from the same population, where the candidate with a better fitness value replaces the parent individual. This is a rather greedy selection scheme that often outperforms traditional EAs [1-8].

2.1. Initialization.

Like others EAs, the DE algorithm starts with initial population vectors that are randomly generated when no preliminary experimental knowledge about the searching space is available. To start the DE, we must decide the population size P_S, the maximum generation G, and D-dimensional parameters vectors, and we must generate parameters randomly. The pth vector of the population at generation g (or iteration) is denoted as $\Theta_p(g) = (\theta_{p1}, \theta_{p2}, \ldots, \theta_{pD})$, $p = 1, \ldots, P_S$, $g = 1, 2, \ldots, G$, where D is the parameter dimension of the optimization problem. After that, the evaluation is implemented.

2.2. Mutation.

Mutation can be viewed as an operation that combines a differential vector with a population vector. The formula for mutation is that a population vector $\Theta_p(g)$ is mutated into a mutation vector $v_p(g+1)$ by adding the weighted difference of two randomly selected population vector $\Theta_{r_1}(g)$ and $\Theta_{r_2}(g)$, i.e.,

$$v_p(g+1) = \Theta_p(g) + F \cdot (\Theta_{r_1}(g) - \Theta_{r_2}(g)) \tag{1}$$

where $F \in [0,1]$ is the scaling factor (or mutation factor), $r_1, r_2 \in \{1, 2, \ldots, P_S\}$, and $p \neq r_1 \neq r_2$. The scaling factor F is a real constant that controls the amplification of the differential variation.

2.3. **Crossover.** To increase the diversity, crossover is introduced. The goal of implementing the crossover is to obtain a trial vector by replacing certain parameters of the parent with corresponding parameters of a randomly generated donor vector. A vector of solutions is selected randomly from the mutant individuals when rand(d) is less than the value of the crossover probability CR ($0 \leq CR \leq 1$), i.e.,

$$u_p(g) = (u_{p1}(g+1), u_{p2}(g+1), \ldots, u_{pD}(g+1)) \tag{2}$$

$$u_{pd}(g+1) = \begin{cases} v_{pd}(g+1) & \text{if } (\text{rand}(d) \leq CR) \\ \Theta_{pd}(g) & \text{if } (\text{rand}(d) > CR), \end{cases} \quad d = 1, 2, \ldots, D \tag{3}$$

where rand(d) $\in [0,1]$ is a random number.

2.4. **Selection.** Selection is the procedure of producing better offspring. If trial vector $u_p(g+1)$ has an equal or smaller fitness value than that of its parents $\Theta_p(g)$, it replaces the parent vector in the next generation; otherwise, the parent is retained in the population, i.e.,

$$\Theta_p(g+1) = \begin{cases} u_p(g+1) & \text{if } f(u_p(g+1)) \leq f(\Theta_p(g)) \\ \Theta_p(g) & \text{if otherwise,} \end{cases} \quad p = 1, 2, \ldots, P_S. \tag{4}$$

3. **Differential Evolution Using Local Search and a Self-Adaptive Scaling Factor (DELS$_{\mathbf{BP}}$).** Although DE is easy to use for solving the optimization problems, the convergence velocity is slow, and it has low precision. We proposed a novel DE algorithm (called the DELS$_{\text{BP}}$ algorithm) to improve the efficiency of the traditional DE. The DELS$_{\text{BP}}$ consists of a dynamic strategy to determine the best vector, gradient-descent-based local search, and self-adaptive scaling factor. A detailed description for DELS$_{\text{BP}}$ algorithm is introduced below.

The DELS$_{\text{BP}}$ algorithm for optimization problems is in the form of

Minimize $f(\Theta)$

subject to $\Theta \in S$, $\quad S = \left\{\Theta \in \Re^D \,|\theta_d \in \Re,\ d = 1, \ldots, D\right\}$

where D is dimension of the problem and $f(\Theta)$ is the function to be minimized. Each vector Θ represents a solution.

3.1. **Initialization.** The procedure is similar to the traditional DE algorithm. The difference is that the best vector Θ_{best} must be obtained. Afterward, the DELS$_{\text{BP}}$ algorithm process begins.

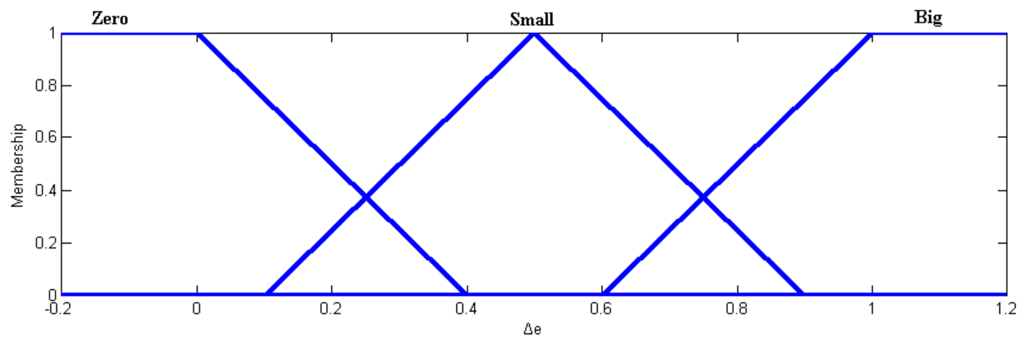
3.2. **Mutation with self-adaptive scaling factor *F*.** The mutation strategy of the traditional DE algorithm is based on each population vector $\Theta_p(g)$ and a weighted differential vector, which would search for the optimal solution slowly. Herein, a dynamic strategy is used to provide the optimal solution updating instantaneously. The difference between the traditional strategy and the dynamic strategy is that the $\Theta_p(g)$ is replaced by the best vector Θ_{best} (Θ_{best} keeps the current best fitness value in the population). This modification provides the current best information for other vectors and leads to the optimal solution. For each population vector $\Theta_p(g)$, $p = 1, 2, 3, \ldots, P_S$, $g = 1, 2, \ldots, G$, a mutation vector is generated as follows:

$$v_p(g+1) = \Theta_{best} + F \cdot (\Theta_{r_1}(g) - \Theta_{r_2}(g)) \tag{5}$$

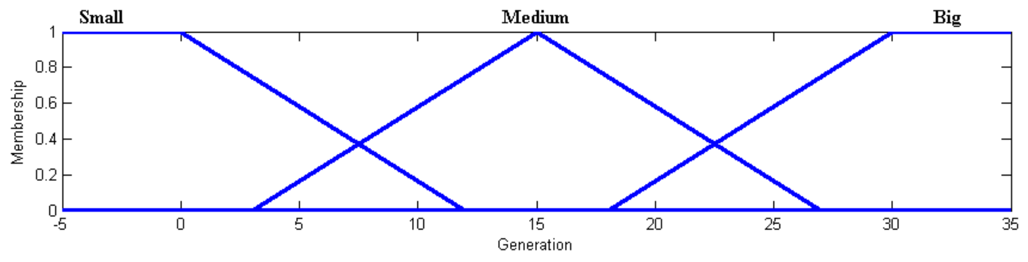
where $F \in [0,1]$ is the scaling factor (or mutation factor), $r_1, r_2 \in \{1, 2, \ldots, P_S\}$, and $p \neq r_1 \neq r_2$. The scaling factor F is a real constant that controls the amplification of the differential variation. The differential vector is the deviation between two randomly selected population vectors $\Theta_{r_1}(g)$ and $\Theta_{r_2}(g)$. The generated vector $v_p(g+1)$ is the mutation vector. The selection of an appropriate value of F is important in the DE optimization process. Herein, we propose a novel self-adaptive strategy for selecting the scaling factor based on fuzzy logic systems. The inputs of the fuzzy system are changes of fitness value (Δe) and generation number. The corresponding output is ΔF. The fuzzy system is constructed by the following fuzzy-rules table shown in Table 1.

