A Switched Parameter Differential Evolution for Large Scale Global Optimization – Simpler May Be Better

Swagatam Das, Arka Ghosh and Sankha Subhra Mullick

Abstract In this article we present two very simple modifications to Differential Evolution (DE), one of the most competitive evolutionary algorithms of recent interest, to enhance its performance for the high-dimensional numerical functions while still preserving the simplicity of its algorithmic framework. Instead of resorting to complicated parameter adaptation schemes or incorporating additional local search methods, we present a simple strategy where the values of the scale factor (mutation step size) and crossover rate are switched in a uniformly random way between two extreme corners of their feasible ranges for different population members. Also each population member is mutated either by using the DE/rand/1 scheme (where the base vector to be perturbed is a randomly chosen member from the population) or by using the DE/best/1 scheme (where the base vector is the best member of the population). The population member is subjected to that mutation strategy which was responsible for the last successful update at the same population index under consideration. Our experiments based on the benchmark functions proposed for the competitions on large-scale global optimization with bound constraints held under the IEEE CEC (Congress on Evolutionary Computation) 2008 and 2010 competitions indicate that the basic DE algorithm with these simple modifications can indeed achieve very competitive results against the currently best known algorithms.

Keywords Continuous optimization · Differential evolution · Large scale optimization · Success counter

S. Das (✉) · A. Ghosh · S.S. Mullick
Indian Statistical Institute, 203 B. T. Road, Kolkata 700 108, India
e-mail: swagatam.das@isical.ac.in

A. Ghosh
e-mail: arka_t@isical.ac.in

S.S. Mullick
e-mail: mullicksankhasubhra@gmail.com

1 Introduction

Over the past few decades, several families of evolutionary computing algorithms have been proposed for solving bound-constrained global optimization problems. Performances of these algorithms remain considerably good for problems with moderate number of decision variables or dimensions. However, most of them face difficulties in locating the global optimum with sufficient accuracy and without consuming too much Function Evaluations (FEs) as the number of dimensions of the search space increases beyond 100 or so. This is not surprising and is primarily caused by the exponential increase of the search volume with dimensions. Consider placing 100 points onto a real interval, say [0,1]. To obtain a similarly dense coverage, in terms of distance between adjacent points, the 10-dimensional space $[0,1]^{10}$ would require $100^{10} = 10^{20}$ points. The previously mentioned 100 points now appear as isolated points in a vast empty space. Usually the distance measures break down in higher dimensionalities and a search strategy that is valuable in small dimensions might be useless in large or even moderate dimensional search spaces.

Many real world problems demand optimization of a large number of variables. A few typical examples of such problems are shape optimization [1, 2], high-dimensional waveform inversion [3], and large scale economic load dispatch (involving 140 units or more) [4]. Recently researchers have been paying attention to the issue of designing scalable nature-inspired optimization techniques for optimizing very high dimensional functions. The ongoing interest of the scientific community is also evident from participations in the competitions on large scale single objective global optimization with bound constraints held under the IEEE CEC (Congress on Evolutionary Computation) 2008 and 2010 [5, 6].

The evolutionary methods for high-dimensional global optimization problems can be roughly categorized into three classes: the cooperative co-evolutionary methods, the micro Evolutionary Algorithms (EAs) and the Local Search (LS) based methods. Cooperative Co-Evolutionary Algorithms (CCEAs) [7–11] are well-known for solving high dimensional optimization problems and they result from an automatic divide and conquer approach. Recently some promising Cooperative Co-evolutionary (CC) algorithms were proposed like the CC versions of the Particle Swarm Optimization (CCPSO and CCPSO2) [10] and CC with Differential Grouping [11]. Micro–EAs (see for example [12–15]) are instances of typical EAs characterized by small population size and often simple fitness functions. Different forms of Memetic Algorithms (MAs) [16–19] developed by combining an LS method with a global evolutionary optimizer have been frequently applied to solve large scale function optimization problems.

Differential Evolution (DE) [20, 21] currently stands out as a very competitive evolutionary optimizer for continuous search spaces. Several attempts have been made to improve the performance of DE for moderate to high dimensional function optimization problems. Some noted DE-variants of current interest involve success-history based parameter adaptation strategies (like SaDE [22], JADE [23]), new mutation and crossover strategies (like Pro-DE [24], MDE-pBX [25]), and combining various offspring generation strategies (CoDE [26], EPSDE [27] etc.). For high-dimensional problems (more than 500 dimensions) DE has been adopted by methods encompassing all the three algorithmic philosophies outlined above. Owing to its inherent simplicity, DE was used as the base optimizer in Yang et al.'s first work [9] on random grouping based CCEAs. Zamuda et al. [28] extended DE by log-normal self-adaptation of its control parameters and by using cooperative co-evolution as a dimensional decomposition mechanism. Parsopoulos developed a cooperative micro-DE [29] for large scale global optimization. The self-adaptive DE was hybridized with MTS for large scale optimization by Zhao et al. [30]. Some other approaches of improving DE for high-dimensional function optimization can be found in [31–34].

We can see that in order to cope with the growing complexity of the problems to be solved, DE has been subjected to several modifications. However, despite the reported performance improvements, the improved DE algorithms are very often lacking one very important thing, that is the simplicity of the DE framework – the very reason why DE was and is loved by the practitioners of evolutionary computation. The present work is motivated by the question that can we improve DE for very high dimensional search spaces by simple parameter control strategies and by combining the basic ingredients of DE without any additional computation overheads (likely to be caused by external achieves, proximity and rank based parent selection schemes, additional local search schemes, keeping the long records of successful individuals etc.).

In this paper we present a simple DE scheme where the two crucial control-parameters of DE, namely the scale factor (equivalent to the mutation step size) F and the crossover rate Cr are switched between their respective limiting values in a uniformly random manner for each offspring generation process. Also each population member is mutated using either the DE/best/1 strategy or the DE/rand/1 strategy. The difference between these two strategies lies in the selection of the base vector to be perturbed. In case of DE/best/1, the base vector that has to be perturbed with the scaled difference of any two distinct population members is the best vector in the population yielding greatest fitness (i.e. smallest objective function value for a minimization problem). On the other hand, for the DE/rand/1 scheme, the base vector is a randomly chosen member from the current population. Each individual undergoes either of the two possible mutation strategies based on which strategy generated a successful offspring (which replaced the parent during selection) last time for the same population index. Thus, the choice of the mutation strategy depends on a unit length success memory of the record of just the last successful update. The proposed algorithm requires no tunable control parameter and is very easy to implement.

Switching of the scale factor between two extreme values (here 0.5 and 2) provides scopes for coarse search of large regions as well as refined search of smaller basins of attraction. Similarly by switching Cr values between 0 and 1, a balance between coordinate-wise search and generation of rotationally invariant search moves can be stricken. Our experiments indicate that the simple parameter switching coupled with the mixing of DE/best/1 and DE/rand/1 strategies can significantly improve the performance of DE on the high-dimensional function optimization problems. This conclusion is reached through a rigorous performance comparison of the proposed DE scheme with that of some of the most well-known large-scale optimizers including the winners of the two CEC competitions. While it is very hard (if not impossible) to analytically justify the suitability of the simple changes made to DE, we undertake some empirical studies based on the population spread and diversity to highlight the effectiveness of each of the modifications suggested.

2 The DE Algorithm

The initial generation of a standard DE algorithm consists of the four basic steps – initialization, mutation, recombination or crossover, and selection, of which, only last three steps are repeated into the subsequent DE generations. The generations continue till some termination criterion (such as exhaustion of maximum functional evaluations) is satisfied.

