Structured Population Size Reduction Differential Evolution with Multiple Mutation Strategies on CEC 2013 Real Parameter Optimization

Aleš Zamuda, Janez Brest
Faculty of Electrical Engineering and Computer Science
University of Maribor
Smetanova ul. 17, 2000 Maribor, Slovenia
Email: ales.zamuda@uni-mb.si

Efrén Mezura-Montes
Artificial Intelligence Department
University of Veracruz
Sebastián Camacho 5, Center, Xalapa, Veracruz 91000, MEXICO
Email: emezura@uv.mx

***Abstract*—This paper presents a differential evolution (DE) algorithm for real-parameter optimization. The algorithm includes the self-adaptive jDE algorithm with one of its strongest extensions, population reduction, combined with multiple mutation strategies using a structured population. The two mutation strategies used are run dependent on the population size, which is reduced with growing function evaluation number. The population is structured with a separate part where only DE/best strategy is executed and then the best vectors are exchanged with the main population part. Algorithm performance assessment results are presented for 10, 30, and 50 dimension settings for all of the 28 problems included in the Problem Definitions and Evaluation Criteria for the CEC 2013 Special Session and Competition on Real-Parameter Optimization.**

I. Introduction

In this paper, a Structured Population Size Reduction Differential Evolution with Multiple Mutation Strategies, denoted SPSRDEMMS, is proposed. SPSRDEMMS, which is an extension of $\text{jDE}_{\text{NP,MM}}$ [1], uses multiple mutation (subscript mark MM) strategies with another jDE extension, the population size (subscript mark NP) reduction [2]. The proposed algorithm is assessed on 10, 30, and 50 dimensions for all of the 28 problems included in the Problem Definitions and Evaluation Criteria for the CEC 2013 Special Session and Competition on Real-Parameter Optimization [3].

In the following section, the related work is described. In Section III, the proposed optimization algorithm is defined. After that, Section IV presents the optimization results for all test problems. Finally, in Section V, the conclusions of this work are drawn and guidelines for further work are given.

II. Related Work

In this section, the related work is described, first introducing the DE algorithm, followed by three of its extensions, jDE [4], jDE with population size reduction [2], and $\text{jDE}_{\text{NP,MM}}$ [1].

A. Differential Evolution Algorithm

Differential Evolution (DE) [5] is a global optimization algorithm which evolves a floating-point encoded population of vectors [6], which works on continuous and also discrete variables. Main performance advantages of DE over other evolutionary algorithms lie in floating-point encoding and a good combination of evolutionary operators, the mutation step size adaptation, and elitist selection [4], [7]. DE was initially proposed by Storn and Price [8], [5] and since then, it has been modified and extended several times with new early improvement versions proposed [9]-[13], followed by introducing several more strategies including control parameter adaptation and self-adaptation [14]-[24]. Among these, and ever more also the following DEs tried to structure the DE population so that the different DE mechanism operated and were selected based on e.g. target vector position (i.e. its index in population) within the specialized structure of the population [25]-[36]. Variants of DE have also won evolutionary algorithm competitions [37], [38]. DE was also introduced for multi-objective optimization [39]-[45]. Among real world problems tackled using jDE, there is the reconstruction of procedural tree models [46]-[49], optimized using differential evolution [50], [51].

The DE algorithm optimizes a population of vectors incrementally, i.e. it evolves the population towards more optimal solutions. The incremental optimization in DE is represented by its main evolution loop in which a population of vectors is computed for each its iteration step, denoted a generation. For each vector $\mathbf{x}_i$, $\forall i \in \{1, ..., NP\}$ in the current population, DE employs optimization mechanisms (i.e. evolutionary operators), namely mutation, crossover, and selection, to produce a trial vector (offspring) and to select a vector with the best fitness value. NP denotes current population size.

In a current generation G, using mutation operator DE generates a mutant vector $\mathbf{v}_{i,G+1}$ for each corresponding population vector. Among many proposed and referred to above, one of the most popular DE mutation strategies are the *'rand/1'* [9], [5]:

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

and the *'best/1'*:

$$\mathbf{v}_{i,G+1} = \mathbf{x}_{best,G} + F(\mathbf{x}_{r_1,G} - \mathbf{x}_{r_2,G}),$$

where the mutually different integers r_1, r_2, and r_3 randomly generated within the range $\{1, ..., NP\}$ represent the mutually different indexes and different from index i. The $\mathbf{x}_{best,G}$

denotes the currently best vector. F denotes an amplification factor of the difference vector within the interval $[0, 2]$, but usually less than 1. Vector at index r_1 is named a base vector. The term $\mathbf{x}_{r_2,G} - \mathbf{x}_{r_3,G}$ denotes a difference vector which after multiplication with F, is named amplified difference vector.

After creating the mutant vector $\mathbf{v}_{i,G+1}$, it is taken into recombination (crossover) process with the target vector $\mathbf{x}_{i,G}$. Crossover creates a trial vector $u_{i,j,G+1}$. One of the crossover operators in DE is the binary crossover. The binary crossover operates as follows:

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

where $j \in \{1, ..., D\}$ is the j-th search parameter of D-dimensional search space, $rand(0, 1) \in [0, 1]$ is a uniformly distributed random number. The j_{rand} is a uniform randomly chosen index of the search parameter, which is always exchanged to prevent cloning of target vectors. CR denotes the crossover rate for which the influence has been thoroughly studied in [52].

