Soft Comput (2010) 14:1187–1207
DOI 10.1007/s00500-009-0510-5

ORIGINAL PAPER

Scale factor inheritance mechanism in distributed differential evolution

Matthieu Weber · Ville Tirronen · Ferrante Neri

Published online: 14 October 2009

Abstract This article proposes a distributed differential evolution which employs a novel self-adaptive scheme, namely scale factor inheritance. In the proposed algorithm, the population is distributed over several sub-populations allocated according to a ring topology. Each sub-population is characterized by its own scale factor value. With a probabilistic criterion, that individual displaying the best performance is migrated to the neighbor population and replaces a pseudo-randomly selected individual of the target sub-population. The target sub-population inherits not only this individual but also the scale factor if it seems promising at the current stage of evolution. In addition, a perturbation mechanism enhances the exploration feature of the algorithm. The proposed algorithm has been run on a set of various test problems and then compared to two sequential differential evolution algorithms and three distributed differential evolution algorithms recently proposed in literature and representing state-of-the-art in the field. Numerical results show that the proposed approach seems very efficient for most of the analyzed problems, and outperforms all other algorithms considered in this study.

Keywords Differential evolution · Distributed evolutionary algorithms · Evolutionary algorithms · Continuous optimization

M. Weber · V. Tirronen · F. Neri (✉)
Department of Mathematical Information Technology, University of Jyväskylä, P.O. Box 35, 40014 Agora, Finland
e-mail: ferrante.neri@jyu.fi

M. Weber
e-mail: matthieu.weber@jyu.fi

V. Tirronen
e-mail: ville.tirronen@jyu.fi

1 Introduction

Differential evolution (DE, see, Storn and Price 1995; Price et al. 2005; Chakraborty 2008) is a reliable and versatile function optimizer which displays a solid performance for diverse continuous optimization problems. DE is a very interesting population-based metaheuristic having mixed features. Due to its recombination and selection features DE can be seen as an evolutionary algorithm (EA). On the other hand, a DE structure tends to generate an individual with an above average performance which leads the exploration search, similar to swarm intelligence algorithms (SIA). In addition, DE, unlike EAs, generates offspring by perturbing the solutions with a scaled difference of two randomly selected population vectors, instead of recombining the solutions by means of a probabilistic criterion. Finally, DE employs a very peculiar survivor selection scheme, namely one to one spawning. This selection scheme allows replacement of an individual only if the offspring outperforms its corresponding parent.

Regardless of its classification, DE has proven to have a very good performance on various real-world problems. For example, in Joshi and Sanderson (1999) a DE application to the multisensor fusion problem is given. In Su and Lee (2003), and Chiou et al. (2004) DE applications to power electronics are presented. In Wang and Jang (2000) an application of DE to chemical engineering is proposed. In Storn (2005) a filter design is carried out by DE.

Reasons for success of the DE can be found in its simplicity and ease of implementation, while at the same time demonstrating reliability and high performance. In addition, the fact that only three parameters require tuning greatly contributes to the rapid diffusion of DE schemes among computer scientists and practitioners.

Although the DE undoubtedly has a great potential, setting of the control parameters is not a trivial task, since it has a heavy impact on the algorithmic performance. Thus, over the years, the DE community has intensively investigated the topic of parameter setting. Several studies have been reported, e.g., in Storn and Price (1997), Price and Storn (1997), Liu and Lampinen (2002), Rönkkönen et al. (2005), and Zielinski et al. (1857), and lead to contradictory conclusions. In other words, in accordance with the No Free Lunch Theorem, (Wolpert and Macready 1997), an efficient DE parameter setting is prone to problems, as the studies in Gämperle et al. (2002), Liu and Lampinen (2002), and Mallipeddi and Suganthan (2008) confirm.

To overcome the problem of the setting, some algorithms which employ adaptive and self-adaptive parameter settings have recently been proposed in literature. In Teo (2005, 2006), a variable population size is presented. The adaptive population size approach has been recently improved in two different implementations reported in Teng et al. (2009). Another scheme, which proposes a progressive population size reduction in DE, has been proposed in Brest and Maučec (2008). A variable population size DE, based on a fitness diversity adaptation is proposed in Tirronen and Neri (2009). An adaptive scheme for the DE scale factor is presented in Ali and Törn (2004). An automatic update of the scale factor has been proposed in Das et al. (2005). By following a similar line of thought, in Rönkkönen and Lampinen (2003), Omran et al. (2005), and Salman et al. (2007), a normal distribution is employed in order to perform a self-adaptation on the parameters F and CR. A Cauchy distribution in the self-adaptive scheme proposed in Soliman et al. (2007), Soliman and Bui (2008). An alternative kind of self-adaptation which employs the so called chaos mutation is proposed in Zhenyu et al. (2006). A controlled randomization of scale factor and crossover rate has been proposed in Brest et al. (2006).

In addition, since the DE algorithm suffers from many real world conditions, e.g., high dimensionality and noisy problems, some modifications on the standard DE scheme can significantly improve upon its performance. Modern DE-based algorithms can be divided into the two following categories:

1. DE integrating an extra component. This class includes those algorithms which use the DE as an evolutionary framework in which it is assisted by additional algorithmic components, e.g., local searchers or extra operators (see, Fan and Lampinen 2003; Liu and Lampinen 2005; Noman and Iba 2008; Tirronen et al. 2008; Caponio et al. 2009), and (Neri and Tirronen 2009). The algorithms belonging to this class can be clearly decomposed as a DE framework and additional components.
2. Modified structures of DE. This class includes those algorithms which make a substantial modification within the DE structure, in the search logic, the selection etc. Some examples are given in Qin et al. (2009) and Das et al. (2009).

A popular way to enhance the DE performance by structurally modifying the algorithmic functioning is through employment of structured populations. In other words, the population individuals are distributed over several sub-populations which evolve independently and interact by exchanging data and information details and contribute to a unique simultaneous evolution.

In Kwedlo and Bandurski (2006) a distributed DE scheme employing a ring topology (the cores are interconnected in a circle and the migrations occur following the ring) has been proposed for the training of a neural network. In Salomon et al. (2005), an example of DE parallelization is given for a medical imaging application. A few famous examples of distributed DE are presented in Zaharie (2002, 2003), Zaharie and Petcu (2003); in these papers the migration mechanism as well as the algorithmic parameters are adaptively coordinated according to criterion based on genotypical diversity. In paper, Zaharie (2004), a distributed DE for preserving diversity in the niches is proposed in order to solve multi-modal optimization problems. In Tasoulis et al. (2004), a distributed DE characterized by a ring topology and the migration of individuals with the best performance, to replace random individuals of the neighbor sub-population, has been proposed. An application of the algorithm in Tasoulis et al. (2004) for training of a neural network has been presented in Pavlidis et al. (2005). Following similar logic, paper (Kozlov and Samsonov 2006) proposes a distributed DE, where the computational cores are arranged according to a ring topology and, during migration, the best individual of a sub-population replaces the oldest member of the neighboring population. In De Falco et al. (2007a, b, c) a distributed DE has been designed for the image registration problem. In these papers, a computational core acts as a master by collecting the best individuals detected by the various sub-populations running in slave cores. The slave cores are connected in a grid and a migration is arranged among neighbor sub-populations. In Apolloni et al. (2008), a distributed DE which modifies the scheme proposed in Tasoulis et al. (2004) has been presented. In Apolloni et al. (2008), the migration is based on a probabilistic criterion depending on five parameters. It is worthwhile mentioning that some parallel implementations of sequential (without structured population) DE are also available in literature (see, Nipteni et al. 2006). An investigation of DE parallelization is given in Lampinen (1999).

This paper focuses on distributed differential evolutions and proposes a novel distributed algorithm. The proposed algorithm distributes its individuals within sub-populations arranged according to a ring topology. Each sub-population is characterized by its own scale factor. According to a simple probabilistic criterion, the migration of individual with the best performance and its associated scale factor occurs between neighbor sub-populations (following the ring topology). At each migration, the scale factor is also perturbed by means of a normal distribution. This paper is based on the idea that the scale factor is a determinant element within the DE search strategy. Thus, a successful search strategy can be inherited by the other sub-populations and propagated throughout the ring. A probabilistic perturbation enhances the exploration pressure of the algorithm.

The remaining of this paper is organized as follows. Section 2 describes the working principles of DE. Section 3 gives a short description of recently presented distributed versions of DE and introduces algorithms employed for comparison in the experimental section. Section 4 describes the proposed algorithm and discusses its algorithmic principle of functioning. Section 5 shows the experimental setup and numerical results of the present study. Section 6 gives the conclusions of this paper.

2 Sequential differential evolution

To clarify the notation used throughout this chapter we refer to the minimization problem of an objective function $f(x)$, where x is a vector of n design variables in a decision space D.

According to its original definition given in Storn and Price (1995), the DE consists of the following steps. An initial sampling of S_{pop} individuals is performed pseudo-randomly with a uniform distribution function within the decision space D. At each generation, for each individual x_i of the S_{pop}, three individuals x_r, x_s and x_t are pseudo-randomly extracted from the population. According to the DE logic, a provisional offspring x'_{off} is generated by mutation as:

$$x'_{\text{off}} = x_t + F(x_r - x_s) \tag{1}$$

where $F \in [0, 1+[$ is a scale factor which controls the length of the exploration vector $(x_r - x_s)$ and, thus, determines how far from point x_i the offspring should be generated. With $F \in [0, 1+[$, it is meant here that the scale factor should be a positive value which cannot be much greater than 1 (see, Price et al. 2005). While there is no theoretical upper limit for F, effective values are rarely greater than 1.0. The mutation scheme shown in Eq. 1 is also known as DE/rand/1. Other variants of the mutation rule have been subsequently proposed in literature (see, Qin and Suganthan 2005):

- DE/best/1: $x'_{\text{off}} = x_{\text{best}} + F(x_r - x_s)$
- DE/cur-to-best/1: $x'_{\text{off}} = x_i + F(x_{\text{best}} - x_i) + F(x_s - x_t)$
- DE/best/2: $x'_{\text{off}} = x_{\text{best}} + F(x_s - x_t) + F(x_u - x_v)$
- DE/rand/2: $x'_{\text{off}} = x_t + F(x_r - x_s) + F(x_u - x_v)$
- DE/rand-to-best/2: $x'_{\text{off}} = x_t + F(x_{\text{best}} - x_t) + F(x_r - x_s) + F(x_u - x_v)$

where x_{best} is the solution with the best performance among the individuals of the population, x_u and x_v are two additional pseudo-randomly selected individuals. It is worthwhile to mention the rotation invariant mutation shown in Qin and Suganthan (2005):

- DE/current-to-rand/1 $x_{\text{off}} = x_i + K(x_t - x_i) + F'(x_r - x_s)$

where K is the combination coefficient, which, as suggested in Price (1999), should be chosen with a uniform random distribution from [0, 1] and $F' = K \cdot F$. For this special mutation, the mutated solution does not undergo the crossover operation described below.

Recently, in Price et al. (2005), a new mutation strategy has been defined. This strategy, namely DE/rand/1/either–or, consists of the following:

$$x'_{\text{off}} = \begin{cases} x_t + F(x_r - x_s) & \text{if} \quad \text{rand}(0,1) < p_F \\ x_t + K(x_r + x_s - 2x_t) & \text{otherwise} \end{cases} \tag{2}$$

where for a given value of F, the parameter K is set equal to 0.5 $(F + 1)$.

When the provisional offspring has been generated by mutation, each gene of the individual x'_{off} is exchanged with the corresponding gene of x_i with a uniform probability and the final offspring x_{off} is generated:

$$x_{\text{off},j} = \begin{cases} x_{i,j} & \text{if} \quad \text{rand}(0,1) < CR \\ x'_{\text{off},j} & \text{otherwise} \end{cases} \tag{3}$$

where rand(0, 1) is a random number between 0 and1; j is the index of the gene under examination. The crossover described in Eq. 3 is known as binary crossover (simply indicated with "bin") and is the most common crossover scheme. It can be remarked that also the exponential crossover (see, Price et al. 2005), is used in some cases.

The resulting offspring x_{off} is evaluated and, according to a one-to-one spawning strategy, it replaces x_i if and only if $f(x_{\text{off}}) \leq f(x_i)$; otherwise no replacement occurs. It must be remarked that although the replacement indexes are saved one by one, during generation, actual replacements occur all at once at the end of the generation. For the sake of clarity, the pseudo-code highlighting working principles of the DE is shown in Fig. 1.

```
generate S_pop individuals of the initial population pseudo-randomly
while budget condition do
  for i = 1 : S_pop do
    compute f(x_i)
  end for
  for i = 1 : S_pop do
    {** Mutation **}
    select three individuals x_r, x_s, and x_t
    compute x'_off = x_t + F(x_r − x_s)
    {** Crossover **}
    x_off = x'_off
    for j = 1 : n do
      generate rand(0, 1)
      if rand(0, 1) < CR then
        x_off,j = x_i,j
      end if
    end for
    {** Selection **}
    if f(x_off) ≤ f(x_i) then
      save index for replacement x_i = x_off
    end if
  end for
  perform replacements
end while
```

Fig. 1 Pseudo-code of DE/rand/1/bin

3 Distributed differential evolution: recently developed algorithms

This section describes three distributed algorithms based on a DE structure recently proposed in literature. The algorithms described in this section are, according to our judgement, representative of the state-of-the-art structured DE algorithms and have been included in the benchmark for comparing the performance of the proposed approach. Although the notation can generate some confusion, i.e., all algorithms are distributed and can easily be parallelized, we decided to indicate these according to original terminology defined by their respective authors.

