Available online at www.sciencedirect.com
ScienceDirect
European Journal of Operational Research 181 (2007) 1137–1147
EUROPEAN JOURNAL OF OPERATIONAL RESEARCH
www.elsevier.com/locate/ejor

Differential evolution with preferential crossover

M.M. Ali *

School of Computational and Applied Mathematics, Witwatersrand University, Wits-2050, Johannesburg, South Africa

Received 27 October 2004; accepted 16 June 2005
Available online 18 April 2006

Abstract

We study the mutation operation of the differential evolution algorithm. In particular, we study the effect of the scaling parameter of the differential vector in mutation. We derive the probability density function of points generated by mutation and thereby identify some drawbacks of the scaling parameter. We also visualize the drawbacks using simulation. We then propose a crossover rule, called the preferential crossover rule, to reduce the drawbacks. The preferential crossover rule uses points from an auxiliary population set. We also introduce a variable scaling parameter in mutation. Motivations for these changes are provided. A numerical study is carried out using 50 test problems, many of which are inspired by practical applications. Numerical results suggest that the proposed modification reduces the number of function evaluations and cpu time considerably.

Keywords: Global optimization; Differential evolution; Mutation; Crossover; Auxiliary population set; Continuous variable; Probability density function

1. Introduction

The global optimization problem in this paper follows the form:

$$\text{minimize } f(x) \quad \text{subject to } x \in \Omega, \tag{1}$$

where x is a continuous variable vector with domain $\Omega \subset \mathbb{R}^n$, and $f(x)$:$\Omega \mapsto \mathbb{R}$ is a continuous real-valued function. The domain Ω is defined by specifying upper (u^j) and lower (l^j) limits of each component j. We denote the global optimal solution by x^*, with its corresponding global optimal function value $f(x^*)$ or f^* for a short hand notation. The paper is concerned with the differential evolution (DE) algorithm [1]. The DE algorithm is a population algorithm [2] and is purely heuristic. All population direct search methods use a population set S. The initial set

$$S = \{x_1, x_2, \ldots, x_N\} \tag{2}$$

consists of N random points in Ω. A contraction process is then used to drive these points to the vicinity of the global minimizer. The contraction process involves replacing bad point(s) in S with better point(s), per generation. In particular, DE attempts to replace all points in S by new points at each generation. It progresses in an epoch or era base. During each epoch, N new function values are evaluated on N trial points. Trial points are generated using mutation and crossover. A brief description of DE is given below.

* Tel.: +27 11 717 6139; fax: +27 11 717 6149.
E-mail address: mali@cam.wits.ac.za

0377-2217/$ - see front matter
doi:10.1016/j.ejor.2005.06.077

2. A brief description of DE

The DE algorithm attempts to replace each point in S with a new better point. Therefore, in each generation, N competitions are held to determine the members of S for the next generation. The ith ($i = 1, 2, \ldots, N$) competition is held to replace x_i in S. Considering x_i as the target point, a trial point y_i is found from two points (parents), the point x_i, i.e., the target point and the mutated point $\hat{x}_i$ determined by the mutation operation. In its mutation phase, DE randomly selects three distinct points $x_{p(1)}$, $x_{p(2)}$ and $x_{p(3)}$, with replacement, from the current set S. None of these points should coincide with the current target point x_i. The weighted difference of any two points is then added to the third point which can be mathematically described as

$$\hat{x}_i = x_{p(1)} + F(x_{p(2)} - x_{p(3)}), \tag{3}$$

where $F > 0$ is a scaling parameter, and $x_{p(1)}$ is known as the 'base vector'. If the point $\hat{x}_i \notin \Omega$ then the mutation operation is repeated. The trial point y_i is found from its parents x_i and $\hat{x}_i$ using the following crossover rule:

$$y_i^j = \begin{cases} \hat{x}_i^j & \text{if } R^j \leqslant C_R \text{ or } j = I_i, \\ x_i^j & \text{if } R^j > C_R \text{ and } j \neq I_i, \end{cases} \tag{4}$$

where I_i is an integer randomly chosen with replacement from the set I, i.e., $I_i \in I = \{1, 2, \ldots, n\}$; the superscript j represents the jth component of respective vectors; $R^j \in (0, 1)$, drawn uniformly for each j. The ultimate aim of the crossover rule (4) is to obtain the trial point y_i with components coming from the components of the target point x_i and mutated point $\hat{x}_i$. This is ensured by introducing C_R and the set I. Notice that for $C_R = 1$ the point y_i is the replica of the mutated point $\hat{x}_i$. The effect of C_R has been studied in [2,3] and it was found that $C_R = 0.5$ is a good choice. The targeting process continues until all members of S are considered. After all N trial points y_i have been generated, acceptance is applied. In the acceptance phase, the function value at the trial point, $f(y_i)$, is compared to $f(x_i)$, the value at the target point. If $f(y_i) < f(x_i)$ then y_i replaces x_i in S, otherwise, S retains the original x_i. Reproduction (mutation and crossover) and acceptance continue until some stopping conditions are met. It can be seen from (3) that mutation is the core point generation mechanism of DE. This operation calculates the coordinates of new points. The crossover operation (4) chooses the coordinates of a trial point from the known coordinates of two points using a distribution controlled by C_R.

