Self-adaptive Differential Evolution Algorithm for Numerical Optimization

A. K. Qin
School of Electrical and Electronic Engineering,
Nanyang Technological University
50 Nanyang Ave., Singapore 639798
qinkai@pmail.ntu.edu.sg

P. N. Suganthan
School of Electrical and Electronic Engineering,
Nanyang Technological University
50 Nanyang Ave., Singapore 639798
epnsugan@ntu.edu.sg

Abstract- In this paper, we propose a novel Self-adaptive Differential Evolution algorithm (SaDE), where the choice of learning strategy and the two control parameters F and CR are not required to be pre-specified. During evolution, the suitable learning strategy and parameter settings are gradually self-adapted according to the learning experience. The performance of the SaDE is reported on the set of 25 benchmark functions provided by CEC2005 special session on real parameter optimization

1 Introduction

Differential evolution (DE) algorithm, proposed by Storn and Price [1], is a simple but powerful population-based stochastic search technique for solving global optimization problems. Its effectiveness and efficiency has been successfully demonstrated in many application fields such as pattern recognition [1], communication [2] and mechanical engineering [3]. However, the control parameters and learning strategies involved in DE are highly dependent on the problems under consideration. For a specific task, we may have to spend a huge amount of time to try through various strategies and fine-tune the corresponding parameters. This dilemma motivates us to develop a Self-adaptive DE algorithm (SaDE) to solve general problems more efficiently.

In the proposed SaDE algorithm, two DE's learning strategies are selected as candidates due to their good performance on problems with different characteristics. These two learning strategies are chosen to be applied to individuals in the current population with probability proportional to their previous success rates to generate potentially good new solutions. Two out of three critical parameters associated with the original DE algorithm namely, CR and F are adaptively changed instead of taking fixed values to deal with different classes of problems. Another critical parameter of DE, the population size NP remains a user-specified variable to tackle problems with different complexity.

2 Differential Evolution Algorithm

The original DE algorithm is described in detail as follows: Let $\mathbf{S} \subset \Re^n$ be the n-dimensional search space of the problem under consideration. The DE evolves a population of NP n-dimensional individual vectors, i.e. solution candidates, $\mathbf{X}_i = (x_{i1},\ldots,x_{in}) \in \mathbf{S}$, $i = 1,\ldots,NP$, from one generation to the next. The initial population should ideally cover the entire parameter space by randomly distributing each parameter of an individual vector with uniform distribution between the prescribed upper and lower parameter bounds x_j^u and x_j^l.

At each generation G, DE employs the mutation and crossover operations to produce a trial vector $\mathbf{U}_{i,G}$ for each individual vector $\mathbf{X}_{i,G}$, also called target vector, in the current population.

a) Mutation operation

For each target vector $\mathbf{X}_{i,G}$ at generation G, an associated mutant vector $\mathbf{V}_{i,G} = \{v_{1i,G}, v_{2i,G},\ldots,v_{ni,G}\}$ can usually be generated by using one of the following 5 strategies as shown in the online availbe codes [] :

"DE/rand/1": $\mathbf{V}_{i,G} = \mathbf{X}_{r_1,G} + F \cdot (\mathbf{X}_{r_2,G} - \mathbf{X}_{r_3,G})$

"DE/best/1": $\mathbf{V}_{i,G} = \mathbf{X}_{best,G} + F \cdot (\mathbf{X}_{r_1,G} - \mathbf{X}_{r_2,G})$

"DE/current to best/1":

$$\mathbf{V}_{i,G} = \mathbf{X}_{i,G} + F \cdot (\mathbf{X}_{best,G} - \mathbf{X}_{i,G}) + F \cdot (\mathbf{X}_{r_1,G} - \mathbf{X}_{r_2,G})$$

"DE/best/2":

$$\mathbf{V}_{i,G} = \mathbf{X}_{best,G} + F \cdot (\mathbf{X}_{r_1,G} - \mathbf{X}_{r_2,G}) + F \cdot (\mathbf{X}_{r_3,G} - \mathbf{X}_{r_4,G})$$

"DE/rand/2":

$$\mathbf{V}_{i,G} = \mathbf{X}_{r_1,G} + F \cdot (\mathbf{X}_{r_2,G} - \mathbf{X}_{r_3,G}) + F \cdot (\mathbf{X}_{r_4,G} - \mathbf{X}_{r_5,G})$$

where indices r_1, r_2, r_3, r_4, r_5 are random and mutually different integers generated in the range $[1, NP]$, which should also be different from the current trial vector's index i. F is a factor in $[0,2]$ for scaling differential vectors and $\mathbf{X}_{best,G}$ is the individual vector with best fitness value in the population at generation G.

b) Crossover operation

After the mutation phase, the "binominal" crossover operation is applied to each pair of the generated mutant vector $\mathbf{V}_{i,G}$ and its corresponding target vector $\mathbf{X}_{i,G}$ to generate a trial vector: $\mathbf{U}_{i,G} = (u_{1i,G}, u_{2i,G}, \ldots, u_{ni,G})$.

$$u_{j,i,G} = \begin{cases} v_{j,i,G}, & \text{if } (rand_j[0,1] \le CR) \text{ or } (j = j_{rand}) \\ x_{j,i,G}, & \text{otherwise} \end{cases}, \quad j = 1, 2, \ldots, n$$

where CR is a user-specified crossover constant in the range $[0,1)$ and j_{rand} is a randomly chosen integer in the range $[1,NP]$ to ensure that the trial vector $\mathbf{U}_{i,G}$ will differ from its corresponding target vector $\mathbf{X}_{i,G}$ by at least one parameter.

c) Selection operation

If the values of some parameters of a newly generated trial vector exceed the corresponding upper and lower bounds, we randomly and uniformly reinitialize it within the search range. Then the fitness values of all trial vectors are evaluated. After that, a selection operation is performed. The fitness value of each trial vector $f(\mathbf{U}_{i,G})$ is compared to that of its corresponding target vector $f(\mathbf{X}_{i,G})$ in the current population. If the trial vector has smaller or equal fitness value (for minimization problem) than the corresponding target vector, the trial vector will replace the target vector and enter the population of the next generation. Otherwise, the target vector will remain in the population for the next generation. The operation is expressed as follows:

$$\mathbf{X}_{i,G+1} = \begin{cases} \mathbf{U}_{i,G}, & \text{if } f(\mathbf{U}_{i,G}) < f(\mathbf{X}_{i,G}) \\ \mathbf{X}_{i,G} & \text{otherwise} \end{cases}$$

The above 3 steps are repeated generation after generation until some specific stopping criteria are satisfied.