Finally after mutation and crossover, the selection operator propagates the fittest individual between (parent) target and trial vector to the new generation (for a minimization problem):

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

After the DE main loop is finished (e.g. a termination condition is satisfied, such as total number of function evaluations allowed or error precision is reached), the best vector is returned as a result of the algorithm.

B. The jDE Algorithm

The original DE algorithm [5] keeps all three control parameters fixed during the optimization process. However, finding reasonably good values for the control parameters of a given function is still an existing problem [53], [54]. As observed from the experiments in [4], the necessity of changing control parameters during the optimization process was confirmed.

In the jDE algorithm [4] which extends the original DE algorithm, a self-adaptive control mechanism is introduced to change the control parameters F and CR during the evolutionary process. The NP control parameter is kept unchanged in [4] and no additional structuring of the population is done. However, for encoding the population, each individual in the jDE population is extended using the values of F and CR control parameters. The better values for these (encoded) control parameters lead to better individuals which, in turn, are more likely to survive and produce offspring and, hence, propagate these better parameter values [4]. New control parameters $F_{i,G+1}$ and $CR_{i,G+1}$ are calculated for jDE as in [4]:

$$F_{i,G+1} = \begin{cases} F_{\mathrm{l}} + rand_1 \times F_{\mathrm{u}} & \text{if } rand_2 < \tau_1 \\ F_{i,G} & \text{otherwise} \end{cases}$$

$$CR_{i,G+1} = \begin{cases} rand_3 & \text{if } rand_4 < \tau_2 \\ CR_{i,G} & \text{otherwise.} \end{cases}$$

This produces control parameters F and CR in a new vector. The $rand_j \in [0, 1]$, $j \in \{1, 2, 3, 4\}$ are uniform random values. τ_1 and τ_2 represent the probabilities of adjusting control parameters F and CR, respectively. $\tau_1, \tau_2, F_{\mathrm{l}}, F_{\mathrm{u}}$ are taken fixed values as proposed in [4]. The new F takes a random value from $[0.1, 1.0]$. The new CR also takes a random value from $[0, 1]$. $F_{i,G+1}$ and $CR_{i,G+1}$ are obtained before the mutation operation is performed using these control parameters for a new vector. So the new F and CR influence the mutation, crossover, and selection operations of the new vector $\mathbf{x}_{i,G+1}$.

C. Population Size Reduction

Population sizing in DE has been studied by Teo et al. [54], [55]. As later reported in [28], population size reduction [2] is one of the best extensions for jDE. In [2], DE reduces population size to half at certain generations when the generation number exceeds the ratio between maximum number of function evaluations allowed and population size:

$$G_p > \frac{N_{\text{max\_Feval}}}{p_{\max} NP_p},$$

where $N_{\text{max\_Feval}}$ is the total number of function evaluations and NP_p is the population size reduced as of current generation in $p = \{1, ..., p_{\max}\}$ iteration of reduction. Vectors are compared and discarded index neighbor pairwise (i-th vector competes against $(i + NP)$-th vector for the new population size NP), as explained and drawn in [2].

D. Multiple Mutation Strategies

Zamuda et al. [1] introduced multiple mutation strategies into population size reduction differential evolution in $\text{jDE}_{\text{NP,MM}}$ algorithm. The $\text{jDE}_{\text{NP,MM}}$ algorithm extends the dynNP-DE [2] algorithm so that, besides the population reduction enhancement on top of jDE algorithm [4], the approach selects one among two mutation strategies to mutate one individual vector. The mutation strategies used in $\text{jDE}_{\text{NP,MM}}$ [1] are rand/1 and best/1.

The first mutation strategy, rand/1, is executed always when the population size is $NP \geq 100$, or also randomly for three in four mutation vector computations. When the population size is $NP < 100$, in one out of four cases, the best/1 strategy is executed. Therefore, the two mutation strategies adopted are selected based on the population size, which is reduced with a growing function evaluation number.

As noted in [1], the $\text{jDE}_{\text{NP,MM}}$ algorithm adds very few to the computational cost of the dynNP-DE (one random number generation and an if statement). Also, as noted in [2], the dynNP-DE follows the inspiration of the original DE and does not require many additional operations with respect to the original DE.

III. Structured Population Size Reduction Differential Evolution with Multiple Mutation Strategies

The proposed Structured Population Size Reduction Differential Evolution with Multiple Mutation Strategies, denoted SPSRDEMMS – is an extension of $\text{jDE}_{\text{NP,MM}}$ variant.