3.1 Parallel differential evolution

In Tasoulis et al. (2004), the problem of parallelization for DE schemes has been studied through an experimental analysis and an algorithm, namely parallel differential evolution (indicated here with PDE) has been proposed.

The original PDE implementation uses the parallel virtual machine (PVM), allowing multiple computers (called *nodes*) to be organized as a cluster and exchange arbitrary messages. The PDE algorithm is organized around one master node and m sub-populations running each on one node, and organized as a unidirectional ring, as illustrated in Fig. 2. It must be noted that although the logical topology is a ring which does not contain the master node, the actual topology is a star, where all communications (i.e., the migrations of individuals) are passing through the master.

The S_{pop} individuals constituting the populations are distributed over the m sub-populations composing the ring. Each sub-population is composed of $\frac{S_{\text{pop}}}{m}$ individuals. Each

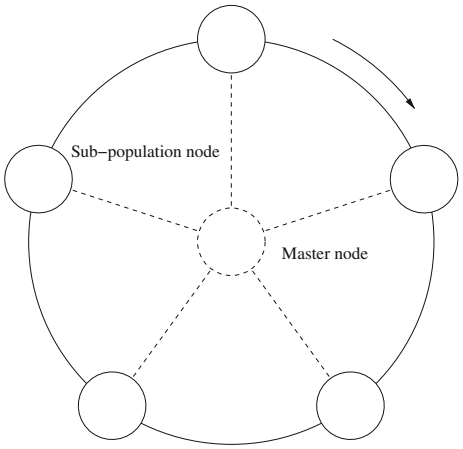


Fig. 2 Unidirectional ring topology in the parallel differential evolution algorithm

sub-population runs a regular DE algorithm while the master node coordinates the migration of individuals between sub-populations. On each generation, the sub-population has a given probability ϕ to send a copy of its best individual to its next neighbor sub-population in the ring. When migration occurs, the migrating individual replaces a pseudo-randomly selected individual belonging to the target sub-population. Figure 3 describes the behavior of both the master node and the sub-populations in more detail.

The DE variant run by each sub-population is the same across all the sub-populations. In Tasoulis et al. (2004), six mutation strategies have been compared, namely DE/best/1, DE/rand/1, DE/cur-to-best/1, DE/best/2, DE/rand/2 described in Sect. 2, as well as the trigonometric operator described in Fan and Lampinen (2003). Each strategy is used with different values of the migration constant ϕ and compared over seven test functions the dimensions of which vary between 2 and 30. The results in Tasoulis et al. (2004) showed that DE/best/1 is the most efficient mutation strategy and quite stable across different values of ϕ for the low dimensional problems analyzed.

3.2 Island based distributed differential evolution

In Apolloni et al. (2008) a distributed DE algorithm, namely island based distributed differential evolution (IBDDE) has been proposed. The IBDDE algorithm is a modified version of PDE described in Subsect. 3.1. In IBDDE, the population, having size S_{pop}, is structured in m sub-populations. Thus, each sub-population is composed of $\frac{S_{\text{pop}}}{m}$ individuals. The migration policy is then defined as a five-tuple $\mathcal{M} = (\gamma, \rho, \phi_s, \phi_r, \tau)$. $\gamma \in \mathbb{N}$ is the number of generations between two migrations, $\rho \in \mathbb{N}$ is the number of individuals which migrate from a sub-population P during each migration, ϕ_s is the selection function which,

Fig. 3 Pseudo-code of PDE. **a** At the master node, **b** At each sub-population

```
spawn N sub-populations, each one on a different processor
for each generation do
   receive an individual from each sub-population
   for each received individual do
      if rand(0, 1) < φ then
         send the individual to the next sub-population in the ring
      end if
   end for
   if the stop criterion for the objective function is met then
      send a termination signal to all the sub-populations
   end if
end for
```

(a)

```
for each generation do
   perform a DE generation
   send a copy of the best individual to the master node
   if a migrated individual has been received then
      replace a random individual, different from the best, by this migrated individual
   end if
   if a termination signal has been received then
      terminate the execution
   end if
end for
```

(b)

applied to a sub-population, returns the migrating individuals $v_g'^i$, ϕ_r is the replacement function that selects individuals to be replaced by the immigrants in the receiving sub-population, and τ is the topological rule, which selects the target sub-population Q. The individuals to be migrated are pseudo-randomly (uniformly) chosen by the selection function ϕ_s. Incoming individuals from other sub-populations replace pseudo-randomly chosen local individuals, only if the former are better, by the replacement function ϕ_r.

Figure 4 describes the algorithm as pseudo-code.

In Apolloni et al. (2008), the experiments have been run with a population size S_{pop} equal to 20. The population is divided into two sub-populations of 10 individuals in one experiment, and into four sub-populations of 5 individuals in a second experiment. The migration parameters are set to $\gamma = 100, \rho = 1$, the functions ϕ_s and ϕ_r are defined to randomly select an individual, the topology τ is a unidirectional ring very similar to the logical topology used by PDE (see, Subsect. 3.1). The mutation strategy for DE is DE/rand/1, and the algorithm is tested on 25 different test functions in 30 and 50 dimensions, for a total of 50 test functions.

3.3 Distributed differential evolution

In De Falco et al. (2007a, b, c), in order to solve some image registration problems a distributed DE (indicated here with DDE) has been proposed. This algorithm differs from PDE and IBDDE by the topology it uses. Instead of a unidirectional ring, DDE uses a locally connected topology, where each node is connected to μ other nodes.

```
initialize(P)
while the stopping condition is not met do
   perform a DE generation
   if the last migration was γ generations ago then
      for each of the ρ individuals to send do
         v'^i_g ← φ_s(P)
         send v'^i_g to Q chosen by τ
      end for
   end if{** Asynchronous communication **}
   while individuals are arriving do
      receive v'^i_g from P
      replace an individual chosen from φ_r(Q) by v'^i_g
   end while
end while
```

Fig. 4 Pseudo-code of IBDDE for the sub-population P

Figure 5 represents such a topology, where the nodes are arranged in a mesh folded into a torus.

In De Falco et al. (2007a, b, c), it has been proposed to set $\mu = 4$, i.e., each node (such as the black disc in the Fig. 5) has exactly four nearest neighbors (represented by the four gray discs). In DDE, each node represents one processor running a DE algorithm with a DE/rand/1 mutation strategy on a sub-population. Every M_I generations (the migration interval), each sub-population is allowed to exchange S_I (the migration rate) individuals with its nearest neighbors. In the experimental setup, each node sends a copy of its best individual to its neighbors. Figure 6 describes the algorithm as pseudo-code.

DDE also makes use of a master node, the role of which is to collect the best solutions found in each sub-population and to present these results to the user.

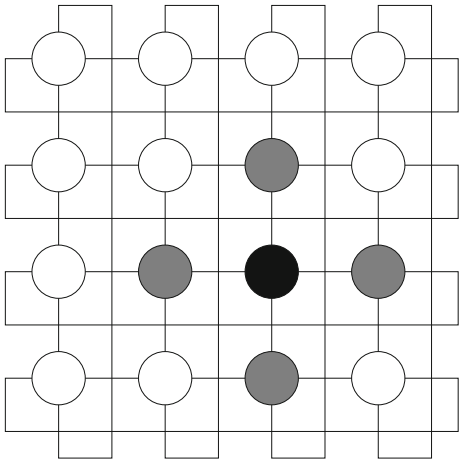


Fig. 5 Torus topology in distributed differential evolution

```
initialize the sub-population
while the stopping condition is not met do
  perform a DE generation
  if the last migration was M_I generations ago then
    send a copy of the best individual to each neighbor
  end if
  if there are incoming individuals then
    replace the worst S_I × μ individuals by the S_I × μ incoming ones
  end if
end while
```

Fig. 6 Pseudo-code of the DDE algorithm at a sub-population

4 Distributed differential evolution with scale factor inheritance

The proposed algorithm enhances the PDE structure described in Subsect. 3.1 by means of the implementation, in a distributed logic, of the self-adaptive parameter control proposed in Brest et al. (2006). More specifically, an adaptive control of the scale factor "F" is proposed here. The novel mechanism proposed here in this article is named scale factor inheritance. The proposed algorithm, namely "F" adaptive control parallel differential evolution (FACPDE) consists of the following steps.

At the beginning of the optimization process, S_{pop} individuals are pseudo-randomly sampled within the decision space D. These S_{pop} are distributed over m sub-populations; each sub-population is composed of $\frac{S_{\text{pop}}}{m}$ individuals. For each (generic hth) sub-population a scale factor F^k, for $k = 1, \ldots, m$, is assigned. Each scale factor is initially generated as pseudo-random by sampling a value from a uniform distribution between -1 and 1. The sub-populations are then arranged according to ring topology, as with the PDE topology represented in Fig. 2.

At each generation, each sub-population performs a DE scheme. For each individual x_i of the $\frac{S_{\text{pop}}}{m}$, three individuals x_r, x_s and x_t are pseudo-randomly extracted from the population. The provisional offspring x'_{off} is generated by mutation as:

$$x'_{\text{off}} = x_t + F^k(x_r - x_s). \tag{4}$$

It is clear that a scale factor taking on a negative value means that the search direction is inverted.

When the provisional offspring has been generated by mutation, each gene of the individual x'_{off} is exchanged with the corresponding gene of x_i with a uniform probability and the final offspring x_{off} is generated:

$$x_{\text{off},j} = \begin{cases} x_{i,j} & \text{if} \quad \text{rand}(0,1) < CR \\ x'_{\text{off},j} & \text{otherwise} \end{cases} \tag{5}$$

where, as for the sequential DE, rand(0, 1) is a random number between 0 and 1; j is the index of the gene under examination. The standard one-to-one spawning is then applied and, at the end of each generation, the scheduled replacements are performed.

For each sub-population, between two subsequent generations, a pseudo-random number rand(0, 1) is generated by means of a uniform distribution. Analogous to what was explained about the PDE in Subsect. 3.1, this pseudo-random number is then compared with a constant value ϕ, namely migration constant. If $\text{rand}(0,1) < \phi$, the individual with the best performance is selected to undergo migration. Thus, for the generic kth sub-population, the individual x^k_{best} is duplicated and then replaces a pseudo-randomly selected individual of the neighbor (in the ring) sub-population. The scale factor inheritance mechanism occurs contextually with the migration. More specifically, when the migration occurs, performance of the individual x^k_{best} is compared with that of the best individual belonging to the target sub-population, indicated here with x^{k+1}_{best}. If the new immigrant has a better performance than the best individual of the target sub-population, i.e., if $f\left(x^k_{\text{best}}\right) < f\left(x^{k+1}_{\text{best}}\right)$, the $(k+1)$th sub-population inherits the scale factor F^k after a perturbation. More specifically, the scale factor F^{k+1} related to the $(k+1)$th sub-population is updated according to the following formula:

$$F^{k+1} = F^k + \alpha\mathcal{N}(0,1) \tag{6}$$

where $\mathcal{N}(0,1)$ is a pseudo-random value sampled from a normal distribution characterized by a zero mean and variance equal to 1. The constant value α has the role of controlling the range of perturbation values $\alpha\mathcal{N}(0,1)$. It must be observed that we did not impose any bounds for the variation of F. On the contrary, we decided to allow an unbounded variation of the control parameter and rely on the self-adaptation mechanism.

If the new immigrant does not outperform the best individual of the target sub-population, i.e., $f\left(x^k_{\text{best}}\right) \geq f\left(x^{k+1}_{\text{best}}\right)$, the scale factor inheritance mechanism is not activated and, thus, only the migration of the individual x^k_{best} occurs.

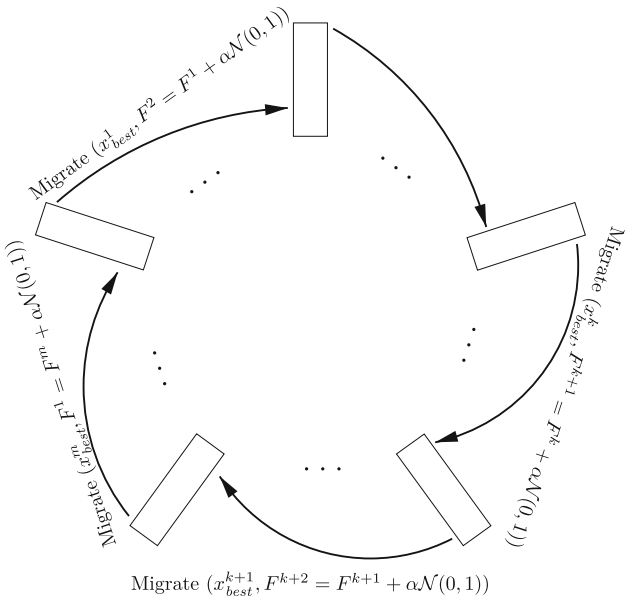


Fig. 7 Graphical representation of FACPDE

The described operations are repeated until the budget conditions are satisfied.

A graphical representation of FACPDE is given in Fig. 7.

