Large Scale Global Optimization Using Differential Evolution With Self-adaptation and Cooperative Co-evolution

Aleš Zamuda, *Student Member, IEEE*, Janez Brest, *Member, IEEE*,
Borko Bošković, *Student Member, IEEE*, Viljem Žumer, *Member, IEEE*

***Abstract*— In this paper, an optimization algorithm is formulated and its performance assessment for large scale global optimization is presented. The proposed algorithm is named DEwSAcc and is based on Differential Evolution (DE) algorithm, which is a floating-point encoding evolutionary algorithm for global optimization over continuous spaces. The original DE is extended by log-normal self-adaptation of its control parameters and combined with cooperative co-evolution as a dimension decomposition mechanism. Experimental results are given for seven high-dimensional test functions proposed for the Special Session on Large Scale Global Optimization at 2008 IEEE World Congress on Computational Intelligence.**

I. Introduction

THE objective of global optimization is to find the search parameter values $\vec{x} = \{x_1, x_2, ..., x_D\}$ (D being dimensionality of a problem tackled) such that the evaluation function value $f(\vec{x})$ is optimal, i.e. $\vec{x}$ has better evaluation function value than any other parameter setting. In this paper, we will treat optimization task as a minimization.

Most reported studies on differential evolution (DE) are performed using low-dimensional problems, e.g., smaller than 100, which are relatively small for many real-world problems [1].

This paper gives performance analysis of a new variant of DE algorithm, on large-scale (100, 500, and 1000 dimension) test functions. Test functions were prepared for the Special Session on Large Scale Global Optimization at 2008 IEEE World Congress on Computational Intelligence [2]. Test function set includes well known functions, included in previous benchmark function suites [3], [4].

The new variant of DE is based on Differential Evolution (DE) algorithm, which is a floating-point encoding evolutionary algorithm for global optimization over continuous spaces. The original DE is extended by log-normal self-adaptation of its control parameters and combined with cooperative co-evolution as a dimension decomposition mechanism. We have already used a similar differential evolution self-adaptation mechanism (without cooperative co-evolution mechanism) for multiobjective optimization in [5].

This work provides the following contributions: (1) an application of a self-adaptive mechanism from evolution strategies and cooperative co-evolution mechanism for dimension decomposition to the original DE algorithm to construct a new version of log-normal self-adaptive DE algorithm; (2) performance analysis of the newly constructed algorithm on three differently dimensioned sets of seven test functions with combinations of unimodal, multimodal, shifted, and fractal properties.

This paper is organised as follows. The next section explains related work on differential evolution, self-adaptation, and cooperative co-evolution. Third section outlines the proposed algorithm. The fourth section presents obtained simulation results on given test function set. The last, fifth section, concludes the paper and gives few guidelines for future work.

Aleš Zamuda, Janez Brest, Borko Bošković, and Viljem Žumer are with the Faculty of Electrical Engineering and Computer Science, University of Maribor, Smetanova ul. 17, 2000 Maribor, Slovenia (email: ales.zamuda@uni-mb.si).

This work was supported in part by the Slovenian Research Agency under program P2-0041 – Computer Systems, Methodologies, and Intelligent Services.

II. Related Work

This work is related to the differential evolution algorithm, evolution strategies, self-adaptation, and cooperative co-evolution. Therefore, all related topics are shortly described.

A. Differential Evolution

In this section we give some background of the DE algorithm [6], [7] that is important for understanding the idea of our new DE variant.

Differential Evolution (DE) is a floating-point encoding evolutionary algorithm for global optimization over continuous spaces [8], [9], which can also work with discrete variables. It was proposed by Storn and Price [7] and since then it has been modified and extended several times with new versions being proposed [10], [11], [12], [13].

In [12], a new DE mutation operator was introduced. Ali and Törn in [14] also suggested some modifications to the classical DE to improve its efficiency and robustness.

DE was also introduced for multiobjective optimization [15], [16], [5], [17].

B. DE Algorithm Strategies

DE is a population based algorithm and vector $\vec{x}_{i,G},\ i = 1, 2, ..., NP$ is an individual in the population. NP denotes population size and G the generation. During one generation for each vector, DE employs mutation, crossover and selection operations to produce a trial vector (offspring) and to select one of those vectors with best fitness value.

By mutation for each population vector a mutant vector $\vec{v}_{i,G+1}$ is created. One of the most popular DE mutation strategy is *'rand/1/bin'* [8], [7]:

$$\vec{v}_{i,G+1} = \vec{x}_{r_1,G} + F \cdot (\vec{x}_{r_2,G} - \vec{x}_{r_3,G}), \qquad (1)$$

978-1-4244-1823-7/08/$25.00©2008 IEEE

where the indexes r_1, r_2, r_3 represent the random and mutually different integers generated within the range $[1, NP]$ and also different from index i. F is an amplification factor of the difference vector within the range $[0, 2]$, but usually less than 1. Another DE strategy is *'best/1/bin'* [8], [7]:

$$\vec{v}_{i,G+1} = \vec{x}_{best,G} + F \cdot (\vec{x}_{r_2,G} - \vec{x}_{r_3,G}), \quad (2)$$

where $\vec{x}_{best,G}$ denotes the best evaluated vector in generation G. Newer DE strategy *'rand/1/either-or'* [8] is:

$$\vec{v}_{i,G+1} = \vec{x}_{r_1,G} + 0.5(F_{i,G+1} + 1) \cdot (\vec{x}_{r_2,G} + \vec{x}_{r_3,G} - 2\vec{x}_{r_1,G}). \quad (3)$$

The original DE algorithm is well described in literature [8], [7], and therefore, we will skip detailed description of the whole DE algorithm.

C. Self-adaptation in DE

Evolution strategies [18], [19], [20], [21] include a self-adaptive mechanism [22], [23], [24], encoded directly in each individual of the population to control some of the parameters [25] in the evolutionary algorithm. Self-adaptation allows an evolution strategy to adapt itself to any general class of problems, by reconfiguring itself accordingly, and to do this without any user interaction [19], [21].