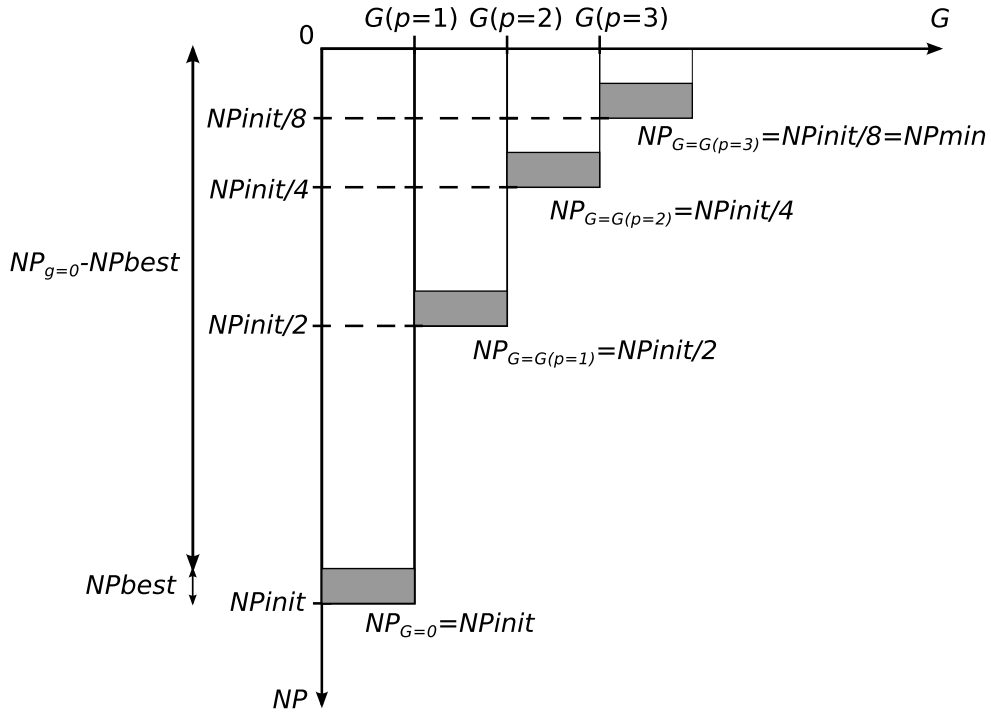

Fig. 1. Encoding aspect for population reduction and structuring of the population.

The SPSRDEMMS, for a fixed part of the population ($NPbest$ number of individuals at end of the entire population), executes only the best/1 strategy. This part of population (which might be seen as a sub-population) has a separate best vector index, $\mathbf{x}_{\text{best\_bestpop}}$. The first part of the population (**mainpop**) operates on target vectors $\mathbf{x}_i \in \{\mathbf{x}_1 ... \mathbf{x}_{NP-NPbest}\}$ and second part (**bestpop**) operates on target vectors $\mathbf{x}_i = \{\mathbf{x}_{NP-NPbest+1} ... \mathbf{x}_{NP}\}$. Both strategies generate mutation vectors using all vectors of the population $\mathbf{x}_1 ... \mathbf{x}_{NP}$, i.e. $r_1, r_2, r_3 \in \{1..NP\}$ The population structure with added population reduction mechanism is seen in Fig. 1.

The best vector is migrated from **mainpop** to the **bestpop** population to replace its best vector $\mathbf{x}_{\text{best\_bestpop}}$, each time before start of second population part generation processing, only if it is better. If migrated, this vector is also saved as $\mathbf{x}_{\text{xchg}}$ so that it can later be used for comparison for any improvement made in sub-population **bestpop**.

The **bestpop** best vector $\mathbf{x}_{\text{best\_bestpop}}$ is potentially migrated back to the first (**mainpop**) population part, only if *it is better than the best vector of the first population part* and *it has changed significantly since being migrated*, i.e. differs significantly from $\mathbf{x}_{\text{xchg}}$ vector. The significance of the difference between $\mathbf{x}_{\text{xchg}}$ and $\mathbf{x}_{\text{best\_bestpop}}$, when $\mathbf{x}_{\text{best\_bestpop}}$ is considered as a migration candidate back to the main population, is calculated as a distance ratio in the search space:

$$d = \frac{1}{D} \sum_{j=1}^{D} \frac{x^g_{\text{bestbestpop},j} - x_{\text{min},j}}{x^g_{\text{xchg},j} - x_{\text{min},j}}$$

where the upper and lower part of the fraction in the sum part could be divided by x_j min/max interval span, but since for both, it is eliminated. Then, if $d \notin [1 - d_\epsilon, 1 + d_\epsilon]$, the vector is migrated, replacing the best vector in the first population.

IV. Results

The proposed SPSRDEMMS algorithm is assessed considering the total number of function evaluations allowed, differing for the 10, 30, and 50 dimensions, for all of the 28 problems (84 total, considering varying dimensions) included in the Problem Definitions and Evaluation Criteria for the CEC 2013 Special Session and Competition on Real-Parameter Optimization [3], based on existing functions [56], [57], [58]. The search range for the functions is set [-100,100]^D and the initial population is initialized at random with a Uniform distribution. The implementation used to run these functions (total 1.2852e+9 evaluation calls) is included in Linux toolkit for [3]. In this paper, functions implementation for C++ programming language is used.

