Soft Comput (2011) 15:2175–2185
DOI 10.1007/s00500-010-0645-4

FOCUS

Self-adaptive differential evolution with multi-trajectory search for large-scale optimization

Shi-Zheng Zhao · Ponnuthurai Nagaratnam Suganthan · Swagatam Das

Published online: 14 September 2010

Abstract In this paper, self-adaptive differential evolution (DE) is enhanced by incorporating the JADE mutation strategy and hybridized with modified multi-trajectory search (MMTS) algorithm (SaDE-MMTS) to solve large-scale continuous optimization problems. The JADE mutation strategy, the "DE/current-to-*pbest*" which is a variation of the classic "DE/current-to-*best*", is used for generating mutant vectors. After the mutation phase, the binomial (uniform) crossover, the exponential crossover as well as no crossover option are used to generate each pair of target and trial vectors. By utilizing the self-adaptation in SaDE, both trial vector generation strategies and their associated control parameter values are gradually self-adapted by learning from their previous experiences in generating promising solutions. Consequently, suitable offspring generation strategy along with associated parameter settings will be determined adaptively to match different phases of the search process. MMTS is applied frequently to refine several diversely distributed solutions at different search stages satisfying both the global and the local search requirement. The initialization of step sizes is also defined by a self-adaption during every MMTS step. The success rates of both SaDE and the MMTS are determined and compared; consequently, future function evaluations for both search algorithms are assigned proportionally to their recent past performance. The proposed SaDE-MMTS is employed to solve the 19 numerical optimization problems in special issue of soft computing on scalability of evolutionary algorithms for large-scale continuous optimization problems and competitive results are presented.

Keywords Differential evolution · Large scale continuous optimization · Self-adaptation · Strategy adaptation · Parameter adaptation · Multi-trajectory search · JADE mutation

S.-Z. Zhao (✉) · P. N. Suganthan
School of Electrical and Electronic Engineering,
Nanyang Technological University, 50 Nanyang Ave.,
Singapore 639798, Singapore
e-mail: zh0047ng@ntu.edu.sg

P. N. Suganthan
e-mail: epnsugan@ntu.edu.sg

S. Das
Department of Electronics and Telecommunication,
Jadavpur University, Kolkata, India
e-mail: swagatamdas19@yahoo.co.in

1 Introduction

Differential evolution (DE) algorithm, proposed by Storn and Price (1997), is a simple, yet powerful population-based stochastic search technique for solving global optimization problems in continuous search domain. DE generates offspring by perturbing the solutions with a scaled difference of selected population vectors, and employs parent–offspring comparison which allows replacement of a parent only if the offspring is better or equivalent to its parent. In DE, there exist many trial vector generation strategies and crucial control parameters that significantly influence the optimization performance of the DE. Therefore, in order to successfully solve a specific optimization problem, it is generally required to perform a time-consuming and computationally demanding trial-and-error search for the most appropriate strategy and to tune the associated parameter values. Moreover, during the search process, the DE population may evolve through different regions in the search space thereby requiring different strategies and parameter settings. Hence, it is desirable to adaptively determine appropriate strategies and associated parameter values at different stages of evolution. In this paper, we apply the self-adaptive DE (SaDE) (Qin et al. 2009; Qin and Suganthan 2005; Huang et al. 2006) to make both crossover strategies and their associated parameters self-adaptive by learning from their previous experiences in generating improved solutions. Consequently, suitable generation strategies along with parameter settings can be determined adaptively to match different search phases. At each generation, a strategy together with associated parameter values will be assigned to each parent according to the selection probabilities learnt from the previous generations.

The JADE (Zhang and Sanderson 2009) introduces a new mutation strategy, "DE/current-to-*pbest*" with optional external archive. As a generalization of DE/current-to-best, DE/current-to-*pbest* utilizes not only the best solution information but also the information of other good solutions of the current population. To be specific, any of the top $p\%$, $p \in (0, 100]$ solutions will be randomly chosen in DE/current-to-*pbest* to play the role designed exclusively for the single best solution in DE/current-to-best. In addition, the difference between the archived solutions and a member in the current population can be incorporated into the mutation operation. This strategy diversifies the population and provides approximate direction of evolution.

In this paper, the SaDE is enhanced by utilizing modified multi-trajectory search (MMTS), JADE mutation strategy (SaDE-MMTS) and the binomial (uniform) crossover, the exponential crossover and without crossover to test the 19 problems from the soft computing special session and competition on large scale global optimization. The step sizes are also self-adapted during every MMTS step. The success rates of both SaDE and the MMTS are computed and consequently, future function evaluations for both search algorithms are assigned according to their recent past performance. The rest of the paper is organized as follows: Sect. 2 presents an overview of the original SaDE, the JADE and MMTS before proceeding on to discuss the details of the proposed SaDE-MMTS. Section 3 presents experimental results with analysis. The paper is concluded in Sect. 4 with a brief statement on future research directions.

2 SaDE with MMTS

In this section, the standard DE is first introduced followed by the SaDE, the JADE mutation, and the original multi-trajectory search (MTS). Finally, we explain how we combine them to form the SaDE with A MMTS (SaDE-MMTS).