3 SaDE: Strategy and Parameter Adaptation

To achieve good performance on a specific problem by using the original DE algorithm, we need to try all available (usually 5) learning strategies in the mutation phase and fine-tune the corresponding critical control parameters CR, F and NP. Many literatures [4], [6] have pointed out that the performance of the original DE algorithm is highly dependent on the strategies and parameter settings. Although we may find the most suitable strategy and the corresponding control parameters for a specific problem, it may require a huge amount of computation time. Also, during different evolution stages, different strategies and corresponding parameter settings with different global and local search capability might be preferred. Therefore, we attempt to develop a new DE algorithm that can automatically adapt the learning strategies and the parameters settings during evolution. Some related works on parameter or strategy adapation in evolutionary algorithms have been done in literatures [7], [8].

The idea behind our proposed learning strategy adaptation is to probabilistically select one out of several available learning strategies and apply to the current population. Hence, we should have several candidate learning strategies available to be chosen and also we need to develop a procedure to determine the probability of applying each learning strategy. In our current implementation, we select two learning strategies as candidates: "rand/1/bin" and "current to best/2/bin" that are respectively expressed as:

$$\mathbf{V}_{i,G} = \mathbf{X}_{r_1,G} + F\cdot(\mathbf{X}_{r_2,G} - \mathbf{X}_{r_3,G})$$
$$\mathbf{V}_{i,G} = \mathbf{X}_{i,G} + F\cdot(\mathbf{X}_{best,G} - \mathbf{X}_{i,G}) + F\cdot(\mathbf{X}_{r_1,G} - \mathbf{X}_{r_2,G})$$

The reason for our choice is that these two strategies have been commonly used in many DE literatures [] and reported to perform well on problems with distinct characteristics. Among them, "rand/1/bin" strategy usually demonstrates good diversity while the "current to best/2/bin" strategy shows good convergence property, which we also observe in our trial experiments.

Since here we have two candidate strategies, assuming that the probability of applying strategy "rand/1/bin" to each individual in the current population is p_1, the probability of applying another strategy should be $p_2 = 1 - p_1$. The initial probabilities are set to be equal 0.5, i.e., $p_1 = p_2 = 0.5$. Therefore, both strategies have equal probability to be applied to each individual in the initial population. For the population of size NP, we can randomly generate a vector of size NP with uniform distribution in the range $[0,1]$ for each element. If the j^{th} element value of the vector is smaller than or equal to p_1, the strategy "rand/1/bin" will be applied to the j^{th} individual in the current population. Otherwise the strategy "current to best/2/bin" will be applied. After evaluation of all newly generated trial vectors, the number of trial vectors successfully entering the next generation while generated by the strategy "rand/1/bin" and the strategy "current to best/2/bin" are recorded as ns_1 and ns_2, respectively, and the numbers of trial vectors discarded while generated by the strategy "rand/1/bin" and the strategy "current to best/2/bin" are recorded as nf_1 and nf_2. Those two numbers are accumulated within a specified number of generations (50 in our experiments), called the "learning period". Then, the probability of p_1 is updated as:

$$p_1 = \frac{ns1\cdot(ns2+nf2)}{ns2\cdot(ns1+nf1)+ns1\cdot(ns2+nf2)}, \quad p_2 = 1 - p_1$$

The above expression represents the percentage of the success rate of trial vectors generated by strategy "rand/1/bin" in the summation of it and the successful rate of trial vectors generated by strategy "current to best/2/bin" during the learnng period. Therefore, the probability of applying those two strategies is updated, after the learning period. Also we will reset all the counters ns_1, ns_2, nf_1 and nf_2 once updating to avoid the possible side-effect accumulated in the previous learning stage. This adaptation procedure can gradually evolve the most suitable learning strategy at different learning stages for the problem under consideration.

In the original DE, the 3 critical control parameters ***CR***, ***F*** and ***NP*** are closely related to the problem under consideration. Here, we keep ***NP*** as a user-specified value as in the original DE, so as to deal with problems with different dimensionalities. Between the two parameters ***CR*** and ***F***, ***CR*** is much more sensitive to the problem's property and complexity such as the multi-modality, while ***F*** is more related to the convergence speed. According to our initial experiments, the choice of ***F*** has a larger flexibility, although most of the time the values between $(0,1]$ are preferred. Here, we consider allowing ***F*** to take different random values in the range $(0,2]$ with normal distributions of mean 0.5 and standard deviation 0.3 for different individuals in the current population. This scheme can keep both local (with samll ***F*** values) and global (with large ***F*** values) search ability to generate the potential good mutant vector throughout the evolution process. The control parameter ***CR***, plays an essential role in the original DE algorithm. The proper choice of ***CR*** may lead to good performance under several learning strategies while a wrong choice may result in performance deterioration under any learning strategy. Also, the good ***CR*** parameter value usually falls within a small range, with which the algorithm can perform consistently well on a complex problem. Therefore, we consider accumulating the previous learning experience within a certain generation interval so as to dynamically adapt the value of ***CR*** to a suitable range. We assume ***CR*** normally distributed in a range with mean ***CRm*** and standard deviation 0.1. Initially, ***CRm*** is set at 0.5 and different ***CR*** values conforming this normal distribution are generated for each individual in the current population. These ***CR*** values for all individuals remain for several generations (5 in our experiments) and then a new set of ***CR*** values is generated under the same normal distribution. During every generation, the ***CR*** values associated with trial vectors successfully entering the next generation are recorded. After a specified number of generations (25 in our experiments), ***CR*** has been changed for several times (25/5=5 times in our experiments) under the same normal distribution with center ***CRm*** and standard deviation 0.1, and we recalculate the mean of normal distribution of ***CR*** according to all the recorded ***CR*** values corresponding to successful trial vectors during this period. With this new normal distribution's mean and the standard devidation 0.1, we repeat the above procedure. As a result, the proper ***CR*** value range for the current problem can be learned to suit the particular problem and. Note that we will empty the record of the successful ***CR*** values once we recalculate the normal distribution mean to avoid the possible inappropriate long-term accumulation effects.

