European Journal of Operational Research 216 (2012) 33–46

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Continuous Optimization

Differential Evolution algorithm with Separated Groups for multi-dimensional optimization problems

Adam P. Piotrowski [*], Jaroslaw J. Napiorkowski, Adam Kiczko

Institute of Geophysics, Polish Academy of Sciences, Ksiecia Janusza 64, 01-452 Warsaw, Poland

ARTICLE INFO

Article history:
Received 15 March 2011
Accepted 22 July 2011
Available online 29 July 2011

Keywords:
Metaheuristics
Evolutionary Computation
Differential Evolution
Particle Swarm Optimization
Evolution Strategies

ABSTRACT

The classical Differential Evolution (DE) algorithm, one of population-based Evolutionary Computation methods, proved to be a successful approach for relatively simple problems, but does not perform well for difficult multi-dimensional non-convex functions. A number of significant modifications of DE have been proposed in recent years, including very few approaches referring to the idea of distributed Evolutionary Algorithms. The present paper presents a new algorithm to improve optimization performance, namely DE with Separated Groups (DE-SG), which distributes population into small groups, defines rules of exchange of information and individuals between the groups and uses two different strategies to keep balance between exploration and exploitation capabilities. The performance of DE-SG is compared to that of eight algorithms belonging to the class of Evolutionary Strategies (Covariance Matrix Adaptation ES), Particle Swarm Optimization (Comprehensive Learning PSO and Efficient Population Utilization Strategy PSO), Differential Evolution (Distributed DE with explorative-exploitative population families, Self-adaptive DE, DE with global and local neighbours and Grouping Differential Evolution) and multi-algorithms (AMALGAM). The comparison is carried out for a set of 10-, 30- and 50-dimensional rotated test problems of varying difficulty, including 10- and 30-dimensional composition functions from CEC2005. Although slow for simple functions, the proposed DE-SG algorithm achieves a great success rate for more difficult 30- and 50-dimensional problems.

1. Introduction

The global single-objective unconstrained optimization problems considered in this paper may be defined in the following form – for a real valued function $f(\boldsymbol{x})$, find the global optimum vector $\boldsymbol{x}^*$, such that:

$$f(\mathbf{x}^*) = \min_{\mathbf{x}\in\mathbf{\Omega}} f(\mathbf{x}) \tag{1}$$

where $\boldsymbol{x}$ is a M-dimensional parameter vector, $\boldsymbol{x} = \{x^1,\dots,x^M\}$, also called an individual, with domain $\Omega \subset R^M$.

Among a number of Evolutionary Computation methods proposed during the last twenty years, the Differential Evolution (DE) algorithm suggested by Storn and Price (1995, 1997) gained significant popularity. The DE method was successfully applied to some practical problems (Price et al., 2005; Rowiński and Piotrowski, 2008; Zhang et al., 2010; Cruz et al., 2010; Beynon et al., 2010; Piotrowski et al., 2011) and simple benchmark functions (discussion on difficulty of test problems may be found for example in Clerc, 2006). However, for a number of problems (Ilonen et al., 2003; Langdon and Poli, 2007; Mendes et al., 2009) DE performs poorly, especially when applied to rotated multimodal problems with higher dimensionality (Neri and Tirronen, 2010). Therefore, a number of modifications of DE have been proposed (Price et al., 2005; Qin et al., 2009; Neri and Tirronen, 2010; Das and Suganthan, 2011).

The performance of the DE algorithm frequently depends on the control parameter setting. Moreover, DE has only a limited flexibility to adapt to specific problems and frequently prematurely converges to local optimum. To overcome these problems many modifications of DE aims at self-adaptation of DE parameters (Liu and Lampinen, 2005; Brest et al., 2006; Salman et al., 2007; Qin et al., 2009), development of new mutation and crossover strategies (Price et al., 2005; Das et al., 2009) or hybridization of DE with other algorithms (Zhang and Xie, 2003; Omran et al., 2009). Distributed DE methods have recently been proposed based on distributed Evolutionary Algorithms and Island Models (Tanese, 1989; Gustafson and Burke, 2006), including Distributed DE algorithm with Explorative-Exploitative Families (DDE, Weber et al., 2009) and Grouping DE algorithm (GDE, Piotrowski and Napiorkowski, 2010). These algorithms distribute the population into a few sub-populations (or groups) and define simple communication rules between them. However, the idea of distributing a population into many small sub-populations and defining various contact mechanisms between them has not been developed within a DE

[*] Corresponding author. Tel.: +48 22 6915858; fax: +48 22 6915915.
E-mail address: adampp@igf.edu.pl (A.P. Piotrowski).

framework. Similar to the self-adaptation of control parameters or the use of different mutation and crossover strategies, the distribution of a population into small groups with various contact mechanisms between them should improve adaptation capability of the algorithm to various types of problems improving optimization performance.

The present paper proposes a new algorithm pertaining to distributed DE methods, called DE with Separated Groups (DE-SG). The DE-SG is based on the GDE algorithm proposed by the authors in Piotrowski and Napiorkowski (2010), but some its features were inspired by other distributed and self-adaptive DE methods, as well as Island Models. However, DE-SG differs from the other algorithms in its structure and its mechanisms aimed at sorting and exchanging of information.

Unlike in DDE and GDE, the population of individuals in DE-SG is divided into halves (rules of migration of individuals are different in each half) and then each half is further divided into small groups (or sub-populations) that operate independently. Because the exchange of information within a small group is quicker, small groups are able to speed up exploitation. To facilitate exploration, communication between sub-populations is introduced in a few ways. Communication is understood here as gaining information about the location of individuals belonging to different groups and the migration of individuals between groups. In DE-SG, like in GDE, if a particular group has no improvement, the group's members are allowed to communicate with the whole population for some time; doing so allows them to get the necessary knowledge needed to escape from local optimum. One elite group has constant access to information collected by the whole population. However, as the groups are small, relying only on information exchange turns out to be insufficient; hence, the mechanism of migration of individuals between groups is introduced. Rules governing the migration of individuals between groups within each half differ. Within one half, the best individuals migrate relatively quickly to an elite group, while within the other, the best individuals migrate slowly and are distributed more widely among various groups.

On the basis of the Self-adaptive DE (SaDE, Qin et al., 2009) and GDE, in DE-SG an offspring may be produced by one of two strategies of different nature. The first one is expected to perform better exploration, the second exploitation. Although DE-SG introduces a set of new parameter values, only the speed of communication and exchange of individuals between groups may require some tuning by the user, depending on the number of function calls allowed. All results presented in this paper were obtained with the fixed recommended parameter values.

The proposed method is compared to eight state of the art EC algorithms based on 19 rotated 10-to 50-dimensional test problems. The paper is organized as follows: Section 2 describes the DE algorithm and the overview of methods developed to improve its performance; Section 3 describes the DE-SG algorithm; Section 4 provides benchmark results and includes a discussion of the performance of all considered algorithms; Section 5 concludes the paper.

2. Differential Evolution algorithm (DE)

To find the optimal solution $\boldsymbol{x}^*$ in the M-dimensional space, the original DE algorithm (Storn and Price, 1995) randomly initializes population P with individuals $\boldsymbol{x}_i$, $i = 1, \ldots, K$ within Ω.

At each iteration, for every individual $\boldsymbol{x}_i$, ($i = 1,\ldots,K$), three other distinct parameter vectors $\boldsymbol{x}_a$, $\boldsymbol{x}_b$, and $\boldsymbol{x}_c$ are randomly chosen from the population P. Then a new vector $\boldsymbol{u}_i$ is generated by an operation called mutation:

$$\mathbf{u}_i = \mathbf{x_a} + F \cdot (\mathbf{x_b} - \mathbf{x_c}) \tag{2}$$

where F is a predefined scaling parameter. The mutated vector $\boldsymbol{u}_i$ and the target (initial) vector $\boldsymbol{x}_i$ form two parents. In order to form the trial vector (offspring) $\boldsymbol{y}_i$, the crossover between $\boldsymbol{u}_i$ and $\boldsymbol{x}_i$ is defined component-wise as:

$$y_i^j = \begin{cases} u_i^j & \text{if } rand(0,1)_j \leqslant Cr \text{ } or \text{ } j = j_{rand} \\ x_i^j & \text{otherwise} \end{cases} \tag{3}$$

where Cr is a predefined crossover parameter, j_{rand} is a random integer variable from $[1,M]$ and $rand(0,1)_j$ is a uniform random real variable from range $[0,1]$. Note that as a number of mutation and crossover strategies were proposed since 1995 (see Price et al., 2005; Mezura-Montes et al., 2006; Mishra, 2006; Qin et al., 2009; Das et al., 2009), the initial version defined above was named DE/rand/1/bin. Then, if

$$f(\mathbf{y_i}) \leqslant f(\mathbf{x_i}) \tag{4}$$

$\boldsymbol{y}_i$ replaces $\boldsymbol{x}_i$ – this operation is called selection. The entire algorithm terminates when one of the stopping criteria is met. In basic DE, three control parameters, namely scaling factor F, crossover rate Cr and population size K must be specified by the user.

Population size is probably the most problem-dependent control parameter. Storn and Price (1997) suggested that population size K should equal from $5M$ up to $10M$. Different opinions about proper population size may be found in the literature, ranging from $K < M$ to $K > 10M$ (see review in Das et al., 2009; Qin et al., 2009; Weber et al., 2009 or Das and Suganthan, 2011 for detailed discussion). Frequently K is left to be decided by the user.

The value of Cr also heavily depends on the problem (Price et al., 2005), while desirable value differs for separable (lower Cr) and non-separable problems (higher Cr). The value of F was suggested to equal 0.5 by Storn and Price (1997).

Looking for the improvement of original DE, many researchers considered various control parameter values (Gamperle et al., 2002; Rokkonen et al., 2005; Chakraborty et al., 2006; Kaelo and Ali, 2006), for example suggesting different values of F, generally between 0.4 and 1. More recently a number of algorithms with self-adaptive parameters were proposed (Zaharie and Pectu, 2003; Liu and Lampinen, 2005; Omran et al., 2005; Qin and Suganthan, 2005; Qin et al., 2009; Brest et al., 2006; Al-Anzi and Allahverdi, 2007; Zhang and Sanderson, 2009). The results of several self-adaptive DE algorithms were compared in Brest et al. (2007) and Qin et al. (2009)).