2.1 Initialization

DE begins search for the global optima in the D dimensional real parameter space by initiation of a random population of Np real-valued vectors whose components represent the D parameters of the optimization problem. A generalized notation used to identify the ith solution (real parameter vector) of the present generation G can be shown as:

$$\vec{X}_{i,G} = [x_{1,i,G}, x_{2,i,G}, \ldots x_{D,i,G}].$$

Given the decision space bounds, $\vec{X}_{\max} = [x_{1,\max}, x_{2,\max}, \ldots x_{D,\max}]$ and $\vec{X}_{\min} = [x_{1,\min}, x_{2,\min}, \ldots x_{D,\min}]$, the jth dimension of ith individual can be initialized as:

$$x_{i,j} = x_{j,\min} + rand_{i,j} \times \left(x_{j,\max} - x_{j,\min}\right), \tag{1}$$

where $rand_{i,j}$ is a uniformly distributed random number lying in the range [0, 1] and it is instantiated anew for each ordered pair (i, j).

2.2 Mutation

In DE terminology, a population member (say i) of the current generation, known as the target vector, is chosen and is *differentially* mutated with the scaled difference vector(s) $(\vec{X}_{r_1,G} - \vec{X}_{r_2,G})$ to produce a mutant or *donor vector*. It is to be noted that the indices r_1 and r_2 are sampled from {1,2,…, Np} are different from the running index i and $(r_1, r_2 \in \{1, 2, \ldots Np\} \backslash \{i\})$. A scaling factor F, usually lying in the range [0.4, 2], scales the difference vector(s). Two commonly used DE mutation strategies are listed below:

$$DE/rand/1: \vec{V}_{i,G} = \vec{X}_{r_1,G} + F\left(\vec{X}_{r_2,G} - \vec{X}_{r_3,G}\right), \tag{2a}$$

$$DE/best/1: \vec{V}_{i,G} = \vec{X}_{best,G} + F.\left(\vec{X}_{r_1,G} - \vec{X}_{r_2,G}\right), \tag{2b}$$

where r_1, r_2, r_3 are mutually exclusive indices that are stochastically selected from $\{1, 2, \ldots, Np\}$.

2.3 Crossover

In DE, the crossover step aims to combine the individual components of the parent and the mutant vector into a single offspring commonly known as *trial vector* $\vec{U}_{i,G} = [u_{1,i,G}, u_{2,i,G}, \ldots u_{D,i,G}]$ DE primarily employs either of the two crossover strategies: *exponential* (two-point modulo) and *binomial* (uniform). Binomial crossover is preferred since it does away with the inherent representational bias in n-point crossover by simulating D random trials. Moreover a recent work [35] attributing to the sensitivity of crossover to population size has reported the exponential variant to be more prone as compared to its binomial counterpart. Owing to the aforesaid observations, here we employ binomial crossover to form the trial vector.

In order to implement the binomial crossover, the control parameter *Crossover rate* (Cr) is set to a fixed value lying in the range [0,1] and then D independent random numbers, between 0 and 1, are sampled uniformly and compared with Cr to decide which component is to be included in the trial vector. The method is outlined as:

$$u_{j,i,G} = \begin{cases} \{v_{j,i,G}, & if\ rand_{i,j} < CR\ and\ j = j_r, \\ & x_{j,i,G}\ otherwise, \end{cases} \tag{3}$$

where j_r is a randomly chosen index from {1, 2, …, D} and it ensures that at least one component from the mutant vector is present in the offspring produced.

2.4 Selection

Finally, a selection process is performed through a one-to-one competition between the parent and the offspring to maintain a constant population size. The selection process can be described as:

$$\vec{X}_{i,G+1} = \begin{cases} \vec{U}_{\mathrm{i}} \; \text{if} \; f\left(\vec{U}_{\mathrm{i}}\right) \le f\left(\vec{X}_i\right), \\ \vec{X}_i \; \text{otherwise}, \end{cases} \tag{4}$$

where $f(.)$ is the objective function to be minimized.

3 The Proposed Method

DE has 3 primary control parameters: the scale factor *F*, and the crossover rate *Cr*, and the population size *Np*. The performance of DE largely depends on *F* and *Cr*. Several efforts have been made in the past to control and adapt the value of these two parameters so that the algorithm may strike a balance between its explorative and exploitative behaviours on different fitness landscapes. Both of these parameters have their own allowable ranges. It is easy to see that *Cr* is similar to a probability value and hence it should lie in [0, 1]. Zaharie [36] derived a lower limit of *F* and her study revealed that if *F* be sufficiently small, the population can converge even in the absence of selection pressure. Ronkkonen et al. [37] stated that typically $0.4 < F < 0.95$ with $F = 0.9$ can serve as a good first choice. They also opine that *Cr* should lie in (0, 0.2) when the function is separable, while in (0.9, 1) when the function's variables are dependent. As is evident from [21] and the therein, numerous approaches have been proposed to improve the performance of DE by controlling or self-adapting these two control parameters and also by automating the choice of the appropriate offspring generation strategy. However, very often such methods may necessitate additional computational burdens, which, somewhat sacrifice the simplicity of DE. Thus, the question that naturally comes up is whether we can retain efficient search behaviour and adequate exploration-exploitation trade-off by doing something very simple? Can such easy modifications still result into a very competitive performance against the existing state-of-the-art? This article presents a humble contribution in this context.

The purpose of scaling factor *F* is to add weight to the difference vector and add it to base vector to produce mutant/donor vector. It is established in the study that *F* is strictly positive and greater than zero. A large value of *F* will support exploration, i.e. more of the feasible search volume can be covered. This property can be often desirable for solving high-dimensional optimization problems, since they possess a large search space. But, exploration is not helpful for converging and fine tuning of the solutions, necessary for detecting the optimum. A small value of *F* will serve the purpose of exploitation and support the convergence towards a solution.

In our proposal, for each population member, the *F* value is switched between 0.5 and 2 in a uniformly randomized way. Note that $F = 2$ is somewhat an unusual choice since this extreme value has not been reported for DE in commonly available papers. However, our experiments indicate that for the large scale problems this value can indeed enhance the performance than switching F between 0.5 and 1. When F is 2, the difference vector gets a higher importance. Consequently, the newly generated mutant point will be thrown relatively far from current base point, thereby enhancing the chances of venturing unexplored regions of the search space. When F takes a value of 0.5, the newly generated mutant point lies near by the base point as the difference vector gets very less weight. Thus it enhances the certainty of local neighborhood search. In Fig. 1 two possible distribution of the donor vectors have been shown around two mutant points generated by perturbing a base vector $\vec{X}_{base}$ with two values of the scale factor, $F = 0.5$ and $F = 2$ on a 2D search space.

Similarly for each individual, the Cr value is switched between 0 and 1. Note that this means in our proposed DE variant can either the mutant/donor vector is directly accepted as the final offspring (for $Cr = 1$) or the final offspring differs from the target (parent) vector at a single index determined by j_r (for $Cr = 0$). While the former situation corresponds to the generation of rotationally invariant points, the latter implies an axis parallel movement of the solution points.

Note that a plethora of DE variants has been published with different kinds of parametric (like sampling from a Cauchy or Gaussian distribution, see e.g. [22, 23]) and non-parametric (sampling from a uniform distribution like [31]) randomization in the tuning of F and Cr. However, such switching between only two extreme values has never been proposed earlier. In what follows we name the resulting DE variant as SWDE (Switching DE).