We introduce the above learning strategy and parameter adaptation schemes into the original DE algorithm and develop a new Self-adaptive Differential Evolution algorithm (SaDE). The SaDE does not require the choice of a certain learning strategy and the setting of specific values to critical control parameters ***CR*** and ***F***. The learning strategy and control parameter ***CR***, which are highly dependent on the problem's characteristic and complexity, are self-adapted by using the previous learning experience. Therefore, the SaDE algorithm can demonstrate consistently good performance on problems with different properties, such as unimodal and multimodal problems. The influence on the performance of SaDE by the number of generations during which previous learning information is collected is not significant. We further investigate this now.

To speed up the convergence of the SaDE algorithm, we apply the local search procedure after a specified number of generations which is 200 generations in our experiments, on 5% individuals including the best individual found so far and the randomly selected individuals out of the best 50% individuals in the current population. Here, we employ the Quasi-Newton method as the local search method. A local search operator is required as the prespecified MAX_FES are too small to reach the required level accuracy.

4 Experimental Results

We evaluate the performance of the proposed SaDE algorithm on a new set of test problems includes 25 functions with different complexity, where 5 of them are unimodal problems and other 20 are multimodal problems. Experiments are conducted on all 25 10-D functions and the former 15 30D problems. We choose the population size to be 50 and 100 for 10D and 30D problems, respectively.

For each function, the SaDE is run 25 runs. Best functions error values achieved when FES=1e+2, FES=1e+3, FES=1e+4 for the 25 test functions are listed in Tables 1-5 for 10D and Tables 6-8 for 30D, respectively. Successful FES & Success Performance are listed in Tables 9 and 10 for 10D and 30D, respectively.

Table 1. Error Values Achieved for Functions 1-5 (10D)

10D		1	2	3	4	5
1e+3	1st	814.1681	3.1353e+003	6.0649e+006	2.7817e+003	6.6495e+003
	7th	1.4865e+003	6.0024e+003	2.2955e+007	6.2917e+003	8.4444e+003
	13th	2.0310e+003	7.3835e+003	3.4010e+007	7.8418e+003	9.1522e+003
	19th	2.4178e+003	9.1189e+003	5.3783e+007	9.5946e+003	9.4916e+003
	25th	3.2049e+003	1.1484e+004	8.4690e+007	1.5253e+004	1.0831e+004
	M	1.9758e+003	7.3545e+003	3.9124e+007	8.0915e+003	8.9202e+003
	Std	651.2718	2.4077e+003	2.1059e+007	3.1272e+003	999.5368
1e+4	1st	1.1915e-005	7.9389	2.3266e+005	29.7687	126.9805
	7th	2.6208e-005	14.1250	7.7086e+005	57.3773	165.4529
	13th	3.2409e-005	19.6960	1.0878e+006	70.3737	184.6404
	19th	4.9557e-005	30.4271	1.7304e+006	91.9872	228.7035
	25th	9.9352e-005	45.1573	2.9366e+006	187.8363	437.7502
	M	3.8254e-005	23.2716	1.2350e+006	83.1323	203.5592
	Std	2.0194e-005	10.7838	6.8592e+005	43.7055	66.1114
1e+5	1st	0	0	0	0	1.1133e-006
	7th	0	0	0	0	0.0028
	13th	0	0	0	0	0.0073
	19th	0	0	9.9142e-006	0	0.0168
	25th	0	2.5580e-012	1.0309e-004	3.5456e-004	0.0626
	M	0	1.0459e-013	1.6720e-005	1.4182e-005	0.0123
	Std	0	5.1124e-013	3.1196e-005	7.0912e-005	0.0146

Table 2. Error Values Achieved for Functions 6-10 (10D)

10D		6	7	8	9	10
1e+3	1st	1.7079e+007	113.7969	20.3848	36.9348	45.2123
	7th	3.5636e+007	191.6213	20.5603	49.4287	69.0149
	13th	4.9869e+007	206.4133	20.7566	53.2327	77.9215
	19th	7.6773e+007	235.1666	20.8557	60.5725	82.2402
	25th	1.4553e+008	421.4129	20.9579	70.0434	94.8549
	M	5.6299e+007	227.6164	20.7176	54.3968	75.7973
	Std	3.4546e+007	82.5769	0.1696	7.5835	11.6957
1e+4	1st	10.2070	0.2876	20.3282	3.8698	24.1745
	7th	15.5318	0.6445	20.4420	5.8920	26.9199
	13th	23.6585	0.6998	20.5083	6.5883	32.2517
	19th	31.4704	0.7328	20.5607	7.2996	36.3790
	25th	93.9778	0.7749	20.6977	9.3280	42.5940
	M	29.7719	0.6696	20.5059	6.6853	32.2302
	Std	23.5266	0.1072	0.0954	1.2652	5.4082
1e+5	1st	0	4.6700e-010	20.0000	0	1.9899
	7th	4.3190e-009	0.0148	20.0000	0	3.9798
	13th	5.1631e-009	0.0197	20.0000	0	4.9748
	19th	9.1734e-009	0.0271	20.0000	0	5.9698
	25th	8.0479e-008	0.0369	20.0000	0	9.9496
	M	1.1987e-008	0.0199	20.0000	0	4.9685
	Std	1.9282e-008	0.0107	5.3901e-008	0	1.6918