A. Parameter Settings

The parameter settings used for SPSRDEMMS are as follows. The d_ϵ takes value 0.5. Also, since of two populations now, minimal population size $NPmin$ for reduction is kept at 10 and $NPbest$ is always assured 4 or above. The initial population size is set 100 and initial $NPbest$ is set to 6. The p_{max} is set to 4 and for $p \in \{0, 1, 2, 3\}$, the population size takes values of $NP_p \in \{100, 50, 25, 12\}$. Reductions are executed in generations differing among $N_{\text{max\_Feval}}$, i.e. $G_p(D = 10) \in \{250, 500, 1000\}$, $G_p(D = 30) \in \{750, 1500, 3000\}$, $G_p(D = 30) \in \{1250, 2500, 5000\}$ for $D \in \{10, 20, 30\}$, respectively.

B. Function error values

The results of 51 independent runs for our algorithm on these functions are gathered, i.e. a total of 4284 SPSRDEMMS algorithm runs. For each function, three different stopping

TABLE I. BEST, WORST, MEDIAN, AVERAGE, AND STANDARD DEVIATION FOR 51 INDEPENDENT RUNS ON 28 CEC 2013 PROBLEMS FOR DIMENSION 10.

Fun.	FES	Best	Worst	Median	Average	Std. dev.
F1	100000	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00
F2	100000	5.9171e-02	4.1989e+03	2.6937e+02	6.8876e+02	1.0151e+03
F3	100000	0.0000e+00	9.7415e+01	2.1116e-02	5.9733e+00	1.7551e+01
F4	100000	1.2188e-05	3.9763e-01	9.8397e-03	3.8703e-02	7.8677e-02
F5	100000	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00
F6	100000	0.0000e+00	9.8124e+00	9.8124e+00	8.6580e+00	3.1929e+00
F7	100000	1.4673e-03	1.3469e+00	5.6118e-02	1.8732e-01	2.7361e-01
F8	100000	2.0134e+01	2.0467e+01	2.0355e+01	2.0343e+01	7.7078e-02
F9	100000	4.5892e-01	5.1789e+00	2.5868e+00	2.7311e+00	1.1258e+00
F10	100000	7.3960e-03	2.7063e-01	9.5967e-02	1.0346e-01	6.2944e-02
F11	100000	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00
F12	100000	1.9899e+00	1.2934e+01	6.9647e+00	7.5821e+00	2.8126e+00
F13	100000	1.9899e+00	2.4145e+01	1.1525e+01	1.1042e+01	5.4391e+00
F14	100000	0.0000e+00	2.4982e-01	6.2454e-02	8.3273e-02	7.2476e-02
F15	100000	3.1037e+02	1.4624e+03	7.2909e+02	7.6305e+02	2.7109e+02
F16	100000	5.5593e-01	1.6109e+00	1.1419e+00	1.0981e+00	2.4198e-01
F17	100000	1.0122e+01	1.0155e+01	1.0122e+01	1.0127e+01	8.7131e-03
F18	100000	1.1815e+01	3.7132e+01	2.0086e+01	2.1369e+01	5.6251e+00
F19	100000	1.4657e-01	4.1864e-01	3.0112e-01	2.9026e-01	5.9477e-02
F20	100000	1.8134e+00	3.5117e+00	2.4922e+00	2.5032e+00	3.9304e-01
F21	100000	2.0000e+02	4.0019e+02	4.0019e+02	3.9234e+02	3.9246e+01
F22	100000	5.0156e+00	1.3042e+02	1.0000e+02	6.6219e+01	4.2508e+01
F23	100000	1.5447e+02	1.4088e+03	9.6324e+02	9.3077e+02	2.7164e+02
F24	100000	1.1340e+02	2.1702e+02	2.0810e+02	2.0442e+02	1.9832e+01
F25	100000	1.1048e+02	2.1465e+02	2.0734e+02	2.0473e+02	1.6309e+01
F26	100000	1.0497e+02	2.0002e+02	2.0002e+02	1.6886e+02	3.8350e+01
F27	100000	3.0020e+02	5.1646e+02	4.8536e+02	4.7300e+02	4.6988e+01
F28	100000	1.0000e+02	3.0000e+02	3.0000e+02	2.8431e+02	5.4305e+01

TABLE II. BEST, WORST, MEDIAN, AVERAGE, AND STANDARD DEVIATION FOR 51 INDEPENDENT RUNS ON 28 CEC 2013 PROBLEMS FOR DIMENSION 30.