For the sake of clarity, the pseudo-code illustrating working principles of FACPDE is shown in Fig. 8.

4.1 Scale factor inheritance: algorithmic philosophy

As shown in Sect. 2, DE is based on a very simple idea, i.e., a search by means of adding vectors and a one-to-one spawning for the survivor selection. Thus, DE is very simple to implement/code and contains a limited number of parameters to tune (only S_{pop}, F, and CR).

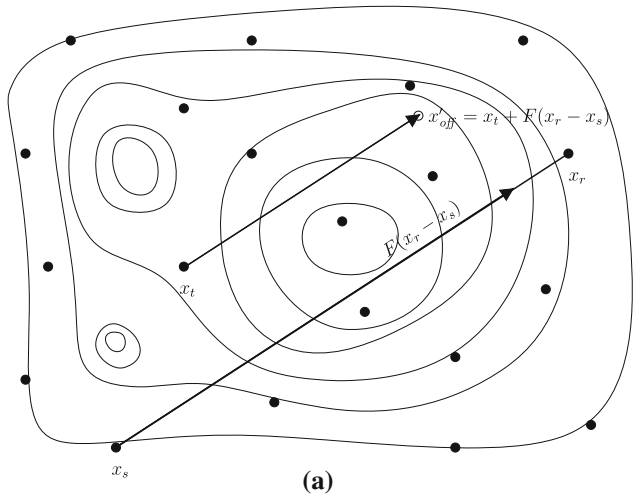
From an algorithmic viewpoint, reasons for the success of DE have been highlighted in Feoktistov (2006): success of DE is due to an implicit self-adaptation contained within the algorithmic structure. More specifically, since, for each candidate solution, the search rule depends on other solutions belonging to the population (e.g., x_t, x_r, and x_s), the capability of detecting new promising offspring solutions depends on the current distribution of the solutions within the decision space. During early stages of the optimization process, solutions tend to be spread out within the decision space. For a given scale factor value, this implies that the mutation appears to generate new solutions by exploring the space by means of a large step size (if x_r and x_s are distant solutions, $F(x_r - x_s)$ is a vector characterized by a large modulus). During the optimization process, the solutions of the population tend to concentrate on specific parts of the decision space. Therefore, step size in the mutation is progressively reduced and the search is performed in the neighborhood of the solutions. In other words, due to its structure, a DE scheme is highly explorative at the beginning of the evolution and subsequently becomes more exploitative during optimization.

Although this mechanism seems at first glance to be very efficient, it hides a limitation. If for some reason, the algorithm does not succeed in generating offspring solutions which outperform the corresponding parent, the search is repeated again with similar step size values and will likely fail by falling into an undesired stagnation condition (see, Lampinen and Zelinka 2000). Stagnation is the undesired effect which occurs when a population-based algorithm does not converge to a solution (even suboptimal) and the population diversity is still high. In the case of the DE, stagnation occurs when the algorithm does not manage to improve upon any solution of its population for a prolonged number of generations. In other words, the main drawback of the DE is that the scheme has, for each stage of the optimization process, a limited amount of exploratory moves. If these moves are not enough for generating new promising solutions the search can be heavily compromised.

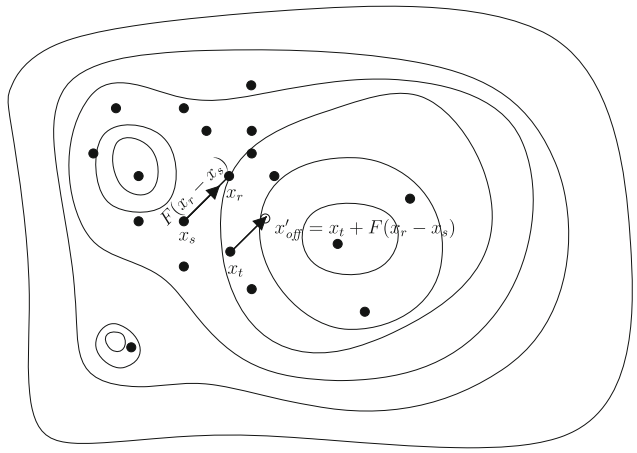
Thus, in order to enhance the DE performance, alternative search moves should support the original scheme and promote a successful continuation of the optimization process. The use of multiple populations in distributed DE algorithms allows an observation of the decision space

Fig. 8 Pseudo-code of the FACPDE algorithm

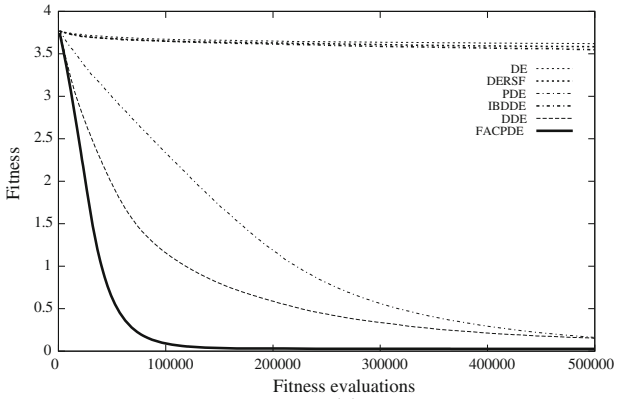
```
while budget conditions do
  for each generation do
    for each sub-population k = 1 : m do
      if rand(0,1) < φ then
        select and copy x^k_best
        migrate x^k_best into the (k+1)^th sub-population by replacing a pseudo-randomly
        selected individual
        if f(x^k_best) < f(x^{k+1}_best) then
          F^{k+1} = F^k + αN(0,1)
        end if
      end if
    end for
  end for
end while
```



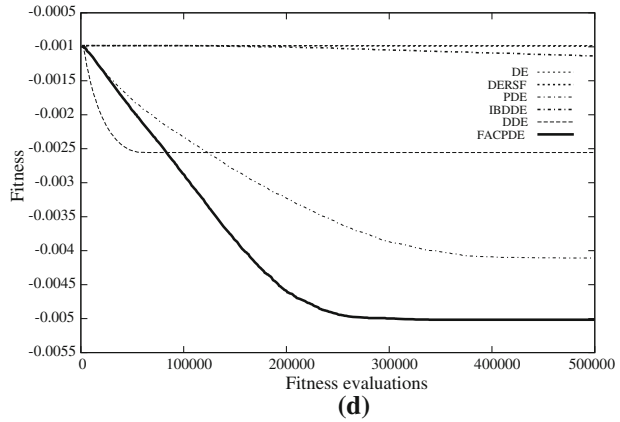
(a)



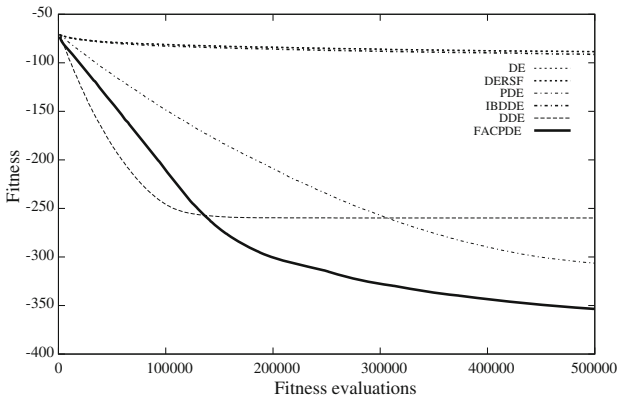
(b)



(c)



(d)



(e)

Fig. 9 Search mechanism in differential evolution performance trends in 500 dimensions. **a** Too explorative conditions, **b** Too exploitative conditions, **c** Ackley, **d** DropWave, **e** Michalewicz, **f** Rastrigin, **g** Rotated Michalewicz, **h** Rotated pathological, **i** Rotated rastrigin, **j** Schwfel, **k** Tirronen

from various perspectives and, most importantly, decreases the risk of stagnation since each sub-population imposes a high exploitation pressure. In addition, the migration mechanism ensures that solutions with a high performance are included within the sub-populations during their evolution. This fact is equivalent to modifying the set of search moves. If the migration gives privilege to the best individuals, the new search moves promote the detection of new promising search directions and, thus, allow the DE search structure to be periodically "refurbished". Thus, migration is supposed to mitigate the risk of DE (sub-)populations stagnating and to enhance the global algorithmic performance.

As a countermeasure against stagnation, several recent DE versions propose a randomization in the search logic which increases the amount of potential exploration moves. For example, in Brest et al. (2006), scale factor and crossover rates are periodically updated by generating new random values. This simple operation seems to have a very relevant effect on the algorithmic performance. Also the

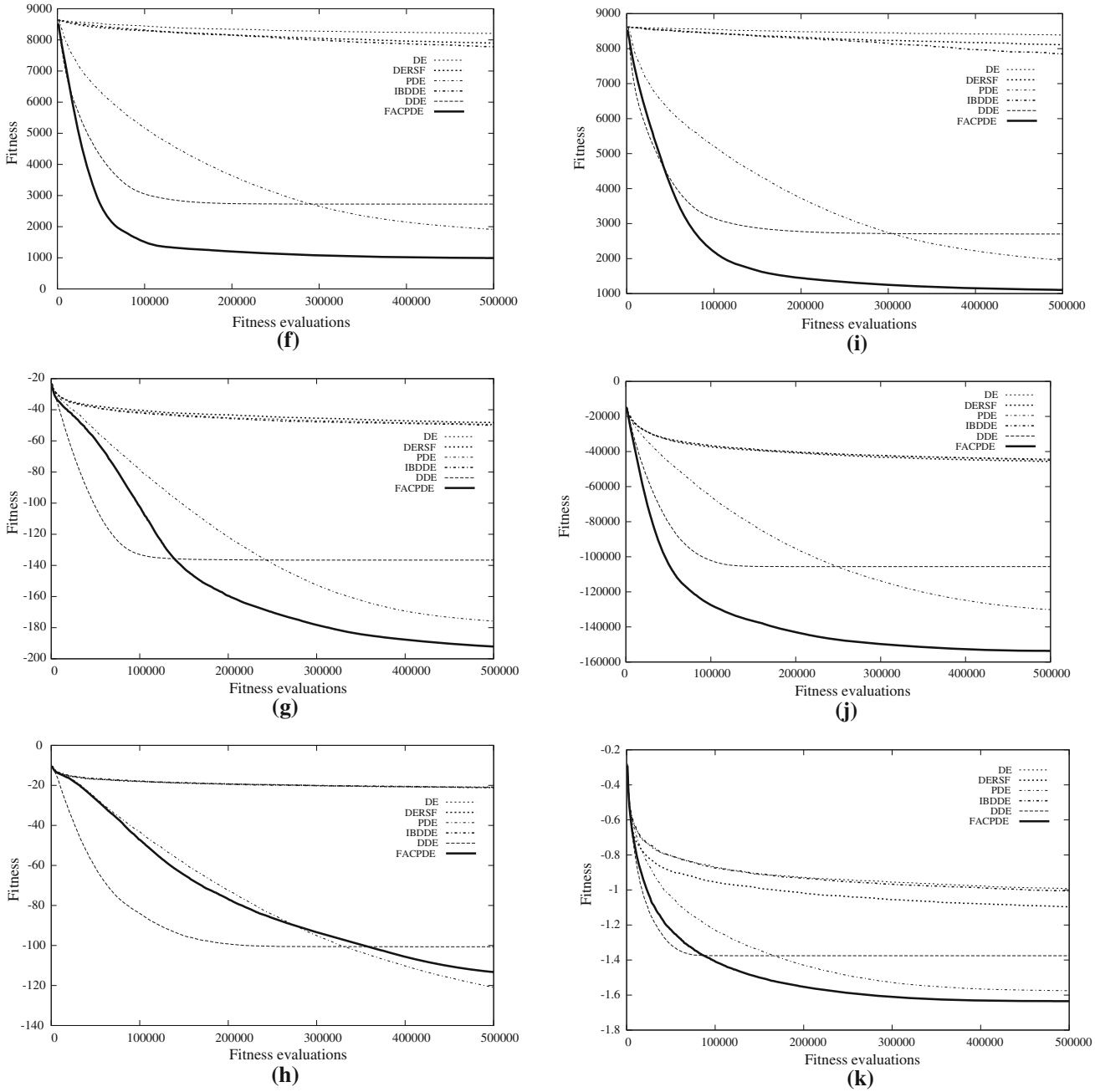


Fig. 9 continued

DE scheme proposed in Qin et al. (2009) enhances a previously proposed algorithm (presented in Qin and Suganthan 2005) by introducing a randomization of the control parameters. Thus, two operations are needed in order to obtain significant improvements in DE performance which seems to be the updating and randomization of control parameters.

In this paper we focus on the scale factor dynamics. Although DE schemes are characterized by the implicit self-adaptation described above, employment of a unique and constant scale factor value can be improper since the exploratory moves depend on distribution of the solution within the decision space. For example, for a highly multimodal problem, a scale factor $F \approx 1$ can generate very long moving vectors $F(x_r - x_s)$ if the population is spread out within the decision space (see, Fig. 9a) and very short moving vectors if the population is concentrated in some areas of the decision space (see, Fig. 9b). This fact may lead to an excessively explorative behavior during some stages of the evolution and an excessively exploitative behavior during other stages. These two behaviors can cause, respectively, stagnation due to an incapability to

detect a promising search direction and a premature convergence due to the excessive exploitation of a suboptimal basin of attraction. In other words, a proper choice of the scale factor depends not only on the optimization problem but also on the stage of the evolution.