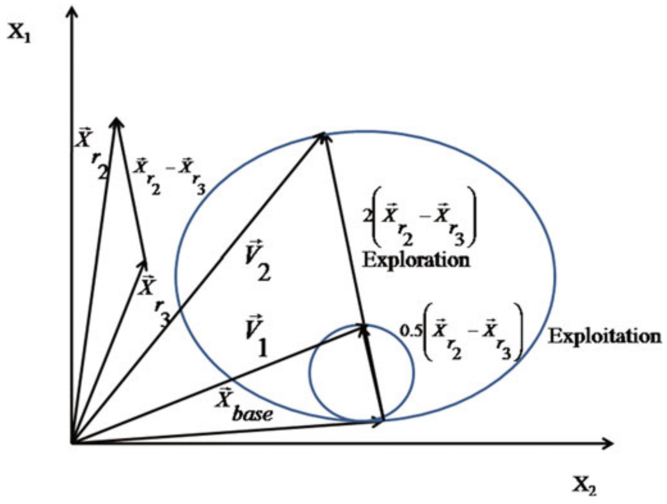
X_1
X_2
$\vec{X}_{r_2}$
$\vec{X}_{r_2} - \vec{X}_{r_3}$
$\vec{X}_{r_3}$
$\vec{V}_2$
$\vec{V}_1$
$\vec{X}_{base}$
$2(\vec{X}_{r_2} - \vec{X}_{r_3})$
Exploration
$0.5(\vec{X}_{r_2} - \vec{X}_{r_3})$
Exploitation

Fig. 1 Effect of $F = 2$ and $F = 0.5$ in DE mutation

Each individual in SWDE can be mutated by any of the commonly used mutation strategies in DE. It is now well-known that DE/best/1 induces somewhat greedy search behaviour and hence can be recommended for unimodal functions.

On the other hand, DE/rand/1 introduces more randomization (since different base vectors are perturbed to generate the mutant points for different individuals) and explorability and is, hence, suitable for multimodal functions. In our proposal we use a simple strategy for choosing any one of these two basic mutation schemes based on the success record of just the last successful update at the same population index. This means, at the first generation an individual is mutated either by DE/rand/1 or DE/best/1 chosen randomly. If the corresponding target individual is replaced by the trial (offspring), then the mutation scheme can be considered as a good choice, and will be used for that individual (of same index) in the next generation as well. If the trial is not selected for the individual, then the mutation strategy is unsuccessful, and in the next generation the other mutation scheme will be used for that individual.

We refer to this DE variant that combines the parameter switching scheme with a success-based selection of the mutation strategies as SWDE_Success (SWitching DE with Success-based selection of mutation scheme). Note that SWDE_Success does not require any control parameter to tune, except for the population size *Np*, which, however, is not really treated as a control parameter and is kept constant for most of the representative literature on DE. There is no need to fix any initial values for *F* and *Cr*, knowing only their feasible ranges would be fine. There is no parametric probability distribution whose parameters (like mean, variance, offset etc.) are required to be tuned.

Before moving on to the comparative study on standard benchmark suites, we would like to illustrate that DE with these simple modifications can indeed better preserve the population diversity, and thus can be helpful in retaining the useful information about promising search regions in a better way. Measurement of diversity level of the total population during the optimization work is another important aspect of empirical analysis, because maintaining diversity along the search process is another important aspect of large scale optimization. In proposed work "distance-to-average point" measurement for diversity of the population P_G at generation *G*, as presented in [38], is used as follows:

$$diversity(P_G) = \frac{1}{Np \times L} \sum_{i=1}^{Np} \sqrt{\sum_{j=1}^{D} \left(x_{ij} - \bar{x}_j\right)^2}, \tag{5}$$

where *Np* represents population size, *L* is the length of the longest diagonal in the search space of dimension *D* and $\bar{x}_j$ is the average value of *j*-th dimension of the vector. Variation of population diversity with respect to iteration as defined in (6) for the population is plotted in Fig. 2 for the Rastrigin's function in 50D. It is clear from the figure that the population of SWDE_Success never loses diversity prematurely. In this figure the red line depicts population diversity for SWDE_Success and blue one represents the same for standard DE.

In Fig. 3 we illustrate the distribution of population members of standard DE/best/1/bin (F = 0.8 and Cr = 0.8) and the proposed SWDE_Success on a 2D parameter space of the Rastrigin's function. The figures present screenshots of the

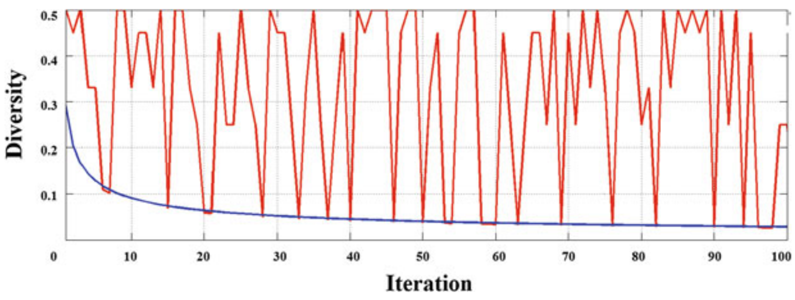

Fig. 2 Comparison of variation of population diversity with number of iterations between SWDE_Success and standard DE/best/1/bin

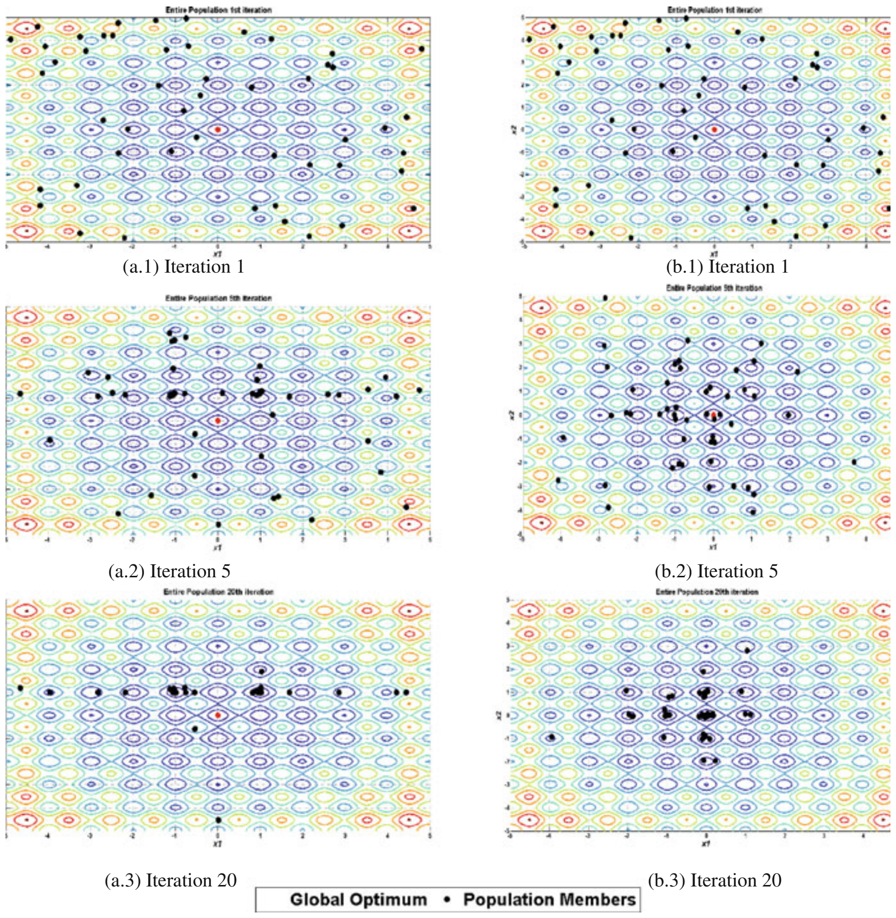

Fig. 3 Screen-shots of the evolving populations on the iso-contours of the 2D Rastrigin's function for (a) standard DE/best/1 scheme with $F = 0.8$ and $Cr = 0.8$ (b) SWDE_Success

population at 1st, 5-th and 20-th iterations. Note that for the two algorithms, the iterations were started from the same initial population, so that the different spread of the evolving populations may be attributed to their internal search mechanisms only. It can be seen that the population members following the SWDE_Success scheme can capture the global optimum more efficiently while still preserving the population diversity.