Fun.	FES	Best	Worst	Median	Average	Std. dev.
F1	300000	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00
F2	300000	2.1343e+04	3.3170e+05	9.7919e+04	1.0157e+05	5.3019e+04
F3	300000	2.1473e+03	5.7884e+07	6.7165e+06	1.0951e+07	1.3944e+07
F4	300000	2.0301e-02	1.4685e+01	1.2318e+00	2.4061e+00	3.2842e+00
F5	300000	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00
F6	300000	8.6100e+00	6.8926e+01	1.2242e+01	1.7463e+01	1.1434e+01
F7	300000	2.4500e+00	2.5063e+01	8.8577e+00	1.1038e+01	6.1823e+00
F8	300000	2.0820e+01	2.1025e+01	2.0965e+01	2.0950e+01	5.0122e-02
F9	300000	1.6308e+01	2.9929e+01	2.5720e+01	2.4903e+01	3.4466e+00
F10	300000	7.3960e-03	2.2653e-01	4.4288e-02	5.3974e-02	4.0192e-02
F11	300000	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00
F12	300000	1.9899e+01	8.2454e+01	4.2783e+01	4.2650e+01	1.3596e+01
F13	300000	3.1302e+01	1.3662e+02	7.9245e+01	7.9763e+01	2.0901e+01
F14	300000	8.3277e-02	4.1341e+01	1.2841e+00	3.2550e+00	6.1293e+00
F15	300000	2.7540e+03	6.5708e+03	4.4099e+03	4.4226e+03	6.9797e+02
F16	300000	7.2146e-01	2.9264e+00	2.3275e+00	2.2801e+00	3.7448e-01
F17	300000	3.0434e+01	3.0459e+01	3.0437e+01	3.0440e+01	7.5842e-03
F18	300000	4.7969e+01	1.2977e+02	9.2933e+01	8.9310e+01	2.0696e+01
F19	300000	5.3421e-01	1.5004e+00	1.2035e+00	1.1639e+00	1.7879e-01
F20	300000	1.0024e+01	1.2276e+01	1.1179e+01	1.1236e+01	5.2488e-01
F21	300000	2.0000e+02	4.4354e+02	3.0000e+02	2.8466e+02	6.9430e+01
F22	300000	1.2321e+01	1.3731e+02	1.1335e+02	7.6606e+01	4.8794e+01
F23	300000	2.1334e+03	6.8424e+03	4.8773e+03	4.7713e+03	7.7505e+02
F24	300000	2.3705e+02	2.7283e+02	2.5176e+02	2.5330e+02	8.9761e+00
F25	300000	2.4513e+02	2.8247e+02	2.6435e+02	2.6408e+02	8.3439e+00
F26	300000	2.0000e+02	2.0002e+02	2.0001e+02	2.0001e+02	4.9445e-03
F27	300000	6.8716e+02	1.0970e+03	8.8741e+02	8.8779e+02	9.1952e+01
F28	300000	3.0000e+02	3.0000e+02	3.0000e+02	3.0000e+02	3.9382e-13

TABLE III. BEST, WORST, MEDIAN, AVERAGE, AND STANDARD DEVIATION FOR 51 INDEPENDENT RUNS ON 28 CEC 2013 PROBLEMS FOR DIMENSION 50.

Fun.	FES	Best	Worst	Median	Average	Std. dev.
F1	500000	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00
F2	500000	2.5465e+05	1.3973e+06	5.3195e+05	5.6496e+05	2.5644e+05
F3	500000	1.2823e+06	2.1525e+08	2.6915e+07	4.4377e+07	4.5567e+07
F4	500000	4.9308e-01	2.1442e+01	3.9574e+00	5.1748e+00	4.7243e+00
F5	500000	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00
F6	500000	4.3447e+01	4.9127e+01	4.3447e+01	4.3678e+01	1.1124e+00
F7	500000	1.0482e+01	5.3925e+01	3.0492e+01	3.1677e+01	9.2013e+00
F8	500000	2.1005e+01	2.1187e+01	2.1138e+01	2.1132e+01	4.1905e-02
F9	500000	3.3732e+01	5.8776e+01	5.1229e+01	5.1236e+01	4.9666e+00
F10	500000	0.0000e+00	1.6976e-01	4.6796e-02	5.8648e-02	3.8884e-02
F11	500000	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00
F12	500000	5.5718e+01	1.3930e+02	8.5566e+01	8.7415e+01	2.1639e+01
F13	500000	1.1664e+02	2.2781e+02	1.5702e+02	1.5857e+02	2.5973e+01
F14	500000	3.8071e+00	5.9535e+01	1.5021e+01	2.0690e+01	1.3473e+01
F15	500000	6.4234e+03	1.0591e+04	8.6445e+03	8.6298e+03	8.4749e+02
F16	500000	1.0699e+00	3.9705e+00	2.9339e+00	2.8300e+00	6.0148e-01
F17	500000	5.0786e+01	5.0904e+01	5.0805e+01	5.0813e+01	2.4401e-02
F18	500000	8.5021e+01	2.3909e+02	1.5610e+02	1.5727e+02	3.6453e+01
F19	500000	1.3298e+00	2.5355e+00	1.9956e+00	1.9584e+00	3.1293e-01
F20	500000	1.8976e+01	2.1918e+01	2.0500e+01	2.0555e+01	7.9209e-01
F21	500000	2.0000e+02	1.1224e+03	2.0000e+02	6.0597e+02	4.4225e+02
F22	500000	2.1628e+01	1.5740e+02	3.0921e+01	3.9409e+01	2.9742e+01
F23	500000	6.7776e+03	1.0525e+04	8.8580e+03	8.9068e+03	9.5617e+02
F24	500000	2.8150e+02	3.3938e+02	3.1048e+02	3.1105e+02	1.4298e+01
F25	500000	3.0723e+02	3.5924e+02	3.3517e+02	3.3510e+02	1.2868e+01
F26	500000	2.0002e+02	4.5188e+02	2.0013e+02	2.8720e+02	1.0995e+02
F27	500000	1.2702e+03	1.8119e+03	1.5494e+03	1.5281e+03	1.4526e+02
F28	500000	4.0000e+02	3.4499e+03	4.0000e+02	7.5373e+02	9.7844e+02

TABLE IV. COMPUTATIONAL COMPLEXITY.