Table 1. Fuzzy rules table for the self-adaptive scaling factor F

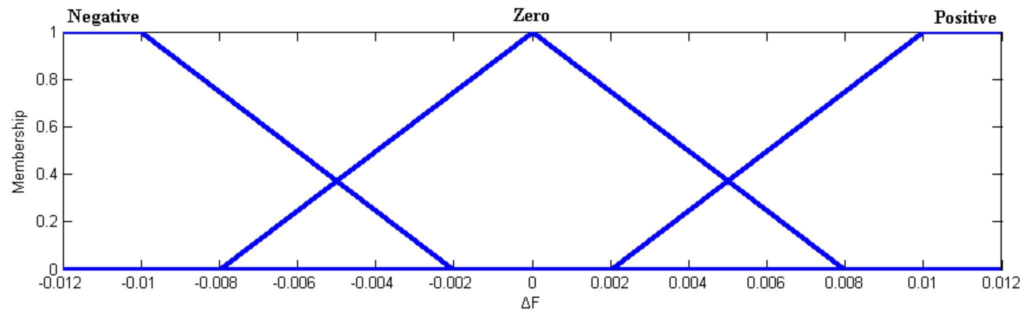
Δe / Generation	Zero	Small	Large
Small	Positive	Positive	Zero
Medium	Positive	Zero	Negative
Large	Zero	Negative	Negative



(a)



(b)



(c)

Figure 1. Membership functions of the fuzzy system: (a) input variable – Δe, (b) input variable – generation, and (c) output variable – ΔF

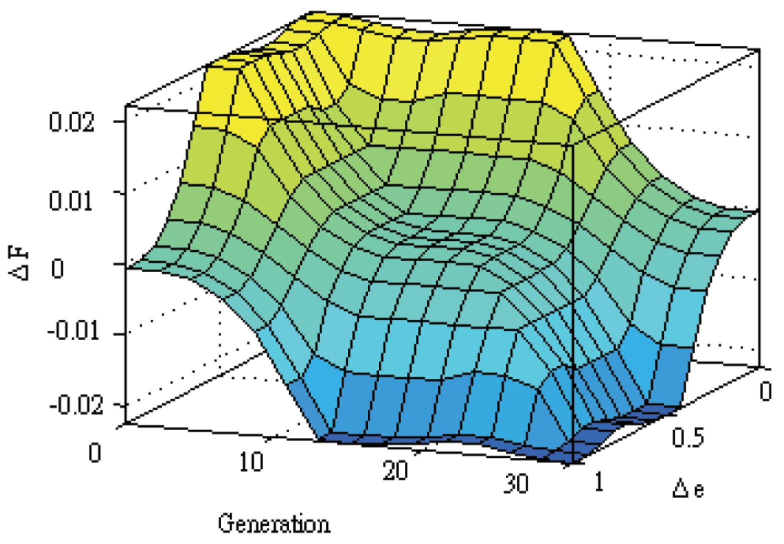


FIGURE 2. The corresponding surface of ΔF for a self-adaptive scaling factor F

The corresponding membership functions for inputs Δe and generation and output ΔF are shown in Figures 1(a), 1(b), and 1(c), respectively. The consequence ΔF is deduced from Δe and generation by taking the max-min composition. The center-of-area (COA) method is used as the defuzzification strategy. Figure 2 shows the corresponding change surface ΔF of an adaptive scaling factor F.

3.3. **Local search by the back-propagation technique.**

To improve the capabilities of the traditional DE algorithm, gradient-descent-based local search was adopted [1,17]. The back-propagation algorithm was integrated as an operator in the global search to increase the convergence rate of the optimization process.

First, the learning rate η has to be determined properly, and corresponding error cost function $E(v_p(g+1))$ is defined as

$$E(v_p(g+1)) = \frac{1}{2}\sum_{k=1}^{T} e^2(k) = \frac{1}{2}\sum_{k=1}^{T} [y(k) - \hat{y}(k)]^2 \tag{6}$$

where $e(k)$ is the error between the two outputs of the actual system and the estimated system and T is the data number. By the gradient-descent method, the update law is

$$v'_{pd}(g+1) = v_{pd}(g+1) + \Delta v_{pd}(g+1) = v_{pd}(g+1) + \eta_d \left(-\frac{\partial E(v_p(g+1))}{\partial v_{pd}(g+1)} \right) \tag{7}$$

where $d = 1, 2, \ldots, D$. Equation (7) provides the update of the mutation vector $v'_p(g+1)$. In our experience, this hybridized approach can speed up the convergence and easily find the optimal solution. Experimental results from several standard test functions and an illustrated example of applications for digital filter coefficients estimation will be introduced in Section 4, Examples 4.1 and 4.2, to demonstrate the effectiveness and performance of our algorithm.

3.4. **Crossover (recombination).**

The goal of implementing the crossover is to obtain a trial vector by replacing certain parameters of the parent with the corresponding parameters of a randomly generated donor vector. A vector of solutions is selected randomly from the mutant individuals when rand(d) is less than the value of the crossover probability

CR $(0 \leq CR \leq 1)$, i.e.,

$$u_p(g+1) = (u_{p1}(g+1), u_{p2}(g+1), \ldots, u_{pD}(g+1)) \tag{8}$$

$$u_{pd}(g+1) = \begin{cases} v'_{pd}(g+1) & \text{if } (\text{rand}(d) \leq CR) \\ \Theta_{pd}(g) & \text{if } (\text{rand}(d) > CR), \end{cases} \quad d = 1, 2, \ldots, D \tag{9}$$

where $\text{rand}(d) \in [0, 1]$ is a random number.