On the basis of this consideration, the scale factor inheritance mechanism has been designed. The initial pseudo-random assignment of the scale factors (one per sub-population) allows each FACPDE sub-population to explore the decision space from complementary perspectives. Subsequently, each evolving sub-population can be seen as a separate searcher which cooperates and competes with the other searcher, analogous to memetic algorithms (see, Ong and Keane 2004; Tang et al. 2007a; Caponio et al. 2007), and (Neri et al. 2007). The sub-populations cooperate with each other by means of migration of the individual with the best performance which can be seen as a suggestion of a promising search direction. The sub-populations compete with each other by means of the scale factor migration; this fact can be seen like the imposition of the most successful search strategy to other sub-populations employing a weaker strategy. The ring topology assures propagation of this successful strategy to all sub-populations. On the other hand, this propagation is rather slow, so as to avoid too greedy an algorithmic behavior. In addition, randomization of the scale factor when the migration occurs guarantees a certain diversity of scale factors even in the event that a propagation throughout the entire ring occurs. Thus, we avoid that all the sub-population are characterized by the same scale factor.

In summary, the combined action of an exploration of the decision space from diverse and complementary perspectives, the cooperation mechanism with its suggestion of promising search directions, the competitive procedure of the most successful search strategies and their randomized updates should, together, compose a robust DE-based algorithm which efficiently balances its explorative and exploitative resources in order to efficiently and robustly solve complex optimization problems.

5 Experimental results

To prove the viability of the FACPDE and test its performance with respect to modern distributed DE-based algorithms, the following numerical experiments have been performed. The test problems listed in Table 1 have been considered in this study.

The rotated version of some of the test problems listed in Table 1 have been included into the benchmark set. These rotated problems have been generated through the multiplication of the vector of variables by a randomly generated orthogonal rotation matrix. In total, 24 test problems have been considered in this study with both $n = 500$ and $n = 1{,}000$. The FACPDE has been tested with these 24 test problems and its behavior and performance have been compared with those obtained by standard sequential DE/rand/1/bin (simply indicated here as DE), the improved DE algorithm employing random scale factor proposed in Das et al. (2005) and indicated as

Table 1 Test problems

Test problem	Function	Decision space
Ackley	$-20 + e + 20\exp\left(-\frac{0.2}{n}\sqrt{\sum_{i=1}^{n} x_i^2}\right)$ $-\exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi \cdot x_i)x_i\right)$	$[-1, 1]^n$
Alpine	$\sum_{i=1}^{n} \lvert x_i \sin x_i + 0.1x_i\rvert$	$[-10, 10]^n$
Axis-parallel hyper-ellipsoid	$\sum_{i=1}^{n} ix_i^2$	$[-5.12, 5.12]^n$
DeJong	$\lvert\lvert x\rvert\rvert^2$	$[-5.12, 5.12]^n$
DropWave	$-\frac{1+\cos\left(12\sqrt{\lvert\lvert x\rvert\rvert^2}\right)}{\frac{1}{2}\lvert\lvert x\rvert\rvert^2+2}$	$[-5.12, 5.12]^n$
Griewangk	$\frac{\lvert\lvert x\rvert\rvert^2}{4000} - \prod_{i=0}^{n}\cos\frac{x_i}{\sqrt{i}} + 1$	$[-600, 600]^n$
Michalewicz	$-\sum_{i=1}^{n}\sin x_i\left(\sin\left(\frac{i \cdot x_i^2}{\pi}\right)\right)^{20}$	$[0, \pi]^n$
Pathological	$\sum_{i=1}^{n-1}\left(0.5 + \frac{\sin^2\left(\sqrt{100x_i^2 + x_{i+1}^2} - 0.5\right)}{1+0.001*\left(x_i^2 - 2x_i x_{i+1} + x_{i+1}^2\right)^2}\right)$	$[-100, 100]^n$
Rastrigin	$10n + \sum_{i=0}^{n}\left(x_i^2 - 10\cos(2\pi x_i)\right)$	$[-5.12, 5.12]^n$
Rosenbrock valley	$\sum_{i=1}^{n-1}\left(100\left(x_{i+1} - x_i^2\right)^2 + (1 - x_i)^2\right)$	$[-2.048, 2.048]^n$
Schwefel	$\sum_{i=1}^{n} -x_i \sin\left(\sqrt{\lvert x_i\rvert}\right)$	$[-500, 500]^n$
Sum of powers	$\sum_{i=1}^{n} \lvert x_i\rvert^{i+1}$	$[-1, 1]^n$
Tirronen	$3\exp\left(-\frac{\lvert\lvert x\rvert\rvert^2}{10n}\right) - 10\exp\left(-8\lvert\lvert x\rvert\rvert^2\right)$ $+\frac{2.5}{n}\sum_{i=1}^{n}\cos\left(5\left(x_i + (1 + i \bmod 2)cos\left(\lvert\lvert x\rvert\rvert^2\right)\right)\right)$	$[-10, 5]^n$

Table 2 Average final fitness values ± standard deviations for 500-dimension problems

	DE	DERSF	PDE	IBDDE	DDE	FACPDE
Ackley	3.62e + 00 ± 9.69e − 03	3.58e + 00 ± 1.40e−02	1.62e − 01 ± 1.67e − 02	3.55e + 00 ± 2.96e − 02	1.51e − 01 ± 6.96e − 02	**2.67e − 02 ± 1.25e − 02**
Alpine	1.30e + 03 ± 1.19e + 01	1.24e + 03 ± 1.53e + 01	8.88e + 01 ± 1.26e + 01	1.21e + 03 ± 3.43e + 01	1.50e + 02 ± 4.35e + 01	**3.09e + 01 ± 1.60e + 01**
Ax.-par. hyp.-ell.	8.67e + 05 ± 1.07e + 04	7.06e + 05 ± 1.50e + 04	3.68e + 03 ± 6.84e + 02	7.27e + 05 ± 3.70e + 04	4.09e + 03 ± 4.73e + 03	**3.28e + 02 ± 2.96e + 02**
DeJong	3.86e + 03 ± 5.73e + 01	3.34e + 03 ± 4.98e + 01	1.92e + 01 ± 3.57e + 00	3.19e + 03 ± 2.67e + 02	1.61e + 01 ± 1.15e + 01	**1.80e + 00 ± 2.02e + 00**
DropWave	−9.83e − 04 ± 2.23e − 05	−9.90e − 04 ± 1.91e − 05	−4.11e − 03 ± 3.96e − 04	−1.13e − 03 ± 8.03e − 05	−2.56e − 03 ± 2.69e − 04	**−5.02e − 03 ± 8.78e − 04**
Griewangk	1.38e + 04 ± 2.15e + 02	1.19e + 04 ± 1.73e + 02	5.62e + 02 ± 1.09e + 01	1.13e + 04 ± 7.65e + 02	5.68e + 02 ± 4.86e + 01	**5.04e + 02 ± 1.19e + 01**
Michalewicz	−8.84e + 01 ± 1.61e + 00	−8.85e + 01 ± 9.05e − 01	−3.06e + 02 ± 5.68e + 00	−9.13e + 01 ± 2.09e + 00	−2.60e + 02 ± 8.56e + 00	**−3.54e + 02 ± 3.12e + 01**
Pathological	−1.05e + 02 ± 1.96e + 00	−1.21e + 02 ± 2.40e + 00	−3.34e + 02 ± 5.98e + 00	−3.34e + 02 ± 7.40e + 00	−3.06e + 02 ± 7.28e + 00	**−3.55e + 02 ± 3.05e + 01**
Rastrigin	8.21e + 03 ± 5.26e + 01	7.90e + 03 ± 6.55e + 01	1.91e + 03 ± 9.94e + 01	7.78e + 03 ± 1.55e + 02	2.73e + 03 ± 2.28e + 02	**9.91e + 02 ± 3.93e + 02**
Rosenbrock	1.94e + 05 ± 5.44e + 03	1.79e + 05 ± 4.47e + 03	2.11e + 03 ± 9.94e + 01	1.35e + 05 ± 1.61e + 04	1.82e + 03 ± 6.14e + 02	**1.63e + 03 ± 6.18e + 02**
Schwefel	−4.42e + 04 ± 7.07e + 02	− 4.57e + 04 ± 9.54e + 02	−1.30e + 05 ± 3.17e + 03	−4.48e + 04 ± 8.15e + 02	−1.06e + 05 ± 4.19e + 03	**−1.54e + 05 ± 1.83e + 04**
Sum of powers	1.82e + 00 ± 3.99e − 01	1.82e + 00 ± 3.99e − 01	1.06e − 05 ± 5.19e − 05	3.09e − 01 ± 1.67e − 01	1.02e − 03 ± 2.51e − 03	**4.82e − 06 ± 1.15e − 05**
Tirronen	−9.92e − 01 ± 2.02e − 02	−1.10e + 00 ± 1.77e − 02	−1.57e + 00 ± 3.44e − 02	−1.01e + 00 ± 1.90e − 02	−1.38e + 00 ± 6.76e − 02	**−1.63e + 00 ± 3.41e − 02**
Rt. Ackley	3.65e + 00 ± 1.54e − 02	3.56e + 00 ± 1.99e − 02	2.15e−01±2.50e−02	3.53e + 00 ± 3.82e − 02	1.94e − 01 ± 7.21e − 02	**6.92e − 02 ± 1.92e − 02**
Rt. Alpine	1.32e + 03 ± 1.60e + 01	1.28e + 03 ± 1.54e + 01	1.03e + 02 ± 9.74e + 00	1.23e + 03 ± 4.13e + 01	1.39e + 02 ± 2.64e + 01	**6.26e + 01 ± 2.87e + 01**
Rt. Ax.−par. hyp.−ell.	8.41e + 05 ± 1.53e + 04	7.43e + 05 ± 1.43e + 04	4.90e + 03 ± 7.92e + 02	7.12e + 05 ± 3.97e + 04	3.87e + 03 ± 2.20e + 03	**1.07e + 03 ± 4.10e + 02**
Rt. Griewangk	1.28e + 04 ± 2.16e + 02	1.12e + 04 ± 2.05e02	5.66e + 02 ± 1.03e + 01	1.10e + 04 ± 6.18e + 02	5.56e + 02 ± 3.75e + 01	**5.10e + 02 ± 5.04e + 00**
Rt. Michalewicz	−4.93e + 01 ± 1.90e + 00	−4.82e + 01 ± 1.18e + 00	−1.76e + 02 ± 7.76e + 00	−4.99e + 01 ± 1.62e + 00	−1.37e + 02 ± 7.98e + 00	**−1.92e + 02 ± 1.20e + 01**
Rt. Pathological	−2.11e + 01 ± 1.36e + 00	−2.09e + 01 ± 7.80e − 01	**−1.21e + 02 ± 7.30e + 00**	−2.11e + 01 ± 1.05e + 00	−1.01e + 02 ± 1.20e + 01	−1.13e + 02 ± 1.26e + 01
Rt. Rastrigin	8.39e + 03 ± 6.89e + 01	8.11e + 03 ± 5.68e + 01	1.95e + 03 ± 1.51e + 02	7.85e + 03 ± 2.27e + 02	2.70e + 03 ± 2.60e + 02	**1.10e + 03 ± 1.85e + 02**
Rt. Rosenbrock	1.95e + 05 ± 6.04e + 03	1.75e + 05 ± 4.47e + 03	1.66e + 03 ± 1.53e + 02	1.47e + 05 ± 1.56e + 04	1.45e + 03 ± 4.68e + 02	**9.74e + 02 ± 1.74e + 02**
Rt. Schwefel	−5.16e + 04 ± 1.80e + 03	−5.45e + 04 ± 1.35e + 03	−1.65e + 05 ± 4.74e + 03	−5.31e + 04 ± 1.87e + 03	−1.27e + 05 ± 7.80e + 03	**−1.67e + 05 ± 1.22e + 04**
Rt. Sum of powers	3.81e + 16 ± 1.28e + 17	8.50e + 16 ± 4.18e + 17	**1.06e − 05 ± 5.20e − 05**	2.13e + 15 ± 7.15e + 15	2.27e − 03 ± 6.51e − 03	2.47e − 04 ± 1.71e − 03
Rt. Tirronen	−5.06e − 01 ± 2.09e − 02	−5.02e − 01 ± 2.34e − 02	−1.24e + 00 ± 7.73e − 02	−5.08e − 01 ± 2.45e−02	−9.31e − 01 ± 9.41e − 02	**−1.28e + 00 ± 7.60e − 02**

Table 3 Average final fitness values ± standard deviations for 1,000-dimension problems