An important issue that needs to be addressed is the value of the scaling parameter F in (3). To the best of our knowledge, no optimal choice of the scaling parameter F has been suggested in the literature of DE. In the original DE [1], F was chosen to lie in $(0, 2]$. Other empirical choices of F were values close to 0.8 [4] and to 1 [2]. In a recent study using 50 test problems the value 0.5 was found to be a good choice [3]. It appears that the choice of F depends upon the problem at hand. Indeed, we observed that a value of F which can be a good choice for a problem but a bad choice for a different problem in the set of 50 problems. Besides, our numerical experiments found that mutation often generates trial points outside the feasible region Ω and the number of points that fall outside Ω varies from problem to problem. We also observed that the larger the F, the higher the number of such points is. On the other hand, the smaller the F, the higher the probability of DE getting trapped in a local minimizer. The choice of F is therefore a delicate issue.

In this paper, we derive the probability density function of the mutated points and show how these points can fall outside Ω. We also visualize this phenomenon by simulation. We then propose an alternative approach that can dispense with the fixed choice of F. Firstly, we replace the fixed scaling parameter value with a variable one and secondly we introduce the preferential crossover rule. We also introduce an auxiliary set of points. These points are normally discarded in DE. The preferential crossover rule uses the auxiliary set in generating trial points within the feasible region.

This paper is divided into seven sections. The next section derives the probability density function (pdf) of the point generation using (3). In Section 4, we present the simulation of mutated points to visualize the effect of F. In Section 5, we present the new algorithm. Results are presented in Section 6 and conclusions are made in Section 7.

We denote the pdf of a random variable (RV), say X by f_X and the joint pdf for RVs, say X and Y by f_{XY}.

3. Probability density of trial points

In this section, we derive the pdf of mutated points generated by (3). We use $F = 1$ for this purpose. The motivation for this choice is as follows. If we take $F > 1$ then the differential vector $(x_{p(2)} - x_{p(3)})$ in (3)

is scaled up and if we take $F<1$ then the vector is scaled down. We take a compromise value, i.e., $F=1$ which neither scales up or scales down the differential vector (3). The pdf with this value will give us a feel of the pdfs with $F>1$ and $F<1$.

To see the effect of F in the pdf, it is enough to derive the pdf of a single coordinate of $\hat{x}_i$ in (3). Let the jth coordinate be our coordinate of interest, $j=1,2,\ldots,n$. Our derivation of the pdf is based on the uniform distribution of points. Initially, points in S are generated uniformly from Ω and therefore points in all coordinate directions are independently uniform. However, as the contraction process continues, points in S will not be uniform over Ω. We consider three stages of the contraction process of S in DE: the initial stage, the intermediate stage and the final stage. The initial stage is when S is uniform or near uniform in Ω. This may hold for only the initial few generations of DE. At this stage, the jth coordinate of points in S distributed uniformly in $[l^j,u^j]$. The intermediate stage is the stage when points in S will be non-uniform in Ω in that points will be distributed lower in the valley of the regions of attraction of different local minimizers. At the final stage, the points in S will be distributed within the region of attraction of the global minimizer. If the function f is well behaved in that it can be approximated well by a quadratic near a minimizer, then the regions of attraction of the minimizer will likely be of a elliptical or spherical shape. Under this assumption, it is reasonable to assume that the points in a particular region of attraction are locally uniform within the region. At the final stage, the jth coordinate of points in S are distributed uniformly in $[x_l^j,x_u^j]$, where $x_l^j(\geqslant l^j)$ and $x_u^j(\leqslant u^j)$ are defined by $x_l^j=\min\{x_i^j\}$ and $x_u^j=\max\{x_i^j\}$ for all x_i in S. Points in S are unlikely to be uniform in Ω at the intermediate stage. However, we assume that subsets of points in S are locally uniform within their respective region of attractions. Under the above considerations, we derive the pdf of the jth coordinate of $\hat{x}_i$. Without loss of generality, let the jth coordinate be defined on $[0,1]$. Let the random variables be X_1 and $X_2\sim U(0,1)$. We also define $Y_1=X_1$ and $Y_2=(X_2-X_1)$. Notice that $F=1$ is taken as the scaling parameter in (3). By defining $y_1=u_1(x_1,x_2)=x_1$ and $y_2=u_2(x_1,x_2)=(x_2-x_1)$, and by using the change of variable technique we can write $x_1=v_1(y_1,y_2)=y_1$ and $x_2=v_2(y_1,y_2)=y_1+y_2$. Here, the values x_i and y_i are, respectively, the realizations of the RVs X_i and Y_i, $i=1,2$. The joint pdf $f_{Y_1Y_2}$ is given by

$$\begin{aligned} f_{Y_1Y_2}(y_1,y_2) &= |J| f_{X_1X_2}(v_1(y_1,y_2),v_2(y_1,y_2)) \\ &= |J| f_{X_1}(v_1(y_1,y_2)) f_{X_2}(v_2(y_1,y_2)) \\ &= 1, \end{aligned} \tag{5}$$

where J is the determinant of the Jacobian matrix

$$\begin{pmatrix} \frac{\partial x_1}{\partial y_1} & \frac{\partial x_1}{\partial y_2} \\ \frac{\partial x_2}{\partial y_1} & \frac{\partial x_2}{\partial y_2} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}. \tag{6}$$