4 Experiments and Results

A popular choice for evaluating the performance of the DE algorithm is to use the benchmark suite proposed for the IEEE CEC (Congress on Evolutionary Computation) competitions. These suites contain collection of functions of diverse nature, which can successfully validate the performance of an optimization algorithm in a variety of scenarios.

The CEC 2008 [5] and CEC 2010 [6] test suites are specially designed with large scale minimization problems (i.e. of dimensions $D = 100$, 500, and 1000), and thus, useful for testing our proposed DE variant (SWDE_Success). Following standard procedure, the mean and standard deviation of the error value is used to measure the performance of an algorithm. The error is calculated as the difference between the actual value of the global optimum and the obtained value of the optimum. Only for function F7 of CEC 2008, the absolute value of the obtained optimum is recorded and compared, because for that function, the globally optimal function value is unknown. The population size has been taken as 100 for SWDE_Success in all the cases. To evaluate the scalability of the algorithm in the worse condition, the comparison is rendered on 1000 and 2000 dimensional problems.

For all the CEC 2008 benchmark problems and for all algorithms compared, the maximum number of Function Evaluations (FEs) corresponding to each run was taken to be $5000 \times D$, where D denotes the dimensionality of the functions following [5]. Similarly for the CEC 2010 benchmarks, the maximum number of FEs corresponding to each run was fixed to $3000 \times D$ [6, 18]. The parametric settings for all the peer algorithms were kept similar to their respective literatures. For SWDE_Success, the population size was fixed to $Np = 100$ for all 1000D problems and $Np = 150$ for the 2000D problems. We find that this choice (which is standard and straight forward) provides consistently good performance on the used benchmarks, and little improvement takes place with increased execution time if we increase the population size any further.

A non-parametric statistical test called Wilcoxon's rank sum test for independent samples [39] is conducted at the 5 % significance level in order to judge whether the results obtained with the best performing algorithm differ from the final results of rest of the competitors in a statistically significant way. In all result tables the statistical test results are summarized in the following way. If the final error yielded by an algorithm is statistically significantly different from that of the best performing algorithm on a particular function, then the mean error of the former is marked with a † symbol. If the difference of the error values found by one algorithm is not statistically significant, as compared to its best competitor, then the mean of this algorithm is marked with a ≈. The best performing algorithm in each case is marked with boldface.

In order to demonstrate how the proposed parameter switching scheme works harmoniously with the success-based mutation scheme, we begin with a comparative study among the proposed SWDE_Success, the standard DE/best/1/bin scheme with fixed F and Cr ($F = 0.8$, $Cr = 0.9$) and a SWDE that uses only DE/best/1 mutation scheme for all its population members. The obtained results are demonstrated in Table 1, which shows that SWDE provides much better results than DE/best/1 with fixed F and Cr values. Also it can be observed that SWDE_Success yields statistically better results as compared to both SWDE and DE/best/1 on all the 7 functions of the CEC 2008 test bed.

To compare the performance of SWDE_Success against the existing state-of-the-art, we consider the results of five other algorithms customised for large scale global optimization. Two of them are the variants of the Particle Swarm Optimisation (PSO) algorithm namely CCPSO2 [10] and EPUS-PSO (Efficient Population Utilization Strategy for Particle Swarm Optimization) [40], which use a variable grouping technique and an efficient population management scheme respectively. The third one is Sep-CMA-ES [41] a scalable variation of the popular CMA-ES optimization technique, which performs faster and better than the original algorithm, especially on separable functions. The fourth is MTS (Multiple Trajectory Search) [16], a hybrid local search method. The fifth one is MLCC (Multi-Level Cooperative Co-evolution) [42]. The results are listed Table 2, where SWDE_Success outperformed others, for all the functions except F6. For F6 CCPSO2 performed slightly better than SWDE_Success, however, result of the rank sum test indicates that this difference is not statistically significant. Thus, SWDE_Success is found to be more consistent on this benchmark suite. In terms of the average rank SWDE_Success is the clear winner, followed by CCPSO2 and MTS.

In Tables 3, 4 and 5 SWDE_Success results are compared with 11 other evolutionary optimizers including some recent DE-variants (DECC-ML, DECC-CG, DECC-DG, and DE/best/1/bin) on the CEC 2010 benchmarks. Out of the 20 high-dimensional functions SWDE_Success provided statistically significantly better results compared to all its peer algorithms on 14 functions, and ranked second in 3 functions. DECC-ML [43] performed best in three functions, namely F3, F11 and F16, jDELsgo [28] performed better on two functions F6 and F19, and MA-SW-Chains [18] performed better than others only in one case of F12. The rank sum test results indicate that on F3, the result of SWDE_Success is not statistically significantly different from the best result given by DECC_ML. Similarly for F6, the best result yielded by jDElsgo is statistically equivalent to that of SWDE_Success. For CEC 2010 functions also SWDE_Success holds the minimum average rank, the closest followers are jDELsgo, and MA-SW-Chains.

To further test the scalability of the proposed algorithm, three CEC 2008 functions (1 unimodal and 2 multimodals) of 2000 dimensions are used, as was also

Table 1 Performance improvement by the proposed algorithm (SWDE_Success) from DE/best/1/bin and SWDE (D = 1000)

Func.	F1	F2	F3	F4	F5	F6	F7
DE/best/1/bin	9.88e + 05† 1.25e + 05	1.291e + 04† 5.62e + 01	7.10e + 11† 2.35e + 09	6.53e + 03† 2.55e + 02	8.46e + 03† 3.52e + 02	1.76e + 01† 7.89e + 00	−2.13e + 01† 1.15e + 01
SWDE	9.95e − 18† 1.235 − 19	9.046e + 01† 5.46e + 01	6.33e + 05† 5.73e + 01	5.23e − 09† 4.61e − 05	1.61e − 14† 2.25e − 12	2.50e − 09† 1.16e − 07	−7.00e + 03† 1.22e + 01
SWDE_ Success	**0.00e + 00** **0.00e + 00**	**2.24e + 00** **2.01e + 01**	**1.03e − 03** **1.85e + 01**	**0.00e + 00** **0.00e + 00**	**2.74e − 22** **1.25e − 18**	**4.31e − 12** **3.25e − 19**	**−9.85e + 04** **2.56e − 02**

Table 2 Performance of 6 large scale evolutionary optimizers including SWDE_Success on CEC 2010 benchmark suite ($D = 1000$)