2.1 Standard differential evolution

DE algorithm aims at evolving a population of NP D-dimensional parameter vectors $\mathbf{X}_{i,G} = \{x_{i,G}^1,\ldots,x_{i,G}^D\}$, $i = 1,\ldots,\mathrm{NP}$ toward the global optima. The initial population should cover the entire search space by uniformly randomizing individuals within the search space constrained by the prescribed minimum and maximum parameter bounds $\mathbf{X}_{\min} = \{x_{\min}^1,\ldots,x_{\min}^D\}$ and $\mathbf{X}_{\max} = \{x_{\max}^1,\ldots,x_{\max}^D\}$. For example, the initial value of the jth parameter in the ith individual at generation $G = 0$ is generated by:

$$x_{i,0}^j = x_{\min}^j + \mathrm{rand}(0,1)\cdot\left(x_{\max}^j - x_{\min}^j\right) \quad j = 1,2,\ldots,D \tag{1}$$

where rand(0, 1) represents a uniformly distributed random variable within the range [0, 1]. After initialization, DE employs a mutation operation to produce a mutant vector $\mathbf{V}_{i,G} = \{v_{i,G}^1, v_{i,G}^2,\ldots,v_{i,G}^D\}$ with respect to each individual $\mathbf{X}_{i,G}$, so-called target vector, in the current population where G is the current generation. The five most frequently used mutation strategies implemented in the original DE (Brest et al. 2006; Das et al. 2009; Price 1999; Price et al. 2005; Rahnamayan et al. 2008) are

$$\text{``DE/rand/1'':}\ \mathbf{V}_{i,G} = \mathbf{X}_{r_1^i,G} + F\cdot\left(X_{r_2^i,G} - X_{r_3^i,G}\right) \tag{2}$$

$$\text{``DE/best/1'':}\ \mathbf{V}_{i,G} = \mathbf{X}_{best,G} + F\cdot\left(\mathbf{X}_{r_1^i,G} - \mathbf{X}_{r_2^i,G}\right) \tag{3}$$

$$\text{``DE/rand-to-best/1'':}\ \mathbf{V}_{i,G} = \mathbf{X}_{r_1^i,G} + F\cdot\left(\mathbf{X}_{best,G} - \mathbf{X}_{r_2^i,G}\right) + F\cdot\left(X_{r_3^i,G} - X_{r_4^i,G}\right) \tag{4}$$

$$\text{``DE/best/2'':}\ \mathbf{V}_{i,G} = \mathbf{X}_{best,G} + F\cdot\left(\mathbf{X}_{r_1^i,G} - \mathbf{X}_{r_2^i,G}\right) + F\cdot\left(\mathbf{X}_{r_3^i,G} - \mathbf{X}_{r_4^i,G}\right) \tag{5}$$

$$\text{``DE/rand/2'':}\ \mathbf{V}_{i,G} = \mathbf{X}_{r_1^i,G} + F\cdot\left(\mathbf{X}_{r_2^i,G} - \mathbf{X}_{r_3^i,G}\right) + F\cdot\left(\mathbf{X}_{r_4^i,G} - \mathbf{X}_{r_5^i,G}\right) \tag{6}$$

The indices r_1^i, r_2^i, r_3^i, r_4^i, r_5^i are mutually exclusive integers randomly generated within the range [1, NP], which are also different from the index i. All these indices are randomly generated at a time for each mutant vector. The scaling factor F is a positive control parameter for scaling the difference vector. $\mathbf{X}_{best,G}$ is the best individual vector with the best fitness value in the population at generation G.

After the mutation phase, crossover operation is applied to each pair of the target vector $\mathbf{X}_{i,G}$ and its corresponding mutant vector $\mathbf{V}_{i,G}$ to generate a trial vector (offspring): $\mathbf{U}_{i,G} = (u_{i,G}^1, u_{i,G}^2,\ldots,u_{i,G}^D)$. The binomial (uniform) crossover employed in the original DE is defined as follows:

$$u_{i,G}^{j} = \begin{cases} v_{i,G}^{j}, & \text{if (with probability of CR) or } (j = j_{\text{rand}}) \\ x_{i,G}^{j}, & \text{otherwise} \quad j = 1, 2, \ldots, D \end{cases} \tag{7}$$

In Eq. 7, the crossover rate CR is a user-specified real value within the range [0, 1]. CR controls the fraction of parameter values copied from the mutant vector. j_{rand} is a randomly generated integer in the range [1, D]. The binomial crossover operator copies the jth parameter value from the mutant vector $\mathbf{V}_{i,G}$ to the corresponding element in the trial vector $\mathbf{U}_{i,G}$ if $\text{rand}_j[0, 1) \le \text{CR}$ or $j = j_{\text{rand}}$. Otherwise, it is copied from the corresponding target (or parent) vector $\mathbf{X}_{i,G}$.

Another commonly used one is exponential crossover operator, in which the parameters of trial vector $\mathbf{U}_{i,G}$ are inherited from the corresponding mutant vector $\mathbf{V}_{i,G}$ starting from a randomly chosen parameter index, then continuing to copy the parameter values form the mutant vector $\mathbf{V}_{i,G}$ till the jth parameter value satisfying $\text{rand}_j[0, 1) > \text{CR}$. The remaining parameters of the trial vector $\mathbf{U}_{i,G}$ are copied from the corresponding target vector $\mathbf{X}_{i,G}$. The condition $j = j_{\text{rand}}$ is introduced in Eq. 7 to ensure that the trial vector $\mathbf{U}_{i,G}$ will differ from its corresponding target vector $\mathbf{X}_{i,G}$ by at least one parameter. DE's exponential crossover operator is functionally equivalent to the circular two-point crossover operator. In this paper, we make use of these two crossover operators.

If the values of some parameters of a newly generated trial vector exceed the corresponding upper and lower bounds, a uniformly randomized re-initialization is used within the pre-specified range. Then the objective function values of all trial vectors are evaluated. After that, a selection procedure is performed. The objective function value of each trial vector $f(\mathbf{U}_{i,G})$ is compared with that of its corresponding target vector $f(\mathbf{X}_{i,G})$ of the current population. If the trial vector has less or equal objective function value (minimization problems) than the corresponding target vector, the trial vector will replace the target vector in the population for the next generation. Otherwise, the target vector will remain in the population for the next generation. The selection operation is expressed as follows:

$$\mathbf{X}_{i,G+1} = \begin{cases} \mathbf{U}_{i,G}, & \text{if } f(\mathbf{U}_{i,G}) \le f(\mathbf{X}_{i,G}) \\ \mathbf{X}_{i,G}, & \text{otherwise} \end{cases} \tag{8}$$

The aforementioned three steps are repeated generation after generation until one of the specified termination criteria is satisfied.