Detailed discussion of DE-based algorithms developed during last fifteen years is beyond the scope of the present paper, but good review may be found in Das et al. (2009), Qin et al. (2009), Weber et al. (2009), Neri and Tirronen (2010) and Das and Suganthan (2011). Most improvements aimed at an increase in the algorithm's ability to adapt to specific problems and avoid premature convergence. To achieve it, different methods combining DE with different neighborhood topologies (Tomassini, 2005), used for example in Particle Swarm Optimization (PSO, Eberhart and Kennedy, 1995) were proposed to slow down information propagation (Das et al., 2009; Omran et al., 2009). Alternatively, a concept of cooperative coevolution – developed to decompose high-dimensional problems into smaller, simpler to solve components (Yang et al., 2008) – was introduced into the DE approach. Also, an idea of merging advantages of multiple EC methods, including DE, has been developed into an adaptive multi-algorithm AMALGAM (Vrugt et al., 2009).

Another idea was adopted from Island or Species based models (Tanese, 1989; Holland, 2000; Liu et al., 2000). A distribution of individuals into sub-populations that occasionally communicate allows a better exploration of search space and usually slows down the algorithm's convergence to local optimum. Among a few distributed DE algorithms (Tasoulis et al., 2004; Falco et al., 2007; Apolloni et al., 2008) two become more successful (Weber et al., 2009 and Piotrowski and Napiorkowski, 2010). However, when applied to rotated multimodal problems, distributed DE-based algorithms do not show adequate performance (see, for example, the results presented in Weber et al., 2009 for rotated and non-rotated problems). Possible causes include an inadequate selection of mutation strategy; too few groups to allow the spreading of individuals across search space; and simplicity of contact mechanisms between the groups.

3. Differential Evolution with Separated Groups

This paper presents a new algorithm, Differential Evolution with Separated Groups pertaining to distributed DE, which aims to improve performance for difficult problems by distributing individuals into small sub-populations and defining diversified communication rules between them. The main goal is to make the optimization technique less vulnerable to entrapment in a local minimum and to improve the ability of the algorithm to adapt to various kinds of problems, including rotated ones. The detailed features of the proposed DE-SG algorithm for minimization are defined and explained below:

(**1**) Individuals. Initially all K individuals in the population P are randomly generated within Ω and numbered from 1 to $K(\boldsymbol{x}_1,\ldots,\boldsymbol{x}_K)$. Numbers are assigned to individuals at random, without any relation to their position in the search space or fitness values. The population is divided into halves, which have different migration rules for individuals. A binary variable $Lx(i)$, equal to 0 or 1, is assigned to each index i to decide whether or not individual $\boldsymbol{x}_i$ is able to create offspring. Initially, for any individual $Lx(i) = 0$. Both the numbering of individuals and the value of $Lx(i)$ may change during the search process.

(**2**) Groups. The population P is composed of $K = 10M$ individuals $\boldsymbol{x}_i$ divided into M separate groups $G(k)$, $k \in [1,M]$, each with 10 individuals with consecutive indices i, i.e.

$$\begin{aligned} &P = \cup_{k=1}^{M} G(k) \\ &\forall_{k,l\leqslant M;k\neq l} G(k)\cap G(l) = \emptyset \\ &G(k) = [x_{10k-9},\ldots,x_{10k}] \end{aligned} \tag{5}$$

Note that the individuals with numbers i equal $(10k-9)$ to $(10k)$ always belong to the same group. Hence, if individuals from different groups exchange their numbering, they accordingly exchange their group membership. The number of individuals within a group was selected empirically. Because most of the time an individual exchanges information with the members of his own group only, and because three different individuals are needed for each parent $\boldsymbol{x}_i$ to create offspring, the much smaller number would yield pathological behavior in the algorithm. On the contrary, a much bigger number would slow down information exchange within a group and limit the number of groups, like in DDE and GDE. A binary variable $LG(k)$, such that may contain either one or zero, is assigned to each group, deciding whether this group performs searches with or without collecting information from other groups.

(**3**) Strategies. In DDE, only the single DE/rand/1/bin strategy was applied. In GDE, three mutation and crossover strategies were used together, each with specified probability, whereas in SaDE four strategies with adaptive probability were used. In DE-SG offspring $\boldsymbol{y}_i$ is created according to two of them, namely:

A. DE/rand/1/Mod-either-or,

$$y_i^j = \begin{cases} u_i^j & \text{from Eq. 2} \quad \text{if } rand(0,1)_j \leqslant CR \\ v_i^j = x_i^j + \left(x_b^j + x_c^j - 2\cdot x_i^j\right)\cdot RandNorm(0,1) & \text{otherwise} \end{cases} \tag{6}$$

where $RandNorm(0,1)$ is a random real value generated from standardized normal distribution, and

B. DE/rand/1/Exponential, described for clarity by means of a pseudocode

```
$\mathbf{y_i} = \mathbf{x_i}$
$jrand = \mathrm{ceil}(M\cdot rand(0,1));$
$j = jrand$
do{$y_i^j = u_i^j$   from Eq. 2                              (7)
$j = (j+1)\%M$}
while $(rand(0,1)_j < Cr\&\&j! = jrand)$
```

In DE/rand/1/Exponential, either one element or a string of consecutive element is copied to $\boldsymbol{y}_i$ from $\boldsymbol{u}_i$. The first element of the string is found at random, the length of the string is also a random value that depends on the Cr parameter. Both random values are generated anew for each individual and iteration. The rest of elements of $\boldsymbol{y}_i$ remain the same as in $\boldsymbol{x}_i$.

At each iteration for each parent $\boldsymbol{x}_i$ only one of the two above strategies is chosen randomly with equal probability. Both strategies differ significantly in terms of offspring selection.

The first one does not allow offspring to keep any elements from $\boldsymbol{x}_i$ (except the rare situation when $\mathrm{RandNorm}(0,1)\approx 0$). The number of possible offspring for each parent is infinite because of the presence of a random number generated from normal distribution. Offspring may be located in any point of the search space, but locations closer to the parent are more probable. The first strategy is well suited for rotated problems and exploration, but performs slowly in exploitation.

The second one implies that only a few elements of $\boldsymbol{x}_i$ would change, while the number of offspring that can be produced in such a way is always limited. The second strategy does not perform well for exploring rotated problems, but speeds up exploitation and easily deals with separable functions.

Hence, in the first strategy, the parameter Cr decides which equation is used to create particular element of the offspring, whereas in the second, Cr governs the average number of elements that change in the offspring.

(**4**) Global and distributed search. This procedure is similar to the approach applied in GDE algorithm. Individuals from all but one group perform search exchanging information mostly with members of their own group only. Only if the group sticks in the local optimum will its members learn where other individuals in the population are located. As groups are small, the spread of information among group members is quick; however, the transfer of information between most groups is occasional. Only the individuals from one privileged group are able to learn from the whole population all the time. More specifically, if the binary variable associated with particular group $LG(k) = 0$, then for each parent $\boldsymbol{x}_i \in G(k)$, the vectors $\boldsymbol{x}_a$, $\boldsymbol{x}_b$ and $\boldsymbol{x}_c$ are randomly chosen from the group $G(k)$ only. Otherwise, they are chosen from the whole P. For the first group, $G(1)$ by definition $LG(1)$ is always equal to 1. For all other groups, $LG(k)$ is initially set to 0, but if the group $G(k)(k > 1)$ converges to local minima, its corresponding $LG(k)$ value is changed to 1 for a pre-defined number of iterations (PNI). During these PNI iterations, the group tries to get out from local optima. The convergence to the local optimum is defined in the following way: after every PNI iterations, the factor of improvement (GF) is computed as

$$GF = \left(\sum_{k=1}^{M}(GFBEST(k)_{PNI} - GFBEST(k))\right)/100 \tag{8}$$

where $GFBEST(k)$ and $GFBEST(k)_{PNI}$ are the best values of the objective function for the group $G(k)$ at a current iteration and at PNI iterations earlier, respectively. We assume that the $G(k)$ converges to local optimum if the objective function for the best individual (with $GFBEST(k)$, $k > 1$) in a group $G(k)$ fulfils the relation:

$$GFBEST(k)_{PNI} - GFBEST(k) < GF \tag{9}$$

(**5**) Migration of individuals. According to communication rules defined above, all groups but one would learn nothing from other groups until sticking in local optimum, a phenomenon that alone would make an algorithm non-efficient. As information is usually spread quickly among members of small groups, the occasional exchange of individuals between groups, independent of sticking in local optima, is introduced to DE-SG, thereby becoming a kind of information exchange. Moreover, its purpose is not only to exchange information between groups, but also to select the best individuals from one half of the population and the worst individuals from the other half. These individuals are collected within the most elite and the poorest group. Note that both halves of the population are treated in a different way, which adds diversity to the algorithm's search. More specifically, in a particular iteration, a migration of individuals may occur with some small probability (*MigProb*) related to *PNI*. At the beginning of each iteration, random value *rand* is generated. If *rand* < *MigProb*, the migration starts from the individual $\boldsymbol{x}_1$. For the first half, it proceeds in the following way:

```
for i = 1, $\lfloor K/2 \rfloor - 1$
if $f(\boldsymbol{x}_i) > f(\boldsymbol{x}_{i+1})$ then (exchange indices i and i + 1)
end for
```

The exchange of indices is immediate: within the above loop, an individual with poor fitness may increase its index up to $\lfloor K/2 \rfloor - 1$ and immediately migrate into the "middle" groups (close to $k = M/2$, when $K = 10M$), whereas an individual with good fitness may decrease its index only by 1. This means that a selection of individuals with best fitness is slow. The individuals with good and moderate fitness are scattered among different groups, but the ones with the poorest fitness are quickly moved into a single poorest group $G(K/2)$.