Function	F1	F2	F3	F4	F5	F6	F7	Avg. Rank
CCPSO2 [10]	4.99e − 13† 9.51e − 14	7.55e + 01† 4.25e + 01	1.30e + 03† 2.15e + 02	1.17e − 03† 3.27e − 03	1.17e − 03† 3.25e − 03	**1.01e − 12** **1.68e − 13**	−1.44e + 04† 8.27e + 01	
RANK	3	4	4	3	4	1	3	3.14
Sep-CMA-ES [41]	7.79e − 15† 1.22e − 15	3.10e + 02† 9.22e + 00	9.09e + 02† 4.22e + 01	5.25e + 03† 2.48e + 02	3.91e − 04† 1.96e − 03	2.16e + 01† 3.19e − 01	−1.24e + 04† 9.36e + 01	
RANK	2	6	3	5	5	8	5	4.85
EPUS-PSO [40]	5.49e + 02† 2.82e + 01	4.55e + 01† 4.00e − 01	8.31e + 05† 1.56e + 05	7.56e + 03† 1.50e + 02	5.80e + 00† 3.92e − 01	1.84e + 01† 2.49e + 00	−6.68e + 03† 3.18e + 01	
RANK	6	3	6	7	6	5	6	5.57
MLCC [42]	8.25e − 13† 5.59e − 14	1.28e + 02† 4.75e + 00	1.77e + 03† 1.25e + 02	1.42e − 10† 3.31e − 10	4.17e − 13† 2.79e − 14	1.06e − 12 ≈ 7.68e − 14	−1.49e + 04† 1.52e + 01	
RANK	4	5	5	2	2	3	2	3.28
MTS [16]	1.21e − 03† 6.14e − 03	4.60e + 01† 1.5e + 00	1.77e − 02† 7.81e − 03	2.81e + 02† 4.74e + 02	9.45e − 08† 2.61e − 07	7.20e − 04† 2.67e − 03	−1.31e + 04† 3.45e + 01	
RANK	5	2	2	4	3	6	4	3.71
SWDE_ Success	**0.00e + 00** **0.00e + 00**	**2.24e + 00** **2.01e + 01**	**1.03e − 03** **1.85e − 02**	**0.00e + 00** **0.00e + 00**	**2.74e − 22** **1.25e − 18**	4.31e − 12 ≈ 3.25e − 19	**−9.85e + 04** **2.56e − 02**	
RANK	1	1	1	1	1	2	1	**1.14**

Table 3 Performance of 12 large scale evolutionary optimizers including SWDE_Success on CEC 2010 benchmark suite (F1 − F7, $D = 1000$)

Functions								
Algorithms		F1	F2	F3	F4	F5	F6	F7
jDElsgo [28]	Mean	8.85e − 20†	1.24e − 01†	3.79e − 12≈	8.01e + 10†	9.71e + 07†	**1.69e − 08**	4.32e − 02†
	SD	4.50e − 20	3.44e − 01	5.02e − 12	3.05e + 10	1.42e + 07	**4.02e − 08**	6.35e − 02
	(Rank)	(7)	(2)	(6)	(2)	(5)	**(1)**	(2)
DECC-ML [43]	Mean	1.36e − 25†	2.15e + 02†	**1.14e − 13**	3.53e + 12†	2.98e + 08†	7.94e + 05†	1.21e + 08†
	SD	1.81e − 25	2.91e + 01	**8.22e − 15**	1.51e + 12	9.32e + 07	3.87e + 06	7.65e + 07
	(Rank)	(3)	(7)	**(1)**	(7)	(10)	(8)	(10)
DASA [45]	Mean	1.51e − 21†	8.44e + 00†	7.21e − 11†	5.04e + 11†	6.21e + 08†	1.98e + 07†	7.75e + 00†
	SD	2.32e − 21	2.50e + 00	8.22e − 12	2.25e + 11	7.81e + 07	4.41e + 04	3.11e + 00
	(Rank)	(5)	(5)	(8)	(5)	(12)	(11)	(3)
DMS-PSO-SHS [46]	Mean	5.55e − 15†	8.52e + 01†	5.51e − 11†	2.41e + 11†	8.35e + 07†	8.25e − 02†	1.95e + 03†
	SD	4.02e − 14	2.02e + 01	3.21e − 10	3.31e + 10	6.15e + 06	9.95e − 01	1.55e + 02
	(Rank)	(6)	(6)	(7)	(3)	(4)	(4)	(6)
SDENS [34]	Mean	5.72e − 06†	2.22e + 03†	2.70e − 05†	5.12e + 12†	1.12e + 08†	2.23e − 04†	1.21e + 08†
	SD	4.42e − 06	8.92e + 01	1.52e − 05	2.12e + 12	2.23e + 07	4.56e − 05	6.52e + 07
	(Rank)	(10)	(10)	(9)	(9)	(6)	(3)	(8)
EOEA [44]	Mean	2.21e − 23†	3.61e − 01†	1.61e − 13≈	3.07e + 12†	2.26e + 07†	3.86e + 06†	1.21e + 02†
	SD	2.81e − 23	6.71e − 01	1.11e − 14	1.66e + 12	5.96e + 06	4.96e + 05	1.51e + 02
	(Rank)	(4)	(3)	(2)	(6)	(3)	(9)	(5)
DECC-G [9]	Mean	2.94e − 07†	1.34e + 03†	1.38e + 00†	1.75e + 13†	2.64e + 08†	4.91e + 06†	1.61e + 08†
	SD	8.64e − 08	3.24e + 01	9.75e − 02	5.34e + 12	8.44e + 07	8.01e + 05	1.31e + 08
	(Rank)	(9)	(9)	(10)	(11)	(9)	(10)	(11)
MLCC [42]	Mean	1.55e − 27†	5.52e − 01†	9.82e − 13≈	9.60e + 12†	3.80e + 08†	1.61e + 07†	6.81e + 05†
	SD	7.62e − 27	2.22e + 00	3.72e − 12	3.40e + 12	6.90e + 07	4.91e + 06	7.31e + 05
	(Rank)	(2)	(4)	(4)	(10)	(11)	(12)	(8)

(continued)

Table 3 (continued)

Functions								
Algorithms		F1	F2	F3	F4	F5	F6	F7
MA-SW-Chains [18]	Mean	2.09e − 14†	8.11e + 02†	7.21e − 13≈	3.52e + 11†	1.67e + 08†	8.13e + 04†	1.03e + 02†
	SD	1.98e − 14	5.81e + 01	3.41e − 13	3.12e + 10	1.07e + 08	2.83e + 05	8.71e + 01
	(Rank)	(7)	(8)	(3)	(4)	(8)	(7)	(4)
DECC-DG [11]	Mean	5.41e + 03†	4.33e + 03†	1.61e + 01†	4.78e + 12†	1.51e + 08†	1.63e + 01†	1.12e + 04†
	SD	2.02e + 04	1.93e + 02	3.31e − 01	1.48e + 12	2.12e + 07	2.72e − 01	7.41e + 03
	(Rank)	(11)	(11)	(11)	(8)	(7)	(5)	(7)
DE/best/1/bin	Mean	3.52e + 05†	9.30e + 03†	3.53e + 03†	5.25e + 19†	5.23e + 06†	2.55e + 01†	1.08e + 11†
	SD	2.69e + 05	1.3e + 04	2.23e + 02	9.23e + 17	9.63 + 05	1.21e + 01	9.08e + 10
	(Rank)	(12)	(12)	(12)	(12)	(2)	(6)	(12)
SWDE_Success	Mean	**1.59e − 30**	**1.21e − 03**	1.21e − 12≈	**1.00e + 06**	**3.39e + 04**	2.33e − 08≈	**1.05e − 03**
	SD	**1.35e − 25**	**2.35e − 06**	7.35e − 12	**2.54e + 02**	**3.37e + 02**	1.01e − 07	**1.01e − 02**
	(Rank)	**(1)**	**(1)**	(5)	**(1)**	**(1)**	(2)	**(1)**

Table 4 Performance of 12 large scale evolutionary optimizers including SWDE_Success on CEC 2010 benchmark suite (F8 − F14, $D = 1000$)