The supports (bounds) of y_1 and y_2 can be obtained from the supports of x_1 and x_2 (see Ref. [5]). By using the bounds of x_1 and x_2, the bounds of y_1 and y_2 can be seen to be $0\leqslant y_1\leqslant 1$ and $-1\leqslant y_2\leqslant 1$, respectively. The other support in the y_1–y_2 plane can be found using the relationships between y_1 and y_2. These relationships are $y_1+y_2=0$ and $y_1+y_2=1$, for $x_2=0$ and $x_2=1$, respectively. The marginal density $f_{Y_2}(y_2)$ can be calculated from the integrals:

$$f_{Y_2}(y_2)=\int_0^{1-y_2} \mathrm{d}y_1=(1-y_2), \quad 0\leqslant y_2\leqslant 1 \tag{7}$$

and

$$f_{Y_2}(y_2)=\int_{-y_2}^{1} \mathrm{d}y_1=(1+y_2), \quad -1\leqslant y_2\leqslant 0. \tag{8}$$

Combining (7) and (8) we write

$$f_{Y_2}(y_2)=(1-|y_2|), \quad -1\leqslant y_2\leqslant 1. \tag{9}$$

We now let $Y_3\sim U(0,1)$ independently of Y_2. We consider the joint pdf of Y_3 and Y_3+Y_2. We denote $Z_1=Y_3$ and $Z_2=Y_3+Y_2$, where $Y_2=(X_2-X_1)$ and $F=1$. The joint pdf $f_{Z_1Z_2}$ is given by

$$\begin{aligned} f_{Z_1Z_2}(z_1,z_2) &= |J| f_{Y_3Y_2}(v_1(z_1,z_2),v_2(z_1,z_2)) \\ &= |J| f_{Y_2}(y_2) f_{Y_3}(y_3) = f_{Y_2}(z_2-z_1), \end{aligned} \tag{10}$$

where $|J|=1$, $v_1(z_1,z_2)=y_3=z_1$ and $v_2(z_1,z_2)=y_2=z_2-z_1$. Therefore, the joint pdf $f_{Z_1Z_2}$ is given by

$$\begin{aligned} &f_{Z_1Z_2}(z_1,z_2)=(1-|z_2-z_1|), \\ &\quad 0\leqslant z_1\leqslant 1,\ z_1-1\leqslant z_2\leqslant z_1+1. \end{aligned} \tag{11}$$

The marginal pdf of Z_2 will be our desired pdf. Therefore,

$$f_{Z_2}(z_2)=\begin{cases} \int_0^{1+z_2} f_{Z_1Z_2}(z_1,z_2)\,\mathrm{d}z_1, & -1\leqslant z_2\leqslant 0, \\ \int_0^{1} f_{Z_1Z_2}(z_1,z_2)\,\mathrm{d}z_1, & 0\leqslant z_2\leqslant 1, \\ \int_{z_2-1}^{1} f_{Z_1Z_2}(z_1,z_2)\,\mathrm{d}z_1, & 1\leqslant z_2\leqslant 2, \end{cases} \tag{12}$$

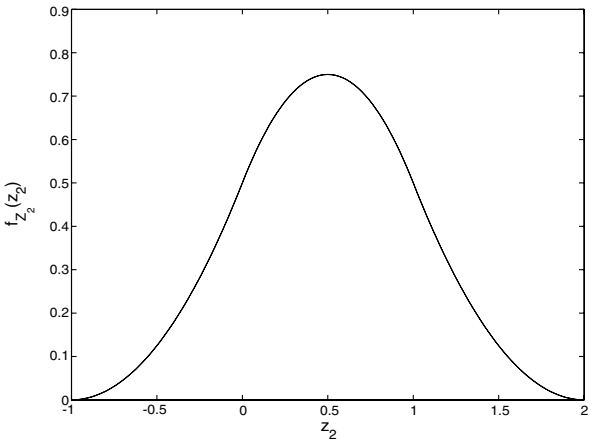

Fig. 1. Probability density function.

where $f_{Z_1Z_2}(z_1,\ z_2) = 1 - |z_2 - z_1|$. Integrating out we get

$$f_{Z_2}(z_2) = \begin{cases} \frac{1}{2} + z_2 + \frac{z_2^2}{2}, & -1 \leqslant z_2 \leqslant 0, \\ \frac{1}{2} + z_2 - z_2^2, & 0 \leqslant z_2 \leqslant 1, \\ 2 - 2z_2 + \frac{z_2^2}{2}, & 1 \leqslant z_2 \leqslant 2. \end{cases} \tag{13}$$

The pdf $f_{Z_2}(z_2)$ is given in Fig. 1. The pdf $f_{Z_2}(z_2)$ is proper in the sense that its integration results in unity. Intuitively speaking, the pdfs corresponding to values of $F > 1$ and $F < 1$ will be of similar shapes as F is a scalar quantity. The tails of the density will extend (shrink) and the height will shrink (extend) corresponding to $F > 1$ ($F < 1$) in order for the area under the pdf to be equal to one.