T_0	Dim.	T_1	$\widehat{T_2}$	$(\widehat{T_2} - T_1)/T_0$
0.06 s	D=10	0.2976 s	0.378 s	21.1694
0.06 s	D=30	0.8904 s	1.1172 s	20.9119
0.06 s	D=50	1.5396 s	1.878 s	20.33

conditions are used, i.e. for three different number of total function evaluations (and dimensions, D) allowed, 100000 ($D = 10$), 300000 ($D = 30$), and 500000 ($D = 50$), respectively. In Tables I–III, best, worst, median, average error values and their standard deviation for our SPSRDEMMS algorithm are reported for all these functions on different dimensions.

C. Algorithm Complexity

The computing system used to run and assess the algorithm is 4-processor Intel(R) Core(TM) i5-3550 CPU @ 3.30GHz with 8GB RAM, Linux 3.2.0-32-generic #51-Ubuntu SMP x86_64, with g++ 4.6.3-1ubuntu5 using compilation flag O3, and algorithm run in serial mode. Table IV shows, according to [3], the timing values T_0, T_1, averaged $\widehat{T}_2$, and complexity $(\widehat{T}_2 - T_1)/T_0$ compared to a simple do-undo arithmetic operations benchmark program from [3]. The C++ implementation for this program used is seen in Fig. 2 and it has been compiled and run on the 64-bit machine as noted in its two starting comment lines.

V. CONCLUSION

A new algorithm SPSRDEMMS was presented in this paper, combining structured population with multiple mutation strategies and population size reduction from self-adaptive differential evolution algorithm jDE. The performance of the algorithm was assessed on 28 real parameter functions on three different dimension settings from a competition at Congress on Evolutionary Computation (CEC) 2013.

In the future research, additional measurements to compare the proposed extension in the new algorithm could be done with existing algorithms. More strategies inclusion like [25] could be addressed. A population size control and best strategy mutation enhancement are two interesting goals to pursue for our algorithm, too. Also, additional real-world applications of the algorithm could be done, especially to tune a forest ecosystem growth factor model [47] to real gathered biological data [59].

ACKNOWLEDGMENT

This work was supported in part by the Slovenian Research Agency under program P2-0041. Workload also run on SLING.SI infrastructure under European Grid Infrastructure.

REFERENCES

[1] A. Zamuda and J. Brest, "Population Reduction Differential Evolution with Multiple Mutation Strategies in Real World Industry Challenges," in *Swarm and Evolutionary Computation*, ser. Lecture Notes in Computer Science, L. Rutkowski, M. Korytkowski, R. Scherer, R. Tadeusiewicz, L. Zadeh, and J. Zurada, Eds. Springer Berlin / Heidelberg, 2012, pp. 154–161.

[2] J. Brest and M. S. Maučec, "Population Size Reduction for the Differential Evolution Algorithm," *Applied Intelligence*, vol. 29, no. 3, pp. 228–247, 2008.

[3] J. J. Liang, Q. B. Y., and S. P. N., "Problem Definitions and Evaluation Criteria for the CEC 2013 Special Session and Competition on Real-Parameter Optimization," Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China And Nanyang Technological University, Singapore, Technical Report 201212, 2013.

```
// g++ -O2 m.cpp
// ./a.out: y=1.0000e+00    t 0.0600
#include <iostream>
#include <iomanip>
#include <cmath>
using namespace std;

double diffclock_min(
  const clock_t &clock1,
  const clock_t &clock2) {
  double diffticks = clock1 - clock2;
  double diffms = double(diffticks)
                  / CLOCKS_PER_SEC;
  return diffms;
}

void CPU_workload(double &x) {
  double y;
  for (int i = 1; i <= 1000000; i++) {
          x = 0.55 + (double) i;
          x = x+x;x = x/2; x=x*x;
          x=sqrt(x); x=log(x);
          x=exp(x); y=x/x;
  }
  x=y;
}

int main() {
  double time;
  clock_t begin, end;
  double x;

  begin = clock();
  CPU_workload(x);
  end = clock();

  time = diffclock_min(end, begin);
  cout << "y=" << setprecision(4)
       << scientific << x;

  cout << setprecision(4) << fixed
       << "\tt " << time << endl;
}
```

Fig. 2. Time estimation T_0 routine in C++.

[4] 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.

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

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

[7] E. Mezura-Montes and B. C. Lopez-Ramirez, "Comparing bio-inspired algorithms in constrained optimization problems," *The 2007 IEEE Congress on Evolutionary Computation*, pp. 662–669, 25-28 Sept. 2007.

[8] 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.

[9] K. V. Price, R. M. Storn, and J. A. Lampinen, *Differential Evolution: A Practical Approach to Global Optimization*, ser. Natural Computing Series. Berlin, Germany: Springer-Verlag, 2005.

