Differential Evolution and Local Search based Monarch Butterfly Optimization Algorithm with Applications

Xingyue Cui , Zhe Chen , Fuliang Yin *

School of Information and Communication Engineering, Dalian University of Technology, Dalian 116023, China

E-mail: xiechoah@mail.dlut.edu.cn, zhechen@dlut.edu.cn, flyin@dlut.edu.cn

Received 23 April, 2018

Accepted 8 August, 2018

Abstract

Global optimization for nonlinear function is a challenging issue. In this paper, an improved monarch butterfly algorithm based on local search and differential evolution is proposed. Local search strategy is first embedded into original monarch butterfly optimization to enhance the searching capability. Then, differential evolution is incorporated with the aim of balancing the exploration and exploitation. To e-valuate the performance of proposed algorithm, some widely-used benchmark functions are tested, and the experiment results show its significant superiority compared with other state-of-the-art methods. In addition, the proposed algorithm is applied to PID tuning and FIR filter design, the superiority of solving practical problems is verified.

Keywords: Monarch butterfly optimization, local search strategy, differential evolution, PID tuning, FIR filter design.

1. Introduction

In many engineering fields, such as automatic control, communication network, signal processing etc, the complex nonlinear optimization is a universal issue. Generally, traditional optimization methods roughly called deterministic algorithms which use a gradient strategy to deal with nonliear problems and will produce the same local solutions when their iterations commence are started from the same initial value. Therefore, their searching capability is limited and may not obtain the optimal solutions when dealing with complex practical problems. In order to solve above issues, stochastic optimization algorithms are emerged. These algorithms usually begin from the initial set of variables and search for the feasible solution based on random walks. Thus, they are less likely to fall into local optimum and the searching capability enhanced to some extent.

Bionic optimization algorithm[1] inspired by some biological behaviors or evolution is a kind of typical stochastic algorithm and has been widely applied in complicated nonlinear global optimizations. Generally, bionic optimization algorithm can be derived into two categories: swarm intelligence algorithms (SIs)[2] and genetic evolution algorithms (EAs)[3]. The common swarm intelligence algorithms, including particle swarm algorithm (PSO)[4], glowworm swarm algorithm (GSA)[5] and biogeography-based optimization (BBO)[6], etc, have been widely used in some engineering fields. More recently emerging swarm intelligence algorithms are bat algorithm (BA)[7], cuckoo search (CS)[8], krill herd algorithm (KH)[9], flower pollination algorithm (FPA)[10],

* Corresponding author.

ant lion optimizer (ACO)[11] and social spider algorithm (SSA)[12]. Besides, the evolution algorithms, like genetic algorithm (GA)[13], differential evolution (DE)[14], artificial immune algorithm (AIA)[15], clonal selection algorithm (CSA)[16], are gradually developed as well.

In 2015, a new intelligent optimization algorithm, called monarch butterfly optimization algorithm (MBO)[17], was proposed, which is motivated by the monarch butterfly migration behavior in America. In MBO, the migration and adjusting operators are implemented to determine the search direction simultaneously. Thus, the MBO method is ideally suited for parallel processing and well capable of making trade-off between intensification and diversification.

In general, the MBO algorithm has acceptable convergence performance and certain degree of searching capability, but sometimes may fail to reach the optimal solution on some tests because of sticking in local optimum. Besides, how to balance the exploration and exploitation is the primary goal in design of an evolutionary algorithm. In the light of this rule, some strategies are introduced to improve its performance. Combining the MBO with greedy strategy and self-adaptive crossover operator, Wang et al.[18] proposed the greedy strategy and self-adaptive crossover operator (GCMBO) algorithm. Ghetas et al.[19] modified MBO into monarch butterfly harmony search (MBHS) by using a harmony search strategy. Although these two algorithms improve the performance of MBO, some issues are still exist, for instance, they are easy to trap into local optimal solution, which will lead to premature convergence, and the searching capabilities need to be strengthened as well.

To solve above problems, an improved monarch butterfly optimization based on differential evolution and local search algorithm is proposed. In the algorithm, with the aim of increasing the population diversity and obtaining the global optimum solution, local search strategy is embedded into the butterfly migration operator. Then, the differential evolution is incorporated into MBO for further enhancing the searching capability and balancing the exploration and exploitation. In order to evaluate the performance of proposed algorithm, some experiments are carrying out, including fourteen benchmark functions, PID tuning and fix-point FIR filter design.

The main contributions of this paper are summarised as follows: (1) An improved monarch butterfly algorithm based on local search and differential evolution (DE-LSMBO) is proposed; (2) A detail study on the superior performance of proposed algorithm is presented, including the selection of parameters and the comparison with other state-of-the-art methods; (3) The capabilities of proposed algorithm dealing with practical problems in PID tuning and FIR filter design are verified.

The residue sections of this paper are organized as follows. Section 2 reviews the principle and procedure of original MBO algorithm. Section 3 describes the proposed algorithm in detail. Section 4 consists of two parts: the experiments validating the superiority of DE-LSMBO and the performances dealing with PID tuning and FIR filter design. Finally, Section 5 concludes the paper.

2. Overview of the Monarch Butterfly Optimization and Differential Evolution

2.1. Monarch Butterfly Optimization (MBO)

In order to establish the model to well-addressing optimization problems, the migration behavior of monarch butterflies is idealized into the following rules[17]: (1) The whole monarch butterfly population is made up by the butterflies in Land 1 and Land 2; (2) Each offspring individual is only generated by the butterfly in Land 1 or Land 2; (3) The number of the population should be kept unchanged during the optimization process; (4) Elitism Strategy is set up, that is the fittest butterfly individuals can not be updated by any operator.

The basic MBO algorithm is mainly composed of butterfly migration operator and butterfly adjusting operator, and its flowchart is briefly presented in Fig. 1[17].