Functions Algorithms		F8	F9	F10	F11	F12	F13	F14
jDElsgo [28]	Mean SD (Rank)	3.12e + 06† 3.23e + 06 (2)	3.13e + 07† 5.02e + 06 (5)	2.59e + 03† 3.18e + 02 (5)	2.21e + 01† 1.53e + 01 (4)	1.22e + 04† 2.05e + 03 (6)	7.12e + 02† 1.39e + 02 (2)	1.67e + 08† 2.09e + 07 (6)
DECC-ML [43]	Mean SD (Rank)	3.41e + 07† 3.52e + 07 (7)	5.93e + 07† 4.71e + 06 (8)	1.21e + 04† 2.60e + 02 (12)	**1.79e − 13** **9.55e − 15** **(1)**	3.56e + 06† 1.35e + 05 (12)	1.12e + 03† 4.30e + 02 (3)	1.70e + 08† 1.45e + 07 (7)
DASA [45]	Mean SD (Rank)	4.97e + 07† 8.94e + 07 (9)	3.61e + 07† 4.78e + 06 (6)	7.26e + 03† 2.61e + 02 (10)	1.97e + 02† 1.52e − 01 (9)	1.70e + 03† 2.24e + 02 (4)	1.20e + 03† 7.34e + 02 (4)	1.01e + 08† 7.85e + 06 (4)
DMS-PSO-SHS [46]	Mean SD (Rank)	1.25e + 07† 1.95e + 06 (4)	8.55e + 06† 6.55e + 05 (2)	5.59e + 03† 5.19e + 02 (8)	3.29e + 01† 2.99e + 00 (6)	6.15e + 02† 6.05e + 01 (3)	1.25e + 03† 1.06e + 02 (5)	1.76e + 07† 1.56e + 06 (2)
SDENS [34]	Mean SD (Rank)	5.15e + 07† 2.15e + 07 (10)	5.61e + 08† 5.71e + 07 (12)	6.81e + 03† 5.61e + 02 (9)	2.22e + 02† 5.02e − 01 (11)	4.12e + 05† 4.22e + 04 (10)	2.13e + 03† 1.03e + 03 (9)	1.83e + 09† 2.33e + 08 (11)
EOEA [44]	Mean SD (Rank)	1.01e + 07† 1.21e + 07 (3)	4.62e + 07† 4.72e + 06 (7)	1.02e + 03† 6.92e + 01 (3)	3.82e + 01† 1.62e + 01 (8)	1.57e + 04† 2.50e + 03 (7)	1.54e + 03† 4.19e + 02 (6)	1.64e + 08† 8.94e + 06 (5)
DECC-G [9]	Mean SD (Rank)	6.43e + 07† 2.81e + 07 (11)	3.20e + 08† 3.36e + 07 (11)	1.05e + 04† 2.94e + 02 (11)	2.30e + 01† 1.75e + 00 (5)	8.91e + 04† 6.81e + 03 (9)	5.11e + 03† 3.91e + 03 (10)	8.01e + 08† 6.02e + 07 (10)
MLCC [42]	Mean SD (Rank)	4.37e + 07† 3.44e + 07 (8)	1.22e + 08† 1.29e + 07 (10)	3.45e + 03† 8.75e + 02 (6)	1.91e + 02† 6.91e − 01 (10)	3.41e + 04† 4.21e + 03 (8)	2.01e + 03† 7.21e + 02 (8)	3.11e + 08† 2.71e + 07 (8)

(continued)

Table 4 (continued)

Functions Algorithms		F8	F9	F10	F11	F12	F13	F14
MA-SW-Chains [18]	Mean	1.41e + 07†	1.42e + 07†	2.02e + 03†	3.81e + 01†	**3.61e − 06**	1.21e + 03†	3.11e + 07†
	SD	3.62e + 07	1.12e + 06	1.42e + 02	7.31e + 00	**5.91e − 07**	5.71e + 02	1.93e + 06
	(Rank)	(5)	(3)	(4)	(7)	**(1)**	(7)	(3)
DECC-DG [11]	Mean	3.01e + 07†	5.91e + 07†	4.53e + 03†	1.01e + 01†	2.52e + 03†	4.58e + 06†	3.46e + 08†
	SD	2.11e + 07	8.12e + 06	1.42e + 02	1.02e + 00	4.85e + 02	2.18e + 06	2.46e + 07
	(Rank)	(6)	(9)	(7)	(3)	(5)	(12)	(9)
DE/best/1/bin	Mean	2.46e + 11†	3.03e + 07†	4.65e + 01†	2.35e + 03†	2.36e + 06†	3.59e + 06†	3.25e + 17†
	SD	2.22e + 10	1.00e + 06	5.11e + 01	1.39e + 03	1.59e + 06	1.09e + 05	1.59e + 15
	(Rank)	(12)	(4)	(2)	(12)	(11)	(11)	(12)
SWDE_Success	Mean	**1.35e + 05**	**2.04e + 04**	**5.15e + 00**	1.55e − 09†	5.00e + 02†	**3.00e + 02**	**8.15e + 06**
	SD	**2.39e + 04**	**1.09e + 02**	**1.11e + 01**	1.35e − 10	3.55e + 01	**2.00e + 00**	**2.15e + 04**
	(Rank)	**(1)**	**(1)**	**(1)**	(2)	(2)	**(1)**	**(1)**

Table 5 Performance of 12 large scale evolutionary optimizers including SWDE_Success on CEC 2010 benchmark suite (F15 – F20, Average Rank, $D = 1000$)

Functions Algorithms		F15	F16	F17	F18	F19	F20	Avg. Rank on all functions F1 – F20
jDElsgo [28]	Mean SD (Rank)	5.85e + 03† 4.46e + 02 (5)	1.40e + 02† 3.42e + 01 (8)	1.00e + 05† 1.25e + 04 (7)	1.86e + 03† 3.11e + 02 (3)	**2.73e + 05** **2.12e + 04** **(1)**	1.51e + 03† 1.32e + 02 (7)	4.3
DECC-ML [43]	Mean SD (Rank)	1.54e + 04† 3.51e + 02 (11)	**5.07e – 02** **2.54e – 01** **(1)**	6.56e + 06† 4.61e + 05 (11)	2.42e + 03† 1.11e + 03 (5)	1.59e + 07† 1.71e + 06 (11)	9.92e + 02† 3.55e + 01 (4)	6.95
DASA [45]	Mean SD (Rank)	1.44e + 04† 3.66e + 02 (10)	3.94e + 02† 2.12e – 01 (9)	1.04e + 04† 8.9e + 02 (4)	4.42e + 03† 2.18e + 03 (7)	8.14e + 05† 5.56e + 04 (3)	1.03e + 03† 1.49e + 02 (6)	6.7
DMS-PSO-SHS [46]	Mean SD (Rank)	4.68e + 03† 2.15e + 02 (4)	6.95e + 01† 4.25e + 00 (4)	3.23e + 03† 4.05e + 02 (3)	2.26e + 03† 1.16e + 02 (4)	1.16e + 06† 1.06e + 05 (6)	3.52e + 02† 4.02e + 01 (2)	4.45
SDENS [34]	Mean SD (Rank)	7.36e + 03† 9.63e + 01 (8)	4.03e + 02† 2.53e + 00 (10)	1.02e + 06† 1.12e + 05 (10)	3.01e + 04† 1.21e + 04 (10)	8.82e + 05† 1.52e + 05 (4)	9.93e + 02† 1.63e + 01 (3)	8.6
EOEA [44]	Mean SD (Rank)	2.14e + 03† 1.24e + 02 (2)	8.26e + 01† 1.68e + 01 (6)	7.93e + 04† 8.80e + 03 (6)	2.94e + 03† 6.92e + 02 (6)	1.84e + 06† 9.97e + 04 (8)	1.97e + 03† 2.35e + 02 (8)	5.35
DECC-G [9]	Mean SD (Rank)	1.26e + 04† 8.91e + 02 (9)	7.65e + 01† 8.15e + 00 (5)	2.85e + 05† 1.98e + 04 (9)	2.45e + 04† 1.05e + 04 (9)	1.15e + 06† 5.15e + 04 (5)	4.05e + 03† 3.66e + 02 (10)	9.15
MLCC [42]	Mean SD (Rank)	7.11e + 03† 1.31e + 03 (7)	3.72e + 02† 4.72e + 01 (11)	1.52e + 05† 1.42e + 04 (8)	7.02e + 03† 4.72e + 03 (8)	1.32e + 06† 7.32e + 04 (9)	2.02e + 03† 1.82e + 02 (9)	8.05