An evolution strategy (ES) has a notation $\mu/\rho, \lambda$-ES, where μ is parent population size, ρ is the number of parents for each new individual, and λ is child population size. An individual is denoted as $\vec{a} = (\vec{x}, \vec{s}, f(\vec{x}))$, where $\vec{x}$ are search parameters, $\vec{s}$ are control parameters, and $f(\vec{x})$ is the evaluation of the individual. Although there are other notations and evolution strategies, we will only need a basic version.

DE creates new candidate solutions by combining the parent individual and a few other individuals of the same population. In each iteration, a candidate replaces the parent only if it has better fitness value. DE has three control parameters: amplification factor of the difference vector – F, crossover control parameter – CR, and population size – NP.

The original DE algorithm keeps all three control parameters fixed during the optimization process. However, there still exists a lack of knowledge how to find reasonably good values for the control parameters of DE, for a given function [11], [26]. Based on the experiment in [27], the necessity of changing control parameters during the optimization process was confirmed. DE with self-adaptive control parameters has already been presented in [26], [28], [29], [30], [31], [32].

In [14] Ali and Törn introduced an auxiliary population of NP individuals alongside the original population. They also introduced a rule for adjusting the control parameter F. Liu and Lampinen [11] adapted the mutation control parameter and the crossover control parameter. Teo self-adapted the population size parameter, crossover, and mutation rates [26]. In [27] Brest et al. also self-adapted F and CR control parameters. In [28] Qin and Suganthan used learning strategy and self-adapted control parameters F and CR. During evolution, suitable learning strategy and parameter settings are gradually self-adapted, according to the learning experience. In [30] a performance comparison of certain selected DE algorithms, which use different self-adapting or adapting control parameter mechanisms, was reported.

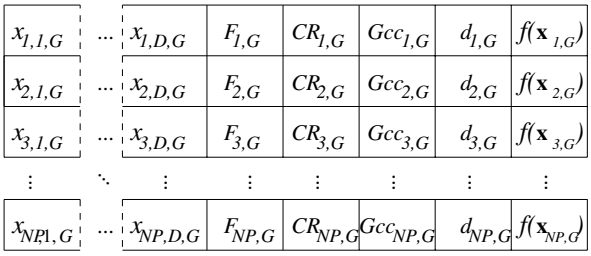

Fig. 1. Encoding of the self-adapting control parameters

D. Cooperative Co-evolution

Cooperative Co-evolution was studied for learning in [33]. Cooperative co-evolution is also a known dimension reduction method [1]. As authors claim in [1], in terms of optimizing high-dimensional problems, cooperative co-evolution with the following divide-and-conquer strategy is the usual and effective choice:

1) Problem decomposition: Splitting the object vectors into some smaller subcomponents.
2) Optimize subcomponents: Evolve each subcomponent with a certain optimizer separately.
3) Cooperative combination: Combine all subcomponents to form the whole system.

III. Differential Evolution With Self-adaptation and Cooperative Co-evolution (DEwSAcc) Algorithm

We use an idea of log-normal self-adaptation mechanism from evolution strategies and apply this idea to the control parameters of the DE algorithm [7]. Our algorithm is also extended with cooperative co-evolution as a dimension decomposition mechanism where only a subcomponent of search parameters is extracted and optimized for certain number of generations. We name the new constructed version of DE, DEwSAcc algorithm.

A. Self-adaptive Control Parameters

Each individual (see Fig. 1) of DEwSAcc algorithm is extended to include the self-adaptive F and CR control parameters to adjust them to the appropriate values during the evolutionary process. The F parameter is a mutation control parameter and CR is the crossover control parameter.

For each individual in the population, a trial vector is composed by mutation and recombination. The mutation procedure is different in the DEwSAcc algorithm in comparison to the original DE.

For adaptation of the amplification factor of the difference vector F_i for trial individual i, from parent generation G

into child generation $G+1$ for the trial vector, the following formula is used:

$$F_{i,G+1} = F_{i,G} \cdot e^{\tau N(0,1)},$$

where τ denotes learning factor and is usually proportional to $\tau = 1/\sqrt{D}$, D being the dimension of a problem. $N(0,1)$ is a random number with a Gauss distribution.

An analogous formula, is used for the CR parameter:

$$CR_{i,G+1} = CR_{i,G} \cdot e^{\tau N(0,1)},$$

where τ is same as in the adaptation of F parameter.

B. Mutation

Mutation strategies used in DEwSAcc algorithm are selected among the DE strategies *'rand/1'* and *'best/1'* used in the original DE [7], as well the *'rand/1/either-or'* strategy [8]. Selection probability in each trial vector creation for the strategies is 0.5, 0.4, and 0.1, respectively. The mutation process for the i-th candidate $\vec{v}_{i,G+1}$ for generation $G+1$ is defined as:

$$\vec{v}_{i,G+1} = \begin{cases} \vec{x}_{r_1,G} + F_{i,G+1} \cdot (\vec{x}_{r_2,G} - \vec{x}_{r_3,G}) & \text{if } r_s < 0.5 \\ \vec{x}_{r_1,G} + 0.5(F_{i,G+1}+1) \cdot (\vec{x}_{r_2,G} & \text{elif } r_s < 0.9 \\ \quad +\vec{x}_{r_3,G} - 2\vec{x}_{r_1,G}) & \\ \vec{x}_{best,G} + F_{i,G+1} \cdot (\vec{x}_{r_1,G} - \vec{x}_{r_2,G}) & \text{otherwise} \end{cases}$$

where $\vec{x}_{r_1,G}$, $\vec{x}_{r_2,G}$, and $\vec{x}_{r_3,G}$ are search parameter values of the uniform randomly selected parents from generation G. The $r_s \in [0,1]$ denotes an uniform randomly generated number which selects among the three mutation strategies.