2.2 Self-adaptive DE

To achieve good performance by applying the conventional DE to a given problem, it is necessary to perform a trial-and-error search for the most appropriate trial vector generation strategy and fine-tune its associated control parameter values, i.e., the values of CR, F, and NP. During different stages of evolution, different trial vector generation strategies coupled with specific control parameter values can be more effective than others. Motivated by these observations, we employed SaDE (Qin et al. 2009), in which both trial vector generation strategies and their associated control parameter values can be gradually self-adapted according to their previous records of generating promising solutions.

SaDE maintains a strategy candidate pool. During evolution, with respect to each target vector in the current population, one strategy will be chosen from the candidate pool according to a probability learnt from its previous experience of generating improved solutions and applied to perform the mutation operation. The probability of being chosen in the current generation to generate offspring is proportional to generating improved offspring in the past iterations.

In SaDE, the population size NP is a user-specified parameter according to the pre-estimated complexity of the given problem. CR is usually more sensitive to problems with different characteristics, while F is closely related to the convergence speed (Price et al. 2005). Hence, the parameter F is approximated by a normal distribution with mean value of 0.5 and standard deviation of 0.3, denoted by $N(0.5, 0.3)$. A set of F values are randomly sampled from the normal distribution and applied to each target vector of the current population. Therefore, both exploitation (with small F values) and exploration (with large F values) are maintained throughout the entire evolution process. The proper choice of CR can lead to successful optimization performance while a wrong choice may deteriorate the performance. Therefore, in SaDE, gradual adaptation of the range of CR values is considered for a given problem according to previous CR values that have generated trial vectors successfully entering the next generation. Details on the updating of probabilities with respect to each strategy, updating the *Success and Failure Memories* and learning period (LP) can be found in the original SaDE work (Qin et al. 2009).

2.3 JADE mutation

Compared with DE/rand, the greedy strategies such as DE/current-to-best and DE/best benefit from their fast convergence property by incorporating best solution information in the evolutionary search. However, the best solution information may also cause premature convergence. As a compromise, a new mutation strategy, named DE/current-to-*pbest* with small-sized archive called *arch*, is proposed in Zhang and Sanderson (2009). Any of the top $p\%$, $p \in (0, 100]$ solutions can be randomly chosen to play the role of the *pbest* solution in DE/current-to-*pbest*. The archive operation is made very simple to avoid computation overhead. The archive is initiated to be empty. Then, after each generation, the parent solutions that fail in the selection process (parent–offspring comparison) are added to the archive. If the archive size exceeds a certain threshold, defined as the population size NP, then some solutions are randomly removed from the archive to keep the archive size as NP. The archive provides information about the evolution path and improves the diversity of the population. In DE/current-to-*pbest*/1 with archive, a mutation vector is generated as follows:

$$\text{``DE/rand-to-}pbest/1\text{''}: \mathbf{V}_{i,G} = \mathbf{X}_{r_1^i,G} + F \cdot \left(\mathbf{X}_{pbest,G} - \mathbf{X}_{r_2^i,G}\right) + F \cdot \left(\mathbf{X}_{r_3^i,G} - \tilde{\mathbf{X}}_{r_4^i,G}\right) \quad (9)$$

where $\mathbf{X}_{r_1^i,G}$,$\mathbf{X}_{r_2^i,G}$, $\mathbf{X}_{r_3^i,G}$ and $\mathbf{X}_{pbest,\,G}$ are randomly selected from the current population named *pops*, while $\tilde{\mathbf{X}}_{r_4^i,G}$ is randomly chosen from the union, $pops \cup arch$, of the current population and the archive.

2.4 MTS

The modified multiple trajectory search (MMTS) was applied to solve the unconstrained real-parameter large-scale optimization problem (Tseng and Chen 2007, 2008). The original MTS employs search agents, which search for better solutions by moving with different step sizes in the parameter space from the original positions in each dimension. The step size was defined within a search range to fit the requirement of global and local search. In this paper, we integrate a MMTS into the SaDE to implement the line search along each dimension one by one. The details will be presented in the next section.

2.5 SaDE-MMTS

Optimization algorithms perform differently when solving different optimization problems due to their distinct characteristics. But, some of them often lose their efficacy when applied to solve complex problem instances with high dimensions. There are two main difficulties for those optimization methods whose performance deteriorate quickly as the dimensionality of the search space increases. First one is the high demand on exploration capabilities of the optimization methods. When the solution space of a problem increases exponentially with the number of problem dimensions, more efficient search strategies are required to explore all promising regions within a given time budget. Second, the complexity of a problem characteristics may increase with increasing dimensionality, e.g., uni-modality in lower dimensions may become multi-modality in higher dimensions for some problems. Due to these reasons, a successful search strategy in lower dimensions may no longer be capable of finding the optimal solution in higher dimension (Tang et al. 2007, 2010).

In order to solve the complex problem instances especially with high dimensions, the JADE mutation strategy with small sized archive is coupled with two basic crossover operators (binomial crossover and exponential crossover) as well as no crossover option. The mutation strategies with exponential crossover and without any crossover are appropriate for solving linked problems. The binomial crossover is effective for solving separable problems when the CR is low while it is also effective for solving non-separable problems when CR is large. SaDE-MMTS also benefits from the self-adaptation of trial vector generation strategies and control parameter adaptation schemes in the SaDE by learning from their previous experiences to fit different characteristic of the problems and different search requirements of evolution phases. In the SaDE-MMTS, with respect to each target vector in the current population, one trial vector generation strategy is selected from the candidate pool according to the probability learned from its success rate in generating improved solutions within a certain number of previous generations. These probabilities are gradually adapted during evolution in the following manner. Assume that the probability of applying the *k*th strategy in the candidate pool to a target vector in the current population is p_k, $k = 1, 2,\ldots,K$, where K is the total number of strategies in the pool. At the beginning, the probabilities with respect to each strategy are initialized as $1/K$, i.e., all strategies have the equal probability to be chosen. At the generation G, after evaluating all the generated trial vectors, the number of trial vectors generated by the *k*th strategy that can successfully enter the population is recorded as $ns_{k,G}$ while the number of trial vectors generated by the *k*th strategy that are discarded is recorded as $nf_{k,G}$. *Success and Failure Memories* are introduced to store these numbers within a fixed number of previous generations hereby named LP. Once two memories overflow after LP generations, the earliest records stored in the memories, i.e., $ns_{G-\mathrm{LP}}$ or $nf_{G-\mathrm{LP}}$ will be removed so that those numbers calculated in the current generation can be stored in the memories.