	DE	DERSF	PDE	IBDDE	DDE	FACPDE
Ackley	3.75e + 00 ± 7.55e − 03	3.72e + 00 ± 7.94e − 03	9.75e − 01 ± 4.77e − 02	3.72e + 00 ± 1.61e − 02	7.25e − 01 ± 8.17e − 02	**2.00e − 02 ± 1.02e − 02**
Alpine	2.77e + 03 ± 2.71e + 01	2.73e + 03 ± 1.66e + 01	6.24e + 02 ± 3.28e + 01	2.66e + 03 ± 6.16e + 01	3.85e + 02 ± 4.99e + 01	**1.05e + 02 ± 2.33e + 02**
Ax.-par. hyp.-ell.	3.96e + 06 ± 4.94e + 04	3.89e + 06 ± 4.28e + 04	1.81e + 05 ± 1.23e + 04	3.20e + 06 ± 1.62e + 05	1.06e + 05 ± 1.86e + 04	**6.43e + 02 ± 9.09e + 02**
DeJong	8.01e + 03 ± 9.20e + 01	8.00e + 03 ± 9.03e + 01	4.66e + 02 ± 2.81e + 01	6.29e + 03 ± 3.74e + 02	2.83e + 02 ± 5.12e + 01	**1.75e + 00 ± 1.94e + 00**
DropWave	−4.84e − 04 ± 7.42e − 06	−4.84e − 04 ± 7.42e − 06	−1.62e − 03 ± 9.29e − 05	−5.20e − 04 ± 2.21e − 05	−1.17e − 03 ± 9.73e − 05	**−2.06e − 03 ± 2.47e − 04**
Griewangk	2.85e + 04 ± 3.16e + 02	2.84e + 04 ± 2.72e + 02	2.58e + 03 ± 1.03e + 02	2.28e + 04 ± 1.55e + 03	1.96e + 03 ± 1.80e + 02	**1.02e + 03 ± 2.12e + 02**
Michalewicz	−1.45e + 02 ± 1.72e + 00	−1.49e + 02 ± 1.72e + 00	−4.18e + 02 ± 9.73e + 00	−1.49e + 02 ± 2.24e + 00	−4.43e + 02 ± 1.22e + 01	**−6.03e + 02 ± 1.57e + 02**
Pathological	−1.57e + 02 ± 2.41e + 00	−1.98e + 02 ± 3.95e + 00	−6.39e + 02 ± 8.43e + 00	−1.87e + 02 ± 3.77e + 01	−5.70e + 02 ± 1.30e + 01	**−7.17e + 02 ± 6.26e + 00**
Rastrigin	1.75e + 04 ± 9.32e + 01	1.71e + 04 ± 7.23e + 01	6.65e + 03 ± 2.16e + 02	1.69e + 04 ± 3.40e + 02	6.18e + 03 ± 3.96e + 02	**1.99e + 03 ± 1.35e + 03**
Rosenbrock	4.06e + 05 ± 7.22e + 03	4.06e + 05 ± 7.19e + 03	1.34e + 04 ± 9.17e + 02	2.90e + 05 ± 3.35e + 04	8.96e + 03 ± 1.82e + 03	**2.94e + 03 ± 6.19e + 02**
Schwefel	−6.43e + 04 ± 1.08e + 03	−6.51e + 04 ± 9.96e + 02	−1.91e + 05 ± 4.04e + 03	−6.45e + 04 ± 1.08e + 03	−1.86e + 05 ± 8.61e + 03	**−3.24e + 05 ± 5.25e + 04**
Sum of powers	1.90e + 00 ± 3.55e − 01	1.90e + 00 ± 3.55e − 01	**3.07e − 06 ± 1.01e − 05**	1.07e + 00 ± 3.69e − 01	4.68e − 04 ± 1.64e − 03	3.50e − 06 ± 1.59e − 05
Tirronen	−8.49e − 01 ± 1.27e − 02	−9.72e − 01 ± 1.46e − 02	−1.45e + 00 ± 2.04e − 02	−8.63e − 01 ± 1.40e − 02	−1.35e + 00 ± 5.11e − 02	**−1.59e + 00 ± 3.57e − 02**
Rt. Ackley	3.78e + 00 ± 1.04e − 02	3.74e + 00 ± 1.21e − 02	9.35e − 01 ± 5.37e − 02	3.69e + 00 ± 4.53e − 02	6.90e − 01 ± 7.83e − 02	**6.45e − 02 ± 1.91e − 02**
Rt. Alpine	2.83e + 03 ± 2.14e + 01	2.79e + 03 ± 2.22e + 01	6.13e + 02 ± 2.58e + 01	2.70e + 03 ± 8.77e + 01	4.01e + 02 ± 4.19e + 01	**1.13e + 02 ± 3.59e + 01**
Rt. Ax.-par. hyp.-ell.	3.87e + 06 ± 5.89e + 04	3.76e + 06 ± 4.55e + 04	1.59e + 05 ± 1.25e + 04	3.31e + 06 ± 2.20e + 05	8.52e + 04 ± 1.60e + 04	**3.31e + 03 ± 1.24e + 03**
Rt. Griewangk	2.80e + 04 ± 3.56e + 02	2.76e + 04 ± 2.48e + 02	2.24e + 03 ± 9.40e + 01	2.33e + 04 ± 1.52e + 03	1.75e + 03 ± 1.60e + 02	**1.01e + 03 ± 5.59e + 00**
Rt. Michalewicz	−7.20e + 01 ± 2.82e + 00	−7.51e + 01 ± 1.83e + 00	−2.45e + 02 ± 7.54e + 00	−7.24e + 01 ± 2.19e + 00	−2.29e + 02 ± 1.23e + 01	**−3.08e + 02 ± 5.12e + 01**
Rt. Pathological	−2.95e + 01 ± 1.45e + 00	−2.98e + 01 ± 9.77e − 01	−1.45e + 02 ± 6.58e + 00	−2.99e + 01 ± 1.56e + 00	−1.80e + 02 ± 1.89e + 01	−1.60e + **02 ± 2.12e + 01**
Rt. Rastrigin	1.76e + 04 ± 1.42e + 02	1.74e + 04 ± 1.06e + 02	6.71e + 03 ± 2.04e + 02	1.66e + 04 ± 5.33e + 02	5.90e + 03 ± 4.07e + 02	**2.18e + 03 ± 6.85e + 02**
Rt. Rosenbrock	4.32e + 05 ± 9.18e + 03	4.28e + 05 ± 8.25e + 03	1.08e + 04 ± 7.14e + 02	3.18e + 05 ± 3.03e + 04	6.95e + 03 ± 9.68e + 02	**2.08e + 03 ± 4.17e + 02**
Rt. Schwefel	−7.09e + 04 ± 2.21e + 03	−7.24e + 04 ± 1.97e + 03	−2.48e + 05 ± 7.65e + 03	−7.18e + 04 ± 1.83e + 03	−2.48e + 05 ± 1.06e + 04	**−3.16e + 05 ± 2.25e + 04**
Rt. Sum of powers	1.14e + 52 ± 4.68e + 52	1.12e + 54 ± 7.06e + 54	**1.00e − 07 ± 4.88e − 07**	8.67e + 50 ± 4.47e + 51	2.02e − 02 ± 8.35e − 02	2.43e − 01 ± 1.70e + 00
Rt. Tirronen	−3.73e − 01 ± 1.49e − 02	−3.98e − 01 ± 1.24e − 02	−9.73e − 01 ± 4.86e − 02	−3.74e − 01 ± 1.89e − 02	−8.12e − 01 ± 6.75e − 02	**−1.04e + 00 ± 6.09e − 02**

Table 4 Results of the Wilcoxon rank-sum test for 500-dimension problems

	DE	DERSF	PDE	IBDDE	DDE
Ackley	+	+	+	+	+
Alpine	+	+	+	+	+
Ax.-par. hyp.-ell.	+	+	+	+	+
DeJong	+	+	+	+	+
DropWave	+	+	+	+	+
Griewangk	+	+	+	+	+
Michalewicz	+	+	+	+	+
Pathological	+	+	+	+	+
Rastrigin	+	+	+	+	+
Rosenbrock	+	+	+	+	=
Schwefel	+	+	+	+	+
Sum of powers	+	+	=	+	+
Tirronen	+	+	+	+	+
Rt. Ackley	+	+	+	+	+
Rt. Alpine	+	+	+	+	+
Rt. Ax.-par. hyp.-ell.	+	+	+	+	+
Rt. Griewangk	+	+	+	+	+
Rt. Michalewicz	+	+	+	+	+
Rt. Pathological	+	+	−	+	+
Rt. Rastrigin	+	+	+	+	+
Rt. Rosenbrock	+	+	+	+	+
Rt. Schwefel	+	+	=	+	+
Rt. Sum of powers	+	=	=	+	+
Rt. Tirronen	+	+	+	+	+

Table 5 Results of the Wilcoxon rank-sum test for 1,000-dimension problems

	DE	DERSF	PDE	IBDDE	DDE
Ackley	+	+	+	+	+
Alpine	+	+	+	+	+
Ax.-par. hyp.-ell.	+	+	+	+	+
DeJong	+	+	+	+	+
DropWave	+	+	+	+	+
Griewangk	+	+	+	+	+
Michalewicz	+	+	+	+	+
Pathological	+	+	+	+	+
Rastrigin	+	+	+	+	+
Rosenbrock	+	+	+	+	+
Schwefel	+	+	+	+	+
Sum of powers	+	+	=	+	=
Tirronen	+	+	+	+	+
Rt. Ackley	+	+	+	+	+
Rt. Alpine	+	+	+	+	+
Rt. Ax.-par. hyp.-ell.	+	+	+	+	+
Rt. Griewangk	+	+	+	+	+
Rt. Michalewicz	+	+	+	+	+
Rt. Pathological	+	+	+	+	−
Rt. Rastrigin	+	+	+	+	+
Rt. Rosenbrock	+	+	+	+	+
Rt. Schwefel	+	+	+	+	+
Rt. Sum of powers	=	=	=	=	=
Rt. Tirronen	+	+	+	+	+

DERSF, PDE introduced in Tasoulis et al. (2004), IBDDE given in Apolloni et al. (2008) and DDE employed in De Falco et al. (2007a, b, c). It must be remarked that all the algorithms chosen for comparison are modern distributed DE algorithms recently proposed in literature, and which are representative of the state-of-the-art in the field. Each algorithm has been run for 500,000 fitness evaluations in the case of $n = 500$ and for 1,000,000 fitness evaluations when $n = 1{,}000$. As much as 50 independent runs have been performed for each algorithm involved in this article.

The algorithms considered in this study have been run with the following parameter setting.

- DE has been run with $F = 0.7$ and $CR = 0.3$ in accordance with the suggestions given in Zielinski et al. (1857), Zielinski and Laur (2008). The population size has been set to $S_{\text{pop}} = 200$ for the 500-dimensional problems and to $S_{\text{pop}} = 400$ for the 1,000-dimensional ones.
- DESRF has been run with $CR = 0.3$. Analogous to DE, the population size has been set to $S_{\text{pop}} = 200$ for the 500-dimensional problems and to $S_{\text{pop}} = 400$ for the 1,000-dimensional ones. In DERSF the scale factor is randomized, at each generation, according to the rule $F = 0.5(1 + \text{rand}(0, 1))$.
- PDE has been run with populations of 200 or 400 individuals divided into 5 sub-populations of 40 or 80 individuals each, for the 500 and 1,000-dimensional problems, respectively. Despite (Tasoulis et al. 2004) showing better performance for the DE/best/1 mutation strategy in 30 and 50 dimensions, it has proven excessively exploitative and has led to premature convergence of the solutions when used on higher dimension problems. In order to perform a fair comparison, we have carried out an analysis on mutation strategies, leading to the choice of DE/rand/1 and setting the migration constant to $\phi = 0.2$. These settings proved to be the best choices in terms of algorithmic performance and have, thus, been chosen for the experiments described below.
- Similar to PDE, IBDDE has been run with populations of 200 or 400 individuals divided into 5 sub-populations of 40 or 80 individuals each, depending on the dimensionality of the test problems. The other parameters have been chosen according to the values in Apolloni et al. (2008): the sub-populations exchange one individual ($\rho = 1$) every 100 generations ($\gamma = 100$). ϕ_s and ϕ_r have been defined so as to

pseudo-randomly select an individual by means of a uniform distribution, and τ has been set to a unidirectional ring.

- For the 500-dimensional problems, the DDE has been run with a population of 200 individuals divided into 16 sub-populations of alternatively 12 or 13 individuals. In the case of the 1,000-dimensional problems, the population has been set to 400 individuals divided into 16 sub-populations of 25 individuals. Following the suggestions in De Falco et al. (2007b) the sub-populations have been organized into a 4×4 grid folded into a torus ($\mu = 4$). Migration occurred in each sub-population with only its best individual ($S_I = 1$) every $M_I = 5$ generations.
- Similar to PDE and IBDDE, FACPDE has been run with populations of 200 and 400 individuals divided into 5 sub-populations of 40 and 80 individuals each for the 500 and 1,000-dimensional cases, respectively. The migration constant ϕ has been set equal to 0.2 and the constant α in formula (6) has been set equal to 0.1.

It is worthwhile commenting on choice of the population sizes $S_{\text{pop}} = 200$ and $S_{\text{pop}} = 400$. Although in Storn and Price (1997) it is suggested that the DE population size be set equal to about 10 times the dimensionality of the problem, this indication is not confirmed by a recent study in Neri and Tirronen (2008), where it is shown that a population size lower than the dimensionality of the problem can be optimal in many cases.

Table 2 shows the average of the final results detected by each algorithm $\pm$ the standard deviations, with the 500 dimension case. Table 3 shows the results for the 1,000-dimension case. The best results are highlighted in boldface.