Within the second half of the population, the process is reversed, i.e.:

```
for i = $\lfloor K/2 \rfloor + 1$, $K - 1$
if $f(\boldsymbol{x}_i) < f(\boldsymbol{x}_{i+1})$ then (exchange indices i and i + 1)
end for
```

```
  Strategy 1: DE/rand/1/Mod-either-or
  Strategy 2: DE/rand/1/Exponential

The DE-SG Algorithm.
Step 0.
Preset mutation parameter F = 0.5, crossover parameter Cr = 0.5, the number of individuals in population K = 10*M; parameters of stopping criterion
PNI = 500 and $\mathcal{E}$ = $10^{-5}$, initial value of $FBEST_{PNI}$ (very high) = $10^{30}$;
  Randomly generate population P of K individuals $\mathbf{x_i}$ from the solution space $\Omega$ and divide P into M groups, each composed of 10 individuals
with consecutive indices i;
  LG(1) = 1 and LG(k) = 0 for k = 2, …M;
  Lx(i) = 0 for any $\mathbf{x_i} \in P$
  Iter = 0                  // iteration number;
  Set StopFlag = 0
  Determine the value of the objective function f($\mathbf{x_i}$) for each $\mathbf{x_i} \in P$

while (the stopping criteria is not satisfied) {
Step 1.
  Iter = Iter +1;                 // add 1 to iteration number
  if ( rand(0,1) >= 0.995 )
   for each vector $\mathbf{x_i}$ ( i = 2,…, floor(K/2) ) // individuals from 1st half of P
    if ( f($x_i$) < f($x_{i-1}$) )        // if individual with higher index i is better than one with lower
          exchange indices i and i+1 of individuals $x_i$ and $x_{i-1}$     // Lx(i) is not exchanged
    end if
   end for
   for each vector $\mathbf{x_i}$ ( i = floor(K/2+1),…,K ) //for consecutive individuals from 2nd half of P
    if ( f($x_i$) < f($x_{i+1}$) )        // if individual with higher index i is better than one with lower
          exchange indices i and i+1 of individuals $x_i$ and $x_{i+1}$     // Lx(i) is not exchanged
    end if
   end for
  end if

Step 2
  for each vector $\mathbf{x_i}$ (i=1, 2,…, K)
  if (Lx(i) = =0)                 //if the individual is not frozen
   IS = ceil(2*rand(0,l)); // select DE/rand/1/Mod-either-or strategy or DE/rand/1/Exponential
   if (LG(k) = =0)   //use only information from the group to which $\mathbf{x_i}$ belongs
    Generate offspring $\mathbf{y_i}$ based on $\mathbf{x_a}$, $\mathbf{x_b}$, $\mathbf{x_c} \in$ G(k) using Strategy IS;  // Strategy 1 or Strategy 2
   else                           //use from all groups
    Generate offspring $\mathbf{y_i}$ based on $\mathbf{x_a}$, $\mathbf{x_b}$, $\mathbf{x_c} \in$ P using Strategy IS;  // Strategy 1 or Strategy 2
   end if
   if ( f($\mathbf{y_i}$) < f($\mathbf{x_i}$) )  $\mathbf{x_i} = \mathbf{y_i}$                    // selection
  end if
  end for
  Find   BEST; FBEST;           // the best vector in P with corresponding function
    value
  Find   GB(k), GFBEST(k);     // the best vector in each group k with corresponding
  function  value
```

Fig. 1. Pseudocode of DE-SG algorithm.

```
Step 3
if(Iter (mod) PNI = = 0)        // after every PNI iterations
 for k=1,…, M
  LG(k) = 0
 end for
Compute the factor of improvement (GF) Eq. (8)
for each group G(k), k = 1,…, M
 if (condition described by Eq. (9) is fulfilled) // group trapped in a local minimum
  LG(k) = 1                    // valid for next PNI iterations
  Lx(GB(k)) = 1  // freeze the index of GB(k) in group G(k)
  if (number of frozen index exceeds the predetermined value)
   release the individual $\mathbf{x}_{if}$ with Lx(if) = 1 frozen for the highest number of iterations
   set Lx(if) = 0
  end if
 end if
end for
end if

Step 4
if(Iter (mod) PNI = = 0)        // after every PNI iterations
 if (FBEST$_{PNI}$ – FBEST) > $\mathcal{E}$
  StopFlag = 0
  FBEST$_{PNI}$ = FBEST
 else
  if (StopFlag = =1)
   Output the best results
   Stop algorithm
  else
   StopFlag = 1
   FBEST$_{PNI}$ = FBEST
   Find GFBEST$_W$ = max GFBEST(k)      // the worst from the best
                                          objective function values of all groups
   if (GFBEST$_W$ – FBEST) > $\mathcal{E}$
    LG(k) = 1 for k = 1,.., M // valid for next PNI iteration
   else
    Stop algorithm
   end if
  end if
 end if
end if

} end while
```

Fig. 1 (*continued*)

Here, the individuals with best fitness are quickly moved to the elitist group $G(M)$. Which half of the population would work more efficiently is problem-dependent; but relying on only one of them almost always provides poorer results. Note that the global and distributed search and migration mechanisms allow the algorithm to have various kinds of groups. One group $G(1)$ has constant access to information from whole population, as in non-distributed DE. The individuals with the best fitness value from the first half slowly migrate towards this group. A number of groups ($G(2)$ to $G(K/2)$) work separately with rare communication and slow exchange of individuals with other groups. A number of groups ($G(K/2+1)$ to $G(K-1)$) behave similarly but quickly lose individuals with the best fitness which are wind up in the elite group $G(K)$. Note that, when multi-modal problem is considered, individuals within elitist group would be scattered around the various, most promising local optima found by half of the population. If some relation in the search space between minima locations exists, they potentially would explore it to find still better ones.

(**6**) Marking local optima by "freezing" indices of some individuals. It is advantageous to remember the previously visited optima, as it can help the algorithm not only to avoid returning to known local optima, but also to use them as $\mathbf{x}_a$, $\mathbf{x}_b$, or $\mathbf{x}_c$ to find the better ones. Hence a kind of memory called freezing mechanism was added to DE-SG. Specifically, at every iteration individuals $\mathbf{x}_i$ with $Lx(i) = 0$ create offspring $\mathbf{y}_i$. The individuals with $Lx(i) = 1$ are "frozen" – i.e. they cannot create offspring, but may be used by other individuals as one of the vectors $\mathbf{x}_a$, $\mathbf{x}_b$ or $\mathbf{x}_c$ in the operation of mutation. Initially, for any individual $Lx(i) = 0$. When the binary variable $LG(k)$ assigned to particular group is changed to 1, then the $Lx(i)$ of index i, which is currently associated with the best individual $\mathbf{x}_i$ within $G(k)$ group, is set to 1. Note that $Lx(i)$ does not move with $\boldsymbol{x}_i$ in the migration process, rather it is fixed to index i. The "frozen" individual may be replaced by another during migration. "Freezing" results in decreasing the number of active vectors that can create offspring. Therefore, no more than a predefined number of individuals (*NFI*) may be "frozen". If the next index i is to be "frozen" and the number of "frozen" individuals equals *NFI*, then the individual with an index *if*, which was already "frozen" for the highest number of iterations is "released", and its corresponding variable *Lx(if)* is set back to 0. Note that precise value of *NFI* has a minor effect on the algorithm's performance, it's suggested value is between $K/20$ and $K/2$. In case of many problems *NFI* have no impact on the results, if groups do not stick in local optima too often. However, in case of some sophisticated problems a high number of individuals could be "frozen", hence the need for a formal remedy from fixing most of the population.

(**7**) Complete access to information before termination. The DE-SG algorithm may terminate according to the handling window idea (Beasley et al., 1993) at which the improvement of the objective function during the predefined number of iterations *PNI* is verified:

$$(FBEST_{PNI} - FBEST) < \varepsilon \tag{10}$$

where FBEST denotes the best objective function value at a present iteration, $\text{FBEST}_{\text{PNI}}$ is the best objective function value obtained *PNI* iterations earlier, and ε defines the threshold value of the required minimum improvement. If the criterion (10) is fulfilled at particular iteration, then the algorithm proceeds for next *PNI* iterations with all $LG(k)$ set to 1. If during that time the improvement in FBEST is obtained, then the $LG(k)$, $k > 1$ are set to 0 and the algorithm continues its search, otherwise it terminates. The idea behind this operation is that when all groups stick in local optima, the full contact between the individuals belonging to different groups may allow them to get out and continue the search.

Most problems have defined bounds of Ω, hence a certain procedure must be applied when an individual attempts to cross the boundary. In DE-SG such individual is simply reflected from this boundary.

The pseudocode of DE-SG algorithm is presented in Fig. 1.

4. Experimental results

Since the work of Salomon (1996) it is well known that EC methods should be tested against rotated problems. In the present paper the performance of the proposed DE-SG algorithm and eight different EC approaches is compared on 19 rotated benchmark problems defined in the Appendix and in CEC2005 (Suganthan et al., 2005). To show performance of DE-SG method in a broad context, its performance is compared on 10-, 30- and 50-dimensional problems with following EC algorithms:

(1) SaDE (Qin et al., 2009), DE-based algorithm with self-adapting control parameters which uses a set of four different DE strategies;
(2) DE with global and local neighbors (DEGL, Das et al., 2009), another method from DE family, which introduces a novel mutation strategy based on neighborhood concept and aims at improving balance between exploration and exploitation abilities, it was recently suggested as the most efficient among EC-based algorithms for neural network training problems (Piotrowski and Napiorkowski, 2011);
(3) DDE, considered as the best distributed DE-based method (Weber et al., 2009), is a concept to divide population into five sub-populations, three with fixed population size organized in a ring topology and two with decreasing population size aiming at exploitation capabilities;
(4) GDE (Piotrowski and Napiorkowski, 2010), one of distributed DE-based algorithms that applies three different mutation strategies and divides population into four sub-populations, including single privileged one;
(5) Comprehensive Learning PSO (CLPSO, Liang et al., 2006), a PSO-based method that modifies velocity adjustment mechanism by using all particles' best information to maintain diversity among the swarm members;
(6) Efficient Population Utilization Strategy PSO (EPUS-PSO, Hsieh et al., 2009), another approach from PSO family, which introduces variable population size and sharing principles mechanism to prevent failing into local optimum;
(7) Covariance Matrix Adaptation ES (CMA-ES, Hansen and Ostermeier, 1996; Hansen et al., 2009), a classical ES method, used in the present paper with different population sizes: default (CMA-ES-op) $K = 4 + \lfloor 3 \cdot \ln M \rfloor$ as suggested in Hansen et al. (2009), $K = 10M$ (CMA-ES-10), $K = 50M$ (CMA-ES-50) and $K = 100M$ (CMA-ES-100);
(8) AMALGAM (Vrugt et al., 2009), a multi-algorithm composed of a set of EC methods, including CMA-ES, that uses a concept of self-adaptive control parameter values; AMALGAM was shown to outperform a number of algorithms, including original DE and a version of CMA-ES with variable population size (IPOP-CMA, Auger and Hansen, 2005).

The population size of all DE-based methods was set to $K = 10M$, with one exception, population size of DDE was increased to 200 for 10-dimensioanl problems, to allow at least 3 stages of population size reductions (see Weber et al., 2009). Population sizes of non DE-based algorithms were set as suggested in source papers, apart from CMA-ES for which four different population sizes were tested, including the one suggested in Hansen et al. (2009). The optimization problems used in the present paper are considered to be solved successfully if the algorithm reaches the global optimum with $\varepsilon = 10^{-5}$ precision. Other control parameters of DE-SG used in the present paper are as follows: population size $K = 10M$, $F = 0.5$, $Cr = 0.5$, $PNI = 500 MigProb = 2.5/PNI$, $NFI = K/8$. The control parameters of other algorithms are set default as defined by the authors of particular approach with the following modifications: in EPUS-PSO the minimum population size is set to 2; in DDE algorithm $F = 0.5$, $CR = 0.5$ and the minimum number of individuals in one group is set to 10. For CLPSO and EPUS-PSO no suggestion about initial setting of $\boldsymbol{v}_i$ was presented (Liang et al., 2006; Hsieh et al., 2009). We follow Das et al. (2008) and generate initial $\boldsymbol{v}_i$ randomly with maximum allowed v_i^j equal to the distance between upper and lower jth bound. If no suggestion on boundary handling was set by authors of particular method, rebounding approach is used as for DE-SG.