Table 3. Error Values Achieved for Functions 11-15 (10D)

10D		11	12	13	14	15
1e+3	1st	8.9358	1.4861e+004	4.4831	3.7675	437.7188
	7th	11.1173	3.9307e+004	6.3099	4.1824	612.0006
	13th	11.5523	6.2646e+004	6.9095	4.2771	659.7280
	19th	12.0657	6.9730e+004	7.5819	4.3973	685.5215
	25th	12.7319	8.1039e+004	9.3805	4.4404	758.4222
	M	11.4084	5.6920e+004	6.9224	4.2598	647.6461
	Std	0.9536	1.8450e+004	1.1116	0.1676	65.1235
1e+4	1st	5.7757	2.5908e+003	0.9800	3.1891	133.4582
	7th	7.3877	7.3418e+003	1.2205	3.7346	159.2004
	13th	7.8938	9.8042e+003	1.4449	3.8886	193.2431
	19th	8.8545	1.0432e+004	1.5457	4.0240	227.7915
	25th	9.5742	1.2947e+004	1.8841	4.0966	444.3964
	M	8.0249	8.8181e+003	1.4318	3.8438	210.5349
	Std	1.0255	2.7996e+003	0.2541	0.2161	80.0138
1e+5	1st	3.2352	1.4120e-010	0.1201	2.5765	0
	7th	4.5129	1.7250e-008	0.1957	2.7576	0
	13th	4.7649	8.1600e-008	0.2170	2.8923	0
	19th	5.3823	3.8878e-007	0.2508	3.0258	2.9559e-012
	25th	5.9546	3.3794e-006	0.3117	3.3373	400
	M	4.8909	4.5011e-007	0.2202	2.9153	32.0000
	Std	0.6619	8.5062e-007	0.0411	0.2063	110.7550

Table 4. Error Values Achieved for Functions 16-20 (10D)

10D		16	17	18	19	20
1e+3	1st	235.2350	307.4325	1.0327e+003	1.0629e+003	1.0183e+003
	7th	281.7288	330.9715	1.0964e+003	1.0936e+003	1.0930e+003
	13th	304.0599	348.7749	1.1120e+003	1.1069e+003	1.1086e+003
	19th	333.1548	405.0067	1.1337e+003	1.1147e+003	1.1347e+003
	25th	367.0937	467.2421	1.1793e+003	1.1524e+003	1.1570e+003
	M	306.5995	366.3721	1.1124e+003	1.1075e+003	1.1108e+003
	Std	36.3082	45.2002	31.4597	23.6555	31.9689
1e+4	1st	142.4128	171.5105	561.9794	543.2119	510.3079
	7th	161.4197	183.9739	800.8610	804.0210	801.3788
	13th	169.3572	200.6682	809.4465	822.0176	815.1567
	19th	173.9672	211.5187	854.3151	850.2155	837.9725
	25th	188.7826	241.7007	970.1451	985.6591	974.6514
	M	168.3112	200.1827	817.4287	832.3296	813.2161
	Std	11.2174	18.7424	97.8982	101.2925	102.1561
1e+5	1st	86.3059	99.0400	300	300	300
	7th	98.5482	106.7286	800.0000	653.5664	800.0000
	13th	101.4533	113.6242	800.0000	800.0000	800.0000
	19th	104.9396	119.2813	800.0000	800.0000	800.0000
	25th	111.9003	135.5105	900.8377	930.7288	907.0822
	M	101.2093	114.0600	719.3861	704.9373	713.0240
	Std	6.1686	9.9679	208.5161	190.3959	201.3396

Table 5. Error Values Achieved for Functions 21-25 (10D)

10D		21	22	23	24	25
1e+3	1st	1.0738e+003	903.5596	1.1912e+003	778.2495	452.5057
	7th	1.2915e+003	970.4664	1.2867e+003	1.0789e+003	608.3791
	13th	1.3148e+003	985.8289	1.3152e+003	1.1394e+003	648.1046
	19th	1.3239e+003	1.0114e+003	1.3239e+003	1.2317e+003	727.7877
	25th	1.3429e+003	1.0863e+003	1.3451e+003	1.3189e+003	1.1262e+003
	M	1.2953e+003	991.2809	1.3031e+003	1.1253e+003	680.7246
	Std	56.6604	42.7061	31.9994	129.1495	151.5888
1e+4	1st	477.2834	412.1654	559.4691	200.0001	383.4268
	7th	502.2285	777.3001	559.4785	200.0003	388.4146
	13th	668.0738	783.8782	614.8667	200.0007	392.8109
	19th	898.5615	786.8441	970.5031	200.0016	393.5933
	25th	1.0735e+003	800.1401	1.1207e+003	200.1128	395.6858
	M	689.0325	768.5931	748.6843	200.0056	390.8016
	Std	203.8093	74.5398	212.1329	0.0224	3.9586
1e+5	1st	300	300.0000	559.4683	200	370.9112
	7th	300.0000	750.6537	559.4683	200	373.0349
	13th	500.0000	752.4286	559.4683	200	375.4904
	19th	500.0000	756.9808	721.2327	200	378.1761
	25th	800.0000	800	970.5031	200	381.5455
	M	464.0000	734.9044	664.0557	200	375.8646
	Std	157.7973	91.5229	152.6608	0	3.1453

Table 6. Error Values Achieved for Functions 1-5 (30D)