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

[11] E. Mezura-Montes, J. Velázquez-Reyes, and C. A. C. Coello, "A comparative study of differential evolution variants for global optimization," in *GECCO 2006: Proceedings of the 8th annual conference on Genetic and evolutionary computation*. New York, NY, USA: ACM Press, 2006, pp. 485–492.

[12] D. Tasoulis, N. Pavlidis, V. Plagianakos, and M. Vrahatis, "Parallel differential evolution," in *Congress on Evolutionary Computation (CEC 2004)*, vol. 2, 2004, pp. 2023–2029.

[13] D. Zaharie, "A multipopulation differential evolution algorithm for multimodal optimization," in *Proceedings of Mendel 2004, 10th International Conference on Soft Computing*, Brno, 2004, pp. 17–22.

[14] 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.

[15] A. Zamuda, J. Brest, B. Bošković, and V. Žumer, "Large Scale Global Optimization Using Differential Evolution with Self Adaptation and Cooperative Co-evolution," in *2008 IEEE World Congress on Computational Intelligence*. IEEE Press, 2008, pp. 3719–3726.

[16] ——, "Differential Evolution with Self-adaptation and Local Search for Constrained Multiobjective Optimization," in *IEEE Congress on Evolutionary Computation 2009*. IEEE Press, 2009, pp. 195–202.

[17] A. K. Qin, V. L. Huang, and P. N. Suganthan, "Differential evolution algorithm with strategy adaptation for global numerical optimization," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 2, pp. 398–417, 2009.

[18] J. Zhang and A. C. Sanderson, "JADE: adaptive differential evolution with optional external archive," *Trans. Evol. Comp*, vol. 13, no. 5, pp. 945–958, 2009.

[19] J. Tvrdík, "Adaptation in differential evolution: A numerical comparison," *Applied Soft Computing*, vol. 9, no. 3, pp. 1149–1155, 2009.

[20] S. Das, A. Abraham, U. Chakraborty, and A. Konar, "Differential Evolution Using a Neighborhood-based Mutation Operator," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 3, pp. 526–553, 2009.

[21] E. Mezura-Montes and A. G. Palomeque-Ortiz, "Parameter control in differential evolution for constrained optimization," in *Evolutionary Computation, 2009. CEC'09. IEEE Congress on*. IEEE, 2009, pp. 1375–1382.

[22] E. Mezura-Montes and A. Palomeque-Ortiz, "Self-adaptive and deterministic parameter control in differential evolution for constrained optimization," *Constraint-Handling in Evolutionary Optimization*, pp. 95–120, 2009.

[23] E. Mezura-Montes, M. E. Miranda-Varela, and R. del Carmen Gómez-Ramón, "Differential evolution in constrained numerical optimization: an empirical study," *Information Sciences*, vol. 180, no. 22, pp. 4223–4262, 2010.

[24] D. Jia, X. Duan, and M. K. Khan, "An efficient binary differential evolution with parameter adaptation," *International Journal of Computational Intelligence Systems*, vol. 6, no. 2, pp. 328–336, 2013.

[25] Y. Wang, Z. Cai, and Q. Zhang, "Differential evolution with composite trial vector generation strategies and control parameters," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 1, pp. 55–66, 2011.

[26] E. Mininno, F. Neri, F. Cupertino, and D. Naso, "Compact Differential Evolution," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 1, pp. 32–54, 2011.

[27] R. Mallipeddi, P. N. Suganthan, Q. K. Pan, and M. F. Tasgetiren, "Differential evolution algorithm with ensemble of parameters and mutation strategies," *Applied Soft Computing*, vol. 11, no. 2, pp. 1679–1696, 2011.

[28] F. Neri and V. Tirronen, "Recent Advances in Differential Evolution: A Survey and Experimental Analysis," *Artificial Intelligence Review*, vol. 33, no. 1–2, pp. 61–106, 2010.

[29] S. Das and P. N. Suganthan, "Differential Evolution: A Survey of

the State-of-the-art," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 1, pp. 4–31, 2011.

[30] A. Ghosh, S. Das, A. Chowdhury, and R. Giri, "An improved differential evolution algorithm with fitness-based adaptation of the control parameters," *Information Sciences*, vol. 181, no. 18, pp. 3749–3765, 2011.

[31] S. Ghosh, S. Roy, S. Islam, S. Das, and P. Suganthan, "A differential covariance matrix adaptation evolutionary algorithm for global optimization," in *Differential Evolution (SDE), 2011 IEEE Symposium on*, 2011, pp. 1 –8.

[32] S. Ghosh, S. Das, A. Vasilakos, and K. Suresh, "On convergence of differential evolution over a class of continuous functions with unique global optimum," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 42, no. 1, pp. 107–124, 2012.

[33] H. Wang, S. Rahnamayan, H. Sun, and M. G. H. Omran, "Gaussian bare-bones differential evolution," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 2012.

[34] L.-A. Gordián-Rivera and E. Mezura-Montes, "A combination of specialized differential evolution variants for constrained optimization," in *Advances in Artificial Intelligence–IBERAMIA 2012*. Springer, 2012, pp. 261–270.