The rather soft question which problems are more difficult than others depend on many factors, for example the geometrical complexity of problems' landscape. However, one may assume a definition (Clerc, 2006) that problems that are rarely or never solved by any method are more difficult than problems that are solved by most of them. Successful optimization of the most difficult problems such defined, some of which are rarely used for comparison of EC algorithms, like rEG, rSH, rWH or rRN (Whitley et al., 2004) is time demanding. Hence the maximum number of function calls is set to 3-, 10- and 20-millions for 10-, 30- and 50-dimensional problems, respectively, with exception of CEC2005 functions. As expected global optima of simpler problems, like rAC, rRO, rGR, rLM1 are usually found much quicker. For 10- and 30-dimensional CEC2005 problems maximum number of function calls is set to 300 and 600-thousand, values much closer to CEC2005 limits (Suganthan et al., 2005). Note that the most difficult of CEC2005 problems were considered here (composition functions CEC18 to CEC23), which according to our knowledge were never solved, hence a slight extension of CEC2005 limits seems reasonable to intensify differences between the performance of different algorithms.

For the 13 non-CEC2005 problems, 100 runs are performed. Rotation angles are generated randomly for each run. For CEC2005 problems, the number of runs is limited to 20 due to much longer computation time. The best and the average fitness values for all runs are reported in Tables 1–3. The statistical significance of the difference between the results obtained by two best algorithms (+significant, −insignificant, NA no difference) for each problem is tested using two-tailed unpaired t-test with 0.05 level of significance. Additionally the resulting *p*-values are given.

4.1. Ten-dimensional problems

According to the results presented in Table 1, almost all considered algorithms easily find global optima of a number of problems, namely rGR, rAC, rRO, rLM1, rLM2, rNU and rRS. However, some methods, namely EPUS-PSO, CMA-ES-op and AMALGAM are 10-to 100-times quicker than other algorithms for the seven problems, mentioned above. Apart from rRO these problems are multi-modal. The plotted fitness landscape of each of them within Ω reveals simple geometrical patterns, hence they may be described as simple ones. Most of them, especially rGR, rRO, rAC and rRS are frequently used for comparing EC methods.

SA is a problem of very simple geometry. Its fitness landscape plot shows exchanging spheres of "good" and "poor" solutions centered on the optimum. The "good" spheres, which are closer to the global optimum have lower fitness, but their volume and probability of finding them, decreases. EPSU-PSO is the only method that find global optimum of 10-dimensional SA problem, the rest of algorithms obtain poorer performance.

The rDP problem is also geometrically "tricky". Its local optimum in $x^j = 0$ with fitness value equal to 2/3 is easy to find, however the global optimum with fitness equal 0 is located at $x^{*j} = 2^{-(2^j-2)/2^j}$. The global optimum is determined at least once by DE-SG, DDE, DEGL, GDE and AMALGAM. Other algorithms stick in the above-mentioned local optimum. On average DE-SG is the clearly most successful with 50% success rate.

The fitness landscape of another four problems, namely rWH, rRN, rEG and rSW show more complicated pattern. Global optima are very difficult to find. The rWH problem is solved at least once by AMALGAM, DE-SG, DDE, GDE and CMA-ES with higher than default population size. AMALGAM enjoys 100% success rate, DE-SG over 75%, but other methods perform significantly poorer. For problems rRN, rEG and rSW most DE-based algorithms enjoy similar performance, much better than the other methods. It is worth to mention that for rSE the GDE reaches 100% success rate, while DE-SG is marginally poorer.

According to our knowledge, the global optima of CEC2005 problems considered in the present paper were never found by EC methods. Three of them, namely CEC18, CEC19 and CEC20 were created by a composition of 5 functions and hence show many similarities with each other. The global optimum of the problem CEC19 has a narrower basin of attraction than CEC18, while CEC20 has global optimum on the bounds. The average best results for the three mentioned CEC2005 problems are found by AMALGAM, followed by DE-SG, all other algorithms perform considerably poorer.

The remaining problems CEC21, CEC22 and CEC23 are created based on composition of another 5 functions. For these three problems the average results obtained by DDE and GDE are superior to other algorithms, followed by DE-SG and AMALGAM. As the average results the of two best methods are similar, the superiority of one of them over another is frequently not statistically

Table 1
Results obtained for 10-dimensional functions in 100 runs for each algorithm. Maximum 3mln iterations (CEC – 300 thousands, 20 runs). The values denote best (upper) and average (lower) solution found. Italic and bold values denote the global optima and the best average solutions, respectively.

Fct	$f(x_i^*)$	DE-SG	DEGL	SaDE	DDE	GDE	CLPSO	EPUS-PSO	CMA-ES-o	CMA-ES-10	CMA-ES-50	CMA-ES-100	AMAL-GAM	Significance/*p*-value
rGR	*0*	0/0.0006	0/0.03	0/0	0/0.01	0/0	0.02/0.32	0/0.06	0/0.009	0/0	0/0	0/0	0/0	NA
rAC	*0*	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0.011	0/0	0/0	0/0	0/0	NA
rRS	*0*	0/0.04	0/4.41	0/0.98	0/1.11	0/0	1.44/8.13	0/1.54	3.98/13.26	0/1.69	0/0.5	0/0	0/0	NA
rRO	*0*	0/0	0/0.36	0/0.16	0/312.3	0/0	0/272.8	0/55.5	0/0.85	0/0.12	0/0.16	0/0.2	0/0	NA
rLM1	*0*	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0.015	0/0	0/0	0/0	0/0	NA
rLM2	*0*	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0.001	0/0	0/0	0/0	0/0	NA
rNU	*−210*	−210/−210	−210/−210	−210/−210	−210/−210	−210/−210	−210/−210	−210/−210	−210/−210	−210/−210	−210/−210	−210/−210	−210/−210	NA
SA	*0*	0.1/0.1	0.1/0.1	0.1/0.1	0.1/0.1	0.1/0.1	0.1/0.1	0/**0.06**	0.2/0.5	0.1/0.108	0.1/0.1	0.1/0.1	0.1/0.1	+3.3E-11
rWH	*0*	0/1.6	2.49/25.83	0.99/6.50	0/9.99	0/2.84	0.02/23.01	19.04/40.80	2.84/37.01	0/9.19	0/2.41	0./1.09	0/**0**	+1.3E-5
rRN	*−512.7*	−484.4/ −419.2	−490.6/ **−445.6**	−471.12/ −432.6	−482.6/ −439.3	−468.2/ −430.5	−294.8/ −229.3	−473.4/ −381.3	−347.9/ −255.5	−380.6/ −291.6	−396.0/ −245.5	−464.3/ −353.8	−464.3/ −401.3	+0.0318
rEG	–	−8291/ −7365	−7954/ −6743	−8291/ **−7595**	−8279/ −7250	−8291/ −7552	−5481/ −3694	−7261/ −5544	−5432/ −3665	−5543/ −4160	−5801/ −4647	−5664/ −4764	−6653/ −5742	−0.458
rDP	*0*	0/**0.3**	0/0.66	0.66/0.66	0/0.61	0/0.65	0.66/0.66	0.66/0.66	0.66/0.66	0.66/0.66	0.66/0.66	0.66/0.66	0/0.66	+4.5E-5
rSW	*−418.9*	−418.9/ −417.7	−418.9/ −124.2	−418.9/ −407.25	−418.9/ −311.3	−418.9/ **−418.9**	−418.9/ 908.2	−300.6/ 432.6	646/1392	35.1/717.5	−418.9/49.2	−418.9/ −285.6	−418.9/ −306.4	−0.319
CEC18	*10*	310/310	310/676.9	310/635	310/427.6	310/360	569.5/666.5	310/816.1	510/933.6	310/765.1	310/785	810/810	310/310	NA
CEC19	*10*	310/313.8	310/666.2	310/560	310/418.3	310/360	310/710	310.1/832.1	510/938.7	310/743	810/814	810/810	310/310	−0.329
CEC20	*10*	310/314.9	310/743.8	310/535	310/380.7	310/360	310/650	310.1/758.9	810/951.7	810/870.2	310/785	810/810	310/310	−0.329
CEC21	*360*	660/740	660/1090	660/935	660/735	660/**660**	660/882	660/1101	860/1380	1160/1182	860/1145	660/1070	860/860	+0.046
CEC22	*360*	660/1040	660/1018	660/1012	660/732	660/730	660/1060	778/1163	1103/1154	1072/1089	1067/1082	1065/1077	660/1058	−0.978
CEC23	*360*	785.2/920.2	919.5/1173	660/960	660/699	660/760	660/1044	715.6/1170	919/1368	1330.5/ 1360	1081/1285	1081/1305	919.5/919.5	−0.263

Table 2
Results obtained for 30-dimensional functions in 100 runs for each algorithm. Maximum 10mln iterations (CEC – 600 thousands, 20 runs). The values denote best (upper) and average (lower) solution found. Italic and bold values denote the global optima and the best average solutions, respectively.