After the initial LP generations, the probabilities of choosing different strategies will be updated at each subsequent generation based on the *Success* and *Failure Memories*. For example, at the generation G, the probability of choosing the *k*th ($k = 1, 2,\ldots,K$) strategy is updated by:

$$p_{k,G} = \frac{S_{k,G}}{\sum_{k=1}^{K} S_{k,G}},$$
$$\text{where } S_{k,G} = \frac{\sum_{g=G-\mathrm{LP}}^{G-1} ns_{k,g}}{\sum_{g=G-\mathrm{LP}}^{G-1} ns_{k,g} + \sum_{g=G-\mathrm{LP}}^{G-1} nf_{k,g}} + \varepsilon,$$
$$(k = 1, 2, \ldots, K; G > \mathrm{LP}) \quad (10)$$

where $S_{k,G}$ represents the success rate of the trial vectors generated by the kth strategy and successfully entering the next generation within the previous LP generations with respect to generation G. The small constant value $\varepsilon = 0.01$ is used to avoid the possible null success rates. To ensure that the probabilities of choosing strategies are always summed to 1, $S_{k,G}$ are further divided by $\sum_{k=1}^{K} S_k$ to obtain $p_{k,G}$. Obviously, the larger the success rate for the kth strategy within the previous LP generations is, the larger the probability of applying it to generate trial vectors in the current generation will be. The JADE mutation strategy contributes *pbest* as well as the evolution path to the SaDE-MMTS.

The MMTS is used periodically for a certain number of function evaluations, which is also determined by an adaptation procedure. At the beginning of optimization procedure, the SaDE and the MMTS are firstly conducted sequentially within one search cycle by using same number of function evaluations. Then the success rates of both SaDE and MMTS are calculated similarly as in Eq. 10). Subsequently, function evaluations are assigned to SaDE and MMTS in each search cycle proportional to the success rates of both search methods.

An adaptation approach is proposed in the paper to adaptively determine the initial step size parameter used in the MMTS every time the MMTS is applied. In each MMTS phase, we calculate the average of all mutual dimension-wise distances between current population members (AveDis), select one of the five linearly reducing factors (LRF) from 1 to 0.02, 5 to 0.02, 10 to 0.02, 20 to 0.02 and 40 to 0.02, and apply this LRF to scale AveDis over the evolution. After that, the step size will be further reduced when a better solution is found along a particular dimension.

The search agents in each MMTS step are selected from the current DE population by using a Clearing procedure (Pétrowski 1996). The number of the search agents in each MMTS step is linearly reduced from 4 to 1, which associates the variation of search requirement from "global" to "local". Consequently, the SaDE-MMTS is proposed to solve 19 problems proposed for the special issue of soft computing on scalability of evolutionary algorithms for large scale continuous optimization problems (Herrera et al. 2010). The proposed SaDE-MMTS is presented in Table 1.

3 Experimental results

Nineteen test functions (F1–F19) that should be used for the experimental study for the Special Issue of soft computing on scalability of evolutionary algorithms and other metaheuristics for large scale continuous optimization problems (Herrera et al. 2010) are considered in the simulation. Here is a summary of the 19 test problems:

The F1–F11 functions are the following:

- Shifted unimodal functions:
  - F1: shifted sphere function
  - F2: Shifted Schwefel's problem 2.21
- Shifted multimodal functions:
  - F3: Shifted Rosenbrock's function
  - F4: Shifted Rastrigin's function
  - F5: Shifted Griewank's function
  - F6: Shifted Ackley's function
- Shifted unimodal functions:
  - F7: Shifted Schwefel's problem 2.22
  - F8: Shifted Schwefel's problem 1.2
  - F9: Shifted extended f10
  - F10: Shifted Bohachevsky
  - F11: Shifted Schaffer

The hybrid composition functions *F12–F19* are obtained combining two functions from the set *F1–F11*.

3.1 Parameter setting

Experiments were conducted on all 19 minimization problems with 50, 100, 200, 500, and 1,000 dimensions. To solve these problems, the population size of SaDE-MMTS is set as 60. The maximum number of fitness evaluations is 5000D. Each run stops when the maximal number of evaluations is exhausted. The size of JADE archive is the same as the population size and the *pbest* selection ratio from the population is set as 0.5 (Zhang and Sanderson 2009), i.e., $p = 50\%$ or top 30 of DE population is used as the *pbest*s. The initial control parameters for binomial crossover and exponential crossover are set as ($F = 0.5$, $\mathrm{CR} = 0.2$) and ($F = 0.5$, $\mathrm{CR} = 0.9$), respectively. Furthermore, the parameter F is approximated by a normal distribution with mean value 0.5 and standard deviation 0.3, denoted by $N(0.5, 0.3)$, and CR obeys a normal distribution with mean value CR_m obtained by self-adaptation and standard deviation 0.1, denoted by $N(\mathrm{CR}_m, 0.1)$ (Qin et al. 2009). The LP is set as 50 generations (Qin et al. 2009).

Table 1 The algorithmic description of SaDE-MMTS

INITIALIZATION: Set the generation counter $G = 0$, and randomly initialize a population (*pops*) of *NP* individuals $\mathbf{P}_G = \{\mathbf{X}_{1,G}, \ldots, \mathbf{X}_{NP,G}\}$ with $\mathbf{X}_{i,G} = \{x^1_{i,G}, \ldots, x^D_{i,G}\}$, $i = 1,\ldots,NP$ uniformly distributed in the range $[\mathbf{X}_{\min}, \mathbf{X}_{\max}]$, where $\mathbf{X}_{\min} = \{x^1_{\min}, \ldots, x^D_{\min}\}$ and $\mathbf{X}_{\max} = \{x^1_{\max}, \ldots, x^D_{\max}\}$. Initialize the mean value of *CR* (CRm_k), strategy probability ($p_{k,G}, k = 1,\ldots,K$, K is the no. of available strategies), learning period (*LP*). Evaluate the initial population.