4. Effect of F in mutation

The scaling parameter F is used to scale the differential vector in the mutation (3). At the early stages scattered points in S result in large differential vectors $F(x_{p(2)} - x_{p(3)})$ in (3) which causes $\hat{x}_i$ to fall outside Ω. At the final stages when the points in S form cluster around the global minimizer, $\hat{x}_i$ will still fall outside Ω if the global minimizer lies close to the boundary of Ω. We face two difficulties. The first difficulty is in choosing a value of F that reduces the phenomenon of points falling outside, but at the same time maintains the exploratory feature of DE. Different empirical values are suggested for F in the literature. There is no rigorous way of choosing the value of F and the suggested values differ from one another due to the differences in the test problem set used in numerical testing.

No doubt, scaling down of the differential vector may prevent some points falling outside Ω but it has a negative impact on the success rate of obtaining the global minimum. On the other hand, scaling up raises the number of function evaluations, increases the cpu times, and causes trial points to fall outside Ω. We would like to see the pattern of the infeasible points when neither scaling up or down are used, i.e., when $F = 1$. We will use the pdf (13) as well as use simulation using $F = 1$. It can be seen that the integration of the pdf (13) in $[0, 1]$ is 2/3 and hence a third of the points fall outside $[0, 1]$. We now show the effect of points falling outside by simulation. For this we took two functions, namely PRD and SBT, each of dimension two. The functions are given in Appendix A. Both functions are defined on $\Omega = \{(x_1, x_2) \mid -10 \leqslant x_1, x_2 \leqslant 10\}$. The global minimizer of PRD is located at the center of Ω while 18 different global minimizers of SBT are distributed around Ω. The DE algorithm solved both problems in 160 generations with $F = 1$. We took S from the 1st, the 50th, the 100th and 150th generation to simulate the mutated points $\hat{x}_i$ with $F = 1$ in (3). The simulated points are presented in Figs. 2(a)–(d) and 3(a)–(d) for PRD and SBT, respectively. Fig. 2 shows that for the PRD function the number of

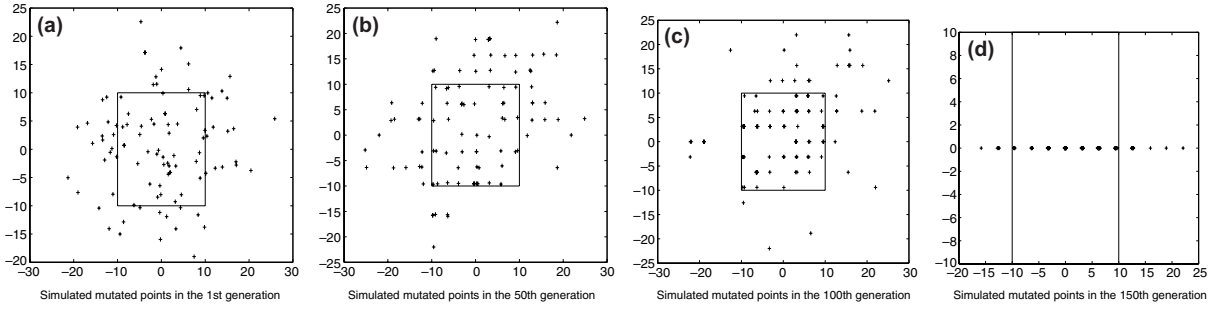

Fig. 2. One hundred mutated points by simulation per figure using PRD.

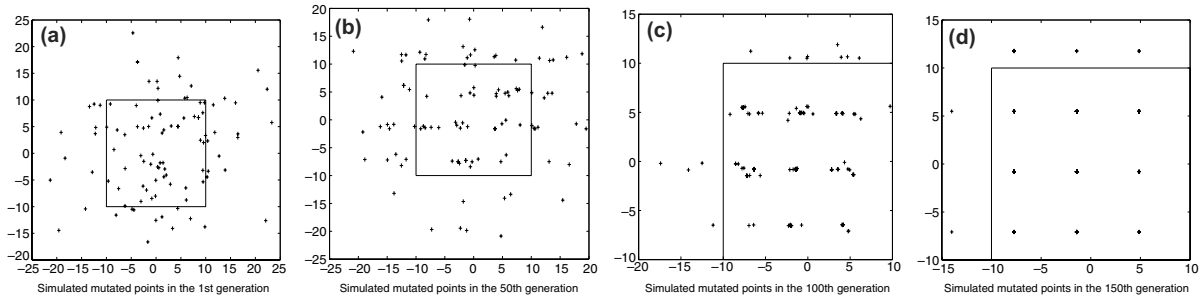

Fig. 3. One hundred mutated points by simulation per figure using SBT.

infeasible points is quite high at the first generation and continues to be high until the 50th generation of the DE algorithm. The number is less at the 100th generation. Although the global minimizer lies at the centre of the search region, it is noticeable that points may fall outside Ω at the final stage of the DE algorithm. It is interesting to note in Fig. 2(d) that the simulated points at the 150th generation of DE on PRD form a number of clusters along the x_1-axis. It is therefore clear that how the points in S will evolve is dependent on the topography of the function being optimized. Fig. 3 shows that the number of infeasible points gradually decreases with each generation. It is also interesting to note that although the SBT function has 18 global minimizer the simulated points show that nine clusters have been formed at the 100th generation of DE. These clusters become dense at the 150th generation. The DE algorithm however gradually drives the points from eight clusters to one cluster around a global minimizer at the final stages. Figs. 2(d) and 3(d) clearly show that the distribution of points in S depends on the function being optimized.