Fct	$f(x_i^*)$	DE-SG	DEGL	SaDE	DDE	GDE	CLPSO	EPUS-PSO	CMA-ES-o	CMA-ES-10	CMA-ES-50	CMA-ES-100	AMAL-GAM	Significance/*p*-value
rGR	*0*	0/0	0/0.003	0/0	0/0	0/0	0/0.004	0/0	0/0.001	0/0	0/0	0/0	0/0	NA
rAC	*0*	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	NA
rRS	*0*	0.99/10.2	12.9/57.8	1.99/10.5	14.58/35.61	2.68/11.0	143/170	0/37.2	21.9/51.3	0/1.68	0/0	0/0	0/0	NA
rRO	*0*	0/0	0/0.558	0/0.11	0/1.89	0/0.04	0/68	7.74/9.14	0/0.95	0/0	0/0	0/0	0/0	NA
rLM1	*0*	0/0	0/0.001	0/0	0/0	0/0	0/0	0/0.005	0/0.007	0/0	0/0	0/0	0/0	NA
rLM2	*0*	0/0	0/0.0001	0/0	0/0	0/0	0/0	0/0.016	0/0.001	0/0	0/0	0/0	0/0	NA
rNU	*−4930*	−4930/ −4930	−4930/ −4930	−4930/ −4930	−4930/ −4930	−4930/ −4930	−4930/ −4928	−4930/ −4930	−4930/ −4930	−4930/ −4930	−4930/ −4930	−4930/ −4930	−4930/ −4930	NA
SA	*0*	0.1/0.21	0.1/0.16	0.1/0.1	0.1/0.12	0.1/0.1	0.1/0.2	0.1/0.1	0.61/1.054	0.1/0./188	0.1/0.106	0.1/0.1007	0.1/0.1	NA
rWH	*0*	0/**1.24**	238.9/372.3	122.3/203.4	236.9/339.9	76.5/143.2	353.1/ 376.1	363/402	227.3/428.3	362.6/363.4	368.7/384.4	373.6/386.3	240.5/354.8	+2.3E-36
rRN	*−512.7*	−445.2/ −**389.6**	−414.3/ −265.5	−411.7/ −378.8	−371.5/ −323.3	−399.1/ −369.2	−156/ −104.6	−431.9/ −313.9	−293.8/ −248.6	−324.3/ −274.5	−378.0/ −304.3	−420.6/ −326.8	−395.0/ −364.0	+0.0052
rEG	–	−24722/ −**20930**	−22476/ −19385	−21974/ −19617	−23322/ −16755	−21005/ −19511	−7246/ −4392	−18759/ −13851	−14470/ −11225	−13567/ −11978	−15294/ −13421	−15153/ −13764	−19330/ −17299	+3.0E−9
rDP	*0*	0/0.66	0.66/0.66	0.66/0.66	0.66/0.66	0.66/0.66	0.66/0.66	0.66/0.66	0.66/0.66	0.66/0.66	0.66/0.66	0.66/0.66	0.66/0.66	NA
rSW	*−418.9*	−418.9/ −210.6	−63.5/1445	−418.9/ 113.6	−418.9/ −136.3	−418.9/ −**227.5**	6135/9494	1752/4070	4161/5416	2206/4240	390.4/ 1508.5	−63.6/570.4	370.9/1275	− 0.591
CEC18	*10*	810/**873.7**	913.4/915.1	913.5/913.9	913.4/914.3	810/910.1	930/981	935/994	810/934.2	810/909.3	913/913.5	913.1/913.4	913.4/913.6	+0.017
CEC19	*10*	810/**886**	913.3/914.4	913.4/914.6	913.6/914.1	913.9/915.8	935/1013	926/996	924/933.1	913.2/914.6	912.9/913.4	913/913.3	913.3/913.5	+0.027
CEC20	*10*	810/**874.7**	913.3/914.6	912.9/914.9	913.6/914.2	914.3/915.8	935/963	949/1001	924.9/935.5	913.2/914.9	913/913.4	913/913.4	913.4/913.5	+0.0049
CEC21	*360*	860/860	1263/1264	1263/1264	1263/1264	1263/1265	1295/1352	1286/1344	860/937	860/860	860/860	860/860	860/860	NA
CEC22	*360*	1260/1289	1264/1265	1263/1264	1263/1264	1263/1265	1287/1352	1299/1351	1196/1247	1116/**1170**	1173/1207	1200/1208	1146/1192	+ 0.012
CEC23	*360*	894.16/ 894.16	1263/1264	1160/1259	1263/1264	1160/1260	1297/1351	1288/1351	894.16/1011	894.16/ 894.16	894.16/ 894.16	894.16/ 894.16	894.16/ 894.16	NA

Table 3
Results obtained for 50-dimensional functions in 100 runs for each algorithm. Maximum 20mln iteraitions (CEC – 1mln, 20 runs). The values denote best (upper) and average (lower) solution found. Italic and bold values denote the global optima and the best average solutions, respectively.

Fct	$f(x_i^*)$	DE-SG	DEGL	SaDE	DDE	GDE	CLPSO	EPUS-PSO	CMA-ES-o	CMA-ES-10	CMA-ES-50	CMA-ES-100	AMAL-GAM	Significance/p-value
rGR	*0*	0/0	0/0.0008	0/0.0004	0/0	0/0	0/0	0/0.0006	0/0	0/0	0/0	0/0	0/0	NA
rAC	*0*	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/1.69	0/0	0/0	0/0	NA
rRS	*0*	0.99/17.5	23.9/120.7	14.9/26.5	22.8/54.9	15.9/30.1	334.8/383.4	0/46.7	58.7/104.8	0/0	0/0	0/0	0/0	NA
rRO	*0*	0/0.03	0/0.598	0/0.598	0/0	0/0	0/657	22.5/25	0/1.33	0/0	0/0	0/0	0/0	NA
rLM1	*0*	0/0	0/0	0/0	0/0	0/0	0/0.61	0/0.002	0/0.006	0/0	0/0	0/0	0/0	NA
rLM2	*0*	0/0	0/0.0002	0/0	0/0	0/0	0/0	0/0.11	0/0.002	0/0	0/0	0/0	0/0	NA
rNU	*−22050*	−22050/ −22050	−22050/ −22050	−22050/ −22050	−22050/ −22032	− 22050/ −22050	−22049.3/ −21930	−22050/ −22050	−22050/ −22050	−22050/ −22050	−22050/ −22050	−22050/ −22050	−22050/ −22050	NA
SA	*0*	0.2/0.33	0.2/0.21	0.2/0.2	0.1/0.19	0.1/0.17	0.2/0.21	0.1/**0.1**	0.9/1.57	0.2/0.2002	0.2/0.2	0.2/0.2	0.2/0.2	+3.0E−59
rWH	*0*	0/**24**	690/1089	263/676	822/985	0/454	1858/1946	1100/1128	783/1272	1007/1034	1037/1093	1075/1100	797/1001	+2.E−107
rRN	*−512.7*	−419.1/ −**391.1**	−397.3/ −200.9	−375.1/ −358.6	−337.6/ −306.9	−367.8/ −347.9	−109.5/−74.1	−372.8/ −300.5	−290.9/ −249.2	−308.4/ −263.9	−383.7/ −310.1	−410.9/ −331.3	−369.7/ −344.8	+1.8E−24
rEG	–	−39323/ −**35095**	−36453/ −31242	−33564/ −31182	−29578/ −25425	−32807/ −30779	−9194/ −5468	−29961/ −23159	−21853/ −18847	−22244/ −19874	−23960/ −22302	−25394/ −23093	−30330/ −28064	+7.0E−17
rDP	*0*	0.66/0.66	0.66/0.66	0.66/0.66	0.66/0.66	0.66/0.66	0.66/0.66	0.66/0.66	0.66/0.66	0.66/0.66	0.66/0.66	0.66/0.66	0.66/0.66	NA
rSW	*−418.9*	−418.9/ −**23.**	647/3547	272/1338	−83./1212	−113./608	14086/15895	3556/7188	7398/9367	7279/8689	1673/4099	1002/2221	2053/2854	+1.2E−21

significant. However, the superiority of two or three best algorithms over the rest of methods is significant.

Overall, distributed-DE or self-adaptive-DE methods show advantage over other algorithms for composition functions CEC21, CEC22 and CEC23, whereas multi-algorithm AMALGAM and DE-SG perform best for CEC18, CEC 19 and CEC20 problems. No algorithm clearly outperforms others for non-CEC2005 problems, AMALGAM enjoys the highest success rate and is the least time-consuming when used for simpler functions, but its results are not encouraging for the majority of more difficult ones.

4.2. Thirty-dimensional problems

Among seven problems considered as "simple" in the previous section (rGR, rAC, rRO, rLM1, rLM2, rNU and rRS), global optima of the first six are also easily found for their 30-dimensional versions (see Table 2) by almost all algorithms. However, only AMALGAM, EPUS-PSO and CMA-ES with higher population size find global minimum of rRS at least once. AMALGAM, CMA-ES-50 and CMA-ES-100 enjoy 100% success rate.

The global optimum of SA problem is never found successfully. The best average results are obtained by means of SaDE, EPUS-PSO and AMALGAM.

For three out of four problems with complicated geometrical pattern (rWH, rRN, rEG) DE-SG clearly outperforms all other algorithms. The difference between its average performance and the average performance of the second best algorithm is always statistically significant. In case of rSW, DE-SG is slightly inferior to GDE on the average, but clearly beats all other methods.

DE-SG is also the only algorithm able to find global optimum of rDP. However, this was achieved only once per 100 runs. Hence, to check the true success rate the DE-SG was applied 1000 times to solve 30-dimensional rDP. The global optimum was found only 3 times, too little to draw a conclusion of DE-SG superiority over other methods.

The comparison of the results for CEC18, CEC19 and CEC20 problems reveal the best performance of DE-SG algorithm. Its advantage over other algorithms is statistically significant.

For CEC21 and CEC23, DE-SG shares the best results with AMALGAM and CMA-ES with higher than default population size. It is only for CEC22 that the two algorithms, namely CMA-ES-10 and AMALGAM, outperform DE-SG.

It is worth to discuss one specific issue here. As in this paper maximum number of function calls is high, when solving most problems better results are obtained when the population size of CMA-ES is higher. This could be expected as the CMA-ES with higher population size better samples the search space around its current average position and less randomness occurs during self-adaptation of control parameters. However, for two problems, namely rWH and CEC18, performance of CMA-ES-o is surprisingly better than for higher population sizes. This result may suggest that the mechanism of self-adaptation of CMA-ES control parameters is unsuccessful for some problems. Probably this is connected with non-elitism of CMA-ES. The impact of the best solutions found during particular iteration on the control parameters self-adaptation is very limited when the population size becomes high. The solutions within basin of attraction of shallow but wide local optimum may be much more frequently sampled than solutions within basin of attraction of deep but narrow one and hence may have more significant impact on control parameter's self-adaptation. As a result, CMA-ES with higher population size moves towards shallow but wide local optimum.

Overall, for the majority of more difficult 30-dimensional problems the DE-SG outperforms other algorithms, or in a few cases shows similar performance to the best among them. There are only two exceptions, SA and CEC22 problems.

4.3. Fifty-dimensional problems

The results obtained for thirteen 50-dimensional problems are shown in Table 3. Contrary to lower-dimensional versions described in the previous sections, results obtained for most of seven “simpler” problems (rGR, rAC, rRO, rLM1, rLM2, rNU and rRS) vary for different algorithms. Two algorithms outperform the others – AMALGAM and CMA-ES with non-default population size, both have 100% success rate and are the fastest (see Fig. 2). Among other methods distributed-DE approaches, the DE-SG, GDE and DDE, show better performance than the other methods. With exception of rRS, results obtained by all three distributed DE-based methods are similar to those obtained by AMALGAM and CMA-ES.

The SA problem is again best solved by EPUS-PSO, followed by DDE. The superiority of EPUS-PSO over the second best algorithm is statistically significant.