30D		1	2	3	4	5
1e+3	1st	4.2730e+004	4.8595e+004	6.5006e+008	5.4125e+004	2.6615e+004
	7th	4.8645e+004	7.7846e+004	7.7457e+008	8.8156e+004	3.1265e+004
	13th	5.3467e+004	8.3764e+004	9.2200e+008	1.0266e+005	3.2998e+004
	19th	5.6481e+004	8.9311e+004	1.0911e+009	1.1412e+005	3.4256e+004
	25th	6.5195e+004	1.0850e+005	1.3928e+009	1.2596e+005	3.5876e+004
	M	5.3182e+004	8.1192e+004	9.6475e+008	9.8651e+004	3.2320e+004
	Std	5.9527e+003	1.4020e+004	2.1207e+008	1.9938e+004	2.5184e+003
1e+4	1st	5.7649e+002	2.3977e+004	7.5955e+007	2.6202e+004	1.0918e+004
	7th	9.2574e+002	3.0457e+004	1.1061e+008	3.4788e+004	1.1863e+004
	13th	9.6939e+002	3.1798e+004	1.2346e+008	3.8316e+004	1.2525e+004
	19th	1.0161e+003	3.3950e+004	1.3413e+008	4.0290e+004	1.3515e+004
	25th	1.2382e+003	4.4482e+004	1.7999e+008	5.3358e+004	1.4761e+004
	M	9.7498e+002	3.1932e+004	1.2425e+008	3.8336e+004	1.2730e+004
	Std	1.3684e+002	4.1549e+003	2.4947e+007	5.5137e+003	1.0810e+003
1e+5	1st	0	2.3302e-004	8.1709e+004	9.7790e+000	1.3264e+003
	7th	0	8.0687e-003	1.3108e+005	7.7998e+001	2.1156e+003
	13th	5.6843e-014	6.9681e-002	2.0066e+005	1.2005e+002	2.5316e+003
	19th	5.6843e-014	4.0714e-001	2.9315e+005	3.0624e+002	2.7938e+003
	25th	5.6843e-014	3.5360e+001	3.1015e+006	1.1099e+003	3.8552e+003
	M	3.1832e-014	2.3574e+000	3.4760e+005	2.4542e+002	2.4449e+003
	Std	2.8798e-014	7.3445e+000	5.8904e+005	2.7869e+002	5.9879e+002
3e+5	1st	0	5.6843e-014	1.8184e+003	4.8044e-010	1.3484e+000
	7th	0	5.6843e-014	7.7336e+003	1.4460e-007	1.7185e+001
	13th	0	1.1369e-013	1.5935e+004	8.5699e-007	6.8808e+001
	19th	0	1.1369e-013	2.9740e+004	4.0090e-006	2.1590e+003
	25th	0	2.4298e-006	8.0315e+005	6.8315e-005	3.5975e+003
	M	0	9.7191e-008	5.0521e+004	5.8160e-006	7.8803e+002
	Std	0	4.8596e-007	1.5754e+005	1.4479e-005	1.2439e+003

Table 7. Error Values Achieved for Functions 6-10 (30D)

30D		6	7	8	9	10
1e+3	1st	5.0916e+009	4.4818e+003	2.0991e+001	3.6522e+002	5.2034e+002
	7th	6.2021e+009	5.5572e+003	2.1156e+001	3.8597e+002	5.7493e+002
	13th	7.4284e+009	5.9274e+003	2.1188e+001	3.9171e+002	6.0359e+002
	19th	8.4641e+009	6.5588e+003	2.1255e+001	4.0845e+002	6.2592e+002
	25th	1.0602e+010	7.1445e+003	2.1302e+001	4.2017e+002	6.9889e+002
	M	7.4005e+009	5.9507e+003	2.1191e+001	3.9462e+002	6.0193e+002
	Std	1.5005e+009	7.3502e+002	7.8238e-002	1.5638e+001	4.4696e+001
1e+4	1st	3.3127e+006	1.7732e+002	2.0980e+001	1.5000e+002	2.3421e+002
	7th	5.9023e+006	2.4483e+002	2.1069e+001	1.7987e+002	2.6193e+002
	13th	7.3526e+006	2.7830e+002	2.1103e+001	1.8706e+002	2.7007e+002
	19th	9.6219e+006	3.0569e+002	2.1125e+001	1.9524e+002	2.7940e+002
	25th	1.5110e+007	3.8775e+002	2.1209e+001	2.0492e+002	2.9881e+002
	M	7.7825e+006	2.7193e+002	2.1096e+001	1.8586e+002	2.6886e+002
	Std	2.8737e+006	5.5928e+001	5.5938e-002	1.3305e+001	1.4686e+001
1e+5	1st	2.2101e+001	3.0941e-005	2.0112e+001	2.0464e-012	2.6864e+001
	7th	2.3734e+001	7.4335e-003	2.0231e+001	1.5111e-002	3.8803e+001
	13th	2.4372e+001	1.0052e-002	2.0309e+001	4.7865e-002	4.5768e+001
	19th	2.5451e+001	2.0582e-002	2.0362e+001	7.3677e-002	5.3728e+001
	25th	9.1559e+001	5.4106e-002	2.0480e+001	2.8997e-001	6.0693e+001
	M	2.7283e+001	1.4565e-002	2.0305e+001	5.8444e-002	4.5763e+001
	Std	1.3445e+001	1.2971e-002	9.2049e-002	6.7432e-002	9.1881e+000
3e+5	1st	3.9866e+000	2.8422e-014	2.0040e+001	0	2.5869e+001
	7th	1.8679e+001	4.1056e-007	2.0096e+001	0	3.0844e+001
	13th	1.9057e+001	7.3960e-003	2.0127e+001	0	3.6813e+001
	19th	2.0072e+001	9.8597e-003	2.0187e+001	0	3.8803e+001
	25th	8.3664e+001	5.1640e-002	2.0259e+001	5.6843e-014	4.6866e+001
	M	2.1248e+001	8.2727e-003	2.0140e+001	2.2737e-015	3.5758e+001
	Std	1.3413e+001	1.1445e-002	5.7258e-002	1.1369e-014	6.0809e+000

Table 8. Error Values Achieved for Functions 11-15 (30D)