C. Cooperative Co-evolution

Based on DECC2 algorithm presented in [1], we have used problem decomposition, so that only some of the function's search parameters were changed by a certain individual for a certain number of generations $Gcc_{i,G}$, adaptive:

$$Gcc_{i,G+1} = Gcc_{i,G} \cdot e^{\tau N(0,1)}.$$

Upon co-evolution, only the target vector's determined decomposed parameters set values are changed for each trial vector. DECC2-based decomposition is used for all generations, where a cc_j paremeter determines whether or not to include the j-th function component in the decomposed part of parameters set:

$$cc_j = \begin{cases} 1 & \text{if } rand(0,1) \le d_{i,G} \\ 0 & \text{otherwise,} \end{cases}$$

where $d_{i,G}$ denotes the decomposition rate for each individual and is adjusted along generations:

$$d_{i,G+1} = d_{i,G} \cdot e^{\tau N(0,1)}.$$

D. Crossover

In our algorithm, the adapted $CR_{i,G+1}$ is used together with cooperative co-evolution mechanism to create a modified candidate $u_{i,j,G+1}$ by binary crossover:

$$u_{i,j,G+1} = \begin{cases} v_{i,j,G+1} & \text{if } cc_j = 1 \text{ and} \\ & (rand(0,1) \le CR_{i,G+1} \text{ or} \\ & j = j_{rand}) \\ x_{i,j,G} & \text{otherwise,} \end{cases}$$

where $j \in [1, D]$ denotes the j-th search parameter, $rand(0,1) \in [0,1]$ denotes a uniformly distributed random number, and j_{rand} denotes an uniform randomly chosen index of the search parameter, which is always exchanged.

E. Selection

Selection mechanism used in DEwSAcc algorithm is not changed from original DE, it is a greedy selection scheme, adapting new trial vectors only if their evaluation function value is better from evaluation function value of the current vector.

It is however worth noting, that the selection principle also helps in adapting F and CR parameters, because only the individuals adapting good control and search parameter values survive. With the more appropriate values for the self-adaptive control parameters, the search process attains better objective space parameters. Therefore, the search process converges faster to better individuals which, in turn, are more likely to survive and produce offspring and, hence, propagate these better parameter values.

IV. Simulation Results

Experiments are conducted using simulations for the given test function set for the Special Session on Large Scale Global Optimization at 2008 IEEE World Congress on Computational Intelligence Benchmark [2].

A. Benchmark Suite

Simulation results were obtained using the two unimodal test functions F1 (Shifted Sphere Function) and F2 (Shifted Schwefels Problem 2.21), and the five multimodal functions F3 (Shifted Rosenbrocks Function), F4 (Shifted Rastrigins Function), F5 (Shifted Griewanks Function), F6 (Shifted Ackleys Function), and F7 (FastFractal DoubleDip Function). Each of them was tested on 25 runs at dimensions $D = 100$, $D = 500$, and $D = 1000$, respectively.

B. Parameters Setting of DEwSAcc Algorithm

In the experiments the following parameter settings were used for the DEwSAcc algorithm. Number of generations $Max\_G = 5000$ was kept fixed for all problems, and therefore NP was $NP = D$, i.e. $NP_{D=100} = 100$, $NP_{D=500} = 500$, and $NP_{D=1000} = 1000$. A global lower and upper bounds for control parameter F were $0 \le F \le 1$, and for control parameter CR they were $1/D \le CR \le 1$.

The τ parameter was proportional with dimension D, $\tau_{100} = 0.2/\sqrt{D}$, $\tau_{500} = 0.2\sqrt{2}/\sqrt{D}$, and $\tau_{1000} = 0.2 \cdot 2\sqrt{2}/\sqrt{D}$.

The co-evolution dimension decomposition rate $d_{i,G} \in [0, 1]$ was $1/D \leq d_{i,G} \leq 1$ for $D = 100$, $D = 500$, and $D = 1000$, respectively. Co-evolution $Gcc_{i,G}$ was constrained to $1 \leq Gcc_{i,G} \leq 100$ bounds for all $D = 100$, $D = 500$, and $D = 1000$, respectively.

Parameter value that could still be adjusted is the adaptation rate τ.

C. Estimated Runtime for the Test Suite

Dimension: $D = 1000$, problems: functions 1–7, algorithm: Differential Evolution, system: Fedora Core 4, language: C++, runs: only one run, Max_FEs: 5000000, PC: Dual Core AMD Opteron(tm) Processor 280, 2390.651GHz, RAM-8G, runtime: 4 hours CPU time (real time was smaller).

D. Discussion

Tables I–III display the 1^{st} (best), 7^{th}, 13^{th} (median), 19^{th}, and 25^{th} (worst) evaluation function values during each function optimization after $0.01Max\_FEs$, $0.1Max\_FEs$, and Max_FEs. The Max_FEs was $500,000$, $2,500,000$, and $5,000,000$ for $D = 100$, $D = 500$, and $D = 1000$, respectively.

For $D = 100$ (see Table I), it can be seen that functions F1, F5, and F6 are optimized quite well. The function F4 is optimized well in the best run, but not so good in the median and on average. Function F2 converges to 8.25 on average at the end of optimization runs. Function F3 seems the hardest to optimize among these, stopping at roughly 145 on average at the end of an optimization run. The best obtained function F7 value for $D = 100$ is -1381.4.

For $D = 500$ (see Table II), it can be seen that function F1 is not optimized as good as for $D = 100$, but its best evaluation function value drops fast after $2.50e+5$ FEs. For function F5, the best evaluation function value is obtained near global optimum, for best optimization run and quite worse in the worse run. Evaluation function value for F6 is not as good as for $D = 100$ as well, but it improves steadily. Functions F2–F4 are again not optimized to such extent, but their evaluation function value keeps improving during FEs too. The best obtained function F7 value for $D = 500$ is -6574.7.

For $D = 1000$ (see Table III), it is seen that function F1 is even harder to optimize as for $D = 500$. F1 and F5 evaluation function values are closest to optimum among this set of functions for $D = 1000$. Function F3 is much optimized during optimization runs, F4 function value is falling as well. For evaluation functions F2 and F6 their best value changes are smallest during optimization runs among $D = 1000$ function set. The best obtained function F7 value for $D = 1000$ is -11508.

Figure 2 shows function error value trough function evaluations for functions 1–6 and $D = 1000$. It can be seen that for functions F1, F3, and F5 the progress is exponential i.e. linear w.r.t. the logarithmic scale. Function F4 is also optimized quite fast. Smallest convergence speed can be observed for functions F2 and F6.

For function 7 and $D = 1000$ Figure 3 depicts evaluation function value. As can be observed here as well, the function values converge more almost all the time of the optimization run.