3.5. **Selection.** Selection is the procedure of producing better offspring. If trial vector $u_p(g+1)$ has an equal or smaller fitness value than that of its parent $\Theta_p(g)$, it replaces the parent vector in the next generation; otherwise, the parent is retained in the population, i.e.,

$$\Theta_p(g+1) = \begin{cases} u_p(g+1) & \text{if } f(u_p(g+1)) \leq f(\Theta_p(g)) \\ \Theta_p(g) & \text{if otherwise} \end{cases} \quad p = 1, 2, \ldots, P_S \tag{10}$$

The flowchart of the proposed DELS$_{BP}$ algorithm is described in Figure 3.

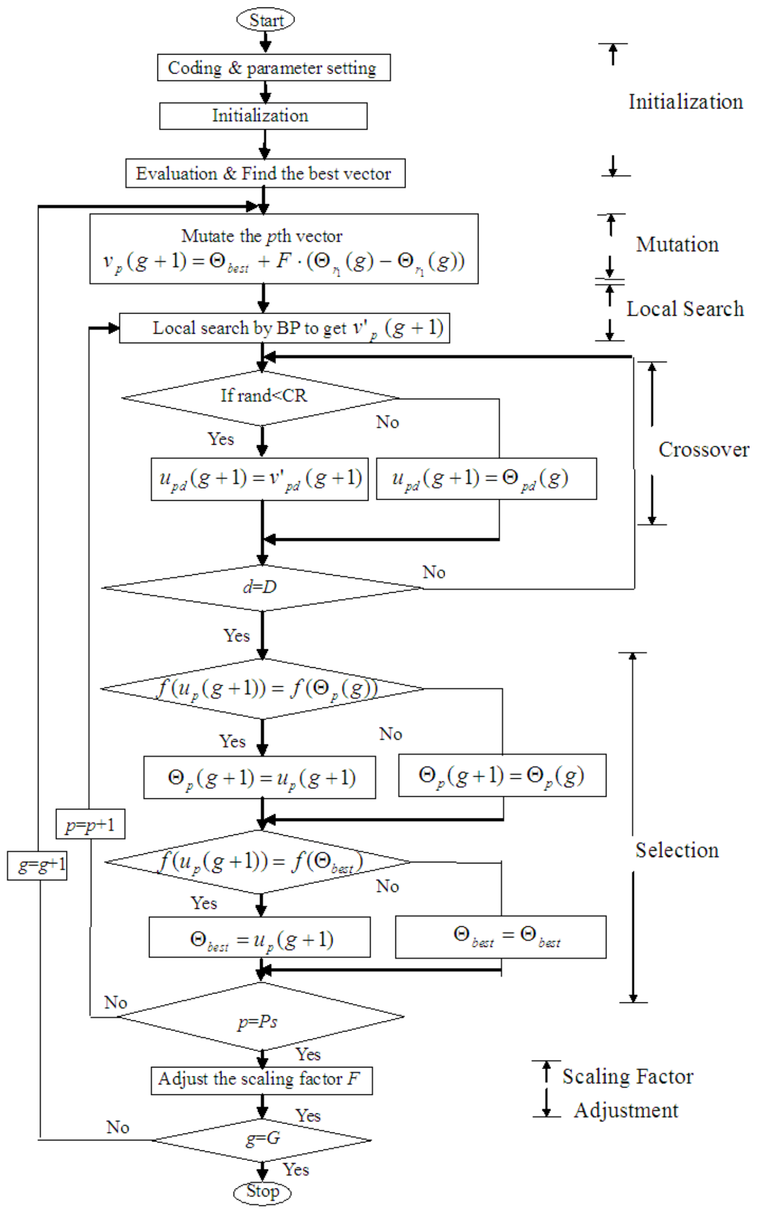

Start
Coding & parameter setting
Initialization
Evaluation & Find the best vector
Mutate the pth vector
$v_p(g+1) = \Theta_{best} + F \cdot (\Theta_{r_1}(g) - \Theta_{r_2}(g))$
Local search by BP to get $v'_p(g+1)$
If rand<CR
Yes
No
$u_{pd}(g+1) = v'_{pd}(g+1)$
$u_{pd}(g+1) = \Theta_{pd}(g)$
d=D
No
Yes
$f(u_p(g+1)) = f(\Theta_p(g))$
Yes
No
$\Theta_p(g+1) = u_p(g+1)$
$\Theta_p(g+1) = \Theta_p(g)$
p=p+1
g=g+1
$f(u_p(g+1)) = f(\Theta_{best})$
Yes
No
$\Theta_{best} = u_p(g+1)$
$\Theta_{best} = \Theta_{best}$
No
p=Ps
Yes
Adjust the scaling factor F
Yes
No
g=G
Yes
Stop
Initialization
Mutation
Local Search
Crossover
Selection
Scaling Factor
Adjustment

FIGURE 3. Flowchart of the proposed DELS$_{BP}$ algorithm

4. **Simulation Results.** In this section, two illustrated examples are introduced to show the performance of this approach, including optima searching of the testing function and infinite impulse response (IIR) filter coefficients estimation by the proposed DELS$_{\text{BP}}$ algorithm. All of the simulations were performed using MATLAB on an Intel Pentium 4 computer with a clock rate of 2.4 GHz and 2.96 GB of main memory.

Example 4.1. ***Optmization of the Testing Functions***
In engineering fields, one is often confronted with the problem of functional optimization. To illustrate the performance and viability of the proposed DELS$_{BP}$ algorithm, a set of standard test functions, given in Table 2, is used. The algorithms used for the comparison are the DELS$_{BP}$ algorithm, the traditional DE algorithm, and the DELS$_{BP}$ algorithm without local search. The population size P_S is set to be 10 for all algorithms and test functions, and no other parameters were adjusted during the evolution. Other parameters are shown in Table 3. For the optimization of these test functions, the performance index is adopted as an error square between the searched value and the optimal solution. In addition, the variable dimension D is also chosen. We choose $n = 1$ for Ackley, Griewangk, Michalewicz, and Parallel Axis and $n = 2$ for Rastrigin, Rosenbrock, and Schwefel. For the statistical analysis, the optimization process is repeated for 50 independent runs for all the algorithms. The comparison results of optimization of testing functions by each algorithm are shown in Table 4. The simulation results show the effectiveness of the DELS$_{BP}$ algorithm and demonstrate that it has the capability for global searching. The DELS$_{BP}$ algorithm clearly outperforms the other algorithms, and the data show that the modifications do indeed improve the performance of the DE algorithms.