INTIAL OPTIMIZATION STEP

Conduct both SaDE and MMTS in one search cycle for a certain number of function evaluations, determine the success rates for both approaches. Assign updated number of function evaluations proportionally according to the success rates of SaDE and MMTS in the next search cycle for both search methods.

OPTIMIZATION LOOP

WHILE stopping criterion is not satisfied.

Conduct both SaDE and MMTS in one search cycle, update the success rates for both approaches. Reassign updated number of function evaluations proportionally according to the success rates of SaDE and MMTS for the next search cycle for both search methods.

```
Step1. Calculate strategy probability $p_{k,G}$ and update the Success and Failure Memory
    IF $G>LP$
        For $k=1$ to $K$
            Update the $p_{k,G}$ by equation (10)
            Remove $ns_{k,G-LP}$ and $nf_{k,G-LP}$ out of the Success and Failure Memory respectively.
        END
    END
Step2. Assign trial vector generation strategy by using stochastic universal sampling to target vector $\mathbf{X}_{i,G}$. Assign control
    parameters to each target vector $\mathbf{X}_{i,G}$. Assign $\mathbf{X}_{pbest,G}$, $\mathbf{X}_{r_1^i,G}$ and $\tilde{\mathbf{X}}_{r_2^i,G}$ for each target vector $\mathbf{X}_{i,G}$.
  /* Assign trial vector generation strategy */
      Using stochastic universal sampling to select one strategy k for each target vector $\mathbf{X}_{i,G}$
  /* Assign control parameter $F$ */
      FOR $i=1$ to $NP$
          $F_i = Normrnd(0.5, 0.3)$
      END FOR
  /* Assign control parameter $CR$ */
      IF $G>=LP$
          FOR $k = 1$ to $K$, $CRm_k = median(CRMemory_k)$, END FOR
      END IF
       FOR $k = 1$ to $K$
          FOR $i=1$ to $NP$
              $CR_{k,i} = Normrnd(CRm_k, 0.1)$
              WHILE $CR_{m,i} < 0$ or $CR_{m,i} > 1$
              $CR_{k,i} = Normrnd(CRm_k, 0.1)$
              END
          END FOR
       END FOR
  /* Assign pbest vector and the other two distinct parameter vectors for each target vector */
       FOR $i=1$ to $NP$
              Randomly select the $\mathbf{X}_{pbest,G}$ fot the target vector $\mathbf{X}_{i,G}$ from the top p% out of the pops.
              Randomly select the $\tilde{\mathbf{X}}_{r_1^i,G}$ for the target vector $\mathbf{X}_{i,G}$ from the pops.
              Randomly select the $\tilde{\mathbf{X}}_{r_2^i,G}$ fot the target vector $\mathbf{X}_{i,G}$ from the $pops \cup arch$.
       END FOR
```

Table 1 continued

```
Step3. Generate a new population where each trial vector $\mathbf{U}_{i,G}^{k}$ is generated according to associated trial vector generation
    strategy k out of the following listed three combinations and generated parameters $F_i$ and $CR_{k,i}$.
            Generation Strategy 1.   JADE mutation with exponential crossover;
            Generation Strategy 2.   JADE mutation with binomial crossover;
            Generation Strategy 3.   JADE mutation with no crossover;
    Randomly reinitialize a value to the variable of trial vector $\mathbf{U}_{i,G}^{k}$ within the search space it is outside its boundaries.
Step4. Selection and updating
    FOR $i=1$ to $NP$
    Evaluate the trial vector $\mathbf{U}_{i,G}^{k}$
      IF $f(U_{i,G}^{k}) \le f(X_{i,G})$
        $X_{i,G+1} = U_{i,G}^{k}$, $f(X_{i,G+1}) = f(U_{i,G}^{k})$
        $ns_{k,G} = ns_{k,G} + 1$
        Store $CR_{k,i}$ into $CRMemory_k$, store $X_{i,G}$ into JADE archive, if the archive size is full, truncate the archive by
        removing a randomly selected archived member.
        IF $f(U_{i,G}) < f(X_{best,G})$
          $X_{best,G} = U_{i,G}$, $f(X_{best,G}) = f(U_{i,G})$
        END IF
      ELSE
        $nf_{k,G} = nf_{k,G} + 1$
      END IF
    END FOR
    Store $ns_{k,G}$ and $nf_{k,G}$ ($k = 1,...,K$) into the Success and Failure Memory respectively.
    Increment the generation count $G = G+1$
Step5.  IF search cycle of SaDE is completed
            FOR each search agent selected by Clearing
              Apply MMTS along each dimension with initial step size as $LRF*AveDis$.
            ENDFOR
        ENDIF
END WHILE
```

According to the requirements (Herrera et al. 2010), solution quality for each function is measured as defined as: $f(x) - f(\mathrm{op})$, when the FEs reaches the maximum evaluations. Here op is the optimal parameters of the function. Maximum, median, minimum, and average error values of the best individual found in each of the 25 runs are presented in Tables 2 and 3. The SaDE-MMTS is compared with the three evolutionary algorithms (DE, Real-coded CHC, and G-CMA-ES) used as the baseline algorithms in the special issue of soft computing on scalability of evolutionary algorithms and other meta-heuristics for large scale continuous optimization problems. Statistical comparison on the average error is carried out by applying non-parametric tests and presented in Table 4. The computational running time of the SaDE-MMTS is also provided in Table 5. The computer system runs Windows XP (SP2) with Pentium(R) 4 3.00 GHz, *RAM*: 2 GB. Matlab 7.1 is used as the programming language.