We note that the number of mutated point generated by (3) is halved because

$$\begin{aligned}\hat{x}_i &= x_{p(1)} + F(x_{p(2)} - x_{p(3)}) \\ &= x_{p(2)} + F(x_{p(1)} - x_{p(3)}),\end{aligned} \tag{14}$$

when $F = 1$. To see if the choice $F = 1$ has any influence in the special structure (i.e., creating repeated mutated points) in Fig. 2(d), we conducted a simulation using $F = 0.5$ on PRD. The simulated points are presented in Fig. 4. Fig. 4 clearly shows that the formation of the cluster along the line at the 150th generation is independent of the choice of F. This further ascertains the fact that the evolution of S is problem dependent.

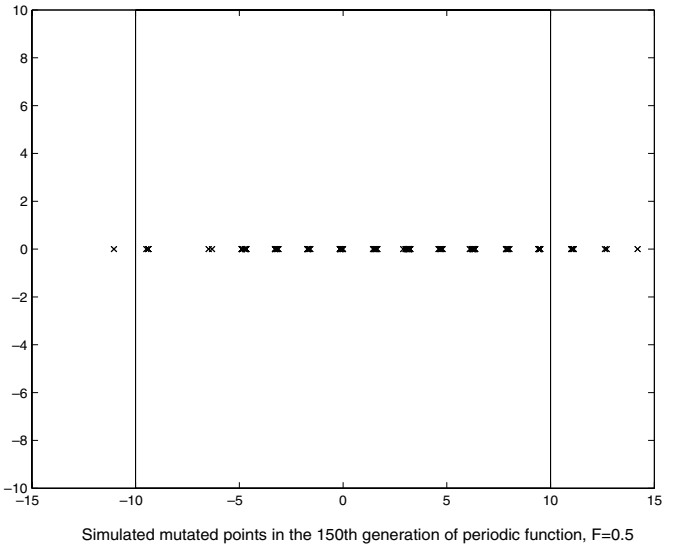

Fig. 4. One hundred mutated points at 150th generation of DE for PRD, $F = 0.5$.

The simulation of mutated points of DE can be shown using other problems as well but we have taken the above problems as representatives. The above simulations show that quite a large number of points are generated unnecessarily outside the search region. This phenomenon slows down the DE algorithm. Next we propose an algorithm that generate trial points using a crossover rule, namely the preferential crossover, as an alternative to (3) and (4). This crossover rule reduces the infeasible points considerably and thereby making DE faster.

5. DE with preferential crossover (DEPC)

In this section, we present a modified DE algorithm, namely the DEPC algorithm. Before presenting the DEPC algorithm, we differentiate between DE and DEPC. There are three differences between DE and DEPC. The first difference between the two algorithms is the way they implement the scaling parameter F. DEPC uses a variable scaling parameter F for each mutated point as opposed to DE which uses a fixed F. The DEPC algorithm uses F_i as a random variable in $[-1,-0.4]\cup[0.4,1]$ for each targeted point x_i. The use of variable F is reported in [3]. The second difference is that DEPC uses two population sets, namely the sets S_1 and S_2 as opposed to the set S in DE. Each of the sets contains N points. We call S_2 the auxiliary population set. The function of S_1 in DEPC is the same as that of S in DE. The function of the auxiliary set S_2 in DEPC is to keep record of the trial points that are discarded in DE. Potential trial points in S_2 are then used for further explorations. The concept of two population sets was first introduced in a DE driven multi-start algorithm [6] for solving large dimensional problem. Thirdly, DEPC differs from DE in that it is now equipped with a new crossover rule, namely the preferential crossover, that always generates feasible trial points.

5.1. Motivation of the changes made to DE

To motivate the first change made to DE, we look at the total number of equiprobable mutated points associated with DE. We note that the replacement of points in DE takes place only after an generation has been completed. DE uses a fixed F in (3), say $F \neq 1$. Hence, it is easy to see that the total number of mutated points in any iteration of DE is given by

$$3! \times ({}^{N-1}C_3 + {}^{N-2}C_{3-1} + {}^{N-3}C_{3-2} + {}^{N-4}C_{3-3}). \quad (15)$$