TABLE 2. Test functions

	Function	Decision Space	Optimal Solution
Ackley	$-20+e+20e^{-\frac{0.2}{n}\sqrt{\sum_{i=1}^{n} x_i^2}} - e^{\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)x_i}$	$[-1,1]^n$	0
Griewangk	$\sum_{i=1}^{n}\frac{x_i^2}{4000} - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1$	$[-200,200]^n$	0
Michalewicz	$-\sum_{i=1}^{n}\sin x_i\left(\sin\left(\frac{i\cdot x_i^2}{\pi}\right)\right)$	$[0,\pi]^n$	-0.87
Parallel Axis	$\sum_{i=1}^{n} i\cdot x_i^2$	$[-5.12,5.12]^n$	0
Rastrigin	$10n+\sum_{i=1}^{n}\left(x_i^2-10\cos(2\pi x_i)\right)$	$[-5.12,5.12]^n$	0
Rosenbrock	$\sum_{i=1}^{n-1}\left(\left(x_{n+1}-x_i^2\right)^2+(1-x)^2\right)$	$[-2.048,2.048]^n$	0
Schwefel	$\sum_{i=1}^{n} x_i\sin\left(\sqrt{\lvert x_i\rvert}\right)$	$[-500,500]^n$	-837.729

TABLE 3. Parameter settings for the optimization of the testing functions

Parameters	P_S	G	CR	F	η
Values	10	10	0.5	0.5	0.1

Example 4.2. ***Application in Coefficient Estimation of IIR Filter***
In time domain filter estimation, filter coefficient identification provides measurements from the input and output signals [33]. Herein, time domain coefficient identification of infinite impulse response (IIR) filters is considered, i.e., the past and present inputs and past output signals are used to generate each new output signal.

TABLE 4. Comparison results for the optimization of the test functions

Test Functions	Traditional DE	DELS$_{BP}$ without BP	DELS$_{BP}$
Ackley	8.0900	2.7689	0
Griewangk	0.1484	0.0676	3.34×10^{-3}
Michalewicz	1.1962×10^{-4}	1.3×10^{-6}	1.0645×10^{-7}
Parallel Axis	1.1047×10^{-4}	1.05×10^{-5}	2.2×10^{-8}
Rastrigin	5.8362	2.3235	0.4011
Rosenbrock	0.12	0.03	4×10^{-8}
Schwefel	344.3316	152.4282	42.7306

Consider the following IIR filter:

$$\begin{aligned} y[n] &= \sum_{k=1}^{N} a_k y[n-k] + \sum_{k=0}^{M} b_k x[n-k] \\ &= a_1 y[n-1] + a_2 y[n-2] + \cdots + a_N y[n-N] + b_0 x[n] + b_1 x[n-1] \\ &\quad + b_2 x[n-2] + \cdots + b_M x[n-M] \end{aligned} \tag{11}$$

where x is the external input signal; y is the output of the actual IIR filter or a signal filtered by this filter; N is the number of past outputs, normally referred to as the order of the filter; M is the number of past inputs; and a_k and b_k are parameters that determine the contribution of each output and input value for each sample point. Herein, the values of N and M of the IIR filter are assumed to be previously known for simplicity. Our objective is to estimate the coefficients a_k and b_k. Therefore, we construct the estimated system

$$\begin{aligned} \hat{y}[n] &= \sum_{k=1}^{N} \hat{a}_k \hat{y}[n-k] + \sum_{k=0}^{M} \hat{b}_k x[n-k] \\ &= \hat{a}_1 \hat{y}[n-1] + \hat{a}_2 \hat{y}[n-2] + \cdots + \hat{a}_N \hat{y}[n-N] + \hat{b}_0 x[n] + \hat{b}_1 x[n-1] \\ &\quad + \hat{b}_2 x[n-2] + \cdots + \hat{b}_M x[n-M] \end{aligned} \tag{12}$$

where $\hat{a}_k$ and $\hat{b}_k$ are the estimated coefficients and $\hat{y}$ is the corresponding output of the estimated filter. By using the proposed DELS$_{BP}$ algorithm, $\hat{a}_k$ and $\hat{b}_k$ will approach the actual values of a_k and b_k. Let $\Theta = [\theta_1, \theta_2, \ldots, \theta_m] = [\hat{a}_1, \hat{a}_2, \ldots, \hat{a}_N, \hat{b}_0, \hat{b}_1, \ldots, \hat{b}_M]$ be a new estimated coefficients vector, where $m = N + M + 1$ is the dimension of the parameters.

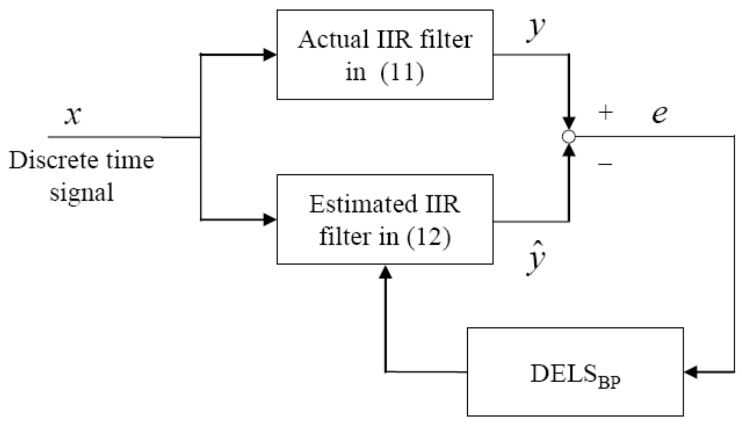


FIGURE 4. The coefficient estimation architecture for IIR filters using DELS$_{BP}$

Figure 4 shows the architecture for filter coefficient estimation using the DELS$_{BP}$ algorithm. For this optimization problem, a cost function value should be defined. The sum of squared error (SSE) is adopted as the cost function

$$\text{SSE} = \sum_{k=0}^{T} (y[k] - \hat{y}[k])^2 = \sum_{n=0}^{T} e^2[k], \tag{13}$$

where T is the sampling data number. The goal of this paper is to find the optimal model coefficients in (12) such that the SSE in (13) is minimized as much as possible. We use the following learning procedure for IIR filter coefficient estimation by applying the DELS$_{BP}$ algorithm.

Learning Procedure DELS$_{\mathbf{BP}}$ for IIR Filter Estimation

Step 1: Set the initial population size P_S, parameter number D, mutation rate F, and crossover rate CR.
Step 2: Define the cost function. In this paper, we use the sum of squared error (SSE) in (13).
Step 3: Evaluate the cost function value for all initial vectors and find the best vector Θ_{best}.
Step 4: Randomly choose two different vectors and mutate the best vector to get the pth mutated vector in (5).
Step 5: Apply local search by using the back propagation algorithm to get $v_p'(g+1)$.
Step 6: Apply the crossover between $\Theta_p(g)$ and $v_p'(g+1)$ to obtain the trial vector $u_p(g+1)$.
Step 7: Evaluate the cost function value of $u_p(g+1)$ and select a better vector between $\Theta_p(g)$ and $u_p(g+1)$.
Step 8: Update the best Θ_{best} if $u_p(g+1)$ is better than Θ_{best}.
Step 9: Repeat Steps 4 to 8 until the entire population is done.
Step 10: Adjust the mutation rate F by using a fuzzy system.
Step 11: Repeat Steps 4 to 10 until the maximum iteration has been reached.