Results in Tables 2 and 3 show that the sequential DE algorithms are outperformed by the distributed algorithms. This result confirms that a structured population can enhance performance of the DE (see, Alba and Tomassini 2002). In addition, FACPDE seems to have a very good performance with the test problems considered in this study since it detects those solutions with best performance for 23 and 21 test problems out of the 24 considered in 500 and 1,000 dimensions, respectively. It must be remarked that in the cases, where FACPDE does not detect the solutions with best performance, the algorithm still detects, in any case, competitive solutions; these solutions usually have, on average, the second best performance. In this sense, FACPDE seems to be a high-quality algorithm for various test problems.

To prove statistical significance of the results, the Wilcoxon Rank-Sum test has also been applied according to the description given in Wilcoxon (1945), where the confidence level has been fixed to 0.95. Tables 4 and 5 show results of the test. A "+" indicates the case in which FACPDE statistically outperforms, for the corresponding test problem, the algorithm mentioned in that column; a "=" indicates that a pairwise comparison leads to success of the Wilcoxon Rank-Sum test, i.e., the two algorithms have the same performance; a "−" indicates that FACPDE is outperformed.

In the case of 500-dimensional problems, the Wilcoxon test results in Table 4 shows that FACPDE, out of the 120 pairwise comparisons performed, loses out only in one case, obtains the same results in four cases, and wins in 91 cases. Thus, FACPDE comes out behind in only 0.83% of the comparisons and comes out ahead in 95.0% of the considered comparisons. In 1,000 dimensions, the Wilcoxon test results displayed in Table 5 show that FACPDE, out of the 120 pairwise comparisons performed, loses in one case, obtains the same results in seven cases, and wins in 112 cases. Thus, FACPDE loses only 0.83% of the comparisons and is shown to be superior in 93.3% of the considered comparisons. The statistical test carried out confirms that FACPDE has a very good performance in regard to the studied test problems. In addition, the comparison between PDE and FACPDE shows that the proposed scale factor inheritance mechanism seems to be very beneficial.

To strengthen the statistical significance of the results, the Holm procedure (Holm 1979) has been applied by following the description in Garcfa et al. (2008). The Holm procedure consists of the following. Considering the results in Tables 2 and 3, the six algorithms under analysis have been ranked on the basis of their average performance

Table 6 Results of the Holm procedure for 500-dimensional problems (FACPDE is the control algorithm)

$N_A - j$	Optimizer	z_{N_A-j}	p_{N_A-j}	$\delta/(N_A - j)$	Hypothesis
5	DE	−8.72e + 00	1.41e − 18	1.00e − 02	Rejected
4	DERSF	−7.33e + 00	1.16e − 13	1.25e − 02	Rejected
3	IBDDE	−5.63e + 00	8.90e − 09	1.67e − 02	Rejected
2	DDE	−3.09e + 00	1.01e − 03	2.50e − 02	Rejected
1	PDE	−2.08e + 00	1.86e − 02	5.00e − 02	Rejected

Table 7 Results of the Holm procedure for 1,000-dimensional problems (FACPDE is the control algorithm)

$N_A - j$	Optimizer	z_{N_A-j}	p_{N_A-j}	$\delta/(N_A - j)$	Hypothesis
5	DE	−8.72e + 00	1.41e − 18	1.00e − 02	Rejected
4	DERSF	−6.71e + 00	9.59e − 12	1.25e − 02	Rejected
3	IBDDE	−5.86e + 00	2.27e − 09	1.67e − 02	Rejected
2	PDE	−2.62e + 00	4.36e − 03	2.50e − 02	Rejected
1	DDE	−2.01e + 00	2.24e − 02	5.00e − 02	Rejected

Table 8 Results of the Q-test for 500 dimensions problems

	THR	DE	DERSF	PDE	IBDDE	DDE	FACPDE
Ackley	2.15e − 01	∞	∞	4.58e + 03	∞	4.65e + 03	**7.37e + 02**
Alpine	9.89e + 01	∞	∞	5.82e + 03	∞	2.54e + 04	**1.79e + 03**
Ax.-par. hyp.-ell.	4.99e + 04	∞	∞	2.14e + 03	∞	1.20e + 03	**4.43e + 02**
DeJong	2.02e + 02	∞	∞	2.36e + 03	∞	1.35e + 03	**4.96e + 02**
DropWave	−4.81e − 03	∞	∞	9.31e + 04	∞	∞	**3.50e + 03**
Griewangk	1.19e + 03	∞	∞	2.34e + 03	∞	1.40e + 03	**5.29e + 02**
Michalewicz	−3.39e + 02	∞	∞	∞	∞	∞	**3.38e + 03**
Pathological	−3.37e + 02	∞	∞	8.43e + 03	1.65e + 04	∞	**1.22e + 03**
Rastrigin	1.38e + 03	∞	∞	∞	∞	∞	**1.23e + 03**
Rosenbrock	1.17e + 04	∞	∞	1.57e + 03	∞	7.23e + 02	**4.71e + 02**
Schwefel	−1.46e + 05	∞	∞	∞	∞	∞	**1.17e + 03**
Sum of powers	1.24e − 01	∞	∞	2.32e + 02	3.52e + 04	**7.97e + 01**	1.81e + 02
Tirronen	−1.55e + 00	∞	∞	5.04e + 03	∞	∞	**2.04e + 03**
Rt. Ackley	2.55e − 01	∞	∞	4.98e + 03	∞	4.59e + 03	**8.62e + 02**
Rt. Alpine	1.31e + 02	∞	∞	4.35e + 03	∞	8.47e + 03	**1.50e + 03**
Rt. Ax.-par. hyp.-ell.	4.97e + 04	∞	∞	2.04e + 03	∞	1.04e + 03	**4.84e + 02**
Rt. Griewangk	1.19e + 03	∞	∞	2.17e + 03	∞	1.18e + 03	**4.91e + 02**
Rt. Michalewicz	−1.83e + 02	∞	∞	2.27e + 04	∞	∞	**3.87e + 03**
Rt. Pathological	−1.15e + 02	∞	∞	**5.24e + 03**	∞	3.16e + 04	1.01e + 04
Rt. Rastrigin	1.49e + 03	∞	∞	∞	∞	∞	**1.76e + 03**
Rt. Rosenbrock	1.17e + 04	∞	∞	1.41e + 03	∞	6.18e + 02	**4.70e + 02**
Rt. Schwefel	−1.60e + 05	∞	∞	4.70e + 03	∞	∞	**3.39e + 03**
Rt. Sum of powers	6.33e + 20	2.27e + 01	1.31e + 01	5.24e + 00	1.44e + 01	4.92e + 00	**4.12e + 00**
Rt. Tirronen	−1.21e + 00	∞	∞	5.21e + 03	∞	∞	**3.72e + 03**

calculated over the 24 test problems. Thus, or in other words, a score R_i for $i = 1, \ldots, N_A$ (where N_A is the number of algorithms under analysis, $N_A = 6$ in our case) has been assigned. With the calculated R_i values, the FACPDE has been taken as a reference algorithm. Indicating with R_0 the rank of FACPDE, and with R_j for $j = 1, \ldots, N_A - 1$ the rank of one of the remaining five algorithms, the values z_j have been calculated as

$$z_j = \frac{R_j - R_0}{\sqrt{\frac{N_A(N_A+1)}{6N_{\mathrm{TP}}}}}$$

where N_{TP} is the number of test problems in consideration ($N_{\mathrm{TP}} = 24$ in our case). By means of the z_j values, the corresponding cumulative normal distribution values p_j have been calculated. These p_j values are then compared with the corresponding $\delta/(N_A - j)$ where δ is the level of confidence, set to 0.05 in our case. Tables 6 and 7 display z_j values, p_j values, and corresponding $\delta/(N_A - j)$. The values of z_j and p_j are expressed in terms of z_{N_A-j} and p_{N_A-j} for $j = 1, \ldots, N_A - 1$. Moreover, it is indicated whether the null-hypothesis (that the two algorithms have indistinguishable performances) is "Rejected" i.e., the FACPDE statistically outperforms the algorithm under consideration, or "Accepted" if the distribution of values can be considered the same (there is no outperformance).

The Holm procedure confirms that the FACPDE displays a significantly better performance with respect to the other algorithms in this study for both 500- and 1,000-dimensional cases.

To carry out a numerical comparison of the convergence speed performance, for each test problem, the average final fitness value returned by the best performing algorithm G has been considered. Subsequently, the average fitness value at the beginning of the optimization process J has also been computed. The threshold value $\mathrm{THR} = J - 0.95(J - G)$ has then been calculated. The value THR represents 95% of the decay in the fitness value of the algorithm with the best performance. If an algorithm succeeds during a certain run to reach the value THR, the run is said to be successful. For each test problem, the average amount of fitness evaluations $\bar{ne}$ required, for each algorithm, to reach THR has been computed. Subsequently, the Q-test (Q stands for Quality) described in Feoktistov (2006) has been applied. For each test problem and each algorithm, the Q measure is computed as:

Table 9 Results of the Q-test for 1,000-dimension problems

	THR	DE	DERSF	PDE	IBDDE	DDE	FACPDE
Ackley	2.10e − 01	∞	∞	∞	∞	∞	**2.14e + 03**
Alpine	2.41e + 02	∞	∞	∞	∞	∞	**2.81e + 03**
Ax.-par. hyp.-ell.	2.02e + 05	∞	∞	9.45e + 03	∞	6.10e + 03	**1.43e + 03**
DeJong	4.11e + 02	∞	∞	∞	∞	7.61e + 03	**1.40e + 03**
DropWave	−1.98e − 03	∞	∞	∞	∞	∞	**1.13e + 04**
Griewangk	2.43e + 03	∞	∞	1.22e + 05	∞	7.43e + 03	**1.45e + 03**
Michalewicz	−5.79e + 02	∞	∞	∞	∞	∞	**7.94e + 03**
Pathological	−6.81e + 02	∞	∞	∞	∞	∞	**4.39e + 03**
Rastrigin	2.78e + 03	∞	∞	∞	∞	∞	**2.59e + 03**
Rosenbrock	2.38e + 04	∞	∞	6.57e + 03	∞	3.37e + 03	**1.54e + 03**
Schwefel	−3.09e + 05	∞	∞	∞	∞	∞	**2.57e + 03**
Sum of powers	1.23e − 01	∞	∞	6.84e + 02	∞	**1.90e + 02**	4.86e + 02
Tirronen	−1.51e + 00	∞	∞	∞	∞	∞	**5.03e + 03**
Rt. Ackley	2.52e − 01	∞	∞	∞	∞	∞	**2.60e + 03**
Rt. Alpine	2.52e + 02	∞	∞	∞	∞	∞	**3.68e + 03**
Rt. Ax.-par. hyp.-ell.	2.02e + 05	∞	∞	8.78e + 03	∞	5.15e + 03	**1.49e + 03**
Rt. Griewangk	2.40e + 03	∞	∞	1.01e + 04	∞	6.09e + 03	**1.37e + 03**
Rt. Michalewicz	−2.94e + 02	∞	∞	∞	∞	∞	**8.87e + 03**
Rt. Pathological	−1.72e + 02	∞	∞	∞	∞	**7.56e + 03**	3.30e + 04
Rt. Rastrigin	2.96e + 03	∞	∞	∞	∞	∞	**4.03e + 03**
Rt. Rosenbrock	2.43e + 04	∞	∞	5.65e + 03	∞	2.52e + 03	**1.30e + 03**
Rt. Schwefel	−3.01e + 05	∞	∞	∞	∞	∞	**6.91e + 03**
Rt. Sum of powers	2.08e + 56	**8.00e + 00**	1.30e + 01	8.00e + 00	8.00e + 00	8.00e + 00	9.28e + 00
Rt. Tirronen	−9.88e − 01	∞	∞	2.10e + 04	∞	∞	**9.92e + 03**

$$Q = \frac{\bar{ne}}{\text{Rob}} \tag{7}$$

where the robustness Rob is the percentage of successful runs. It is clear that, for each test problem, the smallest value equals the best performance in terms of convergence speed. The value "∞" means that Rob = 0, i.e., the algorithm never reached the THR.

Tables 8 and 9 show the Q values for 500-dimensional problems and 1,000-dimensional problems respectively, as well as the associated THR values. The best results are highlighted in boldface.

The Q-test results, listed in Tables 8 and 9, show that in 500 and 1,000-dimensional cases, the proposed FACPDE algorithm has the best performance in terms of convergence speed in 22 and 21 cases out of the 24 considered, respectively. Most importantly, the FACPDE algorithm, throughout all considered test problems, is never characterized by an ∞ value of Q-measure. This fact shows that the proposed algorithm is always competitive with the other algorithms in the benchmark and is never relevantly outperformed. In summary, the algorithmic behavior of FACPDE, thanks to its scale factor inheritance mechanism, is extremely promising in terms of algorithmic robustness.

Figure 9 shows average performance trends of the five considered algorithms over a selection of the test problems listed in Table 1 in 500 dimensions.