When the first point $x_1 \in S$ is targeted, three distinct points (excluding x_1) can be selected from $N-1$ points in ${}^{N-1}C_3$ ways. However, these three distinct points, say $x_{p(1)}$, $x_{p(2)}$ and $x_{p(3)}$ can arranged themselves in the relation (3) in 3! different ways. Therefore, the total number of equiprobable mutated points associated with the targeted point x_1 is $3! \times {}^{N-1}C_3$. For the second targeted point $x_2 \in S$, while the total number of mutated points is the same, i.e., $3! \times {}^{N-1}C_3$, there are only $3! \times {}^{N-2}C_{3-1}$ new mutated points. The remaining are repetitions. The new combinations can be seen as the combinations of three points that contain the point x_1 (x_1 was excluded when it was targeted) but exclude the point x_2. Therefore, the total number of distinct new combinations of three points can be seen as the number of combinations that can be obtained from choosing two points from $N-2$ points, i.e., ${}^{N-2}C_{3-1}$. Hence, the total number of new equiprobable mutated point is $3! \times {}^{N-2}C_{3-1}$, when x_2 is targeted. Similarly, when x_3 is targeted, the new combinations are the ones that must contain x_1 and x_2, i.e., there are ${}^{N-3}C_{3-2}$ of them. Finally, when x_4 is targeted, there is only one (${}^{N-4}C_{3-3}$) new combination, i.e., the combination consists of x_1, x_2 and x_3. For all the subsequent targets, there will not be any new combinations but the repetitions of the combinations corresponding to the first four target points. It is therefore, clear that there is a possibility that the same mutated point (which has been calculated corresponding to a target point) will be re-calculated for a different target point. There is also a possibility that the very same mutated point will be re-calculated again in the next generation, although the possibility will gradually be diminished as the points being replaced per generation. To overcome this shortcoming, we choose a different scaling parameter value for each target point in S. This gives a different mutated point regardless of which combination of three points being selected. Therefore, the use of a variable F is fully justified.

We now motivate the second change made to DE, i.e., the use of two population sets S_1 and S_2 in DEPC as opposed to S in DE. The set S_1 in DEPC is similar to the set S in DE. The set S_2 in DEPC is an information bearing population set. In DE, a trial point is discarded if the point is not better than the target point. The fact that the discarded point (if not better than the target point) may be better than some other points in S and may even lie in a promising region of the search space is ignored in DE. These points are explored

further in DEPC by storing them in the auxiliary set S_2.

Finally, purposes of the preferential crossover rule are (i) to make the DEPC faster by generating feasible trial points and (ii) to make use of the potentials points stored in the set S_2.

We will now explain in detail how DEPC implements these changes.

5.2. Description of the DEPC algorithm

Initially, two sets S_1 and S_2 are generated in the following way; iteratively sample two points from the search region Ω, the best point x_i going to S_1 and the other x_i' to S_2. The process continues until each set has N points. The DEPC procedure then gradually drives both the sets S_1 and S_2 towards the global minimizer through the repeated cycles of preferential crossover, acceptance and replacement. The acceptance is invoked after N trial points are generated by the preferential crossover rule. If at the acceptance phase the ith target point x_i in S_1 is not replaced by the trial point y_i generated by the preferential crossover rule then a new trial point $\tilde{y}_i$ is generated using the mutation (17) and crossover (4) of DE. The trial point $\tilde{y}_i$ then competes with the target x_i in the usual way as in DE. Therefore, the preferential crossover, the mutation and the 'regular' crossover are used in DEPC as opposed to the mutation and the 'regular' crossover in DE.

In each generation at least N competitions are held to determine the members of S_1 and S_2 for the next generation (exactly N competitions are held in DE). The ith ($i = 1, 2, \ldots, N$) competition is held to replace x_i in S_1. At the acceptance phase, two attempts may be made for each targeted point. The first attempt is to replace x_i using the trial point y_i generated by the preferential crossover rule. This attempt is preferential in that the second attempt is made only if the first attempt is unsuccessful in replacing the targeted point in S_1. Therefore, whenever the first attempt fails the second attempt is made.

The trial point y_i associated with the first attempt is obtained in the following way. Considering x_i as the target point a trial point y_i is found from two points (parents), the point x_i in S_1, i.e., the target point and the point a_i, where a_i is a uniform random point (with replacement) in the auxiliary set S_2. The point y_i is obtained from x_i and a_i by preferential crossover defined as follows:

$$y_i^j = \begin{cases} a_i^j & \text{if } R^j \leqslant C_R \text{ or } j = I_i, \\ x_i^j & \text{if } R^j > C_R \text{ and } j \neq I_i. \end{cases} \tag{16}$$

Notice that the preferential crossover operation is the same as in (4) except for a_i is chosen from S_2 with replacement. The objective of the preferential crossover rule is to obtain y_i with components coming from the components of the target point x_i in S_1 and the random point a_i in S_2. At the acceptance phase, if the trial point y_i, corresponding to the target x_i, satisfies the criterion $f(y_i) < f(x_i)$ then the point y_i replaces x_i in S_1, if however, y_i does not satisfy the above criterion then the second attempt is made to obtain another trial point $\tilde{y}_i$. The trial point $\tilde{y}_i$ is found from its parents x_i and $\hat{x}_i$ using the crossover operation (4). The point $\hat{x}_i$ is obtained using

$$\hat{x}_i = x_{p(1)} + F_i(x_{p(2)} - x_{p(3)}), \tag{17}$$

where $x_{p(1)}$, $x_{p(2)}$ and $x_{p(3)}$ are uniform random points in S_1 and F_i is a random variable in $[-1, -0.4] \cup [0.4, 1]$ for each i. Notice that (3) and (17) are the same except a variable F_i is used in (17). The second attempt therefore consists of mutation and crossover of the original DE. Generation of $\tilde{y}_i$ and its acceptance follow the acceptance phase of preferential crossover. The acceptance of $\tilde{y}_i$ is checked in the following way: if the trial point $\tilde{y}_i$, corresponding to the target x_i, satisfies the criterion $f(\tilde{y}_i) < f(x_i)$ then the point $\tilde{y}_i$ replaces x_i in S_1, if however, $\tilde{y}_i$ does not satisfy the above criterion then it is not abandoned altogether, rather it competes with its corresponding target x_i' in the set S_2. If $f(\tilde{y}_i) < f(x_i')$ then $\tilde{y}_i$ replaces x_i' in S_2. Notice that points in S_2 are replaced in the second attempt only. The DEPC procedure continue until some stopping conditions are met. The full detail of the DEPC algorithm is now presented below.