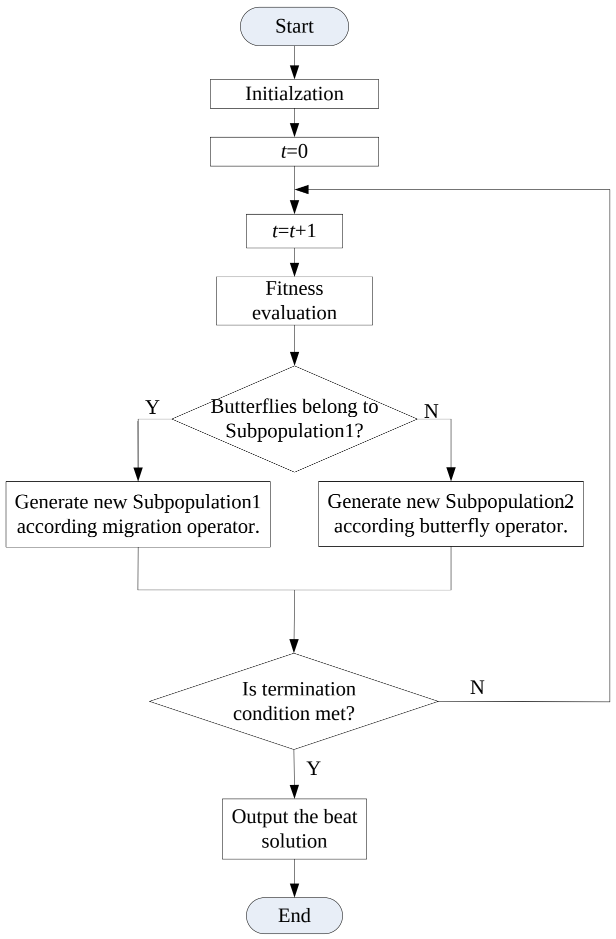
Start
Initialzation
t=0
t=t+1
Fitness evaluation
Butterflies belong to Subpopulation1?
Y
N
Generate new Subpopulation1 according migration operator.
Generate new Subpopulation2 according butterfly operator.
Is termination condition met?
N
Y
Output the beat solution
End

Fig. 1. Schematic flowchart of MBO method

2.1.1. *Butterfly migration operator*

The entire population of monarch butterfly is divided into two parts: Subpopulation1 (living in Land1) and Subpopulation2 (living in Land2). N, N_1 and N_2 denote the numbers of entire population, butterfly in Subpopulation1 and butterfly in Subpopulation2, respectively. Let p be the ratio of butterflies in Subpopulation1, $\lceil\rceil$ be the ceil function. Thus, $N_1 = \lceil pN \rceil$, $N_2 = N - N_1$. For all individuals in Subpopulation1, the migration operation can be expressed as

$$x_{i,k}^{t+1} = \begin{cases} x_{r_1,k}^t & r \leqslant p \\ x_{r_2,k}^t & r > p \end{cases} \tag{1}$$

where $x_{i,k}^{t+1}$ indicates the k-th element of x_i in the generation t+1, $x_{r_1,k}^t$ indicates the k-th element of x_{r_1} in the generation t and $x_{r_2,k}^t$ indicates the k-th element of x_{r_2} in the generation t; r_1 and r_2 are random numbers selected from Subpopulation1 and Subpopulation2, respectively, and r is calculated by

$$r = R_{rand} T_{peri} \tag{2}$$

where T_{peri} is the migration period and R_{rand} is the random number between $[0,1]$.

2.1.2. *Butterfly adjusting operator*

Butterfly adjusting is a critical process for all the individuals in Subpopulation2. The position of each individual can be updated as

$$x_{j,k}^{t+1} = \begin{cases} x_{best,k}^t & R_{rand1} \leqslant p \\ x_{r_3,k}^t & R_{rand1} > p \end{cases} \tag{3}$$

where $x_{j,k}^{t+1}$ indicates the k-th element of x_j in the generation t+1, $x_{best,k}^t$ denotes the k-th element of the global best individual x_{best} in the generation t, and $x_{r_3,k}^t$ indicates the k-th element of x_{r_3} in generation t; The r_3 is randomly selected from subpopulation2 and R_{rand1} is the random number over $[0,1]$.

Under this condition, in order to improve the searching efficiency, a weighting factor α and Levy flight[17] are introduced. Thus, Eq.(3) can be rewritten as

$$x_{j,k}^{t+1} = \begin{cases} x_{best,k}^t & R_{rand1} \leqslant p \\ x_{r_3,k}^t & R_{rand1} > p \\ x_{j,k}^{t+1} + \alpha(d_k - 0.5) & R_{rand1} > p \text{ and } R_{rand2} > R_{BAR} \end{cases} \tag{4}$$

where d is the walk step by computing Levy Flight, R_{BAR} is the butterfly adjusting rate and R_{rand2} is the random number over $[0,1]$ and

$$\alpha = S_{max}/t^2 \tag{5}$$

$$d = \text{Levy}(x_j^t) \tag{6}$$

where S_{max} is the max walk step that a individual can move in one step.

2.2. *Differential Evolution (DE)*

Generally, the original differential evolution consists of three main steps: mutation, crossover and selection. Individuals in differential evolution algorithm can be represented as a D-dimensional parameter vectors

$$x_i^t, i = 1, 2, \ldots, N \tag{7}$$

where t is the number of generation. According to [18], the process of differential evolution algorithm can be described as follows.

(1) Mutation

Mutation is to generate a mutant vector v_i corresponding to each target vector x_{n1}, and the most widely used strategy is

$$v_i^{t+1} = x_{n1}^t + F(x_{n2}^t - x_{n3}^t) \tag{8}$$

where v_i^{t+1} is the mutant vector in the $(t+1)$-th generation, $n1$, $n2$ and $n3 \in \{1, 2, \ldots, N\}$, are random and mutually different integers, $n1 \neq n2 \neq n3 \neq i$, and F is the constant factor over $[0, 2]$.

(2) Crossover

The trial vector $u_i^{t+1} = (u_{i,1}^{t+1}, u_{i,2}^{t+1}, \ldots, u_{i,D}^{t+1})$ is defined as

$$u_{i,j}^{t+1} = \begin{cases} v_{i,j}^{t+1} & R_{randDE} \leqslant C_R \text{ or } j = j_{rand} \\ x_{i,j}^{t+1} & R_{randDE} > C_R \text{ and } j \neq j_{rand} \end{cases} \tag{9}$$

where $j = 1, 2, \ldots, D$, R_{randDE} is the random number over $[0, 1]$, $C_R \in [0, 1]$ is the given crossover constant, j_{rand} is the randomly selected index, $j_{rand} \in [1, D]$; u_i^{t+1} obtains at least one parameter from v_i^{t+1}.

(3) Selection

The greedy strategy is used to choose better fitness offspring for the next generation.

$$x_i^{t+1} = \begin{cases} u_i^{t+1} & f(u_i^{t+1}) \leqslant f(x_i^t) \\ x_i^t & otherwise \end{cases} \tag{10}$$

where $f(\cdot)$ is the fitness function.

3. Monarch butterfly optimization based on differential evolution and local search strategy

The original MBO has the capability of searching the optimization solution, but some problems still need to be solved, for instance, MBO may trap into local optimal solution, which will lead to premature convergence. Besides, how to strengthen the searching capabilities of MBO is a significant issue. In order to solve these problems, two strategies are incorporated into the basic MBO method: local search strategy and differential evolution. For local search strategy, it brings new individuals into the existing population, which would increase the population diversity and improve the evolutionary efficiency. For differential evolution, it can maintain the population diversity while enhance the searching capability, which would overcome the premature convergence. Therefore, in proposed algorithm, local search strategy is embedded into butterfly migration operator and differential evolution is brought into MBO. More details for proposed algorithm are presented in next subsection.