(continued)

Table 5 (continued)

Functions Algorithms		F15	F16	F17	F18	F19	F20	Avg. Rank on all functions F1 – F20
MA-SW-Chains [18]	Mean	2.86e + 03†	9.95e + 01†	1.25e + 00≈	1.34e + 03†	2.84e + 05†	1.05e + 03†	4.6
	SD	1.25e + 02	1.45e + 01	1.25e − 01	4.34e + 02	1.69e + 04	7.25e + 01	
	(Rank)	(3)	(7)	(2)	(2)	(2)	(5)	
DECC-DG [11]	Mean	5.84e + 03†	7.32e + 03†	4.06e + 04†	1.18e + 10†	1.78e + 06†	4.89e + 07†	7.75
	SD	1.04e + 02	5.72e − 14	2.86e + 03	2.08e + 09	9.58e + 04	2.28e + 07	
	(Rank)	(6)	(2)	(5)	(12)	(7)	(12)	
DE/best/1/bin	Mean	6.66e + 05†	2.33e + 03†	3.95e + 07†	3.52e + 07†	2.55e + 09†	3.54e + 05†	10.01
	SD	5.69e + 05	1.11e + 03	6.69e + 06	1.09e + 07	1.59e + 07	9.23e + 04	
	(Rank)	(12)	(12)	(12)	(11)	(12)	(11)	
SWDE_Success	Mean	**1.10e + 03**	8.50e − 01†	**1.13e + 00**	**1.01e + 03**	5.50e + 06†	**3.09e + 00**	**1.85**
	SD	**2.09e + 01**	1.00e − 01	**3.13e − 01**	**2.12e + 01**	2.00e + 05	**1.00e − 01**	
	(Rank)	**(1)**	(3)	**(1)**	**(1)**	(10)	**(1)**	

Table 6 Performance of SWDE_Success, CCPSO2 and Sep-CMA-ES on 3 functions from the CEC 2008 test-suite ($D = 2000$)

Algorithms	SWDE_Success	CCPSO2	Sep-CMA-ES
Functions	Mean (Std. Dev.)	Mean (Std. Dev.)	Mean (Std. Dev.)
F1	1.23e − 14≈ (1.01e − 06)	1.03e − 12† (2.56e − 13)	**7.12e − 15 (6.24e − 15)**
F3	**1.00e + 01 (2.10e + 00)**	2.91e + 03† (6.43e + 02)	1.73e + 03† (7.77e + 01)
F7	**−2.20e + 04 (1.01e + 02)**	−2.28e + 04† (1.90e + 02)	−2.46e + 04† (1.57e + 02)

done in [10]. The performance is compared with two popular evolutionary large scale optimizers of diverse origins, CCPSO2 and Sep-CMA-ES [41]. The obtained results are summarized in Table 6. Sep-CMA-ES only performed better than SWDE_Success in the case of the separable function F1. However, according to the rank sum test, the difference between the final mean errors of sep-CMA-ES and SWDE_Success is not statistically meaningful. For the multimodal non-separable problems F3 and F7, SWDE_Success performed statistically better than both CCPSO2 and sep-CMA-ES. CCPSO2 took the second place for F7 and Sep-CMA-ES did the same for F3.

5 Conclusion

To address the problem of optimizing very high-dimensional numerical functions, this paper presents a new variant of DE, referred here as the SWDE_Success, which uses a simple switching scheme for the two key parameters of DE, the scale factor and the crossover rate. SWDE_Success also employs a success-based selection of either of the two kinds of mutation strategies. The algorithm uses two very common mutation strategies (the greedy DE/best/1 and the explorative DE/rand/1 schemes) and applies a simple scheme selection process, which only depends on the success of a mutation scheme in the previous iteration in terms of generating a successful offspring (one which could replace its parent during the selection).

Exploring a huge search volume (induced by the large number of variables) with a limited population of candidate solutions is challenging and it requires a judicious balance between the explorative and exploitative tendencies of an evolutionary algorithm. This requirement is nicely fulfilled by the random selection of the control parameter values from their extremities. Our results indicate that a combination of the high and unconventional value of F (= 2) with the low value (= 0.5) can be indeed very useful for solving benchmark functions. In contrast to some of the most prominent approaches (like [22, 23, 26, 27]) that sample F values from the interval of (0.4, 1) and Cr values from (0, 1), our results indicate that most of the useful information about F and Cr values can remain attached to the boundaries of their feasible regions. This point requires further analytical and experimental investigations in future.

The future works may also include a detailed study on the dynamics and search procedure of SWDE_Success, alongside an explanation of its success. Also the parameter switching strategy may be further investigated in other optimization scenarios like for moderate dimensional problems, for multi-objective, constrained and dynamic optimization problems.

References

1. Foli, K., Okabe, T., Olhofer, M., Jin, Y., Sendhoff, B.: Optimization of micro heat exchanger: CFD, analytical results and multiobjective evolutionary algorithms. Int. J. Heat Mass Transf. **49**(5–6), 1090–1099 (2006)
2. Sonoda, T., Yamaguchi, Y., Arima, T., Olhofer, M., Sendhoff, B., Schreiber, H.A.: Advanced high turning compressor airfoils for low Reynolds number condition, part I: Design and optimization. J. Turbomach. **126**(3), 350–359 (2004)
3. Wang, C., Gao, J.: High-dimensional waveform inversion with cooperative coevolutionary differential evolution algorithm. IEEE Geosci. Remote Sens. Lett. **9**(2), 297–301 (2012)
4. Das, S., Suganthan, P.N.: Problem definitions and evaluation criteria for CEC 2011 competition on testing evolutionary algorithms on real world optimization problems. Technical Report, Jadavpur University, India and Nanyang Technological University, Singapore (2010)
5. Tang, K., Yao, X., Suganthan, P., MacNish, C., Chen, Y., Chen, C., Yang, Z.: Benchmark functions for the CEC'2008 special session and competition on large scale global optimization. In: Nature Inspired Computat. Applicat. Lab., Univ. Sci. Technol. China, Hefei, China, Tech. Rep. http://nical.ustc.edu.cn/cec08ss.php (2007)
6. Tang, K., Li, X., Suganthan, P., Yang, Z., Weise, T.: Benchmark functions for the CEC'2010 special session and competition on large scale global optimization. In: Nature Inspired Computat. Applicat. Lab., Univ. Sci. Technol. China, Hefei, China, Tech. Rep. http://nical.ustc.edu.cn/cec10ss.php (2009)
7. Potter, M., De Jong, K.: Cooperative coevolution: an architecture for evolving coadapted subcomponents. Evol. Comput. **8**(1), 1–29 (2000)
8. Ray, T., Yao, X.: A cooperative coevolutionary algorithm with correlation based adaptive variable partitioning. In: Proceedings of the IEEE CEC, pp. 983–999, May 2009
9. Yang, Z., Tang, K., Yao, X.: Large scale evolutionary optimization using cooperative coevolution. Inf. Sci. **178**(15), 2986–2999 (2008)
10. Li, X., Yao, X.: Cooperatively coevolving particle swarms for large scale optimization. IEEE Trans. Evol. Comput. **16**(2), 210–224 (2011)
11. Omidvar, M.N., Li, X., Mei, Y., Yao, X.: Cooperative co-evolution with differential grouping for large scale optimization. IEEE Trans. Evol. Comput. **18**(3), 378–393 (2013)
12. Krishnakumar, K.: Micro-genetic algorithms for stationary and non-stationary function optimization, SPIE 1196. Intell. Control Adapt. Syst. (1989). doi:10.1117/12.969927
13. Huang, T., Mohan, A.S.: Micro–particle swarm optimizer for solving high dimensional optimization problems. Appl. Math. Comput. **181**(2), 1148–1154 (2006)
14. Dasgupta, S., Biswas, A., Das, S., Panigrahi, B.K., Abraham, A.: A micro-bacterial foraging algorithm for high-dimensional optimization. In: IEEE Congress on Evolutionary Computation (CEC 2009), pp. 785–792, Tondheim, Norway, May 2009