30D		11	12	13	14	15
1e+3	1st	4.0206e+001	1.0502e+006	8.6240e+001	1.3880e+001	9.0131e+002
	7th	4.4625e+001	1.5267e+006	1.4490e+002	1.4076e+001	1.0109e+003
	13th	4.5066e+001	1.6705e+006	1.9014e+002	1.4208e+001	1.0498e+003
	19th	4.5815e+001	1.7395e+006	2.3156e+002	1.4268e+001	1.1023e+003
	25th	4.7849e+001	1.9462e+006	2.8257e+002	1.4449e+001	1.1602e+003
	M	4.5036e+001	1.6449e+006	1.9328e+002	1.4177e+001	1.0483e+003

	Std	1.4916e+000	2.0222e+005	5.8095e+001	1.3066e-001	6.6584e+001
1e+4	1st	3.9526e+001	6.9444e+005	1.8333e+001	1.3331e+001	5.1155e+002
	7th	4.0650e+001	8.4099e+005	1.9443e+001	1.3731e+001	5.7146e+002
	13th	4.1464e+001	9.0247e+005	2.0457e+001	1.3837e+001	6.0739e+002
	19th	4.3054e+001	9.7931e+005	2.1555e+001	1.3914e+001	6.6392e+002
	25th	4.3636e+001	1.1349e+006	2.3133e+001	1.4049e+001	7.7397e+002
	M	4.1743e+001	9.2214e+005	2.0497e+001	1.3790e+001	6.2072e+002
	Std	1.2503e+000	1.1142e+005	1.3309e+000	1.8812e-001	7.0309e+001
1e+5	1st	2.6526e+001	4.5250e+002	1.5148e+000	1.2497e+001	1.3978e+002
	7th	2.9945e+001	2.9058e+003	1.9457e+000	1.2704e+001	3.0006e+002
	13th	3.1010e+001	5.1056e+003	2.0321e+000	1.2894e+001	3.7037e+002
	19th	3.1861e+001	7.7071e+003	2.1967e+000	1.3015e+001	4.0000e+002
	25th	3.3046e+001	1.4132e+004	2.7691e+000	1.3222e+001	5.0000e+002
	M	3.0807e+001	5.8477e+003	2.0607e+000	1.2870e+001	3.4588e+002
	Std	1.5169e+000	3.9301e+003	3.1533e-001	2.1536e-001	7.7823e+001
3e+5	1st	2.4079e+001	4.3242e+001	9.5408e-001	1.1662e+001	3.6818e+001
	7th	2.5989e+001	1.6940e+002	1.1129e+000	1.2267e+001	3.0000e+002
	13th	2.6639e+001	6.2359e+002	1.1824e+000	1.2457e+001	3.0161e+002
	19th	2.7358e+001	1.3968e+003	1.2981e+000	1.2530e+001	4.0000e+002
	25th	2.8993e+001	3.6680e+003	1.4627e+000	1.2664e+001	5.0000e+002
	M	2.6562e+001	8.7345e+002	1.2070e+000	1.2360e+001	3.2775e+002
	Std	1.1275e+000	9.3383e+002	1.3420e-001	2.5936e-001	9.6450e+001

Table 9. Best Error Functions Values Achieved in the MAX_FES & Success Performance (10D)

F	1st (Min)	7th	13th (Med)	19th	25th (Max)	Mean	Std	Success rate	Success Perf.
1	10126	10126	10126	10126	10126	10126	0	1	1.0126e+004
2	10227	10228	10241	10243	10244	10237	7.2920	1	1.0237e+004
3	28357	31750	36644	-	-	-	-	0.6400	5.2306e+004
4	31040	37822	39110	51796	-	-	-	0.9600	4.5601e+004
5	-	-	-	-	-	-	-	0	0
6	31548	42131	52756	52803	63382	48777	10240	1	4.8777e+004
7	10257	-	-	-	-	-	-	0.2400	1.7197e+005
8	-	-	-	-	-	-	-	0	0
9	14540	15855	17217	17837	20103	17048	1340.9	1	1.7048e+004
10	-	-	-	-	-	-	-	0	0
11	-	-	-	-	-	-	-	0	0
12	10302	20916	31493	42123	52733	31933	12789	1	3.1933e+004
13	-	-	-	-	-	-	-	0	0
14	-	-	-	-	-	-	-	0	0
15	20787	31402	31430	31462	-	-	-	0.9200	3.3165e+004
16	-	-	-	-	-	-	-	0	0
17	-	-	-	-	-	-	-	0	0
18	-	-	-	-	-	-	-	0	0
19	-	-	-	-	-	-	-	0	0
20	-	-	-	-	-	-	-	0	0
21	-	-	-	-	-	-	-	0	0
22	-	-	-	-	-	-	-	0	0
23	-	-	-	-	-	-	-	0	0
24	-	-	-	-	-	-	-	0	0
25	-	-	-	-	-	-	-	0	0

Table 10. Best Error Functions Values Achieved in the MAX_FES & Success Performance (30D)

F	1st (Min)	7th	13th (Med)	19th	25th (Max)	Mean	Std	Success rate	Success Perf.
1	2.0233e+004	2.0233e+004	2.0234e+004	2.0234e+004	2.0234e+004	2.0234e+004	5.0662e-001	1.00	2.0234e+004
2	1.2175e+005	1.3344e+005	1.4174e+005	1.4648e+005	-	-	-	0.96	1.4883e+005
3	-	-	-	-	-	-	-	0	0
4	2.4482e+005	2.8434e+005	2.9639e+005	-	-	-	-	0.52	5.3816e+005
5	-	-	-	-	-	-	-	0	0
6	-	-	-	-	-	-	-	0	0
7	6.9648e+004	8.3422e+004	1.0162e+005	1.6748e+005	-	-	-	0.80	1.3477e+005
8	-	-	-	-	-	-	-	0	0
9	8.2995e+004	1.0351e+005	1.0389e+005	1.0395e+005	1.0396e+005	9.8934e+004	9.0090e+003	1.00	9.8934e+004
10	-	-	-	-	-	-	-	0	0
11	-	-	-	-	-	-	-	0	0
12	-	-	-	-	-	-	-	0	0
13	-	-	-	-	-	-	-	0	0
14	-	-	-	-	-	-	-	0	0
15	-	-	-	-	-	-	-	0	0

The 10D convergence maps of the SaDE algorithm on functions 1-5 , functions 6-10, functions 11-15, functions 16-20, and functions 21-25 are plotted in Figures 1-5 respectively. The 30D convergence maps of the SaDE algorithm on functions 1-5 , functions 6-10, functions 11-15 are illustrated in Figures 6-8, respectively.