When disabling the algorithm's cooperative co-evolution mechanism, we obtained results seen in Table IV. At the end of runs for $D = 100$, $D = 500$, and $D = 1000$, almost all obtained function evaluations were better when using cooperative co-evolution. For these results, t-tests were computed to give evidence for statistically significant improvement of introducing the cooperative co-evolution mechanism. Only for the unbiased optimization function F7 (see Fig. 4), omitting the cooperative co-evolution mechanism always gives better results on average. On other functions introducing the mechanism pays off in the presented better results.

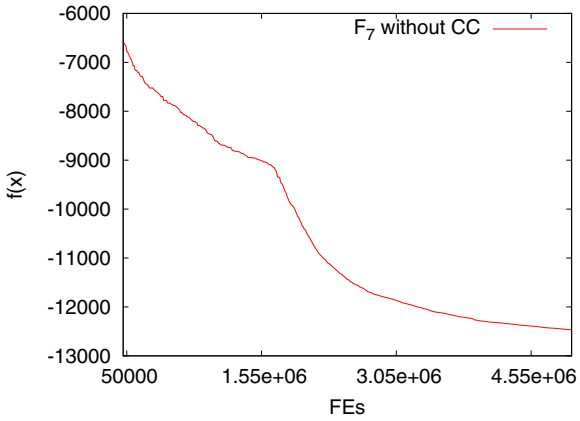

Fig. 4. Convergence Graph for Function 7 when omitting the cooperative co-evolution mechanism

V. Conclusion

The Differential Evolution with Self-adaptation and Cooperative Co-evolution (DEwSAcc) Algorithm was formulated and its performance assessment was given for the seven high-dimensional test functions proposed for the Special Session on Large Scale Global Optimization at 2008 IEEE World Congress on Computational Intelligence. The new algorithm uses DE as the main search technique, but is extended by self-adaptation and cooperative co-evolution.

Further extensions of the algorithm could be hybridisation with a local search (e.g. Sequential Quadratic Programming [34] or Covariance Matrix Adaptation [35]), and DE strategy self-adaptation [36].

As can be observed from results, the worst function value was obtained for F3: Shifted Rosenbrock's Function. For this function we made a few feasibility studies to improve the results by using a local search [37] on the best solution in every tenth generation. Best values obtained for F3 with $D = 100$ were down to 2.0464e-10, for $D = 500$ 1.1024e-07, and 2.3438e-07 for $D = 1000$. This indicates that the results could be improved, but detailed analysis and further improvements are due for further research.

TABLE I

ERROR VALUES ACHIEVED FOR PROBLEMS 1-7, WITH $D = 100$

FES		1	2	3	4	5	6	7
	1^{st} (Best)	8.3828e+04	1.0821e+02	1.7758e+10	1.1226e+03	8.0886e+02	1.9204e+01	-9.3759e+02
	7^{th}	1.0587e+05	1.1488e+02	2.5129e+10	1.2271e+03	8.9186e+02	1.9726e+01	-8.9853e+02
	13^{th} (Median)	1.2259e+05	1.1837e+02	2.9213e+10	1.2663e+03	9.9249e+02	2.0121e+01	-8.9062e+02
$\mathbf{5.00e+3}$	19^{th}	1.3247e+05	1.2374e+02	3.7742e+10	1.2843e+03	1.0451e+03	2.0278e+01	-8.8153e+02
	25^{th} (Worst)	1.5740e+05	1.3191e+02	5.7587e+10	1.3393e+03	1.4357e+03	2.0582e+01	-8.6589e+02
	Mean	1.2067e+05	1.1918e+02	3.2692e+10	1.2504e+03	9.9834e+02	2.0015e+01	-8.9184e+02
	Std	2.1065e+04	5.8203	1.0456e+10	5.2740e+01	1.4831e+02	3.8639e-01	1.5851e+01
	1^{st} (Best)	1.1883	4.8652e+01	6.2478e+03	2.3028e+02	3.1522e-01	1.9315e-01	-1.1567e+03
	7^{th}	9.0535	5.2743e+01	2.9974e+04	2.5919e+02	1.0790	1.3143	-1.1338e+03
	13^{th} (Median)	1.7788e+01	5.4268e+01	4.9348e+04	2.9609e+02	1.1649	1.8422	-1.1234e+03
$\mathbf{5.00e+4}$	19^{th}	2.6560e+01	5.7485e+01	9.8943e+04	3.1490e+02	1.2311	2.2366	-1.1053e+03
	25^{th} (Worst)	1.3695e+02	6.8450e+01	4.6189e+05	4.0082e+02	1.8460	3.2108	-1.0957e+03
	Mean	2.4078e+01	5.5515e+01	8.3301e+04	2.9450e+02	1.1476	1.6941	-1.1234e+03
	Std	2.7276e+01	4.3898	9.3131e+04	4.4907e+01	2.9912e-01	8.0548e-01	1.8137e+01
	1^{st} (Best)	5.6843e-14	3.1796	9.0367e+01	5.6843e-14	2.8421e-14	8.5265e-14	-1.4137e+03
	7^{th}	5.6843e-14	4.6757	9.3053e+01	3.7096e-10	2.8421e-14	1.1368e-13	-1.3814e+03
	13^{th} (Median)	5.6843e-14	6.3670	1.4611e+02	1.9899	2.8421e-14	1.1368e-13	-1.3664e+03
$\mathbf{5.00e+5}$	19^{th}	5.6843e-14	1.0622e+01	1.5296e+02	4.9747	2.8421e-14	1.1368e-13	-1.3460e+03
	25^{th} (Worst)	5.6843e-14	2.7417e+01	3.0627e+02	3.5818e+01	5.6843e-14	1.4210e-13	-1.3204e+03
	Mean	5.6843e-14	8.2500	1.4463e+02	4.3778	3.0695e-14	1.1255e-13	-1.3650e+03
	Std	0.0000	5.3274	5.8483e+01	7.6532	7.8696e-15	1.5306e-14	2.4565e+01