3.1. *Updating migration operator with local search strategy*

In the whole population, each individual has its own feature, which is determined by different states. Thus, the feature information obtained from other individuals will present some new states, which are different from the original states. Based on this, in migration process, instead of updating migration individual $x_{i,k}^t$ with randomly selected $x_{r_1,k}^t$ or $x_{r_2,k}^t$, local search strategy is used to randomly search the new individual $x_{i,k}^{t+1}$ near the original selected $x_{selected,k}^t$, i.e.

$$x_{i,k}^{t+1} = x_{selected,k}^t + R_{rand3} * [x_{selected,k}^t - x_{i,k}^t] \tag{11}$$

where $x_{selected,k}^t$ is the k-th element of $x_{selected}$ (x_{r_1} or x_{r_2}) which is randomly selected from Subpopulation1 or Subpopulation2 in the generation t, R_{rand3} is the uniformly distributed random number over $[-1, 1]$. Thus, the new feature generated by migration operator has characteristic information from $x_{selected,k}^t$ and $x_{i,k}^t$.

Correspondingly, the modified migration operator can be briefly shown in Algorithm 1.

Algorithm 1 Modified migration operator

```
for $i = 1$ to $N_1$ do
    for $k = 1$ to $D$ do
        Generate the offspring individual of subpopulation1 by Eq.(11).
    end for $k$
end for $i$
```

Algorithm 2 Differential evolution and local search based monarch butterfly optimization

```
Step1: Initialization
    Initialize the population of monarch butterfly individuals randomly, the maximum generation $G_{max}$, migration
    ratio $p$, migration period $T_{peri}$, adjusting rate $R_{BAR}$ and max step $S_{max}$.
Step2:
    while $t < G_{max}$ do
        Sort all the butterfly individuals according to their fitness;
        Divide butterfly individuals into two subpopulations (Subpopulation1 and Subpopulation2);
        for $i = 1$ to $N_1$ do
            Generate the offspring individual of subpopulation1 by Algorithm1.
        end for $i$
        for $j = 1$ to $N_2$ do
            Generate the offspring individual of subpopulation1 by Eq.(4).
        end for $j$
        Recombine the two newly-generated subpopulations into one population.
        for $k = N$ do
            Generate new offsprings by mutation operator Eq.(8);
            Generate new offsprings by crossover operator Eq.(9);
            Update whole population by greedy strategy Eq.(10);
        end for $k$
        Evaluate the current butterfly population;
        t = t + 1.
    end while
Step3:
    Output the global best individual.
```

3.2. *New hybrid algorithm: Differential evolution and local search based monarch butterfly optimization*

As above-mentioned in Subsection 2.2, differential evolution has the characteristics of maintaining the diversity of population and well searching capability. Therefore, it can be integrated into MBO algorithm to accelerate the convergence rate while overcome the premature convergence.

Algorithm 2 describes the schematic of differential evolution and local search based monarch butterfly optimization (DE-LSMBO). Local search strategy (Algorithm 1) is embedded to update the individuals in Subpopulation1, and differential evolution is employed after recombining the two newly-generated subpopulations to update the whole population.

From Algorithm 1 and Algorithm 2, the computational complexity of DE-LSMBO algorithm can be obtained. It originates from two phases, LSMBO and DE. In the LSMBO phase, the computational cost of migration operation is $\mathcal{O}(N_1 D)$ and the computational cost of adjusting operation is $\mathcal{O}(N_2 D)$, thus, the computational cost of LSMBO is $\mathcal{O}(ND)$. In the DE phase, similar to LSMBO, its computational cost is $\mathcal{O}(ND)$. Therefore, the whole computational complexity of DE-LSMBO algorithm is $\mathcal{O}(ND)$. Here, it should be pointed out that all the computations refer to one iteration. For G_{max} iterations, the computational complexity of DE-LSMBO algorithm is $\mathcal{O}(NDG_{max})$.

The proposed DE-LSMBO has a strong searching capability and not easy to trap into local optimum, which can be well balanced the exploration and exploitation.

4. Simulation results and discussions

In order to evaluate the performance of the proposed method, several experiments are carried out. Benchmark functions are first employed to evaluate the performance of DE-LSMBO. Then, the proposed method is applied to dealing with PID tuning and fixed-point FIR filter design.

4.1. *Simulation setup*

In the following experiments, to obtain better performance of proposed algorithm, the choice of best parameters are necessary. Here, there are four core parameters in DE-LSMBO algorithm, i.e. population size, dimension of individual, migration ratio (p) and butterfly adjusting rate (R_{BAR}). The parameter selection experiments are based on Ackley function, the minimum value of Ackley function and the number of duplicate in final population are considered as criterions.

(1) Population size

For population size, generally, the larger population size, the more individuals can be regarded as feasible solutions. Thus, the DE-LSMBO algorithm could not be easy to trap into local optimal. But in turn, with the increment of population size, the computational complexity of the algorithm will lead to consume more time. So in this experiment, the population size is changed from 50 to 300 in steps of 50, other parameters p and R_{BAR} are set to 5/12, the dimension of individual is 20. Table 1 shows the result of population change and it is consistent with above theory.

(2) Migration ratio

Migration ratio p, which denotes the ratio of butterflies in Subpopulation1, is one of the main factors affecting the whole optimization results. Therefore, it is necessary to detect the effect of parameter p on the performance of DE-LSMBO algorithm. In this experiment, p is altered from 1/12 to 1/2 in steps of 1/12, the population size is set to 50, parameter R_{BAR} and the dimension of individual are set as 5/12 and 20, respectively.

Table 2 presents the optimization result with different values of p. From Table 2, it can be stated that although the optimization results are better when p is small, the duplicated individuals are emerged in the population, which implies that the obtained results may not be reliable enough. Hence, in the following experiments, 5/12 is chosen as the value of p because the number of duplicate is 0.

(3) Butterfly adjusting rate

In DE-LSMBO, butterfly adjusting rate is another crucial parameter which determines the mode to update the population in Subpopulation2. Thus, this

Table 1. Optimization results of DE-LSMBO with different population sizes.

Population size	50	100	150	200	250	300
Minimum value	5.8367	3.3978	2.5182	2.5788	2.6445	2.4910
Duplicate	0	0	0	0	0	0

Table 2. Optimization results of DE-LSMBO with different migration ratios p.

p	1/12	1/6	1/4	1/3	5/12	1/2
Minimum value	4.4643	5.6599	6.3282	5.7670	5.8367	7.2162
Duplicate	8	2	1	1	0	0

experiment is aimed to observe the performance of DE-LSMBO with different R_{BAR}s. Here, the population size is 50, the dimension of individual is 20, the migration ratio p is 5/12, and the value of R_{BAR}s is increased from 1/12 to 1/2 in steps of 1/12.