3.2 Analysis of the results

In this section, the experimental results obtained with the proposed algorithm are analyzed. In particular, a statistical comparison on the average error is conducted against the original SaDE with JADE mutation as well as the reference algorithms specified for this special issue. Finally, the computational running time of the proposed SaDE-MMTS is reported.

From the experimental results in Tables 2 and 3, we can observe that out of the 19 test functions, problems 1, 3, 5, 6, 7, 10, 12, 15, 16, and 19 can be comparatively easily solved by the proposed SaDE-MMTS irrespective of the scaling of dimensions. Among those 10 functions, functions 1, 6 and 7 are separable; functions 12 and 16 are 2 composition functions which combine separable functions 1 and 9. By performing line search with MMTS, they should be optimized variable by variable. Also benefiting from the binomial crossover, the SaDE-MMTS performs

Table 2 Experimental results of 25 runs obtained by SaDE-MMTS on problem 1–10

	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10
50D										
Best	0	0	0	0	0	0	0	4.4545e−10	8.1081e−02	0
Median	0	0	0	0	0	0	0	3.0454e−09	1.1018e−01	0
Worst	0	0	0	0	0	0	0	1.7532e−08	1.9025e−01	0
Average	0	0	0	0	0	0	0	4.1349e−09	1.3584e−01	0
SD	0	0	0	0	0	0	0	6.1641e−09	1.1641e−01	0
100D										
Best	0	0	0	0	0	0	0	6.1262e−06	1.2856e−01	0
Median	0	0	0	0	0	0	0	2.6778e−04	2.2137e−01	0
Worst	0	0	0	0	0	0	0	3.9786e−04	5.9347e−01	0
Average	0	0	0	0	0	0	0	3.0522e−04	3.1831e−01	0
SD	0	0	0	0	0	0	0	3.1615e−04	2.3154e−02	0
200D										
Best	0	0	0	0	0	0	0	9.2076e+00	5.7459e−01	0
Median	0	1.0229e+00	0	1.9849e−02	0	0	0	2.3691e+01	1.0815e+00	0
Worst	0	3.2684e+00	0	1.9798e−01	0	0	0	5.3190e+01	2.0170e+00	0
Average	0	1.3410e+00	0	8.0899e−02	0	0	0	2.6753e+01	1.2403e+00	0
SD	0	1.4616e+00	0	9.4616e−02	0	0	0	8.5501e+00	2.1961e+00	0
500D										
Best	0	8.6415e+00	0	1.4695e+00	0	0	0	1.5694e+02	4.6115e+00	0
Median	0	1.2131e+01	0	3.1960e+00	0	0	0	2.8418e+02	2.6313e+01	0
Worst	0	1.6156e+01	0	5.8169e+00	0	0	0	5.8474e+02	4.5152e+01	0
Average	0	1.2546e+01	0	3.8563e+00	0	0	0	3.0164e+02	2.8122e+01	0
SD	0	5.8941e+00	0	4.4961e−01	0	0	0	4.5110e+01	4.5113e+00	0
1000D										
Best	0	4.0546e+01	0	2.5841e+01	0	0	0	1.0561e+03	5.4512e+01	0
Median	0	4.6414e+01	0	3.0655e+01	0	0	0	1.2564e+03	8.5612e+01	0
Worst	0	5.5642e+01	0	4.0229e+01	0	0	0	2.8645e+03	1.2348e+02	0
Average	0	4.8875e+01	0	3.2145e+01	0	0	0	1.4628e+03	9.2524e+01	0
SD	0	6.8005e+00	0	4.5145e+00	0	0	0	3.6545e+02	1.5116e+01	0

All the results below 1e−14 have been approximated to 0

perfectly on them. Moreover, with the greedy strategy in mutation phase, high precision of solutions can be achieved on those functions. In contrast, the basic functions 3, 5, and 10 are non-separable functions which can be solved by using the exponential crossover with control parameter adaptation schemes and without any crossover. The SaDE-MMTS performs well on them also. The perfect performance is difficult to be achieved on both separable and highly linked functions by SaDE-MMTS which is a compromise between the dimension-wise search as well as general directional search. Consequently, the performance of the SaDE-MMTS when solving functions 8, 9, and 11 deteriorates when the number of dimensions is increased. Comparatively, the four composition functions, namely 13, 14, 17, and 18, combine the relatively hardest basic functions 3, 4, and 9. The proposed SaDE-MMTS is facing difficulty in solving them, especially when the number of dimensions is greater than 200.

In order to demonstrate the improvement of the proposed SaDE-MMTS when compared with the original SaDE with JADE mutation, four out of the 19 test functions with 1,000 dimensions (F3, F9, F13 and F19) are selected to be solved by both methods. The average errors obtained by both methods on this function group are presented in Table 6. The table shows clearly the superior performance achieved by SaDE-MMTS compared with the original SaDE with JADE mutation.

The indicative results obtained by three reference algorithms specified for this special issue (Herrera et al. 2010) show that DE with exponential crossover achieves the best performance than other two indicative algorithms. Therefore, in Table 4, only a comparison