The parameters used in the DELS$_{\text{BP}}$ operations for the following simulations are listed in Table 5, and the external input signal $x[n]$ is

$$x[n] = \frac{2}{3}\cos\left(n\frac{2\pi}{3}\right) + \frac{1}{6}\cos\left(n\frac{14\pi}{15}\right), \text{ for } n = 0, 1, 2, \ldots, 50. \tag{14}$$

Table 5. Parameter settings used in the DELS$_{\text{BP}}$ operations

Parameters	T	P_S	G	CR	F	η	$(\theta_{\min},\theta_{\max})$
Values	50	35	30	0.5	0.5	0.5	$[-1, 1]$

Case A. Noise-free case

Consider an actual band-pass IIR filter [34]

$$y[n] = -0.5926y[n-1] - 0.1193y[n-2] + 0.4404x[n] - 0.4404x[n-2]. \tag{15}$$

From (15), we know that the actual filter coefficients are $a_l = -0.5926$, $a_2 = -0.1193$, $b_0 = 0.4404$, and $b_2 = -0.4404$. Using the aforementioned design steps and Figure 4 for IIR filter coefficient identification, several numerical simulations and comparisons can be made. To demonstrate the effectiveness, the estimated coefficients of the actual IIR filter are compared with the estimated IIR filter, which is performed with traditional DE, DELS$_{\text{BP}}$ without ΔF, and DELS$_{\text{BP}}$ as listed in Table 6. From the comparison results of Table 6, the proposed DELS$_{\text{BP}}$ can exactly identify the actual IIR coefficients.

In addition, the results of DELS_{BP} without ΔF have smaller error compared with the traditional DE. This shows the performance of DELS_{BP} and the adaptive fuzzy scaling factor. The estimated trajectories of filter coefficients obtained by the proposed novel DELS_{BP} are shown in Figure 5. In addition, Figure 6 shows the convergence trajectories of the cost function of the traditional DE (dashed-line), DELS_{BP} without ΔF (dotted-line), and DELS_{BP} (solid-line). The value of the adaptive scaling factor F is shown in Figure 7. This demonstrates the efficient tuning of the fuzzy scaling factor for improving the optimization. The estimated results (Table 6 and Figures 5 and 6) reveal that all of the filter coefficients are accurately approximated in DELS_{BP} and that the coefficient of the absent order $x[n-1]$ is also successfully identified.

TABLE 6. Comparison between the coefficients of the actual IIR filters and IIR filters estimated by using DELS_{BP}

	a_1	a_2	b_0	b_1	b_2
Actual values	-0.5926	-0.1193	0.4404	0	-0.4404
Traditional DE	-0.9135	-0.0612	0.3747	0.1279	-0.6509
DELS_{BP} without ΔF	-0.5628	-0.1059	0.423	-0.0471	-0.4698
DELS_{BP}	-0.5926	-0.1193	0.4404	0.0000	-0.4404

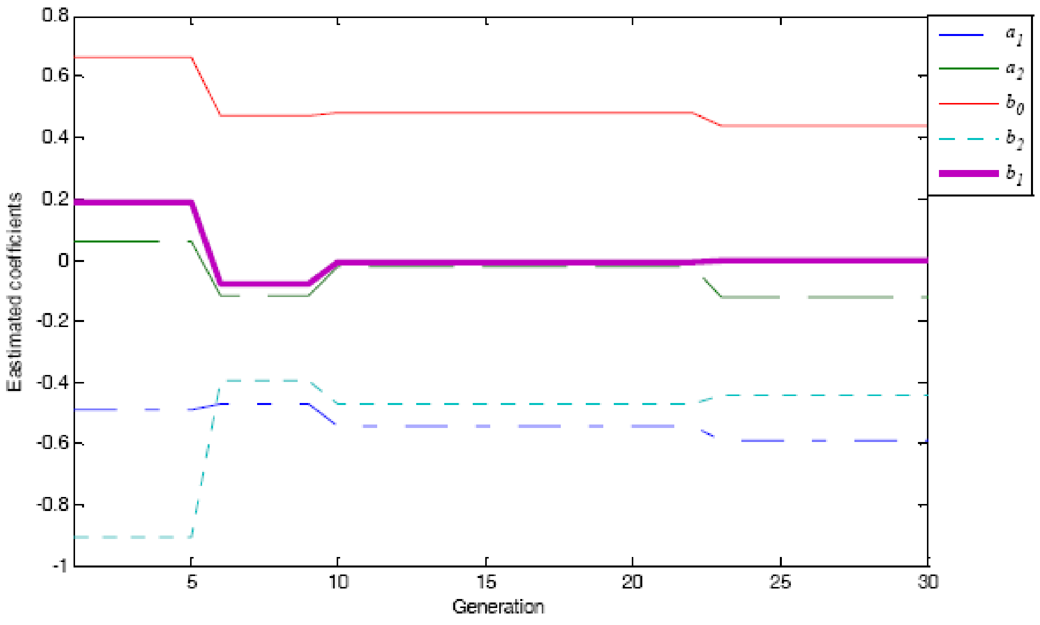


FIGURE 5. Trajectory of all DELS_{BP} estimated coefficients

Case B. Noise disturbed

Consider an IIR filter disturbed by Gaussian noise

$$y[n] = -0.5926y[n-1] - 0.1193y[n-2] + 0.4404x[n] - 0.4404x[n-2] + wgn[n]. \quad (16)$$

The difference between the two examples is that we consider an actual band-pass IIR filter, which is subject to additive white Gaussian noise. The estimated coefficients of the additive noise filter are compared with the estimated IIR filter, which is calculated using traditional DE, DELS_{BP} without ΔF, and DELS_{BP}, as listed in Table 7. The coefficient trajectories estimated by the proposed DELS_{BP} are shown in Figure 8. Figure 9 shows the