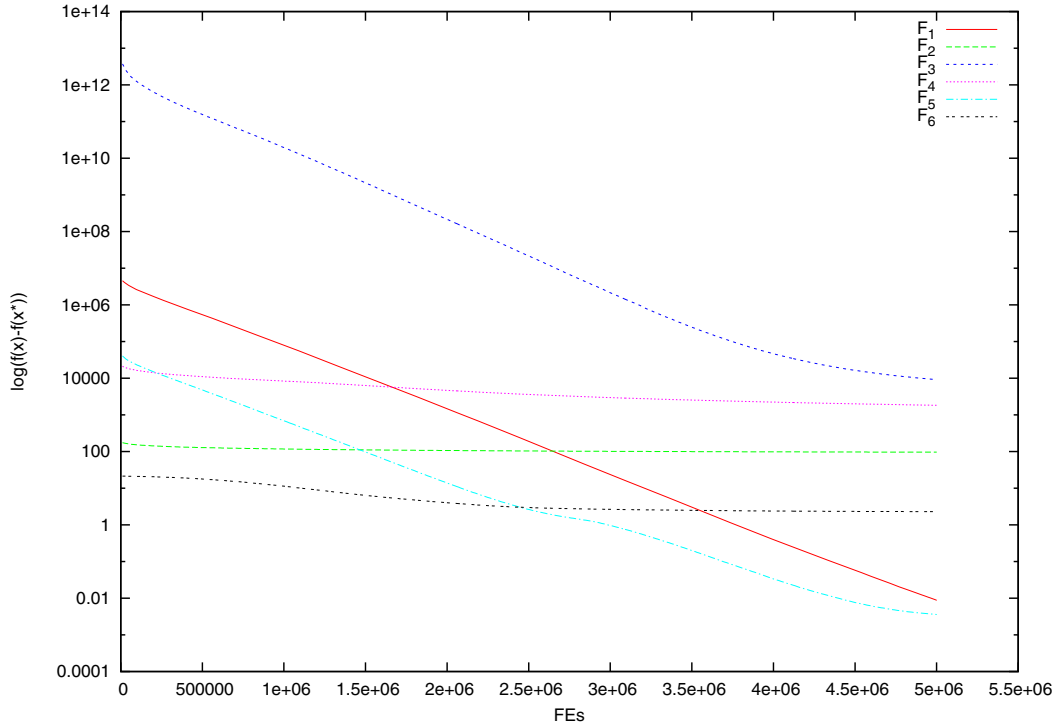

Fig. 2. Convergence Graph for Functions 1–6

TABLE II

ERROR VALUES ACHIEVED FOR PROBLEMS 1-7, WITH $D = 500$

FES		1	2	3	4	5	6	7
	1^{st} (Best)	1.2787e+06	1.4415e+02	5.3248e+11	7.9014e+03	1.1023e+04	2.0797e+01	-3.6612e+03
	7^{th}	1.3741e+06	1.4830e+02	6.9256e+11	8.3009e+03	1.1375e+04	2.0984e+01	-3.6249e+03
	13^{th} (Median)	1.4301e+06	1.5140e+02	7.3432e+11	8.4198e+03	1.1580e+04	2.1033e+01	-3.5965e+03
$\mathbf{2.50e+4}$	19^{th}	1.4713e+06	1.5539e+02	8.0235e+11	8.5105e+03	1.2141e+04	2.1062e+01	-3.5751e+03
	25^{th} (Worst)	1.5263e+06	1.5912e+02	9.9182e+11	8.6877e+03	1.3536e+04	2.1130e+01	-3.5581e+03
	Mean	1.4215e+06	1.5157e+02	7.5376e+11	8.3908e+03	1.1749e+04	2.1019e+01	-3.6030e+03
	Std	6.5764e+04	4.3153	9.2525e+10	1.8565e+02	5.7805e+02	7.5075e-02	3.1426e+01
	1^{st} (Best)	5.3448e+04	1.0648e+02	7.6260e+09	3.9041e+03	4.2257e+02	1.1718e+01	-4.4236e+03
	7^{th}	5.8422e+04	1.1114e+02	9.4815e+09	4.0264e+03	4.8057e+02	1.2121e+01	-4.3635e+03
	13^{th} (Median)	6.2486e+04	1.1277e+02	1.0598e+10	4.1424e+03	5.1518e+02	1.2252e+01	-4.3209e+03
$\mathbf{2.50e+5}$	19^{th}	6.6214e+04	1.1733e+02	1.1713e+10	4.2123e+03	5.7750e+02	1.2751e+01	-4.2938e+03
	25^{th} (Worst)	1.0434e+05	1.2184e+02	1.4378e+10	4.3296e+03	7.9815e+02	1.4105e+01	-4.2581e+03
	Mean	6.5378e+04	1.1375e+02	1.0733e+10	4.1260e+03	5.4453e+02	1.2562e+01	-4.3293e+03
	Std	1.2098e+04	3.9597	1.6030e+09	1.2645e+02	9.2209e+01	6.8792e-01	4.7115e+01
	1^{st} (Best)	5.5194e-11	6.9437e+01	1.3387e+03	2.8699e+02	1.1482e-11	7.0973e-07	-6.5747e+03
	7^{th}	2.5283e-10	7.3939e+01	1.6610e+03	3.2619e+02	3.3054e-11	4.7260e-06	-5.7378e+03
	13^{th} (Median)	8.5242e-10	7.5286e+01	1.7515e+03	3.4896e+02	8.5577e-11	1.2709e-05	-5.7129e+03
$\mathbf{2.50e+6}$	19^{th}	1.1424e-09	7.7271e+01	1.9962e+03	4.0069e+02	2.5801e-10	9.2623e-01	-5.6951e+03
	25^{th} (Worst)	1.8813e-08	8.2208e+01	2.5213e+03	4.6943e+02	9.8572e-03	1.5436	-5.6009e+03
	Mean	2.0958e-09	7.5737e+01	1.8130e+03	3.6403e+02	6.9013e-04	4.8041e-01	-5.7487e+03
	Std	4.6182e-09	3.0465	2.7425e+02	5.2353e+01	2.4149e-03	5.7362e-01	1.8370e+02