The global optimum of rDP problem is not found by any algorithm, all sticks in the local optimum.

For four difficult problems (rWH, rRN, rEG and rSW), the DE-SG algorithm significantly outperforms all other methods. Its advantage is also statistically significant.

Important features of EC methods may be found by means of convergence characteristics plot. It is presented for selected 50-dimensional problems in Fig. 2. From four CMA-ES versions considered, only CMA-ES-100 is included in Fig. 2, because of the overall highest success rate. The proposed DE-SG algorithm is noticeably

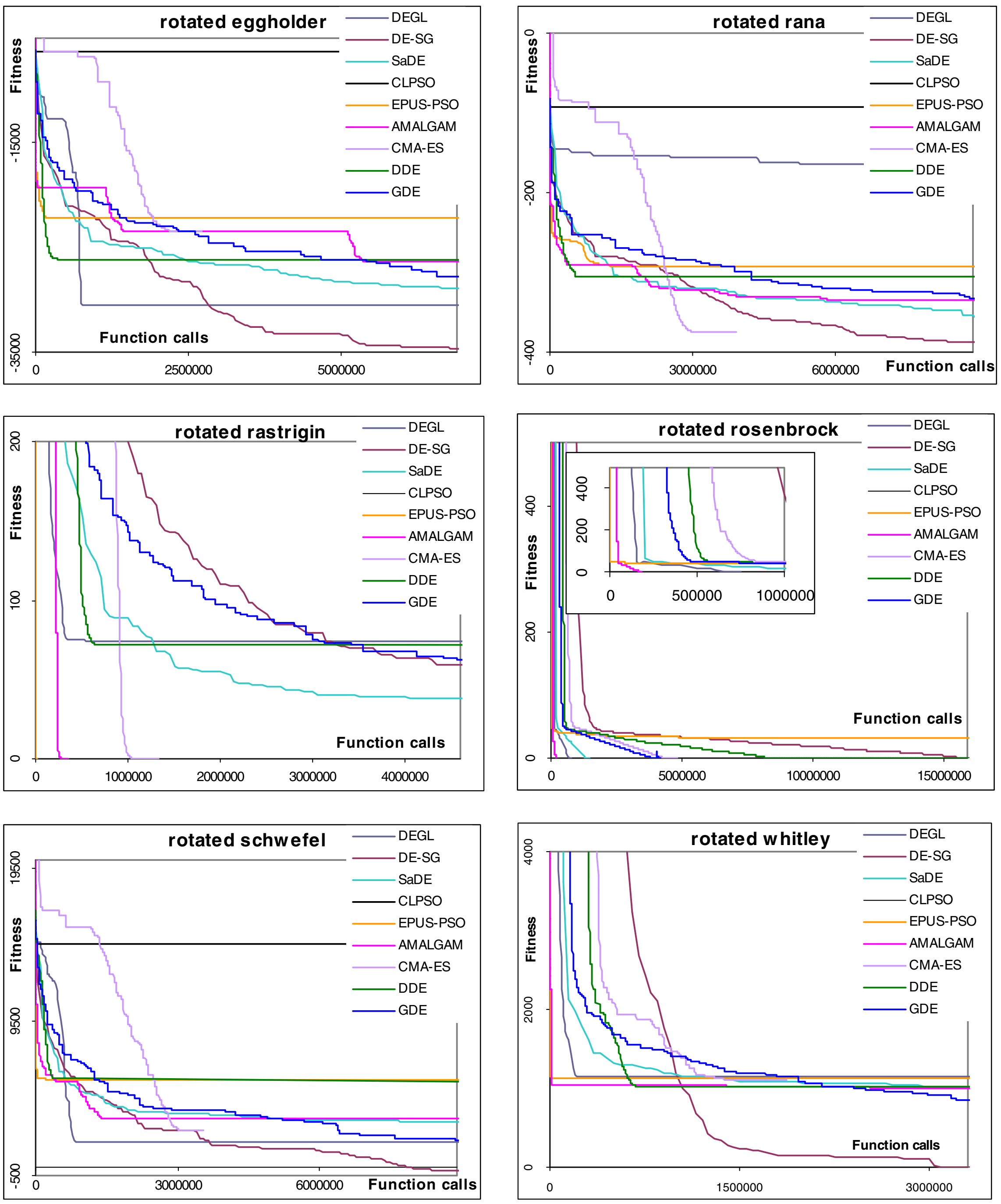

Fig. 2. Example of convergence for 50-dimensional functions.

slower than most algorithms for "simpler" problems like rRS and rRO. However, its convergence speed is similar to competing algorithms for more difficult problems, for example rRN, rSW or rEG, for which it obtains much better results in terms of fitness value. It is worth noting that for example for rWH the DE-SG improves slowly during initial iterations but shows clear advantage once the migration mechanism starts to play an important role, which suggests that DE-SG is better suited for more demanding problems. One may see that some algorithms, for example EPUS-PSO and DEGL, often quickly converge to some (local or global) optima and stop improving. Some other methods, like SaDE, DDE, GDE and DE-SG, are usually able to improve their performance for a significant period of time due to self-adaptation of control parameters, distribution of population into sub-populations and information exchange between them. However, as a consequence they are usually slower.

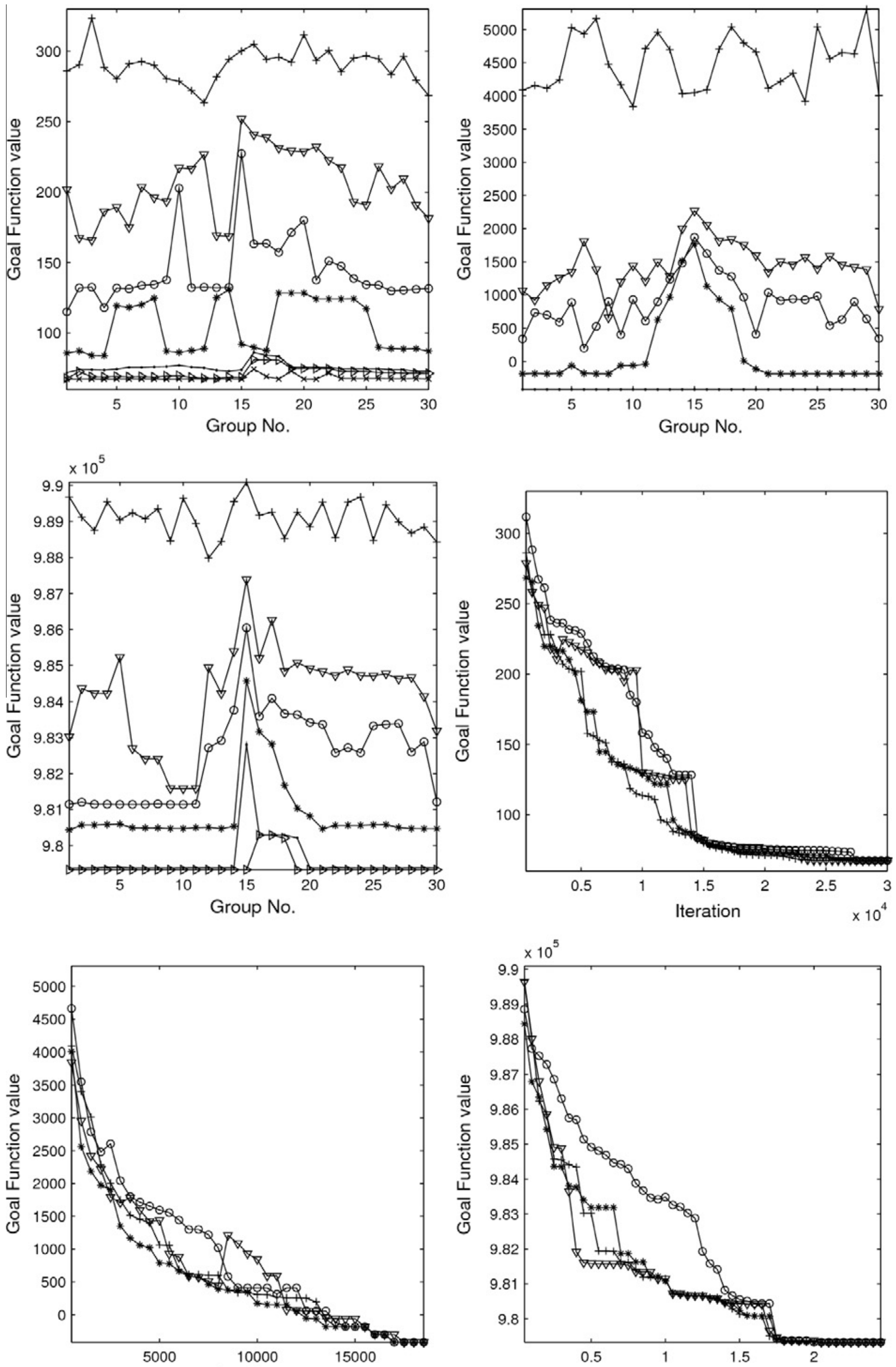

Fig. 3. The best individuals in 30 groups at 500 (+), 1000 (∇), 1500 (O), 2000 (*), 2500 ($\cdot$) 3000 (>) and 3500 ($\times$) iteration for Rana's (top left), Schwefel's (top right), Eggholder's (middle left) goal function and improvement of best function value for group No. 1 (+), 10 (∇), 20 (O) and 30 (*) with iterations for Rana's (middle right), Schwefel's (bottom left) and Eggholder's (bottom right) goal function.

4.4. Discussion on migration of individuals among the groups

The process of migration of individuals used by DE-SG algorithm is illustrated in Fig. 3. This figure shows the change of fitness values of the best individuals within each group with time (on the left) and the change of best function values found by the whole population with time (on right). The migration mechanism is illustrated on 30-dimensional rRN, rEG and rSW problems. Although the rate of exchange of individuals among the given groups is small in current application of DE-SG, after about 1000 iterations the best individual within the right half of the population almost always belongs to the last (elitist) group. The location of the best individuals among left half of the population is much more irregular. Due to migration mechanism, the best fitness value within a group may also deteriorate – examples are seen in Fig. 3 (bottom left) for group number 10 around iteration 9000 and for group number 20 around iteration 2000. This can happen when the best individual within a group is exchanged with the individual from a neighboring group before it shares information about successful location with other participants in the same group. This phenomenon is illustrated in Fig. 3 for rSW problem (top right), where the deterioration of group 8 between iteration 1000 and 1500 and coincidental significant improvement of groups 7 and 6 is observed.

Note that although the consequences of migration of individuals are complicated, technically the mechanism is simple and easy to implement.