Table 3 shows the optimization result with different R_{BAR}s. From Table 3, it can be revealed that after 1/3, the minimum value of Ackley function is converged to 5.7. This indicates that under this condition, parameter R_{BAR} can take the value after 1/3, for instance, 5/12 or 1/2. Therefore, in the following experiments, 5/12 is chosen as the value of R_{BAR}. (4) Dimension of individual

Dimension of individual, in general, depends on different problems. Large dimension of individual usually indicates high complicated problems that is difficult to search the optimal solution. Therefore, in following experiments, the dimension of individual is changed based on different complexity problems.

To make a fair comparison, all implementations of metaheuristic algorithms are executed under the same condition as referred in 20. For DE-LSMBO, as discussed above, the parameters are set as follows: migration ratio p=5/12, butterfly adjusting rate R_{BAR}=5/12, migration period T_{peri}=1.2, and max step $S_{max} = 1.0$, the weighting factor F=0.5 and crossover constant C_R=0.5. For the original MBO, its parameters are set as same as MBO except it has no F and C_R. For the parameters used in other methods, their settings can be showed in Ref. 6 and Ref. 9.

4.2. Benchmark performance

In these experiments, the proposed DE-LSMBO is compared with other five population-based optimization algorithms, including PSO, DE, BBO, SGA and MBO. To search the results more quickly and efficiently, the population size and maximum generation for each algorithm are set to 50, and the dimension of each individual is set to 20. Table 4 shows fourteen high-dimensional benchmark functions which are employed to investigate the performance of algorithms, and the minimum value for each benchmark are considered as a criterion. Besides, an elitism strategy is also implemented by retaining the two best solutions for the next generation.

To obtain representative performances, 100 Monte Carlo simulations of each algorithm on each benchmark have been conducted. Table 5 and Table 6 show the results of six algorithms and the last row of every table presents the total number of benchmark functions on which the particular method has a better performance against others. The optimal solution for each test function is also highlighted. It should note that the normalizations in Table 5 and Table 6 are based on different scales, so values cannot be compared between the two tables.

From Table 5, it can be claimed that, on average, DE-LSMBO has a great advantage in finding the global minimum on eleven out of the fourteen benchmarks (F01-F04, F06-F07, F09-F13) and outperforms the original MBO algorithm on thirteen functions. BBO is the second effective algorithm, performing the best on two of the benchmarks (F05,

Table 3. Optimization results of DE-LSMBO with different butterfly adjusting rates R_{BAR}.

R_{BAR}	1/12	1/6	1/4	1/3	5/12	1/2
Minimum value	7.3160	7.0544	6.9915	5.7288	5.8367	5.6713
Duplicate	0	0	0	0	0	0

F14). Following is MBO, performing the best on F08.

Table 4. Benchmark Functions.

No.	Name		Name
F01	Ackley	F08	Schwefel1.2
F02	Griewank	F09	Schwefel2.21
F03	Penalty #1	F10	Schwefel2.22
F04	Penalty #2	F11	Schwefel2.26
F05	Quartic	F12	Sphere
F06	Rastrigin	F13	Step
F07	Rosenbrock	F14	Fletcher-Powell

Table 6 indicates that DE-LSMBO has the best performance in searching the function minimum on eleven out of fourteen benchmarks (F01, F03-F04, F05-F06, F08-F13) which significantly exceeds other optimization algorithms. BBO, SGA and MBO have the same performance that achieve to find the best solutions on one out of all functions (F02, F14, F08), respectively. Therefore, we can safely say that DE-LSMBO has absolute superiority than others in dealing with complicated nonlinear benchmarks under given conditions.

Furthermore, to validate the ascendancy of DE-LSMBO algorithm, an array of convergence curves are showed as well. Figs. 1-6 represent the comparisons of six intelligent optimization algorithms (PSO, DE, BBO, SGA, MBO, DE-LSMBO) on six out of fourteen benchmarks (F01, F06-F07, F10-F12), while other details are omitted due to space limitations.

From Fig. 2, it can be stated that even though all the algorithms have nearly the same initial values, DE-LSMBO has the fastest convergence rate and obtains the smallest optimal solution during the searching process. SGA and BBO rank two and three, their final fitness value are similar. Besides, PSO gets trapped in local extremum after ten generations and has the worst performance among these algorithms.

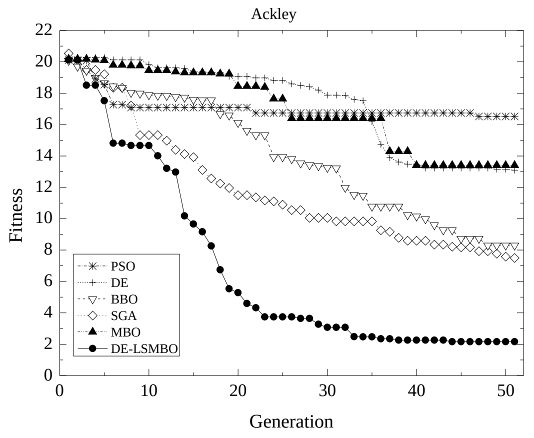

Fig. 2. Performance comparison of six methods on F01 Ackley function

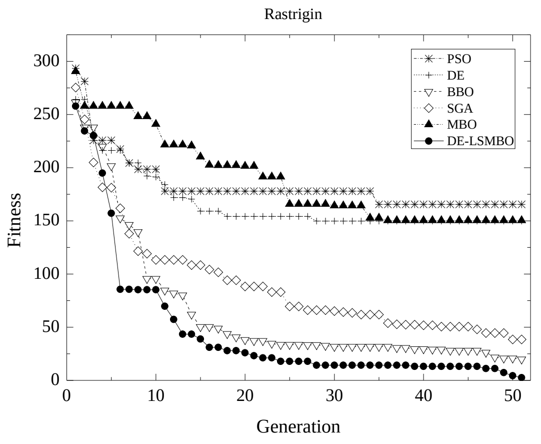

Fig. 3. Performance comparison of six methods on F06 Rastrigin function

In Fig. 3, the convergence curves may be roughly divided into two parts: 1) good performance, including DE-LSMBO, BBO, SGA; 2) poor performance, including DE, PSO and MBO. The former algo-

Table 5. Mean Normalized Optimization Results on Benchmark Functions.

Benchmark	PSO	DE	BBO	SGA	MBO	DE-LSMBO
F01	7.06	5.22	3.36	4.12	4.83	**1.00**
F02	30.45	4.03	1.69	2.72	1.69	**1.00**
F03	3.11E8	1.58E6	4.70E6	5.64E6	8.64E5	**1.00**
F04	1.58E6	1.60E4	1.68E3	3.17E4	3.85E2	**1.00**
F05	89.81	1.21	**1.00**	1.96	46.37	23.84
F06	31.04	19.97	3.21	7.56	19.15	**1.00**
F07	75.51	4.41	2.31	3.32	3.55	**1.00**
F08	8.66	4.87	1.41	1.56	**1.00**	6.25
F09	182.3	14.45	5.87	14.11	10.61	**1.00**
F10	8.76E14	28.06	5.11	10.14	43.01	**1.00**
F11	3.26	2.36	1.20	1.83	2.26	**1.00**
F12	63.03	8.27	4.91	6.32	5.41	**1.00**
F13	28.95	3.87	1.35	2.41	3.92	**1.00**
F14	14.99	3.10	**1.00**	1.76	7.01	7.61
Total	0	0	2	0	1	11

Table 6. Best Normalized Optimization Results on Benchmark Function.