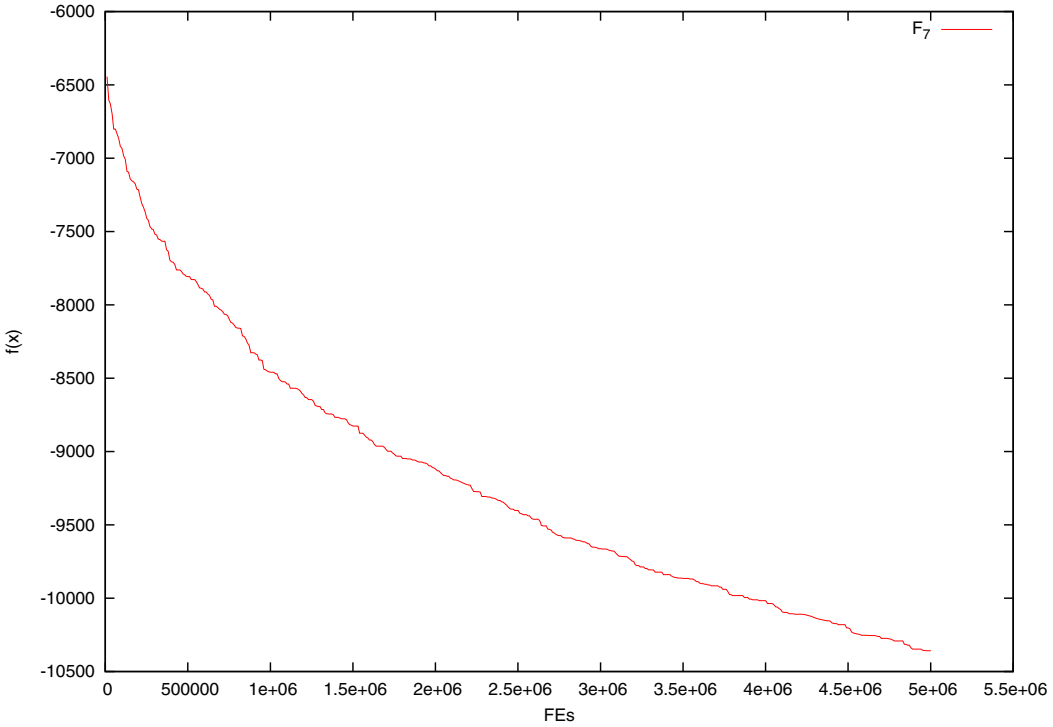

Fig. 3. Convergence Graph for Function 7

TABLE III

ERROR VALUES ACHIEVED FOR PROBLEMS 1-7, WITH $D = 1000$

FES		1	2	3	4	5	6	7
$\mathbf{5.00e+4}$	1^{st} (Best)	3.1517e+06	1.5282e+02	1.6805e+12	1.7615e+04	2.8321e+04	2.1129e+01	-6.8913e+03
	7^{th}	3.2855e+06	1.5926e+02	1.7836e+12	1.7851e+04	2.9085e+04	2.1181e+01	-6.8048e+03
	13^{th} (Median)	3.3457e+06	1.6153e+02	1.9141e+12	1.8020e+04	2.9650e+04	2.1197e+01	-6.7939e+03
	19^{th}	3.3836e+06	1.6326e+02	2.0498e+12	1.8244e+04	3.0190e+04	2.1213e+01	-6.7617e+03
	25^{th} (Worst)	3.5443e+06	1.6593e+02	2.2811e+12	1.8635e+04	3.1439e+04	2.1246e+01	-6.7069e+03
	Mean	3.3445e+06	1.6104e+02	1.9115e+12	1.8055e+04	2.9707e+04	2.1199e+01	-6.7859e+03
	Std	8.7727e+04	3.2054	1.5648e+11	2.4394e+02	8.3760e+02	2.8576e-02	3.9072e+01
$\mathbf{5.00e+5}$	1^{st} (Best)	4.9622e+05	1.2195e+02	1.3986e+11	1.0738e+04	4.2273e+03	1.7338e+01	-7.9377e+03
	7^{th}	5.1087e+05	1.2549e+02	1.4776e+11	1.0927e+04	4.5186e+03	1.7655e+01	-7.8972e+03
	13^{th} (Median)	5.2243e+05	1.2879e+02	1.5336e+11	1.0991e+04	4.7242e+03	1.7759e+01	-7.8451e+03
	19^{th}	5.5952e+05	1.3082e+02	1.6018e+11	1.1109e+04	4.8783e+03	1.7948e+01	-7.8152e+03
	25^{th} (Worst)	6.2241e+05	1.3645e+02	1.8745e+11	1.1471e+04	5.6358e+03	1.8461e+01	-7.7558e+03
	Mean	5.3604e+05	1.2825e+02	1.5510e+11	1.1025e+04	4.7567e+03	1.7799e+01	-7.8549e+03
	Std	3.5192e+04	3.4718	1.1013e+10	1.9433e+02	3.3061e+02	2.6880e-01	5.0918e+01
$\mathbf{5.00e+6}$	1^{st} (Best)	3.1715e-03	9.2531e+01	6.9294e+03	1.5824e+03	2.3156e-04	1.5605	-1.1508e+04
	7^{th}	5.0898e-03	9.4671e+01	8.2666e+03	1.7299e+03	4.3463e-04	2.1076	-1.0520e+04
	13^{th} (Median)	8.0634e-03	9.6356e+01	9.0116e+03	1.8170e+03	6.7930e-04	2.3699	-1.0414e+04
	19^{th}	9.7046e-03	9.7274e+01	1.0035e+04	1.8985e+03	1.5052e-03	2.5210	-1.0358e+04
	25^{th} (Worst)	2.6490e-02	9.9238e+01	1.1629e+04	2.0998e+03	2.0272e-02	2.7450	-1.0268e+04
	Mean	8.7874e-03	9.6058e+01	9.1498e+03	1.8239e+03	3.5826e-03	2.2956	-1.0585e+04
	Std	5.2705e-03	1.8194	1.2605e+03	1.3773e+02	5.7398e-03	2.9834e-01	4.1846e+02