5. Conclusions

In the present paper, the DE with Separated Groups algorithm was proposed for multi-dimensional optimization of continuous real functions. The performance of compared algorithms significantly depends on the problem to be solved. This may be expected according to both “No Free Lunch Theorem” (Wolpert and Macready, 1997) and the practical experiments performed by Langdon and Poli (2007).

Due to almost 100% success rate and quick convergence, CMA-ES and AMALGAM should be considered as the best methods for the “simpler” problems, which are usually solved successfully by most algorithms (rGR, rAC, rRO, rLM1, rLM2 and rNU). Among the problems that pose more difficulty for various EC methods, AMALGAM and CMA-ES show the best results for high-dimensional rRS and CEC22 only.

EPUS-PSO algorithm turns out to be the best method for geometrically specific SA problem.

DE-SG outperforms the other methods for vast majority of problems which are rarely or never solved, with local optima scattered in the search space and frequently without clear geometrical pattern (rWH, rRN, rEG, rDP, rSW, CEC18, CEC19, CEC20). For many among these problems, other DE-based algoritms also perform better than non-DE-based methods.

For problems CEC21 and CEC23 single leader cannot be specified. Four algorithms, namely DE-SG, DDE, CMA-ES and AMALGAM, provided better results than the other methods.

Overall, the numerical results revealed that AMALGAM and Covariance Matrix Adaptation-ES should be regarded as the most efficient for solving relatively simple problems, while the DE with Separated Groups provides the best results for the majority of more difficult problems.

Acknowledgment

This work has been supported by the Foundation for Polish Science.

Appendix A. Test problems

M – Rotation matrix;
f(***x***) – Objective function;
x*** – Global optimum;

1. Griewank function (rGR), Price et al. (2005)

$$f(\mathbf{x}) = \frac{1}{4000}\sum_{j=1}^{M} z_j^2 - \prod_{j=1}^{M}\cos\left(\frac{z_j}{\sqrt{j}}\right) + 1$$

$$\mathbf{z} = \mathbf{M}\cdot\mathbf{x}$$

$$f(\mathbf{x}^*) = 0 z_j^* = x_j^* = 0$$

in this paper $-1000 \leqslant z_j \leqslant 1000$

2. Ackley function (rAC), Price et al. (2005)

$$f(\mathbf{x}) = -20\exp\left(-0.2\sqrt{\frac{1}{M}\sum_{j=1}^{M} z_j^2}\right) - \exp\left(\frac{1}{M}\sum_{j=1}^{M}\cos(2\pi z_j)\right) + 20 + e$$

$$\mathbf{z} = \mathbf{M}\cdot\mathbf{x}$$

$$f(\mathbf{x}^*) = 0 \quad z_j^* = x_j^* = 0 \quad -32 \leqslant z_j \leqslant 32$$

3. Rastrigin function (rRS), Price et al. (2005)

$$f(\mathbf{x}) = \sum_{j=1}^{M}\left(z_j^2 - 10\cos(2\pi z_j) + 10\right)$$

$$\mathbf{z} = \mathbf{M}\cdot\mathbf{x}$$

$$f(\mathbf{x}^*) = 0 \quad z_j^* = x_j^* = 0$$

in this paper $-1000 \leqslant z_j \leqslant 1000$

4. Generalized Rosenbrock function (rRO), Price et al. (2005)

$$f(\mathbf{x}) = \sum_{j=1}^{M-1}\left(100\left(z_{j+1} - z_j^2\right)^2 + \left(z_j - 1\right)^2\right)$$

$$\mathbf{z} = \mathbf{M}\cdot\mathbf{x}$$

$$f(\mathbf{x}^*) = 0 \quad z_j^* = 1$$

in this paper $-1000 \leqslant z_j \leqslant 1000$

5. Levy and Montalvo 1 function (rLM1), Ali et al. (2005)

$$f(\mathbf{x}) = \left(\frac{\pi}{M}\right)\left(10\sin^2(\pi y_1) + \sum_{j=1}^{M-1}(y_j - 1)^2(1 + 10\sin^2(\pi y_{j+1})) + (y_M - 1)^2\right)$$

$$y_j = 1 + 0.25(z_j + 1)$$

$$\mathbf{z} = \mathbf{M}\cdot\mathbf{x}$$

$$f(x^*) = 0 \quad z_j^* = -1 \quad -10 \leqslant z_j \leqslant 10$$

6. Levy and Montalvo 2 function (rLM2), Ali et al. (2005)

$$f(\mathbf{x}) = 0.1\left(\sin^2(3\pi z_1) + \sum_{j=1}^{M-1}(z_j - 1)^2(1 + \sin^2(3\pi z_{j+1})) + (z_M - 1)^2(1 + \sin^2(2\pi z_M))\right)$$

$$\mathbf{z} = \mathbf{M}\cdot\mathbf{x}$$

$$f(\mathbf{x}^*) = 0 \quad z_j^* = 1 \quad -5 \leqslant z_j \leqslant 5$$

7. Neumaier 3 function (rNU), Ali et al. (2005)

$$f(\mathbf{x}) = \sum_{j=1}^{M}(z_j - 1)^2 - \sum_{j=2}^{M} z_j z_{j-1}$$

$$\mathbf{z} = \mathbf{M} \cdot \mathbf{x}$$

$$f(x^*) = -\frac{M(M+4)(M-1)}{6} \quad z_j^* = j(M+1-j) - M^2 \leqslant z_j \leqslant M^2$$

8. Salomon function (SA), Ali et al. (2005)

$$f(x) = 1 - \cos(2\pi\|\mathbf{x}\|) + 0.1\|\mathbf{x}\|$$

$$\|\mathbf{x}\| = \sqrt{\sum_{j=1}^{M} x_j^2}$$

$$f(x^*) = 0 \; x_j^* = 0 \quad -100 \leqslant x_j \leqslant 100$$

9. Whitley function (rWH), Whitley et al. (1996), Price et al. (2005)

$$f(\mathbf{x}) = \sum_{j=1}^{M}\sum_{l=1}^{M}\left(\frac{\left(100\left(z_j^- z_l^2\right)^2 + (1 - z_l)^2\right)^2}{4000} - \cos\left(100\left(z_j^- z_l^2\right)^2 + (1 - z_l)^2\right) + 1\right)$$

$$\mathbf{z} = \mathbf{M} \cdot \mathbf{x}$$

$$f(\mathbf{x}^*) = 0 \quad z_j^* = 1 \quad -100 \leqslant z_j \leqslant 100$$

10. Normalized Rana function (rRN), Whitley et al. (1996), Price et al. (2005), www.it.lut.fi/ip/evo/functions

$$f(\mathbf{x}) = \sum_{j=1}^{M} z_j \sin\left(\sqrt{|z_l + 1 - z_j|}\right)\cos\left(\sqrt{|z_l + 1 + z_j|}\right) + (z_l + 1)\cos\left(\sqrt{|z_l + 1 - z_j|}\right)\sin\left(\sqrt{|z_l + 1 + z_j|}\right)$$

$$l = (j+1) Mod M$$

$$\mathbf{z} = \mathbf{M} \cdot \mathbf{x}$$

$$f(x^*) = -512.7531624 \quad z_j^* = -514.04168$$

when $-520 \leqslant z_j \leqslant 520$

11. Eggholder function (rEG), Whitley et al. (1996), Adorio (2005)

$$f(\mathbf{x}) = \sum_{j=1}^{M-1} -(z_{j+1} + 47)\sin\sqrt{|z_{j+1} + z_j 0.5 + 47|} + \sin\sqrt{|z_j - (z_{j+1} + 47)|}(-z_j)$$

$$\mathbf{z} = \mathbf{M} \cdot \mathbf{x}$$

$$-512 \leqslant z_j \leqslant 512$$

$f(\boldsymbol{x}^*)$ and $\boldsymbol{x}^*$ depend on problem dimensionality

12. Dixon–Price function (rDP), Heddar and Fukushima (2006)

$$f(\mathbf{x}) = (z_1 - 1)^2 + \sum_{j=2}^{M} j\left(2z_j^2 - z_{j-1}\right)^2$$

$$\mathbf{z} = \mathbf{M} \cdot \mathbf{x}$$

$$f(\mathbf{x}^*) = 0 \quad z_j^* = 2^{-\left(\frac{2^j - 2}{2^j}\right)} \quad -10 \leqslant z_j \leqslant 10$$

13. Schwefel function (rSW), Price et al. (2005)

$$f(\mathbf{x}) = -\frac{1}{M}\sum_{j=1}^{M} z_j \sin\left(\sqrt{|z_j|}\right)$$

$$\mathbf{z} = \mathbf{M} \cdot \mathbf{x}$$

$$f(\mathbf{x}^*) = -418.983 \quad z_j^* = 420.9687 \quad \text{when } -500 \leqslant z_j \leqslant 500$$

The detailed description of CEC 2005 problems 18–23 may be found in Suganthan et al. (2005).

References

Adorio, E.P., 2005. MVF – Multivariate Test Functions Library in C for unconstrained global optimization. <www.geocities.com/anyongqing/myLibrary/globalOptimization/unconstrainedTestProblemsMain.html#NoteAdorio2005>.

Al-Anzi, F.S., Allahverdi, A., 2007. A self-adaptive differential evolution heuristic for two-stage assembly scheduling problem to minimize maximum lateness with setup times. European Journal of Operational Research 182 (1), 80–94.

Ali, M.M., Khompatraporn, C., Zabinsky, Z.B., 2005. A numerical evaluation of several stochastic algorithms on selected continuous global optimization test problems. Journal of Global Optimization 31, 635–672.

Apolloni, J., Leguizamon, G., Garcia-Nieto, J., Alba, E., 2008. Island based distributed differential evolution: an experimental study on hybrid testbeds. In: Proceedings of IEEE International Conference on Hybrid Intelligent Systems, pp. 696–701.

Auger, A., Hansen, N., 2005. A restart CMA evolution strategy with increasing population size. In: Proceedings of the IEEE Congress on Evolutionary Computation, CEC2005.

Beasley, D., Bull, D.B., Martin, R.R., 1993. A sequential niche technique for multimodal function optimization. Technical Report No. 93001.

Beynon, M.J., Andrews, R., Boyne, G.A., 2010. Evidence-based modeling of strategic fit: an introduction to RCaRBS. European Journal of Operational Research 207 (2), 886–896.

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 Transactions on Evolutionary Computation 10 (6), 646–657.

Brest, J., Boskovic, B., Greiner, S., Zumer, V., Maucec, M.S., 2007. Performance comparison of self-adaptive and adaptive differential evolution algorithms. Soft Computing 11, 617–629.