Table 3 Experimental results of 25 runs obtained by SaDE-MMTS on problem 11–19

	F11	F12	F13	F14	F15	F16	F17	F18	F19
50D									
Best	3.4100e−05	0	4.0288e−01	5.5227e−09	0	0	3.3926e−01	2.0392e−03	0
Median	4.7037e−05	0	1.4239e+00	3.3544e−08	0	0	3.4407e−01	8.1103e−03	0
Worst	8.8619e−05	0	1.2352e+01	5.0622e−08	0	0	6.9469e−01	1.1422e−02	0
Average	5.1919e−05	0	4.2352e+00	3.9366e−08	0	0	4.7842e−01	9.3882e−03	0
SD	7.1961e−06	0	4.6191e+00	6.2614e−09	0	0	5.4941e−02	1.8441e−03	0
100D									
Best	6.3085e−05	0	5.0705e+00	3.2652e−04	0	0	3.1848e+00	8.1542e−03	0
Median	1.3069e−04	0	2.3406e+01	8.2050e−03	0	0	1.0016e+01	3.1257e−02	0
Worst	4.0392e−04	0	4.5811e+01	1.6313e−02	0	0	3.0398e+01	7.5505e−02	0
Average	2.0017e−04	0	3.3080e+01	1.0282e−02	0	0	1.1719e+01	4.7029e−02	0
SD	2.1949e−04	0	5.1641e+00	6.1658e−03	0	0	5.1648e+00	6.4165e−03	0
200D									
Best	9.9286e−05	0	3.6922e+01	7.0715e−03	0	0	2.4531e+01	7.0611e−02	0
Median	1.9458e−04	0	8.2768e+01	1.6604e−02	0	0	3.0080e+01	2.9554e−01	0
Worst	5.5211e−04	0	9.3254e+01	2.2528e−02	0	0	5.6547e+01	3.6500e−01	0
Average	2.3926e−04	0	8.8902e+01	1.5751e−02	0	0	3.5030e+01	3.3546e−01	0
SD	7.0554e−05	0	1.4615e+01	1.7506e−02	0	0	5.0456e+00	3.5964e−01	0
500D									
Best	9.1615e+00	0	2.5314e+02	5.1634e−02	0	0	6.8454e+01	1.0024e+00	0
Median	2.0164e+01	0	3.0522e+02	2.6440e−01	0	0	8.8405e+01	1.1516e+00	0
Worst	3.6515e+01	0	4.6881e+02	6.4615e−01	0	0	1.6554e+02	1.3578e+00	0
Average	2.5328e+01	0	3.2784e+02	4.0112e−01	0	0	9.8075e+01	1.1898e+00	0
SD	8.5114e+00	0	5.0064e+01	4.5131e−01	0	0	2.5641e+01	2.3564e−01	0
1000D									
Best	6.9116e+01	0	4.5582e+02	9.8451e+01	0	0	1.5451e+02	5.6018e+01	0
Median	9.3347e+01	0	5.4631e+02	1.3401e+02	0	0	1.8556e+02	7.6545e+01	0
Worst	5.1612e+02	0	9.9085e+02	1.5131e+02	0	0	2.0164e+02	9.6299e+01	0
Average	1.8516e+02	0	6.5502e+02	1.4007e+02	0	0	1.9045e+02	8.2054e+01	0
SD	2.1674e+02	0	8.7559e+01	3.0847e+01	0	0	3.4516e+01	1.2610e+01	0

All the results below 1e−14 have been approximated to 0

between SaDE-MMTS and DE on the whole set of functions and dimensions is presented. In Table 4, for each function, the better value out of the two competing average errors obtained by SaDE-MMTS and DE are show in bold, as well it is assigned the rank value as 1, while the other one is given the rank as 2. In the last column of this table, all the rank values are accumulated to show the performance difference of two algorithms. Overall, the SaDE-MMTS performs superior than the reference DE algorithm for all five different dimensions. Accordingly, the proposed SaDE-MMTS is able to achieve compromised performance on both separable and non-separable problems. With increasing number of dimensions, the SaDE-MMTS still performs in a stable manner on most of the test functions demonstrating a good scalability characteristic.

4 Conclusions

In order to solve complex problem instances especially with high dimensions, this paper proposed SaDE-MMTS incorporating the JADE mutation strategy with small-sized archive and two crossover operators (binomial crossover and exponential crossover) as well as no crossover option. Moreover, SaDE-MMTS employs adaptation of trial vector generation strategies and control parameters in the SaDE by learning from their previous experiences to suit different characteristics of the problems and different requirements of evolution phases. Furthermore, a MMTS (i.e., line search) is used to improve the solutions generated by the SaDE.

The SaDE-MMTS hybridization benefits from SaDE, JADE mutation strategy, different crossover operators and

Table 4 Comparison on the average error of 25 runs obtained by four algorithms (A1: DE, A2: SaDE-MMTS)

	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10
50D										
A1	**0.00e+00**	3.29e−01	2.90e+01	1.51e−13	**0.00e+00**	1.42e−13	**0.00e+00**	3.54e+00	2.73e+02	**0.00e+00**
A2	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**4.13e−09**	**1.35e−01**	**0.00e+00**
100D										
A1	**0.00e+00**	4.34e+00	7.81e+01	4.23e−13	**0.00e+00**	3.13e−13	**0.00e+00**	3.47e+02	5.06e+02	**0.00e+00**
A2	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**3.05e−04**	**3.18e−01**	**0.00e+00**
200D										
A1	**0.00e+00**	1.93e+01	1.77e+02	**3.58e−12**	**0.00e+00**	6.54e−13	**0.00e+00**	5.33e+03	1.01e+03	**0.00e+00**
A2	**0.00e+00**	**1.34e+00**	**0.00e+00**	8.08e−02	**0.00e+00**	**0.00e+00**	**0.00e+00**	**2.67e+01**	**1.24e+00**	**0.00e+00**
500D										
A1	**0.00e+00**	5.33e+01	4.74e+02	**9.22e−03**	**0.00e+00**	1.65e−12	**0.00e+00**	6.11e+04	2.52e+03	**0.00e+00**
A2	**0.00e+00**	**1.25e+01**	**0.00e+00**	3.85e+00	**0.00e+00**	**0.00e+00**	**0.00e+00**	**3.01e+02**	**2.81e+01**	**0.00e+00**
1000D										
A1	**0.00e+00**	8.44e+01	9.69e+02	**1.32e+00**	**0.00e+00**	3.30e−12	**0.00e+00**	2.46e+05	5.13e+03	**0.00e+00**
A2	**0.00e+00**	**4.88e+01**	**0.00e+00**	3.21e+01	**0.00e+00**	**0.00e+00**	**0.00e+00**	**1.46e+03**	**9.25e+01**	**0.00e+00**