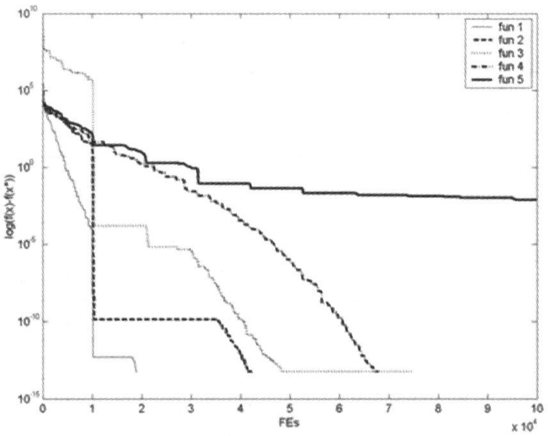

Figure 1. Convergence Graph for Function 1-5

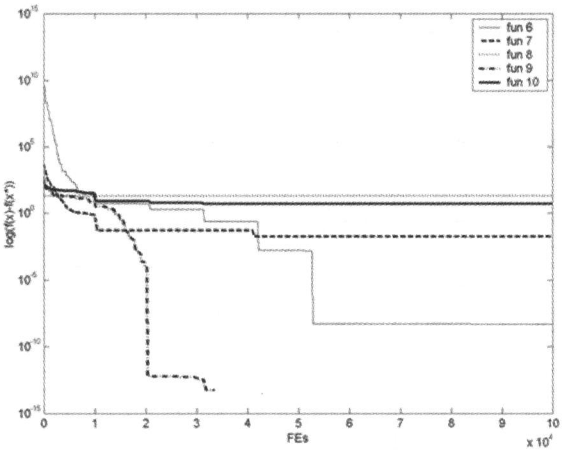

Figure 2. Convergence Graph for Function 6-10

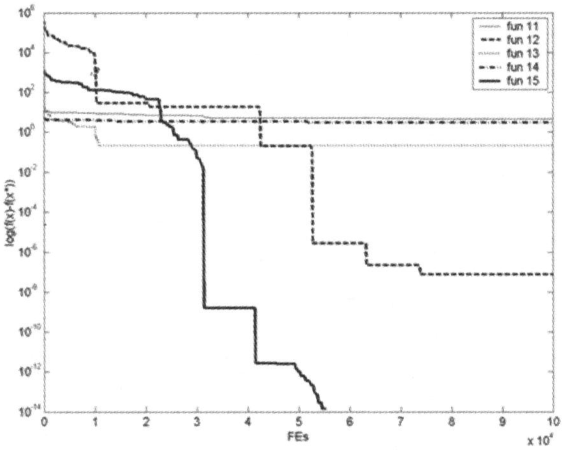

Figure 3. Convergence Graph for Function 11-15

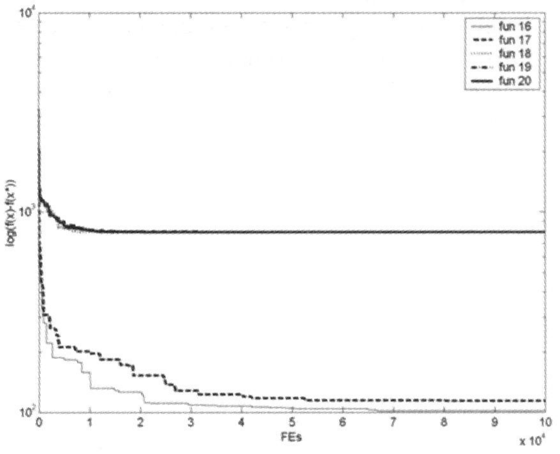

Figure 4. Convergence Graph for Function 16-20

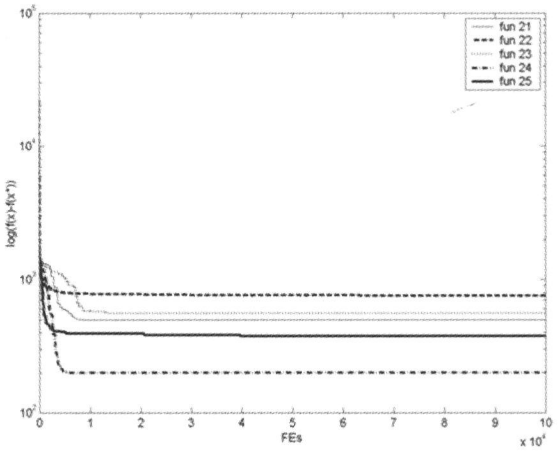

Figure 5. Convergence Graph for Function 21-25

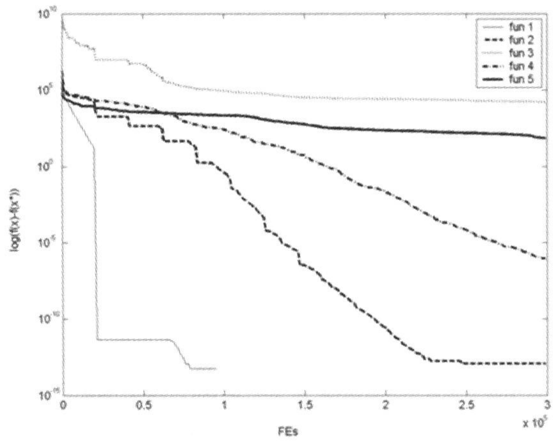

Figure 6. Convergence Graph for Function 1-5

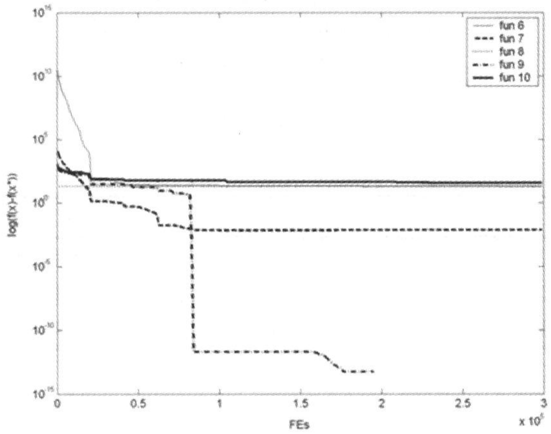

Figure 7. Convergence Graph for Function 6-10

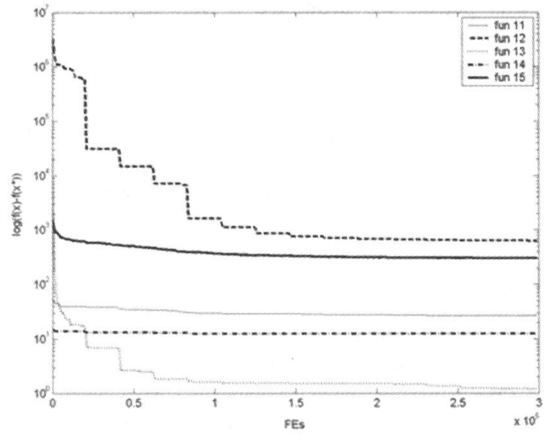

Figure 8. Convergence Graph for Function 11-15

From the results, we could observe that, for 10D problems, the SaDE algorithm can find the global optimal solution for functions 1, 2, 3, 4, 6, 7, 9, 12 and 15 with success rate 1, 1, 0.64, 0.96, 1, 0.24, 1, 1 and 0.92, respectively. For some functions, e.g. function 3, although the success rate is not 1, the final obtained best solutions are very close to the success level; For 30D problems, the SaDE algorithm can find the global optimal solutions for functions 1, 2, 4, 7 and 9 with success rate 1, 0.96, 0.52, 0.8 and 1, respectively. However, from function 16 throughout to 25, the SaDE algorithm cannot find any global optimal solution for both 10D and 30D over the 25 runs due to the high multi-modality of those composite functions and also the local search process asscociated with the SaDE make the algorithm to prematurely converge to a local optimal solution. Therefore, in our paper, we do not list the 30D results for functions 16-25.
The algorithm complexity, which is defined on http://www.ntu.edu.sg/home/EPNSugan/, is calculated on 10, 30, 50 dimensions on function 3, to show the algorithm complexity's relationship with increasing dimensions as in Table 9. We use the Matlab 6.1 to implement the algorithm and the system configurations are listed as follows:

System Configurations
Intel Pentium® 4 CPU 3.00 GHZ
1 GB of memory
Windows XP Professional Version 2002
Language: Matlab

Table 9. Algorithm Complexity