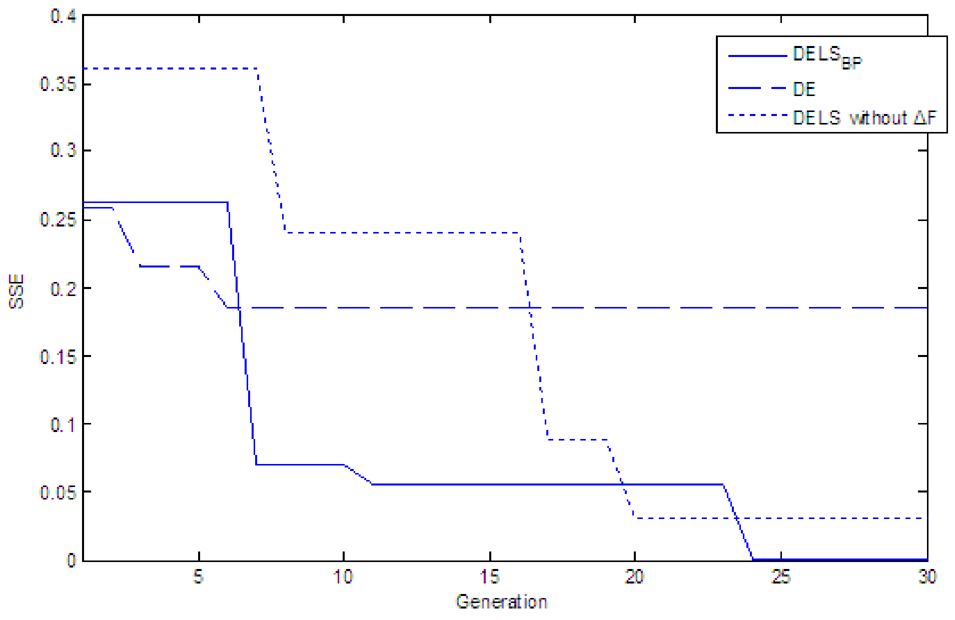


Figure 6. SSE convergence trajectory of Example 4.1 – noise-free case

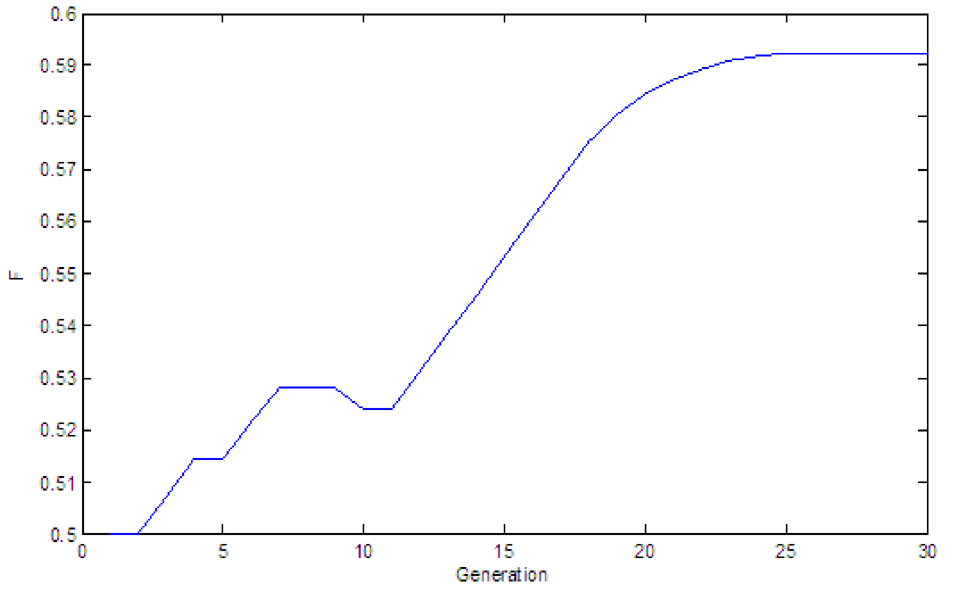


Figure 7. Simulation result of Example 4.1: trajectory of scaling factor F

Table 7. Comparisons between the coefficients of the actual IIR filter and IIR filter estimated by using DELS$_{BP}$ (SNR = 20 dB)

	a_1	a_2	b_0	b_2
Actual values	-0.5926	-0.1193	0.4404	-0.4404
DE	-0.5475	0.2875	0.3668	-0.4621
DELS$_{BP}$ without ΔF	-0.5637	-0.173	0.5027	-0.5287
DELS$_{BP}$	-0.5866	-0.1066	0.4422	-0.4410

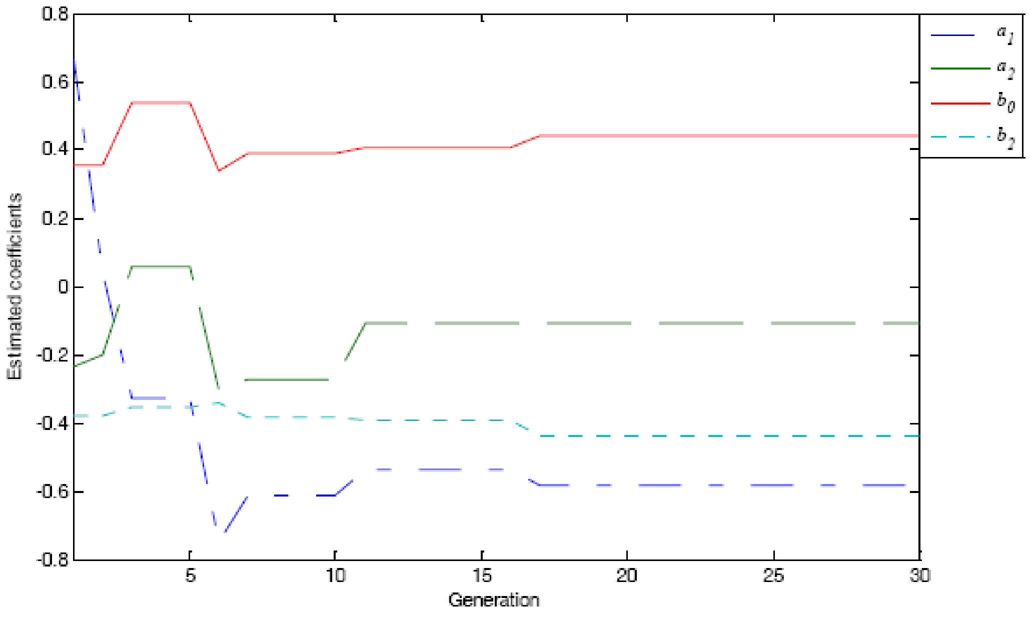


FIGURE 8. Estimated coefficient trajectories by DELS_{BP} for Example 4.2 – noise disturbed