Figure 9c–k show that in 500 dimensions, the serial DE algorithms improve only marginally. IBDDE has a slightly better performance than DE and DERSF (see, for example, Fig. 9i), but is still not competitive compared to PDE, DDE, and FACPDE for the high dimensional problems considered in this study. In Fig. 9d–k, DDE improves its solutions very quickly in the beginning, before ceasing to make any significant improvement. The transition occurs around 100,000 fitness evaluations in all cases except in Fig. 9h, where it occurs around 200,000 fitness evaluations. PDE's improvement rate is slower than DDE's, but contrary to the latter, it continuously improves its solutions and in all but one of the examples listed above, finds a better solution than DDE; Figure 9c is the exception, but PDE's solution is still very close to DDE's. Finally, FACPDE's improvement rate at the onset is steeper than DDE's in Fig. 9c, f, and j, similar in 9i and k, and softer in Fig. 9d, e, g, and h. In all cases, however, FACPDE does not show symptoms of premature convergence as DDE does. Moreover, FACPDE's

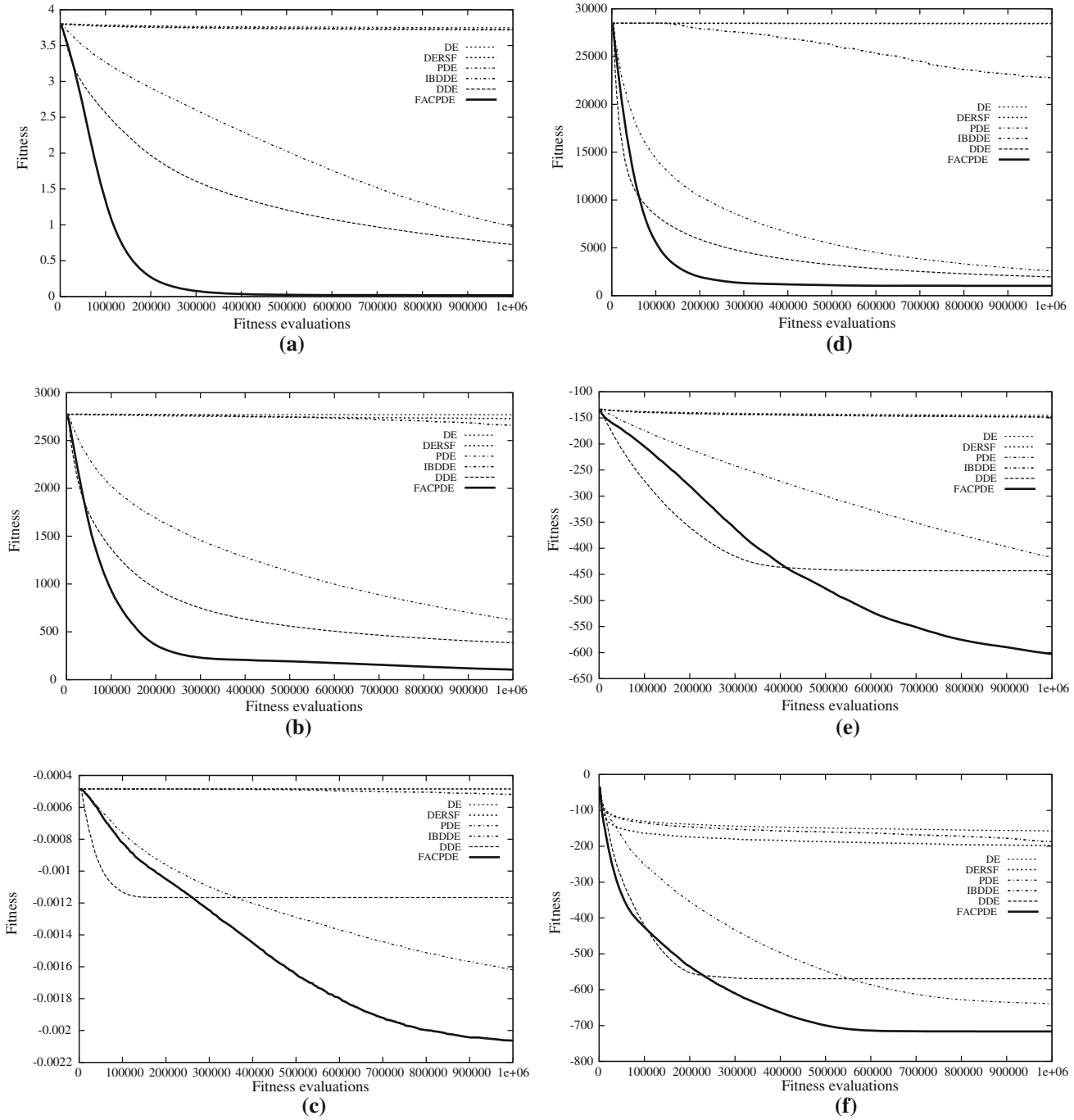
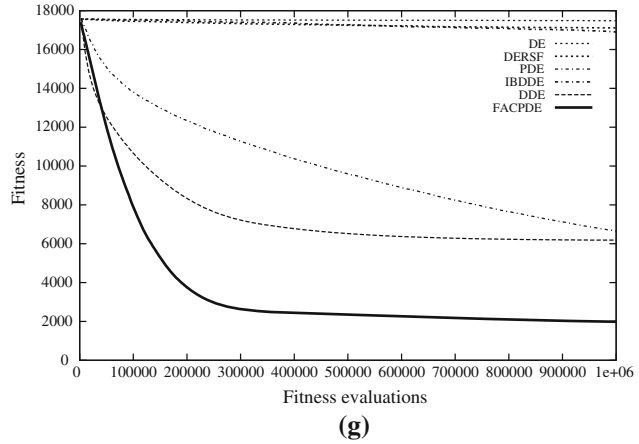


Fig. 10 Performance trends in 1,000 dimensions. **a** Ackley, **b** Alpine, **c** DropWave, **d** Griewangk, **e** Michalewicz, **f** Pathological, **g** Rastrigin, **h** Rotated Ackley, **i** Rotated Michalewicz

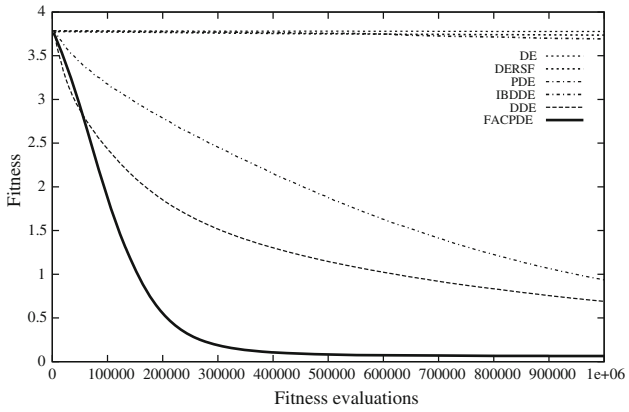
final solutions are, in all above examples except 9h, better than PDE's.

Figure 10 shows average performance trends of the five considered algorithms over a selection of the test problems listed in Table 1 in 1,000 dimensions.

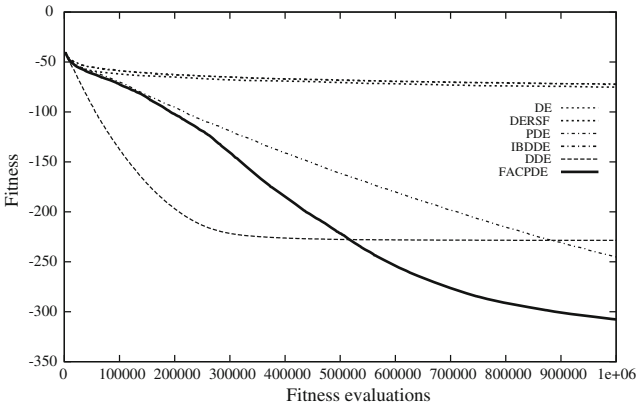
Qualitative results represented in Fig. 10 confirm the findings in 500 dimensions. For large scale problems, the sequential DE seems to suffer from the curse of dimensionality. The IBDDE algorithm succeeds at improving upon the performance of sequential DE and DERSF. The other algorithms demonstrate a better behavior for the high dimensional problems. The DDE has good convergence speed performance during early stages of the evolution since it often manages to detect high quality solutions



(g)



(h)



(i)

Fig. 10 continued

during the initial generations. However, the DDE tends to stop improving upon previous achievements quite quickly, see, e.g., Fig. 10c and i. It is interesting to observe that in 1,000 dimensions PDE is slower at detecting high-quality solutions with respect to the 500-dimensional case e.g., compare the behavior with Michalewicz function in Figs. 9e and 10e. Thus, more often than in the 500-dimensional case, the DDE outperforms PDE in high

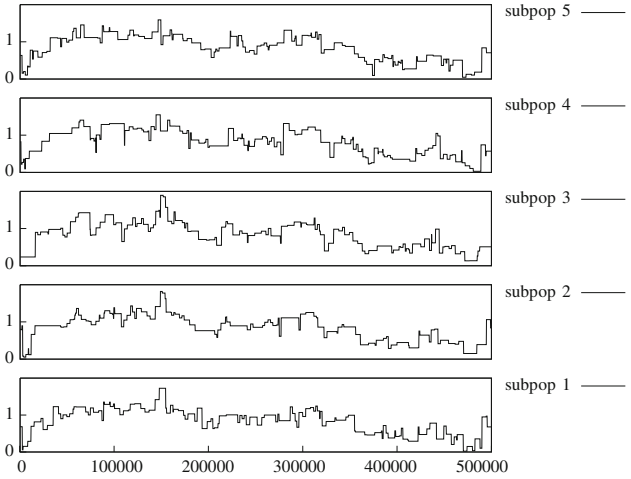


Fig. 11 Example of the scale factor behavior over the five sub-populations

dimensions. The FACPDE algorithm, on the contrary, seems to be very efficient and outperforms, in most cases, both PDE and DDE. In addition, the gap between FACPDE and the second best algorithm performance seems to be systematically larger for the 1,000-dimensional case with respect to the results in 500 dimensions, see e.g., Fig. 10a, c, e, and h. In other words, the scale factor inheritance mechanism seems to be very beneficial in order to tackle large scale problems.

To better understand the working principle of the scale factor inheritance, Fig. 11 has been included. Figure 11 shows an example, based on a single run, of the trend that the absolute value of F takes during the FACPDE evolution for the five distributed sub-populations.

It can be noticed from Fig. 11 how the scale factor inheritance mechanism works. For example, at around 150,000 a sudden increase in the scale factor value occurs throughout all the sub-populations. This phenomenon is clearly visible in sub-populations 1, 2, and 3 and proves how promising scale factor values are propagated within the ring of sub-populations. The most important finding in Fig. 11 is related to the variation of the scale factor values. It can be observed that although the variation of F is, in principle, unbounded, there is no divergence in the trends. On the contrary, the scale factors never take a value greater than 2. In addition, the scale factor trends do not converge to a constant value. On the contrary, the trends continue to oscillate throughout the entire evolution. This fact can be seen as confirmation that there is no optimal scale factor for a given problem, but that a dynamic mechanism is required in order to satisfy the necessities of the evolution. Finally, since the evolution is biased by randomization, according to our interpretation, the scale factor control should also contain some randomization, as empirically shown in many

Table 10 Experimental Results for the IEEE CEC08 benchmark

	$n = 100$	$n = 500$	$n = 1{,}000$
F1	3.94e + 01 ± 1.41e + 02	2.45e + 02 ± 2.42e + 02	3.76e + 02 ± 3.54e + 02
F2	8.97e + 01 ± 7.68e + 00	1.13e + 02 ± 7.20e + 00	1.28e + 02 ± 9.92e + 00
F3	8.49e + 06 ± 3.05e + 07	2.39e + 08 ± 3.40e + 08	3.35e + 08 ± 3.09e + 08
F4	1.89e + 02 ± 7.18e + 01	8.31e + 02 ± 1.66e + 02	1.65e + 03 ± 3.32e + 02
F5	8.64e − 01 ± 1.17e + 00	4.18e + 00 ± 2.89e + 00	1.36e + 01 ± 4.96e + 01
F6	1.01e + 01 ± 3.60e + 00	2.83e + 00 ± 2.24e + 00	4.24e + 00 ± 4.41e +00
F7	−1.32e + 03 ± 5.09e + 01	−6.31e + 03 ± 2.03e + 02	−1.19e + 04 ± 9.28e + 02

papers on DE (e.g., Das et al. 2005, 2009; Brest et al. 2006; Qin et al. 2009).

5.1 Experimental results for a large scale optimization benchmark

The proposed FACPDE has been finally tested on the large scale benchmark settled for the 2008 IEEE Congress on Evolutionary Computation (IEEE CEC08) described in Tang et al. (2007b). Following the suggestions given in Tang et al. (2007b), each function has been considered in 100, 500, and 1,000 dimensions. For each problem, FACPDE has been has been run 25 times (25 independent runs) and the computational budget has been fixed equal to $5,000 \times n$. Regarding the parameter setting, the FACPDE has been in 500 and 1,000 dimensions with the same setting mentioned above. Following the same rate $\frac{n}{S_{\text{pop}}}$, in 100 dimensions, the population size S_{pop} has been set equal to 40. The final results, expressed in terms of fitness difference between the detected value and the actual minimum (± related standard deviation) are given for the seven test problems contained in Tang et al. (2007b) for the three levels of dimensionality in Table 10.