Benchmark	PSO	DE	BBO	SGA	MBO	DE-LSMBO
F01	7.37	5.63	3.89	3.53	6.05	**1.00**
F02	13.54	3.62	**1.00**	1.37	2.80	1.17
F03	2.07E6	1.07E5	4.32E4	83.47	3.13E5	**1.00**
F04	1.69E6	1.77E5	3.60E4	2.14E3	2.66E4	**1.00**
F05	1.12E2	15.47	6.21	**1.00**	9.04E2	1.46E2
F06	61.76	52.99	7.54	18.08	56.81	**1.00**
F07	12.96	6.26	3.65	2.61	2.07	**1.00**
F08	5.83	3.93	1.37	1.09	**1.00**	3.72
F09	115.52	140.73	51.68	90.28	54.13	**1.00**
F10	45.85	20.51	6.54	9.74	21.54	**1.00**
F11	1.84	1.72	1.19	1.28	2.28	**1.00**
F12	48.83	13.55	11.35	5.29	12.86	**1.00**
F13	14.85	3.91	1.03	1.30	6.68	**1.00**
F14	7.23	3.30	1.32	**1.00**	5.91	1.61
Total	0	0	1	1	1	11

rithms have comparable convergent tendency and reliable solutions. By looking in detail, SGA's curve declines rapidly before the 3rd generations, but soon it is overtaken by DE-LSMBO. Finally, DE-LSMBO has better fitness than SGA and BBO. For algorithms with poor performance, all algorithms fail to find the global minimum within the limited iterations.

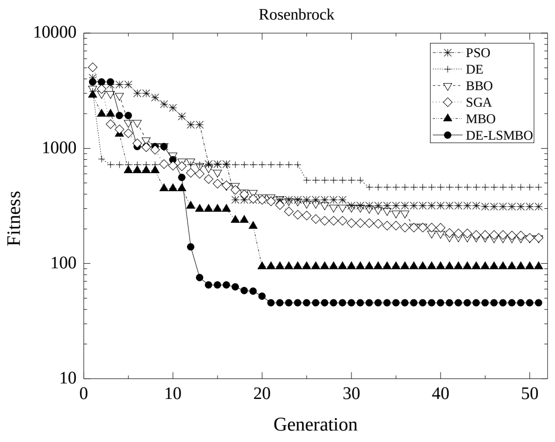

Fig. 4. Performance comparison of six methods on F07 Rosenbrock function

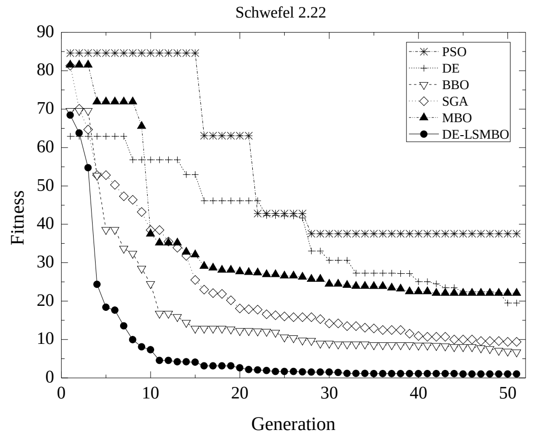

Fig. 5. Performance comparison of six methods on F10 Schwefel 2.22 function

In Fig. 4, at early ten generations, the convergent speed of DE-LSMBO is slower than MBO, DE and SGA, but from 10th generation to 15th one, it decreases sharply and obtains the best value at the 20th generation. Moreover, MBO performs the second best, while SGA and BBO have similar final fitness values, jointly rank the third place.

As shown Fig. 5, the curve of DE-LSMBO converges sharply and finds the best solution at about the 10th generation. During the optimization process, the convergent trajectory of DE-LSMBO is far below other algorithms and this imply that DE-LSMBO has an overwhelming advantage in dealing with F10 function. Further, SGA and BBO also display a good performance as well and have the second best and third rank.

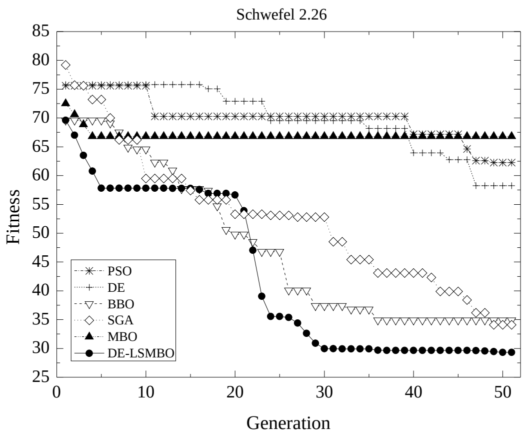

Fig. 6. Performance comparison of six methods on F11 Schwefel 2.26 function

As Fig. 6 is similar to Fig. 3, obviously, the performance of DE-LSMBO, BBO and SGA are superior to PSO, DE and MBO which may trap into local extremum. At first glance, it is clear that DE-LSMBO falls into the local optima from 5th generation to 20th one, but soon it escapes from local and decreases rapidly. Finally, DE-LSMBO outperforms other approaches, while SGA and BBO can be considered the second and third rank.

In Fig. 7, all algorithms have similar convergent trend. DE-LSMBO ranks first and finds the optimal solution at about 20th generation. In final, SGA, MBO, BBO and DE converge to the value that is very close to DE-LSMBOs, which means these algorithms have a well performance in Function 12, while PSO stucks in the local minimum.

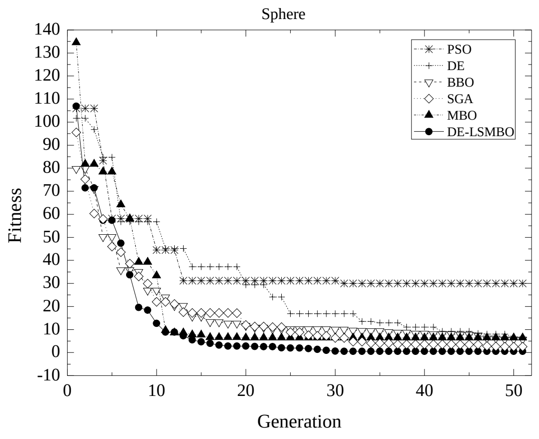

Fig. 7. Performance comparison of six methods on F12 Sphere function

From above analyses about the Figs. 2-7, it can be concluded that the performance of DE-LSMBO is superior to or at least competitive with other five algorithms. The propose algorithm has faster convergence speed and stronger optimization capability which could avoid most of the local optimum and approach the global optimal solution on majority of the test functions.