[35] J. Zhu and X. Yan, "Adaptive variable space differential evolution algorithm based on population distribution," *Memetic Computing*, vol. 5, no. 1, pp. 49–64, 2013.

[36] X. Wang and S. Zhao, "Differential evolution algorithm with self-adaptive population resizing mechanism," *Mathematical Problems in Engineering Mathematical Problems in Engineering*, Accepted 30 January 2013.

[37] J. Brest, A. Zamuda, B. Bošković, M. S. Maučec, and V. Žumer, "Dynamic Optimization using Self-Adaptive Differential Evolution," in *IEEE Congress on Evolutionary Computation 2009*. IEEE Press, 2009, pp. 415–422.

[38] J. Brest, P. Korošec, J. Šilc, A. Zamuda, B. Bošković, and M. S. Maučec, "Differential evolution and differential ant-stigmergy on dynamic optimisation problems," *International Journal of Systems Science*, vol. 44, pp. 663–679, 2013.

[39] 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*, vol. 2. Piscataway, New Jersey: IEEE Service Center, 2001, pp. 971–978.

[40] F. Xue, A. C. Sanderson, and R. J. Graves, "Pareto-based Multi-Objective Differential Evolution," in *Proceedings of the 2003 Congress on Evolutionary Computation*, vol. 2. Canberra, Australia: IEEE Press, 2003, pp. 862–869.

[41] S. Kukkonen and J. Lampinen, "GDE3: The third Evolution Step of Generalized Differential Evolution," in *2005 IEEE Congress on Evolutionary Computation (CEC 2005)*, vol. 1. Edinburgh, Scotland: IEEE Service Center, 2005, pp. 443–450.

[42] 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.

[43] 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.

[44] A. W. Iorio and X. Li, "Incorporating Directional Information within a Differential Evolution Algorithm for Multi-objective Optimization," in *2006 Genetic and Evolutionary Computation Conference (GECCO 2006)*, M. K. et al., Ed., vol. 1. Seattle, Washington, USA: ACM Press. ISBN 1-59593-186-4, 2006, pp. 691–697.

[45] A. Zhou, B.-Y. Qu, H. Li, S.-Z. Zhao, P. N. Suganthan, and Q. Zhang, "Multiobjective evolutionary algorithms: A survey of the state of the art," *Swarm and Evolutionary Computation*, vol. 1, no. 1, pp. 32–49, 2011.

[46] J. P. Weber, "Fast Simulation of Realistic Trees," *Computer Graphics and Applications, IEEE*, vol. 28, no. 3, pp. 67–75, 2008.

[47] A. Zamuda and J. Brest, "Environmental framework to visualize emergent artificial forest ecosystems," *Information Sciences*, vol. 220, pp. 522–540, 2013, DOI 10.1016/j.ins.2012.07.031.

[48] J. Phattaralerphong and H. Sinoquet, "A method for 3D reconstruction of tree crown volume from photographs: assessment with 3D-digitized plants," *Tree Physiology*, vol. 25, no. 10, pp. 1229–1242, 2005.

[49] L. Quan, *Image-Based Modeling*, 1st ed. Springer Publishing Company, Incorporated, 2010.

[50] A. Zamuda, J. Brest, B. Bošković, and V. Žumer, "Differential Evolution for Parameterized Procedural Woody Plant Models Reconstruction," *Applied Soft Computing*, vol. 11, no. 8, pp. 4904–4912, 2011.

[51] A. Zamuda and J. Brest, "Tree Model Reconstruction Innovization Using Multi-objective Differential Evolution," in *2012 IEEE World Congress on Computational Intelligence (IEEE WCCI 2012)*. Brisbane, Australia: IEEE Press, 2012, pp. 2827–2834.

[52] D. Zaharie, "Influence of crossover on the behavior of Differential Evolution Algorithms," *Applied Soft Computing*, vol. 9, no. 3, pp. 1126–1138, 2009.

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

[54] 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.

[55] N. Teng, J. Teo, and M. H. A. Hijazi, "Self-adaptive population sizing for a tune-free differential evolution," *Soft Computing*, vol. 13, no. 7, pp. 709–724, 2009.

[56] 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, URL http://www.ntu.edu.sg/home/EPNSugan, Tech. Rep., 2005.

[57] J. J. Liang, P. N. Suganthan, and K. Deb, "Novel composition test functions for numerical global optimization," in *Proceedings of the IEEE International Swarm Intelligence Symposium*, 2005, pp. 68–75.

[58] R. R. N. Hansen, S. Finck and A. Auger, "Real-Parameter Black-Box Optimization Benchmarking 2010: Noiseless Functions Definitions," INRIA, Tech. Rep. RR-6829, March 24, 2012.

[59] U. Vilhar, M. Starr, K. Katzensteiner, P. Simončič, L. Kajfež-Bogataj, and J. Diaci, "Modelling drainage fluxes in managed and natural forests in the dinaric karst: a model comparison study," *European Journal of Forest Research*, vol. 129, no. 4, pp. 729–740, 2010.