Although the results displayed in Table 10 are outperformed in several cases by the algorithms which took part in the CEC08 competition, FACPDE is still competitive in most cases. This fact is, in our opinion, very relevant since FACPDE, unlike most of those algorithms, does not employ local search components and did not undergo a parameter setting in order to have a high-quality performance for these specific test problems.

6 Conclusion

This paper proposes an adaptive mechanism for the scale factor in distributed differential evolution schemes. The proposed mechanism, namely, scale factor inheritance, consists of the perturbation, by means of a random number and migration, of promising scale factor values throughout sub-populations arranged according to a ring topology. This mechanism is integrated within a distributed differential evolution which employs migration of those individuals demonstrating the best performance.

The resulting algorithm has been tested on a broad and various set of optimization problems and then compared with two sequential differential evolution algorithms and three distributed differential evolution schemes, recently proposed in literature. Numerical results show that the proposed approach is very promising and that the resulting algorithm displays excellent performance in terms of detected final solutions, convergence speed, and robustness for all test problems and comparisons considered in this study.

On the basis of the obtained results and an analysis of the algorithmic working principles, some additional conclusions have been drawn. In distributed differential evolution a cooperative/competitive adaptation of the scale factor is beneficial to algorithmic performance and, more generally, the employment of multiple scale factors, updating of which can greatly improve performance of the distributed algorithm. In addition, a constant scale factor value is inadequate since it restricts the amount of search moves in differential evolution (both sequential and distributed). In other words, for a given problem there is no optimal scale factor since the optimal setting varies during the various stages of evolution. An optimal setting dynamically varies with distribution of the solutions within the decision space during evolution. Although an understanding of a proper control dynamic of the scale factor variation is not yet complete, according to our interpretation, it does not follow a progressive increase or decrease, but takes on an oscillatory behavior. Finally, since the evolution is affected by random events, mainly in the selection of individuals composing the moving vectors within the mutation operation, a certain randomization of the scale factor seems to be beneficial for a successful enhancement of the differential evolution performance.

Acknowledgments This research is supported by the Academy of Finland, Akatemiatutkija 00853, Algorithmic Design Issues in Memetic Computing.

References

Alba E, Tomassini M (2002) Parallelism and evolutionary algorithms. IEEE Trans Evol Comput 6(5):443–462

Ali MM, Törn A (2004) Population set-based global optimization algorithms: some modifications and numerical studies. Comput Oper Res 31(10):1703–1725

Apolloni J, Leguizamón G, García-Nieto J, Alba E (2008) Island based distributed differential evolution: an experimental study on hybrid testbeds. In: Proceedings of the IEEE international conference on hybrid intelligent systems, pp 696–701

Brest J, Maučec MS (2008) Population size reduction for the differential evolution algorithm. Appl Intell 29(3):228–247

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

Caponio A, Cascella GL, Neri F, Salvatore N, Sumner M (2007) A fast adaptive memetic algorithm for on-line and off-line control design of pmsm drives. IEEE Trans Syst Man Cybern B 37(1):28–41

Caponio A, Neri F, Tirronen V (2009) Super-fit control adaptation in memetic differential evolution frameworks. Soft Comput Fusion Found Methodol Appl 13(8):811–831

Chakraborty UK (ed) (2008) Advances in differential evolution, vol 143 of Studies in computational intelligence. Springer

Chiou J-P, Chang C-F, Su C-T (2004) Ant direction hybrid differential evolution for solving large capacitor placement problems. IEEE Trans Power Syst 19(4):1794–1800

Das S, Konar A, Chakraborty UK (2005) Two improved differential evolution schemes for faster global search. In: Proceedings of the 2005 conference on genetic and evolutionary computation. ACM, pp 991–998

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

De Falco I, Della Cioppa A, Maisto D, Scafuri U, Tarantino E (2007a) Satellite image registration by distributed differential evolution. In: Applications of evolutionary computing, vol. 4448 of Lectures notes in computer science. Springer, pp 251–260

De Falco I, Maisto D, Scafuri U, Tarantino E, Della Cioppa A (2007b) 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

De Falco I, Scafuri U, Tarantino E, Della Cioppa A (2007c) A distributed differential evolution approach for mapping in a grid environment. In: Proceedings of the IEEE euromicro international conference on parallel, distributed and network-based processing, pp 442–449

Fan H-Y, Lampinen J (2003) A trigonometric mutation operation to differential evolution. J Glob Optim 27(1):105–129

Feoktistov V (2006) Differential evolution in search of solutions. Springer

Gämperle R, Müller SD, Koumoutsakos P (2002) A parameter study for differential evolution. In: Proceedings of the conference in neural networks and applications (NNA), fuzzy sets and fuzzy systems (FSFS) and evolutionary computation (EC). WSEAS, pp 293–298

Garcfa S, Fernández A, Luengo J, Herrera F (2008) A study of statistical techniques and performance measures for genetics-based machine learning: accuracy and interpretability. Soft Comput 13(10):959–977

Holm S (1979) A simple sequentially rejective multiple test procedure. Scand J Stat 6(2):65–70

Joshi R, Sanderson AC (1999) Minimal representation multisensor fusion using differential evolution. IEEE Trans Syst Man Cybern A 29(1):63–76

Kozlov KN, Samsonov AM (2006) New migration scheme for parallel differential evolution. In: Proceedings of the international conference on bioinformatics of genome regulation and structure, pp 141–144

Kwedlo W, Bandurski K (2006) A parallel differential evolution algorithm. In: Proceedings of the IEEE international symposium on parallel computing in electrical engineering, pp 319–324

Lampinen J (1999) Differential evolution—new naturally parallel approach for engineering design optimization. In: Topping BH (ed) Developments in computational mechanics with high performance computing. Civil-Comp Press, pp 217–228

Lampinen J, Zelinka I (2000) On stagnation of the differential evolution algorithm. In: Oŝmera P (ed) Proceedings of 6th international Mendel conference on soft computing, pp 76–83

Liu J, Lampinen J (2002) On setting the control parameter of the differential evolution algorithm. In: Proceedings of the 8th international Mendel conference on soft computing, pp 11–18

Liu J, Lampinen J (2005) A fuzzy adaptive differential evolution algorithm. Soft Comput Fusion Found Methodol Appl 9:448–462

Mallipeddi R, Suganthan PN (2008) Empirical study on the effect of population size on differential evolution algorithm. In: Proceedings of the IEEE congress on evolutionary computation, pp 3663–3670

Neri F, Tirronen V (2008) On memetic differential evolution frameworks: a study of advantages and limitations in hybridization. In: Proceedings of the IEEE World congress on computational intelligence, pp 2135–2142

Neri F, Tirronen V (2009) Scale factor local search in differential evolution. Memet Comput J 1(2):153–171

Neri F, Toivanen J, Cascella GL, Ong Y-S (2007) An adaptive multimeme algorithm for designing HIV multidrug therapies. IEEE/ACM Trans Comput Biol Bioinform 4(2):264–278

Nipteni MS, Valakos I, Nikolos I (2006) An asynchronous parallel differential evolution algorithm. In: Proceedings of the ERCOFTAC conference on design optimisation: methods and application

Noman N, Iba H (2008) Accelerating differential evolution using an adaptive local search. IEEE Trans Evol Comput 12(1):107–125

Omran MG, Salman A, Engelbrecht AP (2005) Self-adaptive differential evolution. In: Computational intelligence and security, vol 3801 of Lecture notes in computer science. Springer, pp 192–199

Ong Y-S, Keane AJ (2004) Meta-lamarkian learning in memetic algorithms. IEEE Trans Evol Comput 8(2):99–110

Pavlidis NG, Tasoulis DK, Plagianakos VP, Nikiforidis G, Vrahatis MN (2005) Spiking neural network training using evolutionary algorithms. In: Proceedings of the IEEE international joint conference on neural networks, pp 2190–2194

Price KV (1999) Mechanical engineering design optimization by differential evolution. In: Corne D, Dorigo M, Glover F (eds) New ideas in optimization. McGraw-Hill, pp 293–298

Price KV, Storn RM (1997) Differential evolution: a simple evolution strategy for fast optimization. Dr. Dobb's J Softw Tools 22(4):18–24

Price KV, Storn RM, Lampinen J (2005) Differential evolution: a practical approach to global optimization. Springer

Qin AK, Suganthan PN (2005) Self-adaptive differential evolution algorithm for numerical optimization. In: Proceedings of the IEEE congress on evolutionary computation, vol 2, pp 1785–1791

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

Rönkkönen J, Lampinen J (2003) On using normally distributed mutation step length for the differential evolution algorithm. In: Matousek R, Osmera P (eds) Proceedings of ninth international Mendel conference on soft computing, pp 11–18

Rönkkönen J, Kukkonen S, Price KV (2005) Real-parameter optimization with differential evolution. In: Proceedings of IEEE international conference on evolutionary computation, vol 1, pp 506–513

Salman A, Engelbrecht AP, Omran MG (2007) Empirical analysis of self-adaptive differential evolution. Eur J Oper Res 183(2):785–804

Salomon M, Perrin G-R, Heitz F, Armspach J-P (2005) Parallel differential evolution: application to 3-d medical image registration. In: Price KV, Storn RM, Lampinen JA (eds) Differential evolution—a practical approach to global optimization natural computing series. Springer, Chap 7, pp 353–411

Soliman OS, Bui LT (2008) A self-adaptive strategy for controlling parameters in differential evolution. In: Proceedings of the IEEE congress on evolutionary computation, pp 2837–2842

Soliman OS, Bui LT, Abbass HA (2007) The effect of a stochastic step length on the performance of the differential evolution algorithm. In: Proceedings of the IEEE congress on evolutionary computation, pp 2850–2857

Storn RM (2005) Designing nonstandard filters with differential evolution. IEEE Signal Process Mag 22(1):103–106

Storn RM, Price KV (1995) Differential evolution—a simple and efficient adaptive scheme for global optimization over continuous spaces. Tech Rep TR-95-012, ICSI

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

Su C-T, Lee C-S (2003) Network reconfiguration of distribution systems using improved mixed-integer hybrid differential evolution.IEEE Trans Power Deliv 18(3):1022–1027

Tang J, Lim MH, Ong Y-S (2007a) Diversity-adaptive parallel memetic algorithm for solving large scale combinatorial optimization problems. Soft Comput Fusion Found Methodol Appl 11(9):873–888

Tang K, Yao X, Suganthan PN, MacNish C, Chen YP, Chen CM, Yang Z (2007b) 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

Tasoulis DK, Pavlidis NG, Plagianakos VP, Vrahatis MN (2004) Parallel differential evolution. In: Proceedings of the IEEE congress on evolutionary computation, pp 2023–2029

Teng NS, Teo J, Hijazi MHA (2009) Self-adaptive population sizing for a tune-free differential evolution. Soft Comput Fusion Found Methodol Appl 13(7):709–724

Teo J (2005) Differential evolution with self-adaptive populations. In: Knowledge-based intelligent information and engineering systems, vol 3681 of Lecture notes in computer science. Springer, pp 1284–1290

Teo J (2006) Exploring dynamic self-adaptive populations in differential evolution. Soft Comput Fusion Found Methodol Appl 10(8):673–686

Tirronen V, Neri F (2009) Differential evolution with fitness diversity self-adaptation. In: Chiong R (ed) Nature-inspired algorithms for optimisation vol. 193 of Studies in computational intelligence. Springer, pp 199–234

Tirronen V, Neri F, Kärkkäinen T, Majava K, Rossi T (2008) An enhanced memetic differential evolution in filter design for defect detection in paper production. Evol Comput 16:529–555

Wang F-S, Jang H-J (2000) Parameter estimation of a bioreaction model by hybrid differential evolution. In: Proceedings of the IEEE congress on evolutionary computation, vol 1, pp 410–417

Wilcoxon F (1945) Individual comparisons by ranking methods. Biom Bull 1(6):80–83

Wolpert D, Macready W (1997) No free lunch theorems for optimization. IEEE Trans Evol Comput 1(1):67–82

Zaharie D (2002) Parameter adaptation in differential evolution by controlling the population diversity. In: Petcu D et al (ed) Proceedings of the international workshop on symbolic and numeric algorithms for scientific computing, pp 385–397

Zaharie D (2003) Control of population diversity and adaptation in differential evolution algorithms. In: Matousek D, Osmera P (eds) Proceedings of Mendel international conference on soft computing, pp 41–46

Zaharie D (2004) A multipopulation differential evolution algorithm for multimodal optimization. In: Matousek R, Osmera P (eds) Proceedings of Mendel international conference on soft computing, pp 17–22

Zaharie D, Petcu D (2003) Parallel implementation of multipopulation differential evolution. In: Proceedings of the NATO advanced research workshop on concurrent information processing and computing. IOS Press, pp 223–232

Zhenyu G, Bo C, Min Y, Binggang C (2006) Self-adaptive chaos differential evolution. In: Advances in natural computation, vol 4221 of Lecture notes in computer science. Springer, pp 972–975

Zielinski K, Laur R (2008) Stopping criteria for differential evolution in constrained single-objective optimization. In: Chakraborty UK (ed) Advances in differential evolution, vol 143 of Studies in computational intelligence. Springer, pp 111–138

Zielinski K, Weitkemper P, Laur R, Kammeyer K-D (2006) Parameter study for differential evolution using a power allocation problem including interference cancellation. In: Proceedings of the IEEE congress on evolutionary computation. pp 1857–1864