	T0	T1	$\hat{T}2$	$(\hat{T}2 - T1)/T0$
D=10	40.0710	31.6860	68.8004	0.8264
D=30	40.0710	38.9190	74.2050	0.8806
D=50	40.0710	47.1940	85.4300	0.9542

5 Conclusions

In this paper, we proposed a Self-adaptive Differential Evolution algorithm (SaDE), which can automatically adapt its learning strategies and the asscociated parameters during the evolving procedure. The performance of the proposed SaDE algorithm are evaluated on the newly proposed testbed for CEC2005 special session on real parameter optimization.

Bibliography

[1] R. Storn and K. V. Price, “Differential evolution-A simple and Efficient Heuristic for Global Optimization over Continuous Spaces,” Journal of Global Optimization 11:341-359. 1997.

[2] J. Ilonen, J.-K. Kamarainen and J. Lampinen, “Differential Evolution Training Algorithm for Feed-Forward Neural Networks,” In: *Neural Processing Letters* Vol. 7, No. 1 93-105. 2003.

[3] R. Storn, “Differential evolution design of an IIR-filter,” In: *Proceedings of IEEE Int. Conference on Evolutionary Computation ICEC'96*. IEEE Press, New York. 268-273. 1996.

[4] T. Rogalsky, R.W. Derksen, and S. Kocabiyik, “Differential Evolution in Aerodynamic Optimization,” In: *Proc. of 46th Annual Conf of Canadian Aeronautics and Space Institute*. 29-36. 1999.

[5] K. V. Price, “Differential evolution vs. the functions of the 2nd ICEO”, Proc. of 1997 IEEE International Conference on Evolutionary Computation (ICEC '97), pp. 153-157, Indianapolis, IN, USA, April 1997.

[6] R. Gaemperle, S. D. Mueller and P. Koumoutsakos, “A Parameter Study for Differential Evolution”, A. Grmela, N. E. Mastorakis, editors, *Advances in Intelligent Systems, Fuzzy Systems, Evolutionary Computation*, WSEAS Press, pp. 293-298, 2002.

[7] J. Gomez, D. Dasgupta and F. Gonzalez, "Using Adaptive Operators in Genetic Search", *Proc. of the Genetic and Evolutionary Computation Conference (GECCO)*, pp.1580-1581,2003.

[8] Bryant A. Julstrom, "What Have You Done for Me Lately? Adapting Operator Probabilities in a Steady-State Genetic Algorithm" *Proc. of the 6th International Conference on Genetic Algorithms*, pp.81-87,1995.