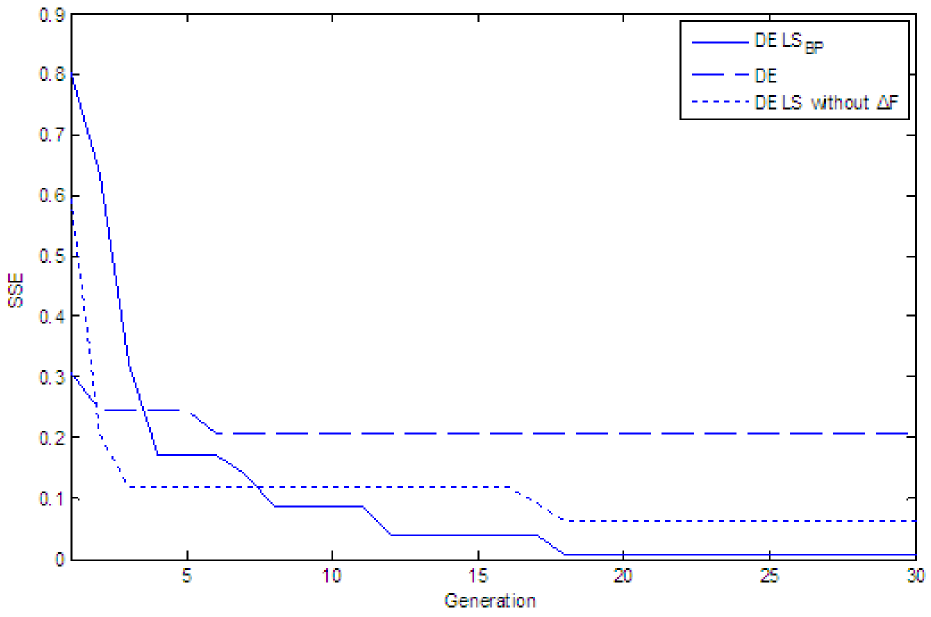


FIGURE 9. SSE convergence trajectory of Example 4.2 – noise disturbed case

convergence trajectories of the cost function of the traditional DE (dashed-line), DELS_{BP} without ΔF (dotted-line), and DELS_{BP} (solid-line). Figure 10 shows the trajectory of the scaling factor F for Example 4.2, and Figure 11 shows the estimation performance (estimated error of coefficients versus signal-to-noise ratio (SNR)) of the simulation results. Herein, the estimated error of the coefficients is $\sum_{i=1}^{2}(a_i - \hat{a}_i)^2 + \sum_{i=0}^{2}(b_i - \hat{b}_i)^2$. Our approach performs well (the coefficients can be estimated exactly) even when the output signal is

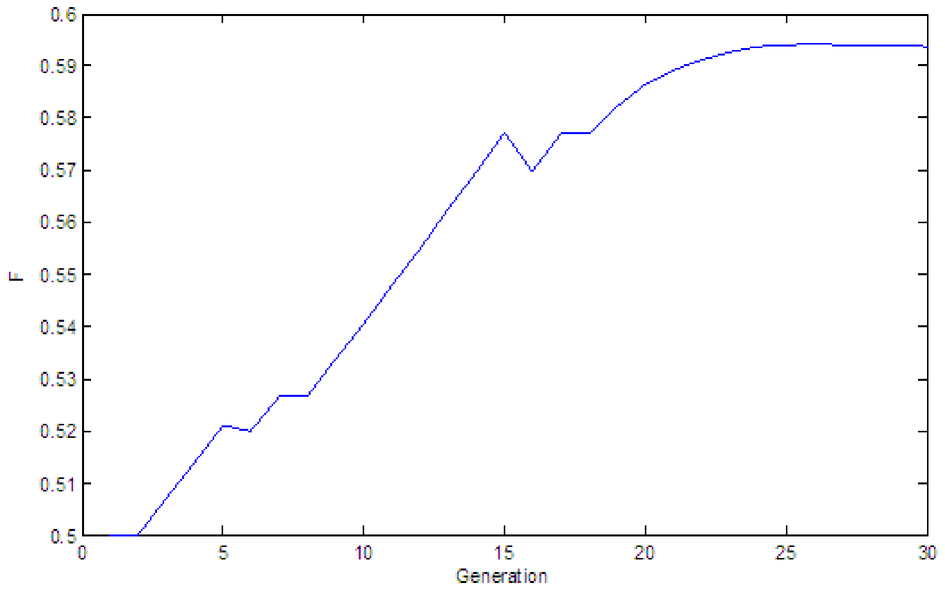


Figure 10. Simulation result of Example 4.2: trajectory of scaling factor F

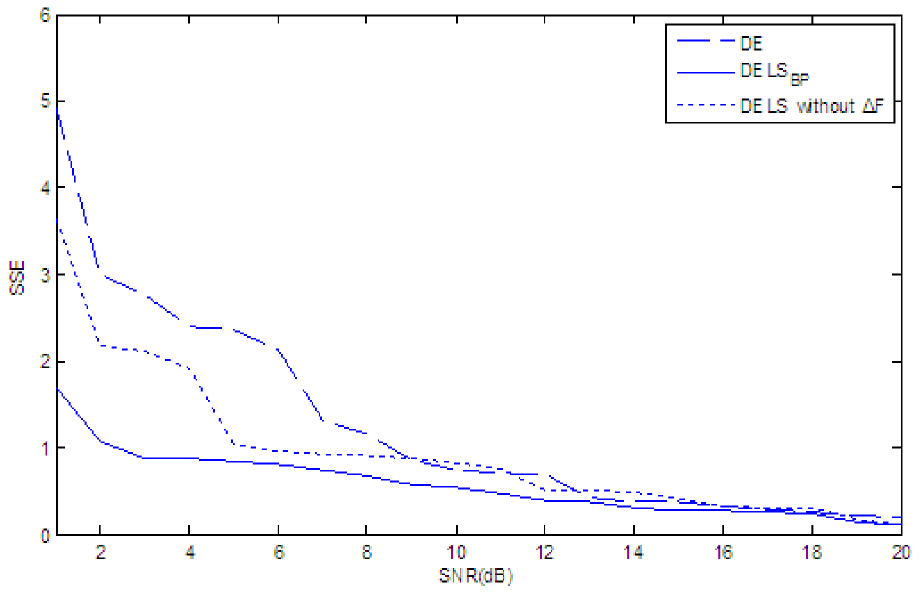


Figure 11. Performance comparisons: SSE of estimated coefficients versus SNR

noisy. This illustrated example demonstrates that the proposed DELS$_{\mathrm{BP}}$ algorithm is an effective, efficient, and robust optimization method for IIR filter coefficient estimation.

5. **Conclusion.** In this paper, we proposed a novel differential evolution algorithm DELS$_{\mathrm{BP}}$ using a dynamic strategy, gradient-descent-based local search, and a fuzzy self-adaptive scaling factor to enhance the performance of the traditional DE algorithm. The DELS$_{\mathrm{BP}}$ includes an instant-update strategy for the best vector, a back-propagation technique, and a self-adaptive scaling factor by using a fuzzy logic system. The technique can more easily determine the optimal solution in the search space and accelerate convergence performance. The performance of the DELS$_{\mathrm{BP}}$ algorithm has been demonstrated using

experimental results from a set of standard test functions and through the estimation of coefficients for an IIR filter with noise. From the experimental results, it can be observed that the proposed approach can accurately estimate the IIR filter coefficients even when the system is noisy. Furthermore, the illustrated example shows that the $\mathrm{DELS}_{\mathrm{BP}}$ has higher accuracy and convergence speed. These results demonstrate the efficacy and performance of the proposed approach.

Acknowledgment. The authors would like to thank the Associate Editor and the anonymous reviewers for their insightful comments and valuable suggestions. This work was partially supported by the National Science Council, Taiwan under contract No: NSC-97-2221-E-155-033-MY3.

REFERENCES

[1] N. Noman and H. Iba, Enhancing differential evolution performance with local search for high dimensional function optimization, *Proc. of 2005 Conference on Genetic and Evolutionary Computation*, pp.967-974, 2005.

[2] W. Gong, Z. Cai and L. Jiang, Enhancing the performance of differential evolution using orthogonal design method, *Applied Mathematics and Computation*, vol.206, pp.56-69, 2008.