	F11	F12	F13	F14	F15	F16	F17	F18	F19	Sum rank
50D										
A1	5.60e−05	5.27e−13	2.44e+01	**2.58e−08**	**0.00e+00**	1.51e−09	6.83e−01	**1.20e−04**	**0.00e+00**	30
A2	**5.19e−05**	**0.00e+00**	**4.23e+00**	3.93e−08	**0.00e+00**	**0.00e+00**	**4.78e−01**	9.38e−03	**0.00e+00**	**21**
100D										
A1	**1.29e−04**	6.18e−11	6.17e+01	**1.30e−07**	**0.00e+00**	3.53e−09	1.28e+01	**2.86e−04**	**0.00e+00**	29
A2	2.00e−04	**0.00e+00**	**3.30e+01**	1.02e−02	**0.00e+00**	**0.00e+00**	**1.17e+01**	4.70e−02	**0.00e+00**	**22**
200D										
A1	2.59e−04	9.36e−10	1.36e+02	**2.71e−07**	**0.00e+00**	7.26e−09	3.70e+01	**4.70e−04**	**0.00e+00**	30
A2	**2.39e−04**	**0.00e+00**	**8.89e+01**	1.57e−02	**0.00e+00**	**0.00e+00**	**3.50e+01**	3.35e−01	**0.00e+00**	**21**
500D										
A1	**6.71e−04**	6.98e−09	3.58e+02	**9.01e−07**	**0.00e+00**	2.05e−08	1.11e+02	**1.22e−03**	**0.00e+00**	29
A2	2.53e+01	**0.00e+00**	**3.27e+02**	4.01e−01	**0.00e+00**	**0.00e+00**	**9.80e+01**	1.18e+00	**0.00e+00**	**22**
1000D										
A1	**1.35e−03**	1.70e−08	7.29e+02	**9.95e−01**	**0.00e+00**	4.19e−08	2.35e+02	**2.37e−03**	**0.00e+00**	29
A2	1.85e+02	**0.00e+00**	**6.55e+02**	1.40e+02	**0.00e+00**	**0.00e+00**	**1.90e+02**	8.20e+01	**0.00e+00**	**22**

All the results below 1e−14 have been approximated to 0

Table 5 Matlab average running time on the 25 runs for the SaDE-MMTS

50D	100D	200D	500D	1000D
5 min	12 min	45 min	250 min	750 min

Table 6 Comparison on the average error of 25 runs obtained by *SaDE-MMTS* and *SaDE with JADE mutation* on four functions

1000D functions	F3	F9	F13	F19
SaDE	1.24e+10	3.51e+03	3.15e+09	6.27e+01
SaDE-MMTS	0	9.25e+01	6.55e+02	0

line search to solve large scale continuous optimization problems with diverse characteristics. Exploration as well as exploitation capabilities are also enhanced simultaneously by combining these schemes. The SaDE-MMTS is simple and easy to implement, possessing fast convergence while diversity is preserved. In our future work, we will investigate more stable parameter settings, integrate additional DE-mutation strategies, and implement modification on this current framework to improve further the scalability of the SaDE-MMTS.

References

Brest J, Greiner S, Boskovic B, Mernik M, Zumer V (2006) Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems. IEEE Trans Evol Comput 10(6):646–657

Das S, Abraham A, Chakraborty UK, Konar A (2009) Differential evolution using a neighborhood based mutation operator. IEEE Trans Evol Comput 13(3):526–553

Herrera F, Lozano M, Molina D (2010) Test suite for the special issue of soft computing on scalability of evolutionary algorithms and other metaheuristics for large scale continuous optimization problems. http://sci2s.ugr.es/eamhco/CFP.php

Huang VL, Qin AK, Suganthan PN (2006) Self-adaptive differential evolution algorithm for constrained real-parameter optimization. In: Proceedings of the IEEE congress on evolutionary computation (CEC 2006), July 2006, pp 17–24

Pétrowski A (1996) A clearing procedure as a niching method for genetic algorithms. In: Proceedings of the IEEE international conference on evolutionary computation, New York, USA, 1996, pp 798–803

Price KV (1999) An introduction to differential evolution. In: Corne D, Dorigo M, Glover F (eds) New ideas in optimization. McGraw-Hill, London, pp 79–108

Price K, Storn R, Lampinen J (2005) Differential evolution—a practical approach to global optimization. Springer, Berlin

Qin AK, Suganthan PN (2005) Self-adaptive differential evolution algorithm for numerical optimization. In: Proceedings of the IEEE congress on evolutionary computation (CEC 2005). IEEE Press, Edinburgh, Scotland, September 2005, pp 1785–1791

Qin AK, Huang VL, Suganthan PN (2009) Differential evolution algorithm with strategy adaptation for global numerical optimization. IEEE Trans Evol Comput 13(2):398–417

Rahnamayan S, Tizhoosh HR, Salama MMA (2008) Opposition-based differential evolution. IEEE Trans Evol Comput 12(1):64–79

Storn R, Price KV (1997) Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. J Glob Optim 11:341–359

Tang K, Yao X, Suganthan PN, MacNish C, Chen YP, Chen CM, Yang Z (2007) Benchmark functions for the CEC'2008 special session and competition on large scale global optimization. Technical Report, Nature Inspired Computation and Applications Laboratory, USTC, China & Nanyang Technological University, Singapore, November 2007

Tang K, Li X, Suganthan PN, Yang Z, Weise T (2009) Benchmark functions for the CEC'2010 special session and competition on large scale global optimization. Technical Report, Nature Inspired Computation and Applications Laboratory, USTC, China & Nanyang Technological University, 2009. http://nical.ustc.edu.cn/cec10ss.php

Tseng LY, Chen C (2007) Multiple trajectory search for multiobjective optimization. In: Proceeding 2007 IEEE congress on evolutionary computation, pp 3609–3616

Tseng LY, Chen C (2008) Multiple trajectory search for large scale global optimization. In: Proceeding 2008 IEEE congress on evolutionary computation, pp 3052–3059

Zhang JQ, Sanderson AC (2009) JADE: adaptive differential evolution with optional external archive. IEEE Trans Evol Comput 13(5): 945–958