4.3. *PID parameter tuning*

The proportional-Integral-Derivative (PID) controller has been widely applied in the process industry due to its simple regular structure, strong robustness and satisfactory control performance. In general, a system controlled by PID controller requires tuning the controller's parameters to make the system stable and achieve the response fast enough. Therefore, many heuristic approaches such as stochastic processing, bionic algorithms, etc., have been used for tuning PID parameters. Ziegler and Nichols[21] proposed the tuning method based on the classical tuning rules, however, it does not always obtain optimal solutions. Some intelligent optimization algorithms emerge for tuning PID because of their remarkable capability of searching optimal solution. PSO algorithm has been verified in many applications[22,23]. Besides, Jacknoon[24] embedded ACO to adjust the linear quadratic regulator (LQR) and PID controller parameters. Sambariya[25] took a BA algorithm to improve the overall control system performance. Some other bionic optimization algorithms like Genetic Algorithm (GA) and BBO were also taken into account, and a survey of these algorithms can be found in Ref. 26-28. In this part, DE-LSMBO algorithm is used for PID tuning, and a closed-loop system is analyzed for comparing proposed algorithm with other three existing optimization algorithms.

Generally, the PID controller transfer function with three terms can be given as

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{\mathrm{d}e(t)}{\mathrm{d}t} \tag{12}$$

where $u(t)$ is the control signal, $e(t)$ is the error between the ideal output and the actual output, K_p, K_i and K_d are the vital control parameters which denote the proportional gain, integral gain and derivative gain, respectively. PID tuning means selecting a set of proper values K_p, K_i and K_d that provide control system desired response.

In this paper, the proposed algorithm is compared with three existing common methods, including Z-N, PSO and MBO, on tuning the PID parameters. All the parameters are set as benchmark function experiments except the population size is 500 and the dimension of each individual is 3, which represent the potential control parameters, i.e. $\mathbf{K}_{Ind}$=[K_p K_i K_d], $i = 1,2,\ldots,N_{popsize}$. Moreover, 50 Monte Carlo simulations of each algorithm have been conducted as well.

The S-domain transfer function in this experiment is

$$G(s) = \frac{500}{s^3 + 18s^2 + 200s} \tag{13}$$

Besides, to obtain a satisfactory tuning value within an appropriate optimization time, a performance criterion is necessary when designing PID. Generally, this criterion can guide different optimization algorithms to converge to the global optimal solution. Therefore, it should be carefully defined before the DE-LSMBO algorithm is executed. Here, the objective function is specified as the integral of time absolute error (ITAE)[25]

$$ITAE = \int_0^\infty t|e(t)|dt \tag{14}$$

In addition, the PID law in this experiment is considered as Eq.(12) and the range of control parameters K_p, K_i and K_d are [0, 20], [0, 50] and [0, 2], respectively. The corresponding results are summarized in Table 7, and the comparative performances of closed-loop system with Z-N, PSO, MBO, DE-LSMBO methods are shown in Fig. 8. From Table 7 and Fig. 8, it can be revealed that to a certain extent, all the methods could find optimal control parameters, providing good voltage step response. However, when looking in details, DE-LSMBO generates a relatively faster convergence rate and eliminates the steady state error by using less iteration times. Moreover, to further compare the performance of these methods and identify the most suitable methods, the statistics overshooting and *ITAE* are calculated and listed in Table 8. From Table 8, it is clear that DE-LSMBO has the smallest overshooting and the highest precision, that is to say, DE-LSMBO-PID controller can obtain the highest quality solution.

Table 7. Comparison of the evaluation value for different methods.

Method	K_p	K_i	K_d
Z-N	4.206	18.693	0.237
PSO	10.361	22.735	0.983
MBO	11.126	23.135	0.942
DE-LSMBO	10.635	24.089	1.203

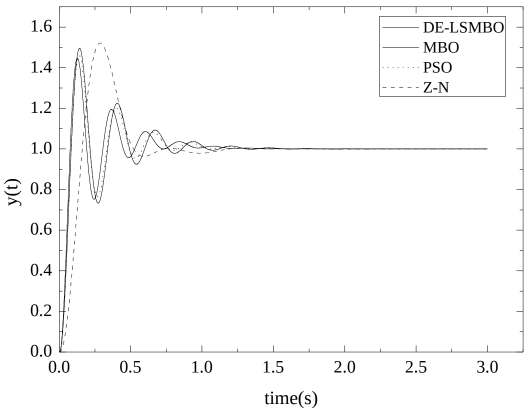

Fig. 8. Close-loop response with different PID tuning methods

Table 8. Simulation results of step response for different methods.

Method	Z-N	PSO	MBO	DE-LSMBO
Overshooting(%)	52.17	45.92	49.57	44.74
ITAE	0.048	0.034	0.040	0.030

4.4. Design of fixed-point FIR Filter

In this subsection, DE-LSMBO algorithm is applied in designing fixed-point FIR filter. It should be point out that this problem can be changed into 0-1 optimization. In general, fixed-point algorithm is used to reduce the hardware complexity and cost, and the most common scheme is to round a floating-point number to its nearest integer. Unfortunately, this will lead to a large quantization error. Therefore, bionic algorithms are emerged to further search the best fixed-point coefficients of FIR filter[29,30]. In this paper, the proposed algorithm is employed to deal with the fixed-point operation and the results are compared with five existing methods, including PSO, MBO and three truncation methods.

In order to show the adaptation of the DE-LSMBO algorithm, three condition changes are taken into account, including type, order and quantization scale. To make a fair comparison, all the parameters of bionic algorithms are set as same as benchmark experiment except population size of each bionic algorithms and maximum generation are changed to 100, and the dimension of each individual depends on the order of FIR filter. In addition, to evaluate the performance of different algorithms, root mean square error (RMSE) of frequency response between the ideal filter and fixed-point filter is considered as a evaluation criterion.

(1) Fixed-point FIR filter design for different types

In this experiment, consider four types of filter: lowpass, highpass, bandpass and bandstop. All the filter order is 400 (band-stop have to be 401 due to the fact that odd order symmetric FIR filter has a gain of zero at Nyquist frequency), the sampling frequency is 8 KHz, quantization scale is 2^{10}(Q10) and the dimension of each individual is 200. More specifically, the pass band ranges for lowpass filter, highpass filter, bandpass filter and bandstop filter are

Table 9. RMSE results of four types FIR filter for six algorithm-s.