[3] R. Sarker, M. Mohammadian and X. Yao, *Evolution Optimization*, Kluwer Academic Pub., 2002.

[4] S. M. Fard, A. Hamzeh and K. Ziarati, A new cooperative co-evolutionary multi-objective algorithm for function optimization, *International Journal of Innovative Computing, Information and Control*, vol.7, no.5(A), pp.2529-2542, 2011.

[5] Y. Guo, X. Cao and J. Zhang, Constraint handling based multiobjective evolutionary algorithm for aircraft landing scheduling, *International Journal of Innovative Computing, Information and Control*, vol.5, no.8, pp.2229-2238, 2009.

[6] T. Uno, H. Katagiri and K. Kato, An evolutionary multi-agent based search method for stackelberg solutions of bilevel facility location problems, *International Journal of Innovative Computing, Information and Control*, vol.4, no.5, pp.1033-1042, 2008.

[7] Z. Wang and M. Li, A hybrid coevolutionary algorithm for learning classification rules set, *ICIC Express Letters*, vol.4, no.2, pp.401-406, 2010.

[8] C. Liu, An evolutionary algorithm for solving dynamic nonlinear constrained optimization, *ICIC Express Letters*, vol.4, no.3(B), pp.1039-1044, 2010.

[9] R. Storn, K. Price and J. Lampinen, *Differential Evolution: A Practical Approach to Global Optimization*, Springer-Verlag, Berlin, 2005.

[10] R. Storn and K. Price, Differential evolution – A simple and efficient heuristic for global optimization over continuous spaces, *Journal of Global Optimization*, vol.11, pp.341-359, 1997.

[11] W. D. Chang, Parameter identification of Chen and Lu systems: A differential evolution approach, *Chaos, Solitons and Fractals*, vol.32, pp.1469-1476, 2007.

[12] W. D. Chang, Parameter identification of Rossler's chaotic system by an evolutionary algorithm, *Chaos, Solitons and Fractals*, vol.29, pp.1047-1053, 2006.

[13] Q. Yang, A comparative study of discrete differential evolution on binary constraint satisfaction problems, *Proc. of 2008 IEEE Congress on Evolutionary Computation*, Hong Kong, China, pp.330-335, 2008.

[14] A. Qing, Dynamic differential evolution strategy and applications in electromagnetic inverse scattering problems, *IEEE Trans. on Geoscience and Remote Sensing*, vol.44, no.1, pp.116-125, 2006.

[15] D. Xu, S. Li and F. Qian, An improved differential evolution algorithm and its application in reaction kinetic parameters estimation, *Proc. of 2007 International Conference on Intelligent Systems and Knowledge Engineering*, 2007.

[16] C. Deng, B. Zhao, A. Deng and R. Hu, New differential evolution algorithm with a second enhanced mutation operator, *Proc. of 2009 International Workshop on Intelligent Systems and Application*, Wuhan, China, pp.1-4, 2009.

[17] B. Subudhi, D. Jena and M. Gutpa, Memetic differential evolution trained neural networks for nonlinear system identification, *Proc. of 2008 IEEE Region 10 Colloquium and the 3rd International Conference on Industrial and Information Systems*, Kharagpur, India, pp.1-6, 2008.

[18] R. Storn and K. Price, *Home Page of Differential Evolution*, http://www.icsi.berkeley.edu/∼storn/code.html, 2003.

[19] J. Liu and J. Lampinen, A fuzzy adaptive differential evolution algorithm, *Soft Computing*, vol.9, pp.448-462, 2005.

[20] B. Alatas, E. Akin and A. Karci, MODENAR: Multi-objective differential evolution algorithm for mining numeric association rules, *Applied Soft Computing*, vol.8, no.1, pp.646-656, 2008.

[21] S. Das, A. Abraham and A. Konar, Automatic clustering using an improved differential evolution algorithm, *IEEE Trans. on Systems, Man, Cybernetics – Part A: Systems and Human*, vol.38, no.1, pp.218-237, 2008.

[22] C. H. Chen, C. J. Lin and C. T. Lin, Nonlinear system control using adaptive neural fuzzy networks based on a modified differential evolution, *IEEE Trans. on Systems, Man, and Cybernetics – Part B: Cybernetics*, vol.39, no.4, pp.459-473, 2009.

[23] X. Zhang, W. Chen, C. Dai and A. Guo, Self-adaptive differential evolution algorithm for reactive power optimization, *Proc. of 2008 IEEE Int. Con. on Natural Computation*, pp.560-564, 2008.

[24] Z. F. Wu, H. K. Huang, B. Yang and Y. Zhang, A modified differential evolution algorithm with self-adaptive control parameters, *Proc. of 2008 IEEE Int. Conf. on Intelligent System and Knowledge Engineering*, vol.1, pp.524-527, 2008.

[25] Z. Yang, K. Tang and X. Yao, Self-adaptive differential evolution with neighborhood search, *Proc. of 2008 IEEE Congress on Evolutionary Computation*, Hong Kong, China, pp.1110-1116, 2008.

[26] F. Neri and V. Tirronen, Scale factor local search in differential evolution, *Journal of Memetic Computing*, vol.1, no.2, pp.153-171, 2009.

[27] A. Caponio, A. V. Kononova and F. Neri, Differential evolution with scale factor local search for large scale problems, in *Computational Intelligence in Expensive Optimization Problems*, Y. Tenne, C.-K. Goh (eds.), Springer, 2009.

[28] C. J. Lin, C. F. Wu and C. Y. Lee, Design of a recurrent functional neural fuzzy network using modified differential evolution, *International Journal of Innovative Computing, Information and Control*, vol.7, no.4, pp.669-683, 2011.

[29] V. Vegh, G. K. Pierens and Q. M. Tieng, A variant of differential evolution for discrete optimization problems requiring mutually distinct variables, *International Journal of Innovative Computing, Information and Control*, vol.7, no.2, pp.897-914, 2011.

[30] F.-T. Lin, Application of differential evolution for fuzzy linear programming, *ICIC Express Letters*, vol.5, no.6, pp.1851-1856, 2011.

[31] F.-T. Lin, Performance comparison of differential evolution and genetic algorithms for the fuzzy transportation problem, *ICIC Express Letters*, vol.4, no.6(B), pp.2469-2474, 2010.

[32] J. Kushida, K. Oba and K. Kamei, Generation alternation model for reducing the number of fitness evaluations on differential evolution, *ICIC Express Letters*, vol.4, no.6(A), pp.2089-2095, 2010.

[33] L. Ljung and T. L. Soderstrom, *Theory and Practice of Recursive Identification*, MIT Press, Cambridge, MA, 1983.

[34] W. D. Chang, Coefficient estimation of IIR filter by a multiple crossover genetic algorithm, *Computers and Mathematics with Applications*, vol.51, no.9-10, pp.1437-1444, 2006.