The DEPC Algorithm

Step 1. Determine the initial sets. Take $N \gg n$, n being the dimension of the function $f(x)$. Set generation counter $k = 0$. Two points are sampled randomly at a time in Ω. The best point x_i going to S_1 and the other x_i' to S_2. Continue until each set has N points.

Step 2. Determine best, worst point in S_1. If the stopping condition such as $|f_{\max} - f_{\min}| \leqslant \epsilon$ is satisfied, then stop.

Step 3. Generate points to replace points in S_1 and S_2 for the next generation. For each $x_i \in S_1$ ($i = 1, 2, \ldots, N$), determine y_i by the following operation:

Preferential crossover: Calculate the trial point y_i corresponding to the target x_i from x_i and a_i using the crossover rule (16). The point a_i is selected uniformly from S_2.

3a. *Acceptance rule to replace points in* S_1. For each $y_i (i = 1, 2, \ldots, N)$, if $f(y_i) < f(x_i)$ then replace x_i in S_1 with y_i; otherwise perform Steps 3b–3d. If $i = N$ then set $k := k + 1$ and go to Step 2.

3b. *Mutation*. Obtain $\hat{x}_i$ using (17). If $\hat{x}_i \notin \Omega$ then the process is repeated.

3c. *Crossover*. Calculate the trial point $\tilde{y}_i$ corresponding to the target x_i from x_i and $\hat{x}_i$ using the crossover rule (4).

3d. *Acceptance rule to replace points in* S_1 *and* S_2. Replace $x_i \in S_1$ with $\tilde{y}_i$ if $f(\tilde{y}_i) < f(x_i)$ otherwise replace $x_i' \in S_2$ with $\tilde{y}_i$ if $f(\tilde{y}_i) < f(x_i')$. If $i = N$ then set $k := k + 1$ and go to Step 2.

Remarks

1. The points $x_{\max}$ and $x_{\min}$ and their function values $f_{\max}$, $f_{\min}$ are such that $f(x_{\max}) = f_{\max} = \max_{x \in S_1} f(x)$ and $f(x_{\min}) = f_{\min} = \min_{x \in S_1} f(x)$.
2. The mutation is the only operation which calculates the coordinates of new points. The auxiliary population set S_2 is updated in Step 3d. We do not update S_2 in Step 3a. Updating in Step 3a may lead to repetition of trial points in preferential crossover.

6. Numerical results

In this section we judge the performance of the new algorithm using a collection of 50 test problems. These problems range from 2 to 20 in dimension and have a variety of inherent difficulty [7,8]. All the problems have continuous variables. A detailed description of each test problem (P) in the collection can be found in [7]. We compare DEPC with DE and DERL. The DERL is a modified version of DE recently proposed in [3]. The algorithms were run 100 times on each of the 50 test problems to determine the success rate (sr) (or percentages of success) of each algorithm. There were 5000 runs in total. We calculated the average number of function evaluations (fe) and cpu times (cpu) for those runs for which the global minima were found. We used sr, fe and cpu as the criteria for comparison. In every case, a run was terminated when the function values of all points in S_1 (S for DE) were identical to an accuracy of four decimal places, i.e.,

$$|f_{\max} - f_{\min}| \leqslant \epsilon = 10^{-4}. \tag{18}$$

A success was counted when the value $f_{\min}$ of a run was such that $f_{\min} - f^* \leqslant 0.009$ where f^* is the known global minimum of the problem being solved. A solution to the problem therefore need not be the global minimum f^* exactly, but may be any value less than or equal to $f^* + 0.009$.

6.1. Parameter selection

The algorithms have some parameter values that are to be provided by the user. We first discuss the parameters that are common to all algorithms. For example, the size N of the population sets S, S_1 and S_2. We took the value of N to be $10n$ where n is the dimension of the problem. This is a heuristic choice. For example the value of N can always be increased for obtaining the global minimum with higher probability. However, the higher the value of N, the higher the number of fe is. Another common parameter is the controlling parameter C_R in the crossover schemes (4) and (16). A parameter of DE is the scaling parameter F in its mutation scheme (3). The values of F and C_R for DE were found empirically using the 50 test problems [9]. The best results obtained by DE using the 50 problems were for $F = 0.5$ and $C_R = 0.5$. The scaling parameter F_i in the mutation (17) of DEPC is a random variable and hence does not require any specific value to be assigned. For example, when x_i is targeted, $\hat{x}_i$ is calculated by choosing F_i randomly from one of the intervals $[-1, -0.4]$ or $[0.4, 1]$. A random number is generated in $(0, 1)$ and if this number is less than 0.5 then we choose F_i randomly in $[-1, -0.4]$, otherwise from $[0.4, 1]$. Therefore, F_i is different for each x_i. DERL also uses the random F_i in (3). The results of DEPC were found to be sensitive to C_R in (16), especially with respect to fe. For some problems a value less than 0.5 worked well while for others a value greater than 0.5 was suitable. Therefore, we used a compromised value of 0.5 for crossover parameter C_R in (16). We do not claim these values to be the optimal for any given problem, in general, but they are good values to choose.