Filter Type	Ceil	Floor	Round	PSO	MBO	DE-LSMBO
Lowpass	4.8250	4.7993	4.8980	4.5362	4.5807	4.4467
Highpass	0.3471	0.3471	0.3404	0.5680	0.4187	0.3675
Bandpass	0.3751	0.3706	0.3557	0.5080	0.3687	0.3277
Bandstop	0.4512	0.4531	0.4610	0.5503	0.3972	0.3657

0-50Hz, 55-4000Hz, 300-3400Hz and 300-340Hz, respectively.

The RMSE results of four filters can be observed in Table 9 and the frequency response of the FIR bandpass filter is given as an example in Fig. 9. From Table 9, it can be claimed that in general, the traditional truncation methods (Floor, Ceil and Round) provide similar values that are almost higher than bionic algorithms except PSO. By looking in details, the proposed DE-LSMBO obtains the least RSME values compared with other five algorithms in three type of filters, including lowpass, bandpass and stoppass. For highpass filter, the performance of DE-LSMBO is little weaker than traditional methods, but it still outperforms MBO and PSO.

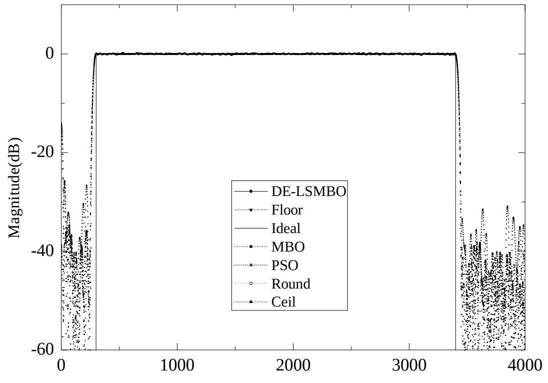

Fig. 9. Frequency response of bandpass FIR filter

(2) Fixed-point FIR filter design for different orders

In order to verify the advantage of DE-LSMBO algorithm in processing complex problems, the fixed-point FIR filter experiment with different order numbers is designed. Here, bandpass FIR filter is chosen and its order is increased from 300 to 800 in steps of 100. Thus, the dimension of each individual is changed from 150 to 400 in steps of 50. Besides, the sampling frequency and quantization value are set as 8KHz and 2^{10}, respectively.

Fig. 10 illustrates the RMSE curves of six algorithms vs order change of bandpass FIR filters. From Fig. 10, it is clear that PSO has the worst performance, while the proposed DE-LSMBO leads to the best performance. More specifically, in order 300, 400, 500, 700 and 800, DE-LSMBO algorithm displays the smallest RSME values obviously, but for order 600, the value obtained by DE-LSMBO is similar to those by traditional methods (Floor and Ceil). Moreover, in this experiment, the whole traditional truncation methods perform the second best and MBO ranks third.

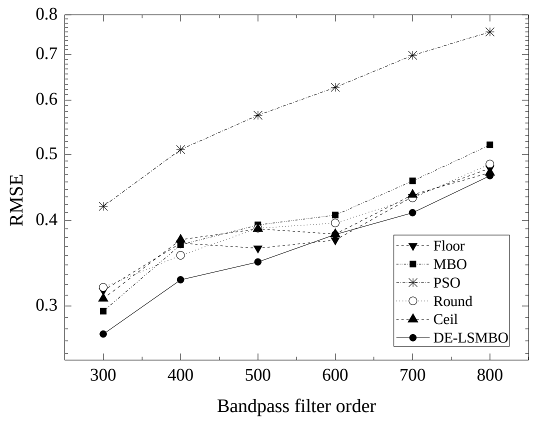

Fig. 10. RMSE results on FIR bandpass filter order change

(3) Fixed-point FIR filter design for different quantization scales

This experiment is aimed to detect the effect of quantization scales on the performance of DE-

LSMBO algorithm. Hence, the quantization scales are altered from 2^{10}(Q10) to 2^{15}(Q15) . Besides, the filter type is bandpass, the order of filter is 400 and the sampling frequency is 8KHz.

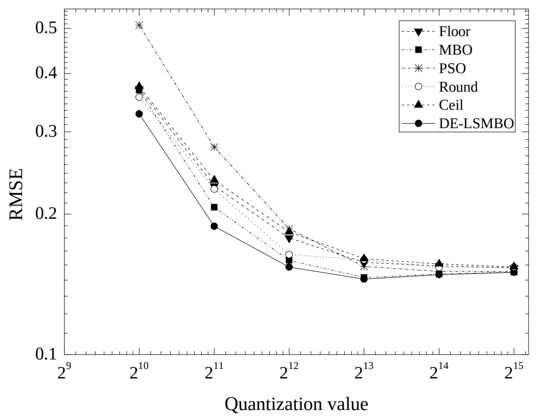

Fig. 11. RMSE results on quantization scale change for FIR bandpass filter

Fig. 11 presents the outputs on quantization scale change for bandpass FIR filters. From Fig. 11, it can be revealed that with the increase of quantization scales, all RMSE curves of six algorithms show downward tendency. By carefully looking at the figure, when quantization scale is set to 2^{10}, 2^{11} and 2^{12}, DE-LSMBO outperfoms other algorithms, while after 2^{13}, the RMSE values of MBO catch up with DE-LSMBO's, and all the RMSE of algorithms are gradually converged to 0.15. In addition, MBO ranks second place and the traditional truncation methods rank third.

From above analyses, it can be concluded that the performance of DE-LSMBO is comparable or even superior than other five algorithms. The proposed algorithm presents good adaptability when dealing with fix-point FIR design under different conditions, which implys that it has strong capability of disposing complex practical problems.

5. Conclusion

In this paper, a local search strategy and differential evolution based monarch butterfly optimization is proposed. In the DE-LSMBO, the local search strategy is embedded to prevent the algorithm from sticking into a local optimum and the differential evolution is incorporated to balance the exploration and exploitation. In order to evaluate the performance of proposed algorithm, 14 common benchmark functions that present different optimization problems are tested. The simulation results show that the proposed DE-LSMBO can provide competitive results and outperforms other algorithms in the majority of test functions. To further evaluate the performance of proposed algorithm in real applications, the DE-LSMBO is applied to tune PID controllers and design fixed-point FIR filters. For the PID problem, with the optimized results obtained by DE-LSMBO algorithm, the output of the close-loop PID system is stable with low overshoot level and fast step response. For the FIR design, the proposed DE-LSMBO algorithm presents good adaptability under different conditions. Therefore, it may be summarized that the proposed DE-LSMBO algorithm has capability of handling different complicated optimization problems.

Acknowledgments

This work was supported by (1) National Natural Science Foundation of China (Nos. 61771091); (2) National High Technology Research and Development Program (863 Program) of China (No. 2015AA016306); (3) Natural Science Foundation of Liaoning Province (No. 20170540159), and (4) Fundamental Research Funds for the Central Universities of China (Nos. DUT17LAB04).