TABLE IV

ERROR VALUES ACHIEVED FOR PROBLEMS 1-7, WITH D =100, 500, OR 1000, WITHOUT COOPERATIVE CO-EVOLUTION

FES		1	2	3	4	5	6	7
$\mathbf{5.00e+5}$ with $D = 100$	1^{st} (Best)	5.6843e-14	2.4974e+01	1.1657e+02	2.0894e+01	2.8421e-14	8.5265e-14	-1.4962e+03
	7^{th}	5.6843e-14	3.0074e+01	1.3728e+02	3.8803e+01	2.8421e-14	1.1368e-13	-1.4681e+03
	13^{th} (Median)	1.1368e-13	3.2951e+01	1.8013e+02	4.7758e+01	5.6843e-14	1.4210e-13	-1.4498e+03
	19^{th}	1.1368e-13	4.1119e+01	2.4199e+02	5.9697e+01	8.5265e-14	1.4210e-13	-1.4368e+03
	25^{th} (Worst)	1.1368e-13	4.8935e+01	3.6559e+02	8.8551e+01	1.4772e-02	1.2697	-1.3944e+03
	Mean	*8.6402e-14*†	3.4855e+01‡	*1.9477e+02*†	*5.0122e+01*†	*2.4643e-03*†	*1.6528e-01*†	-1.4490e+03
	Std	2.8985e-14	7.3395	7.6621e+01	1.7850e+01	4.6641e-03	3.9111e-01	2.4237e+01
$\mathbf{2.50e+6}$ with $D = 500$	1^{st} (Best)	1.1544e-10	8.3223e+01	1.3040e+03	6.5794e+02	1.9809e-11	2.3108	-6.6766e+03
	7^{th}	3.7817e-10	8.7100e+01	1.6455e+03	8.0727e+02	4.8999e-11	2.8004	-6.6335e+03
	13^{th} (Median)	5.2710e-10	8.8862e+01	1.7635e+03	8.6330e+02	1.0174e-10	3.0347	-6.5836e+03
	19^{th}	9.2654e-10	9.0402e+01	1.8518e+03	1.0039e+03	2.0065e-10	3.3967	-6.5305e+03
	25^{th} (Worst)	1.3049e-09	9.3846e+01	2.8263e+03	1.3045e+03	4.1665e-02	3.5607	-6.4326e+03
	Mean	5.9124e-10	8.8905e+01	1.7783e+03	*9.1861e+02*†	*3.1439e-03*†	*3.0407*†	-6.5729e+03
	Std	3.3373e-10	2.5917	2.8353e+02	1.8139e+02	9.1924e-03	3.7318e-01	6.6210e+01
$\mathbf{5.00e+6}$ with $D = 1000$	1^{st} (Best)	2.5110e-01	9.9452e+01	1.9783e+04	2.4684e+03	1.9162e-02	5.7117	-1.2631e+04
	7^{th}	8.5011e-01	1.0312e+02	2.8356e+04	3.0193e+03	3.5632e-02	6.3087	-1.2541e+04
	13^{th} (Median)	1.4251	1.0377e+02	3.6325e+04	3.3757e+03	1.0385e-01	6.5646	-1.2486e+04
	19^{th}	2.7442	1.0465e+02	4.7601e+04	3.6556e+03	2.7126e-01	6.9874	-1.2456e+04
	25^{th} (Worst)	1.5595e+01	1.0849e+02	2.3504e+06	4.3409e+03	1.5691	7.7659	-1.1875e+04
	Mean	2.7763	1.0397e+02	*1.6144e+05*†	*3.3446e+03*†	*2.4106e-01*†	*6.6779*†	-1.2470e+04‡
	Std	3.6131	2.0431	4.7966e+05	4.9581e+02	3.7809e-01	5.5373e-01	1.5047e+02

†,‡ The t value of 24 degree of freedom is significant at 0.01 level of significance by two-tailed t-test. † stands for positive t value, ‡ stands for negative. † means that the original DEwSAcc algorithm is better than the algorithm with omitted cooperative co-evolution mechanism.

ACKNOWLEDGMENT

The authors would like to acknowledge the efforts of the organizers of this special session and competition and its availability of test function suite code. Thanks also to the unknown reviewers for their valuable suggestions and comments.

REFERENCES

[1] Z. Yang, K. Tang, and X. Yao, "Differential Evolution for High-Dimensional Function Optimization," in *Proceedings of the 2007 IEEE Congress on Evolutionary Computation CEC2007*, Singapore, 25-28 September 2007, pp. 3523–3530.

[2] K. Tang, X. Yao, P. N. Suganthan, C. MacNish, Y. P. Chen, C. M. Chen, , and Z. Yang, "Benchmark Functions for the CEC'2008 Special Session and Competition on Large Scale Global Optimization," Nature Inspired Computation and Applications Laboratory, USTC, China, http://nical.ustc.edu.cn/cec08ss.php, Tech. Rep., November 2007.

[3] P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y. P. Chen, A. Auger, and S. Tiwari, "Problem Definitions and Evaluation Criteria for the CEC 2005 Special Session on Real-Parameter Optimization," Nanyang Technological University, Singapore, http://www.ntu.edu.sg/home/EPNSugan, Tech. Rep., 2005. [Online]. Available: citeseer.ist.psu.edu/article/storn95differential.html

[4] C. MacNish, "Towards Unbiased Benchmarking of Evolutionary and Hybrid Algorithms for Real-valued Optimisation," *Connection Science*, vol. 19, no. 4, 2007.

[5] A. Zamuda, J. Brest, B. Bošković, and V. Žumer, "Differential Evolution for Multiobjective Optimization with Self Adaptation," in *The 2007 IEEE Congress on Evolutionary Computation CEC2007*. IEEE Press, 2007, pp. 3617–3624, DOI: 10.1109/CEC.2007.4424941.

[6] R. Storn and K. Price, "Differential Evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces," Berkeley, CA, Tech. Rep. TR-95-012, 1995. [Online]. Available: citeseer.ist.psu.edu/article/storn95differential.html

[7] ——, "Differential Evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces," *Journal of Global Optimization*, vol. 11, pp. 341–359, 1997.

[8] K. V. Price, R. M. Storn, and J. A. Lampinen, *Differential Evolution A Practical Approach to Global Optimization*, ser. Natural Computing Series, G. Rozenberg, T. Bäck, A. E. Eiben, J. N. Kok, and H. P. Spaink, Eds. Berlin, Germany: Springer-Verlag, 2005.

[9] V. Feoktistov, *Differential Evolution: In Search of Solutions (Springer Optimization and Its Applications)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.

[10] J. Lampinen, "A Bibliography of Differential Evolution Algorithm," Lappeenranta University of Technology, Department of Information Technology, Laboratory of Information Processing, Tech. Rep., 2001, online http://www2.lut.fi/ jlampine/debiblio.htm.

[11] J. Liu and J. Lampinen, "A Fuzzy Adaptive Differential Evolution Algorithm," *Soft Comput.*, vol. 9, no. 6, pp. 448–462, 2005.