15. Rajasekhar, A., Das, S., Das, S.: μABC: a micro artificial bee colony algorithm for large scale global optimization. In: Soule, T. (ed.) Proceedings of the 14th Annual Conference Companion on Genetic and Evolutionary Computation (GECCO '12), pp. 1399–1400, ACM, New York, NY, USA. doi:10.1145/2330784.2330951. http://doi.acm.org/10.1145/2330784.2330951
16. Tseng, L.Y., Chen, C.: Multiple trajectory search for large scale global optimization. In: IEEE Congress on Evolutionary Computation (CEC 2008), pp. 3052–3059, Hong Kong, June 2008
17. Zhao, S.Z., Suganthan, P.N., Das, S.: Self-adaptive differential evolution with multi-trajectory search for large scale optimization. Soft. Comput. **15**, 2175–2185 (2011)
18. Molina, D., Lozano, M., Herrera, F.: MA-SW-Chains: memetic algorithm based on local search chains for large scale continuous global optimization. In: IEEE Congress on Evolutionary Computation (CEC 2010), pp. 3153–3160, Barcelona, July, 2010
19. Molina, D., Lozano, M., Sánchez, A.M., Herrera, F.: Memetic algorithms based on local search chains for large scale continuous optimization problems: MA-SSW-Chains. Soft. Comput. **15**, 2201–2220 (2011)
20. Storn, R., Price, K.: Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. J. Glob. Optim. **11**(4), 341–359 (1997)
21. Das, S., Suganthan, P.N.: Differential evolution: a survey of the state-of-the-art. IEEE Trans. Evol. Comput. **15**(1), 4–31 (2011)
22. Qin, A.K., Huang, V., Suganthan, P.: Differential evolution algorithm with strategy adaptation for global numerical optimization. IEEE Trans. Evol. Comput. **13**(2), 398–417 (2009)
23. Zhang, J., Sanderson, A.: JADE: adaptive differential evolution with optional external archive. IEEE Trans. Evol. Comput. **13**(5), 945–958 (2009)
24. Epitropakis, M., Tasoulis, D., Pavlidis, N., Plagianakos, V., Vrahatis, M.: Enhancing differential evolution utilizing proximity based mutation operators. IEEE Trans. Evol. Comput. **15**(1), 99–119 (2011)
25. Islam, S.M., Das, S., Ghosh, S., Roy, S., Suganthan, P.N.: An adaptive differential evolution algorithm with novel mutation and crossover strategies for global numerical optimization. IEEE Trans. Syst. Man Cybern. B Cybern. **42**(2), 482–500 (2012)
26. Wang, Y., Cai, Z., Zhang, Q.: Differential evolution with composite trial vector generation strategies and control parameters. IEEE Trans. Evol. Comput. **15**(1), 55–66 (2011)
27. Mallipeddi, R., Suganthan, P.N.: Differential evolution algorithm with ensemble of parameters and mutation and crossover strategies. In: Proc. Swarm Evol. Memet. Comput., Chennai, India, pp. 71–78 (2010)
28. Zamuda, A., Brest, J., Boˇskovi´c, B., Zumer, V.: Large scale global optimization using differential evolution with self-adaptation and cooperative co-evolution. In: IEEE Congress on Evolutionary Computation (CEC 2008), pp. 3718–3725, Hong Kong, June 2008
29. Parsopoulos, K.E.: Cooperative micro-differential evolution for high-dimensional problems. In: Genetic and Evolutionary Computation Conference 2009 (GECCO 2009), pp. 531–538, Montreal, Canada (2009)
30. Zhao, S.Z., Suganthan, P.N., Das, S.: Self-adaptive differential evolution with multi-trajectory search for large scale optimization. Soft. Comput. **15**, 2175–2185 (2011)
31. Brest, J., Maučec, M.S.: Self-adaptive differential evolution algorithm using population size reduction and three strategies. Soft. Comput. **15**(11), 2157–2174 (2011)
32. Wang, H., Wu, Z., Rahnamayan, S.: Enhanced opposition-based differential evolution for solving high-dimensional continuous optimization problems. Soft. Comput. **15**(11), 2127–2140 (2011)
33. Weber, M., Neri, F., Tirronen, V.: Shuffle or update parallel differential evolution for large-scale optimization. Soft. Comput. **15**(11), 2089–2107 (2011)
34. Wang, H., Wu, Z., Rahnamayan, S., Jiang, D.: Sequential DE enhanced by neighborhood search for large scale global optimization. In: IEEE Congress on Evolutionary Computation (CEC 2010), pp. 4056–4062, Barcelona, July, 2010
35. Zaharie, D.: Influence of crossover on the behavior of the differential evolution algorithm. Appl. Soft Comput. **9**(3), 1126–1138 (2009)

36. Zaharie, D.: Critical values for the control parameters of differential evolution algorithms. In: Proc. 8th Int. Mendel Conf. Soft. Comput., pp. 62–67 (2002)
37. Ronkkonen, J., Kukkonen, S., Price, K.V.: Real parameter optimization with differential evolution. In: The 2005 IEEE Congress on Evolutionary Computation (CEC2005), vol. 1, pp. 506–513. IEEE Press (2005)
38. Hu, J., Zeng, J., Tan, Y.: A diversity-guided particle swarm optimizer for dynamic environments. In: Proceedings of Bio-Inspired Computational Intelligence Applivations, vol. 9, no. 3, pp. 239–247. Lecture Notes in Computer Science (2007)
39. Derrac, J., García, S., Molina, D., Herrera, F.: A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. Swarm Evol. Comput. **1**(1), 3–18 (2011)
40. Hsieh, S.T., Sun, T.Y., Liu, C.C., Tsai, S.J.: Efficient population utilization strategy for particle swarm optimizer. IEEE Trans. Syst. Man Cybern. B Cybern. **39**(2), 444–456 (2009)
41. Ros, R., Hansen, N.: A simple modification in CMA-ES achieving linear time and space complexity. Lect. Notes Comput. Sci. **5199**, 296–305 (2008)
42. Yang, Z., Tang, K., Yao, X.: Multilevel cooperative coevolution for large scale optimization. In: Proc. IEEE Congr. Evol. Comput., pp. 1663–1670, June 2008
43. Omidvar, M.N., Li, X., Yao, X.: Cooperative co-evolution with delta grouping for large scale non-separable function optimization. In: Proc. IEEE Congr. Evol. Comput., pp. 1762–1769, July 2010
44. Wang, Y., Huang, J., Dong, W.S., Yan, J.C., Tian, C.H., Li, M., Mo, W.T.: Two-stage based ensemble optimization framework for large-scale global optimization. Eur. J. Oper. Res. **228**, 308–320 (2013)
45. Korošec, P., Šilc, J.: The differential ant-stigmergy algorithm for large scale real-parameter optimization. In: Ant Colony Optimization and Swarm Intelligence, Lecture Notes in Computer Science, vol. 5217, pp. 413–414, Springer, Berlin Heidelberg (2008)
46. Zhao, S., Liang, J., Suganthan, P.N., Tasgetiren, M.F.: Dynamic multi-swarm particle swarm optimizer with local search for large scale global optimization. In: Proceedings of IEEE Congress on Evolutionary Computation, pp. 3845–3852 (2008)