References

1. E. J. Steele, *Somatic Selection and Adaptive Evolution*, (Springer US, NewYork, 1979).
2. X.S. Yang, S. Deb, S. Fong, X. He, Y.X. Zhao, From swarm intelligence to metaheuristics nature-inspired optimization algorithms, *Computer* 49(9) (2016) 52-59, http://dx.doi.org/10.1109/MC.2016.292.
3. J. Galletly, Evolutionary algorithms in theory and practice : evolution strategies, evolutionary programming, genetic algorithms, *Complexity* 2 (8) (1996) 26-27.
4. Kennedy J, Eberhart R, Particle swarm optimization, in *IEEE Int. Conf. on Neural Networks*, 1995, pp.

1942-1948.

5. K. N. Krishnanand, D. Ghose, Detection of multiple source locations using a glowworm metaphor with applications to collective robotics, in *Proc. IEEE Swarm Intelligence Symposium*, 2005, pp. 84-91.
6. D. Simon, Biogeography-based optimization, *IEEE Trans Evolut. Comput.* 12 (6) (2008) 702C713, http://dx.doi.org/10.1109/TEVC.2008.919004.
7. X.S. Yang, A New Metaheuristic Bat-Inspired Algorithm, in *International Workshop on Nature Inspired Cooperative Strategies for Optimization (NICSO)*, (Springer Berlin Heidelberg, 2010), pp. 65-74.
8. X.S. Yang, S. Deb, Cuckoo search: recent advances and applications, *Neural Comput. Appl.* 24 (1) (2014) 169-174, http://dx.doi.org/doi:10.1007/s00521-013-1367-1.
9. G.G. Wang, L.H. Guo, A.H. Gandomi, Chaotic krill herd algorithm, *Inf. Sci.* 274 (2014) 17-34, http://dx.doi.org/10.1016/j.ins.2014.02.123.
10. X.S. Yang, M. Karamanoglu, X.S. He, Flower pollination algorithm: A novel approach for multiobjective optimization, *Eng. Optimiz.* 46 (9) (2014) 1222-1237, http://dx.doi.org/10.1080/0305215X.2013.832237.
11. M. Seyedali, The ant lion optimizer, *Adv. Eng. Softw.* 83 (2015) 80-98, http://dx.doi.org/10.1016/j.advengsoft.2015.01.010.
12. J.J.Q. Yu, V.O.K. Li, A social spider algorithm for global optimization, *Appl. Soft Comput.* 30 (C) (2015) 614-627, http://dx.doi.org/10.1016/j.asoc.2015.02.014.
13. D.E. Goldberg, *Genetic algorithms in search, optimization and machine learning*, (Addison-Wesley Longman, NewYork, 1989).
14. R.Storn and K. Price, Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces, *J. Global Optim.* 11 (4) (1997) 341-359.
15. A. Watkins, J. Timmis, L. Boggess, Artificial immune recognition system (AIRS): An immune-inspired supervised learning algorithm, *Genet. Program Evol. M.* 5 (3) (2004) 291-317, http://dx.doi.org/10.1023/B:GENP.0000030197.83685.94.
16. L.N.D. Castro, F.J.V. Zuben, The Clonal Selection Algorithm with Engineering Applications, in *Int. Conf. on GECCO* (GECCO-Workshop, 2000), pp. 36-37.
17. G.G. Wang, S. Deb, Z.H. Cui, Monarch butterfly optimization, *Neural Comput. Appl.* (2015) 1-20, http://dx.doi.org/10.1007/s00521-015-1923-y.
18. G.G. Wang, X. Zhao, S. Deb, A Novel Monarch Butterfly Optimization with Greedy Strategy and Self-Adaptive, in *Proc. 2nd Int. Conf. on Soft Computing and Machine Intelligence (ISCMI)*, 2015, pp. 45-50.
19. M. Ghetas, C.H. Yong, P. Sumari, Harmony-based monarch butterfly optimization algorithm, in *IEEE Int. Conf. on Control System, Computing and Engineering (ICCSCE)*, 2015, pp. 156-161.
20. G. Wang, L. Guo, H. Wang, H. Duan, L. Liu, J. Li, Incorporating mutation scheme into krill herd algorithm for global numerical optimization, *Neural Comput. Appl.* 24 (3-4) (2014) 853C871, http://dx.doi.org/10.1007/s00521-012-1304-8.
21. J.G. Ziegler, N.B. Nichols, Optimal settings for automatic controllers, *Trans. ASME* 64 (1942) 759-768.
22. H. Freire, P.B.M. Oliveira, E.J.S. Pires, From single to many-objective PID controller design using particle swarm optimization, *Int. J. Control Autom.* 15 (2) (2017) 918-932, http://dx.doi.org/10.1007/s12555-015-0271-0.
23. R. Patel, V. Kumar, Artificial neuro fuzzy logic PID controller based on BF-PSO algorithm, *Procedia Computer Science* 54 (2015) 463-471, http://dx.doi.org/10.1016/j.procs.2015.06.053.
24. A. Jacknoon, M. A. Abido, Ant Colony based LQR and PID tuned parameters for controlling Inverted Pendulum, in *Int. Conf. on Communication, Control, Computing and Electronics Engineering (ICCCCEE)*, 2017, pp. 1-8.
25. D.K. Sambariya, R. Gupta, Effective PID-PSS design using Bat algorithm for SMIB power system, in *Proc. 6th IEEE Int. Conf. on Power Systems (ICPS)*, 2016, pp. 1-6.
26. M.J. Blondin, P. Sicard, Statistical convergence analysis of ACO-NM for PID controller tuning, in *IEEE Int. Conf. on Industrial Technology (ICIT)*, 2015, pp. 487-492.
27. M.K. Debnath, S. Tripathy, R.K. Mallick, Hybrid DE-PSO optimized fuzzy-PID controller for Automatic Generation Control of a two area multi-unit power system, in *IEEE Uttar Pradesh Section Int. Conf. on Electrical, Computer and Electronics Engineering (UPCON)*, 2016, pp. 537-542.
28. A. Rahman, L.C. Saikia, N. Sinha, Maiden application of hybrid pattern search-biogeography based optimisation technique in automatic generation control of a multi-area system incorporating interline power flow controller, *IET Gener. Transm. Dis.* 10 (7) (2016) 1654-1662, http://dx.doi.org/10.1049/iet-gtd.2015.0945.
29. A.K. Dwivedi , S. Ghosh , N.D. Londhe, Modified artificial bee colony optimisation based FIR filter design with experimental validation using field-programmable gate array, *IET Signal Process.* 10 (8) (2016) 955-964, http://dx.doi.org/10.1049/iet-spr.2015.0214.
30. S. Mondal, D. Chakraborty, R. Kar, D. Mandal, S.P. Ghoshal, Novel particle swarm optimization for high pass FIR filter design, in *IEEE Symposium on Humanities, Science and Engineering Research*, 2012, pp. 413-418.