[12] H.-Y. Fan and J. Lampinen, "A Trigonometric Mutation Operation to Differential Evolution," *Journal of Global Optimization*, vol. 27, no. 1, pp. 105–129, 2003.

[13] M. Ali, "Differential evolution with preferential crossover," *European Journal of Operational Research*, vol. 127, no. 3, pp. 1137–1147, September 2007, available at http://ideas.repec.org/a/eee/ejores/v181y2007i3p1137-1147.html.

[14] M. M. Ali and A. Törn, "Population set-based global optimization algorithms: some modifications and numerical studies," *Comput. Oper. Res.*, vol. 31, no. 10, pp. 1703–1725, 2004.

[15] H. A. Abbass, R. Sarker, and C. Newton, "PDE: A Pareto-frontier Differential Evolution Approach for Multi-objective Optimization Problems," in *Proceedings of the Congress on Evolutionary Computation 2001 (CEC'2001)*, vol. 2. Piscataway, New Jersey: IEEE Service Center, May 2001, pp. 971–978.

[16] T. Robič and B. Filipič, "DEMO: Differential Evolution for Multiobjective Optimization," in *Proceedings of the Third International Conference on Evolutionary Multi-Criterion Optimization – EMO 2005*, ser. Lecture Notes in Computer Science, vol. 3410. Springer, 2005, pp. 520–533.

[17] T. Tušar, P. Korošec, G. Papa, B. Filipič, and J. Šilc, "A comparative study of stochastic optimization methods in electric motor design," *Applied Intelligence*, vol. 2, no. 27, pp. 101–111, 2007, DOI 10.1007/s10489-006-0022-2.

[18] H.-G. Beyer and H.-P. Schwefel, "Evolution strategies — A comprehensive introduction," *Natural Computing*, vol. 1, pp. 3–52, 2002.

[19] T. Bäck, *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. New York: Oxford University Press, 1996.

[20] H.-P. Schwefel, *Evolution and Optimum Seeking*, ser. Sixth-Generation Computer Technology. New York: Wiley Interscience, 1995.

[21] A. E. Eiben and J. E. Smith, *Introduction to Evolutionary Computing (Natural Computing Series)*. Springer, November 2003.

[22] J. Holland, *Adaptation In Natural and Artificial Systems*. Ann Arbor: The University of Michigan Press, 1975.

[23] X. Yao, Y. Liu, and G. Lin, "Evolutionary Programming Made Faster," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 2, pp. 82–102, 1999.

[24] Y. Liu, X. Yao, Q. Zhao, and T. Higuchi, "Scaling up fast evolutionary programming with cooperative coevolution," in *Proceedings of the 2001 Congress on Evolutionary Computation CEC2001*. COEX, World Trade Center, 159 Samseong-dong, Gangnam-gu, Seoul, Korea: IEEE Press, 27-30 2001, pp. 1101–1108. [Online]. Available: citeseer.ist.psu.edu/liu01scaling.html

[25] A. E. Eiben, R. Hinterding, and Z. Michalewicz, "Parameter control in evolutionary algorithms," *IEEE Trans. on Evolutionary Computation*, vol. 3, no. 2, pp. 124–141, 1999. [Online]. Available: citeseer.ist.psu.edu/article/eiben00parameter.html

[26] J. Teo, "Exploring dynamic self-adaptive populations in differential evolution," *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, vol. 10, no. 8, pp. 673–686, 2006.

[27] J. Brest, S. Greiner, B. Bošković, M. Mernik, and V. Žumer, "Self-Adapting Control Parameters in Differential Evolution: A Comparative Study on Numerical Benchmark Problems," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 6, pp. 646–657, 2006, DOI: 10.1109/TEVC.2006.872133.

[28] A. Qin and P. Suganthan, "Self-adaptive Differential Evolution Algorithm for Numerical Optimization," in *Proceedings of the 2005 Congress on Evolutionary Computation (CEC'2005)*, vol. 2. IEEE Press, 2005, pp. 1785– 1791.

[29] J. Brest, V. Žumer, and M. S. Maučec, "Self-adaptive Differential Evolution Algorithm in Constrained Real-Parameter Optimization," in *The 2006 IEEE Congress on Evolutionary Computation CEC2006*. IEEE Press, 2006, pp. 919–926.

[30] J. Brest, B. Bošković, S. Greiner, V. Žumer, and M. S. Maučec, "Performance comparison of self-adaptive and adaptive differential evolution algorithms," *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, vol. 11, no. 7, pp. 617–629, 2007, DOI: 10.1007/s00500-006-0124-0.

[31] J. Brest and M. S. Maučec, "Population Size Reduction for the Differential Evolution Algorithm," *Applied Intelligence*, p. (Online), 2007, DOI: 10.1007/s10489-007-0091-x.

[32] J. Brest, "Differential Evolution with Self-Adaptation," *Encyclopedia of Artificial Intelligence*, pp. –, 2008, accepted.

[33] S. Y. Chong, P. Tino, and X. Yao, "Measuring generalization performance in co-evolutionary learning," *IEEE Transactions on Evolutionary Computation*, accepted in August 2007.

[34] P. Boggs and J. Tolle, "Sequential quadratic programming for large-scale nonlinear optimization," 2000. [Online]. Available: citeseer.comp.nus.edu.sg/265370.html

[35] N. Hansen and A. Ostermeier, "Completely Derandomized Self-Adaptation in Evolution Strategies," *Evolutionary Computation*, vol. 9, no. 2, pp. 159–195, 2001.

[36] V. L. Huang, A. K. Qin, and P. N. Suganthan, "Self-adaptive Differential Evolution Algorithm for Constrained Real-Parameter Optimization," in *2006 Congress on Evolutionary Computation (CEC'2006)*. IEEE Service Center, 2006, pp. 17–24.

[37] C. T. Lawrence, J. L. Zhou, and A. L. Tits, "User's Guide for CFSQP Version 2.5: A C Code for Solving (Large Scale) Constrained Nonlinear (Minimax) Optimization Problems, Generating Iterates Satisfying All Inequality Constraints," Institute for Systems Research, University of Maryland, College Park, MD 20742, Tech. Rep., 1997, TR-94-16r1.