Chakraborty, U.K., Das, S., Konar, A., 2006. Differential evolution with local neighborhood. In: Proceedings of the Congress on Evolutionary Computation. Vancouver, BC, Canada, pp. 042–2049.

Clerc, M., 2006. Particle Swarm Optimization. ISTE Ltd, London, UK.

Cruz, F.R.B., van Woensel, T., MacGregor Smith, J., Lickens, K., 2010. On the system optimum of traffic assignment in M/G/c/c state-dependent queueing networks. European Journal of Operational Research 201, 183–193.

Das, S., Suganthan, P.N., 2011. Differential Evolution: a survey and state-of-the-art. IEEE Transactions on Evolutionary Computation 15 (1), 27–54.

Das, S., Abraham, A., Konar, A., 2008. Particle Swarm Optimization and Differential Evolution Algorithms: technical analysis, applications and hybridization perspectives. Studies in Computational Intelligence, vol. 116/2008. Springer, Berlin/Heidelberg.

Das, S., Abraham, A., Chakraboty, U.K., Konar, A., 2009. Differential evolution using a neighborhood-based mutation operator. IEEE Transactions on Evolutionary Computation 13 (3), 526–553.

Eberhart, R.C., Kennedy, J., 1995. A new optimizer using particle swarm theory. In: Proceedings of the Sixth International Symposium on Micromachine and Human Science, Nagoya, Japan, pp. 39–43.

Falco, I., Maisto, D. Scafuri, U., Tarantino, E., Della Cioppa, A., 2007. Distributed differential evolution for the registration of remotely sensed images. In: Proceedings of the IEEE Euromicro International Conference on Parallel, Distributed and Network-Based Processing, pp. 358–362.

Gamperle, R., Muller, S.D., Koumoutsakos, P., 2002. A parameter study for differential evolution. In: Gremla, A., Mastorakis, N.E. (Eds.), Advances in Intelligent Systems, Fuzzy Systems, Evolutionary Computation. WSEAS Press, Interlaken, Switzerland, pp. 293–298.

Gustafson, S., Burke, E.K., 2006. The Speciating Island Model: an alternative parallel evolutionary algorithm. Journal of Parallel and distributed computing 66 (8), 1025–1036.

Hansen, N., Ostermeier, A., 1996. Adapting arbitrary normal mutation distribution in evolution strategies: the covariance matrix approximation. In: Proceedings of IEEE International Conference on Evolutionary Computation, Nagoya, Japan, pp. 312–317.

Hansen, N., Niederberger, A.S.P., Guzzella, L., Koumoutsakos, P., 2009. A method for handling uncertainty in evolutionary optimization with an application to feedback control of combustion. IEEE Transactions on Evolutionary Computation 13 (1), 180–197.

Heddar, A.R., Fukushima, M., 2006. Tabu search directed by direct search methods for nonlinear global optimization. European Journal of Operational Research 170, 329–349.

Holland, J.H., 2000. Building blocks, cohort genetic algorithms and hyperplane-defined functions. Evolutionary Computation 8 (4), 373–391.

Hsieh, H.T., Sun, T.Y., Liu, C.C., Tsai, S.J., 2009. Efficient population utilization strategy for particle swarm optimizer. IEEE Transactions on Systems, Man and Cybarnetics–Part B: Cybernetics 39 (2), 444–456.

Ilonen, J., Kamarainen, J.K., Lampinen, J., 2003. Differential Evolution training algorithm for feed-forward neural networks. Neural Processing Letters 17, 93–105.

Kaelo, P., Ali, M.M., 2006. A numerical study of some modified differential evolution algorithms. European Journal of Operational Research 169 (3), 1176–1184.

Langdon, W.B., Poli, R., 2007. Evolving problems to learn about particle swarm optimizers and other search algorithms. IEEE Transactions on Evolutionary Computation 11 (5), 561–578.

Liang, J.J., Qin, A.K., Suganthan, P.N., Baskar, S., 2006. Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. IEEE Transactions on Evolutionary Computation 10 (3), 281–295.

Liu, J., Lampinen, J., 2005. A fuzzy adaptive differential evolution algorithm. Soft Computing 9, 448–462.

Liu, Y., Yao, X., Higruchi, T., 2000. Evolutionary ensembles with negative correlation learning. IEEE Transactions on Evolutionary Computation 4 (4), 380–387.

Mendes, S.P., Molina, G., Vega-Rodriguez, M.A., Gomez-Pulido, J.A., Saez, Y., Miranda, G., Segura, C., Alba, E., Isasi, P., Leon, C., Sanchez-Perez, J., 2009. Benchmarking a wide spectrum of metaheuristic techniques for the radio network design problem. IEEE Transactions on Evolutionary Computation 13 (5), 2009.

Mezura-Montes, E., Velazques-Reyes, J., Coello, C.A.C., 2006. A comparative study of differential evolution variants for global optimization. In: Genetic and Evolutionary Conference (GECCO), pp. 485–492.

Mishra, S.K., 2006. Global optimization by Differential Evolution and Particle Swarm methods evaluation on some benchmark functions. Social Science Research Network, Working Papers Series. <http://ssrn.com/abstract=933827>.

Neri, F., Tirronen, V., 2010. Recent advances in differential evolution: a survey and experimental analysis. Artificial Intelligence Review 33, 61–106.

Omran, M.G.H., Salman, A., Engelbrecht, A.P., 2005. Self-adaptive differential evolution. Lecture Notes in Artificial Intelligence 3801, 192–199.

Omran, M.G.H., Engelbrecht, A.P., Salman, A., 2009. Bare bones differential evolution. European Journal of Operational Research 196, 128–139.

Piotrowski, A.P., Napiorkowski, J.J., 2010. Grouping differential evolution algorithm for multi-dimensional optimization problems. Control and Cybernetics 39 (2), 527–550.

Piotrowski, A.P., Napiorkowski, J.J., 2011. Optimizing neural networks for river flow forecasting-Evolutionary Computation methods versus the Levenberg–Marquardt approach. Journal of Hydrology. doi:10.1016/j.jhydrol.2011.06.019.

Piotrowski, A.P., Napiorkowski, J.J., Rowinski, P.M., Wallis, G.G., 2011. Evaluation of temporal concentration profiles for ungauged rivers following pollution incidents. Hydrological Sciences Journal 56 (5), 883–894.

Price, K.V., Storn, R.M., Lampinen, J.A., 2005. Differential Evolution. A Practical Approach to Global Optimization. Springer-Verlag, Berlin, Heidelberg, Germany.

Qin, A.K., Suganthan, P.N., 2005. Self-adaptive differential evolution algorithm for numerical optimization. In: Proceedings of the IEEE Congress on Evolutionary Computation, Edinburgh, UK, pp. 1785–1791.

Qin, A.K., Huang, V.L., Suganthan, P.N., 2009. Differential evolution algorithm with strategy adaptation for global numerical optimization. IEEE Transactions on Evolutionary Computation 13 (2), 398–417.

Rokkonen, J., Kukkonen, S., Price, K.V., 2005. Real parameter optimization with differential evolution. In: Proceedings of the IEEE Congress on Evolutionary Computation, Edinburgh, Scotland, UK, pp. 506–513.

Rowiński, P.M., Piotrowski, A., 2008. Estimation of parameters of transient storage model by means of multi-layer perceptron neural networks. Hydrological Sciences Journal 53 (1), 165–178.

Salman, A., Engelbrecht, A.P., Omran, M.G.H., 2007. Computing, artificial intelligence and information management – empirical analysis of self-adaptive differential evolution. European Journal of Operational Research 183 (2), 785–804.

Salomon, R., 1996. Re-evaluating genetic algorithm performance under coordinate rotation on benchmark functions. A survey of some theoretical and practical aspects of genetic algorithms. BioSystems 39, 263–278.

Storn, R., Price, K.V., 1995. Differential Evolution – A simple and efficient adaptive scheme for global optimization over continuous spaces. Technical Report TR-95-012, International Computer Sciences Institute, Berkeley, CA, USA.

Storn, R., Price, K.V., 1997. Differential Evolution – A simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization 11, 341–359.

Suganthan, P.N., Hansen, N., Liang, J.J., Deb, K., Chen, Y.P., Auger, A., Tiwari, S., 2005. Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization. Nanyang Technol. Univ., Singapore, Tech. Rep. KanGAL #2005005, IIT Kanpur, India.

Tanese, R., 1989. Distributed genetic algorithms. In: Proceedings of the 3rd International Conference on Genetic Algorithms, pp. 434–439.

Tasoulis, D.K., Pavlidis, N.G., Plagianakos, V.P., Vrahatis, M.N., 2004. Parallel differential evolution. In: Proceedings of the IEEE Congress on Evolutionary Computation pp. 2023–2029.

Tomassini, M., 2005. Spatially Structured Evolutionary Algorithms: Artificial Evolution in Space and Time. Natural Computing Series. Springer, Heidelberg.

Vrugt, J.A., Robinson, B.A., Hyman, J.M., 2009. Self-adaptive multimethod search for global optimization in real-parameter spaces. IEEE Transactions on Evolutionary Computation 13 (2), 243–259.

Weber, M., Neri, F., Tirronen, V., 2009. Distributed differential evolution with explorative–exploitative population families. Genetic Programming Evolvable Machines 10, 343–371.

Whitley, D., Rana, S., Dzubera, J., Mathias, K.E., 1996. Evaluating evolutionary algorithms. Artificial Intelligence 85, 245–276.

Whitley, D., Lunacek, M., Knight, J., 2004. Ruffled by Ridges: How Evolutionary Algorithms Can Fail. Lecture Notes in Computer Science, vol. 3103/2004. Springer, Berlin/Heidelberg.

Wolpert, D.H., Macready, W.G., 1997. No free lunch theorems for optimization. IEEE Transactions on Evolutionary Computation 1 (1), 67–82.

Yang, Z., Tang, K., Yas, X., 2008. Large scale evolutionary optimization using cooperative coevolution. Information Sciences 178, 2985–2999.

Zaharie, D., Pectu, D., 2003. Adaptive Pareto differential evolution and its parallelization. In: Proceedings of the 5th International Conference on Parallel Processing and Applied Mathematics, Czestochowa, Poland, pp. 261–268.

Zhang, J.Q., Sanderson, A.C., 2009. JADE: adaptive differential evolution with optional external archive. IEEE Transactions on Evolutionary Computation 13 (5), 945–958.

Zhang, W.J., Xie, X.F., 2003. DEPSO: hybrid particle swarm with differential evolution operator. IEEE International Conference on Systems, Man, and Cybernetics 4 (3816-3821).

Zhang, J.Q., Avasarala, V., Subbu, R., 2010. Evolutionary optimization of transition probability matrices for credit decision-making. European Journal of Operational Research 200 (2), 557–567.