6.2. Comparisons of algorithms

We note that none of the algorithms succeeded in finding the global minimum for two 9-dimensional functions, namely Storn's Tchebychev (ST) and Price's transistor modeling (PTM) and a 10-dimensional function, e.g., Salomon problem (SAL). Except for these functions, all other functions were solved by at least one of the algorithms. Therefore, results for these three functions are not reflected in our presentation. There are however two 10-dimensional functions, namely the Rosenbrock problem

Table 1
Comparison of DEPC, DERL and DE using 47 test problems

$P(n)$	DE		DERL				DEPC			
	fe	ps	fe	sr	fo	cpu	fe	sr	fo	cpu
ACK(10)	26,112	100	21,983	100	728	0.75	29,825	100	318	0.80
AP(2)	676	100	667	100	13	0.01	800	100	4	0.03
BL(2)	970	100	765	100	256	0.01	921	100	88	0.04
B1(2)	996	100	839	100	13	0.03	1137	100	3	0.04
B2(2)	1088	100	883	100	14	0.03	1216	100	3	0.04
BR(2)	1155	100	821	100	344	0.03	1043	100	52	0.03
CB3(2)	872	100	717	100	13	0.04	810	100	4	0.04
CB6(2)	1055	100	766	100	12	0.04	911	100	4	0.04
CM(2)	2219	100	1989	100	13	0.05	2403	100	24	0.05
DA(2)	1185	100	1291	100	213	0.04	1529	100	62	0.04
EP(2)	678	95	567	80	31	0.02	923	78	21	0.02
EM(10)	248,169	73	184,672	51	98,854	8.83	54,872	35	8622	1.86
EXP(10)	9955	100	9422	100	667	0.26	11,238	100	251	0.21
GP(2)	923	100	772	100	22	0.03	1034	100	9	0.04
GW(10)	14,886	100	15,231	100	687	0.67	47,963	100	285	0.75
GRP(3)	2893	96	2407	100	234	0.04	3328	100	144	0.06
H3(3)	1262	100	1032	100	1104	0.03	1354	100	67	0.03
H6(3)	6583	97	4944	97	1536	0.21	7181	96	720	0.17
HV(3)	3942	98	2877	100	43	0.05	5327	100	24	0.07
HSK(2)	554	100	553	100	22	0.02	718	100	8	0.03
KL(2)	1804	100	1498	100	342	0.02	1832	100	209	0.04
LM1(3)	1338	100	1306	100	54	0.04	1652	100	30	0.04
LM2(10)	12,021	100	9568	100	1878	0.28	13,732	100	379	0.19
MC(2)	622	100	5481	100	37	0.02	863	100	17	0.03
MR(3)	3177	74	2521	100	74	0.06	2351	100	108	0.06
MCP(4)	3592	100	2214	100	260	0.09	2153	100	98	0.06
ML(10)	197,336	70	91,326	58	77,382	3.43	118,617	100	14,294	1.04
MRP(2)	1681	64	1005	100	65	0.03	1576	100	4	0.04
MGP(2)	984	68	821	61	15	0.02	1466	66	7	0.04
NF2(4)	82,184	100	43,183	95	3032	1.07	27,286	100	892	0.64
NF3(10)	83,783	100	63,529	100	1672	1.72	85,672	100	725	1.21
OSP(10)	0	0	0	0	0	0.00	1,336,570	4	2954	5.20
PP(10)	14,559	100	11,983	100	2633	0.36	15,818	100	1225	0.25
PRD(2)	1957	90	1284	79	273	0.03	1487	87	136	0.04
PQ(4)	5526	100	4169	100	196	0.10	6842	100	76	0.08
RG(10)	96,839	100	96,428	100	1479	2.11	26,927	100	583	0.42
RB(10)	0	0	198,584	78	6184	7.28	512,165	100	484	10.52
SF1(2)	3204	63	3179	41	34	0.05	2315	100	17	0.04
SF2(2)	1954	100	1735	100	12	0.03	2604	100	3	0.05
SBT(2)	2438	100	2052	96	567	0.05	1955	89	145	0.05
SWF(10)	29,233	100	21,738	100	16,738	0.86	24,046	100	2563	0.37
S5(4)	5843	95	4161	93	172	0.10	6206	100	126	0.09
S7(4)	4952	100	3582	99	134	0.10	5127	98	64	0.09
S10(4)	4859	100	3592	100	155	0.10	5223	100	78	0.09
FX(5)	0	0	0	0	0	0.00	12,768	18	1021	0.19
SIN(20)	62,658	100	49,375	100	14,281	3.42	64,285	100	4471	1.85
WP(4)	15,438	96	11,627	96	143	0.17	19,887	100	72	0.24
Total	964,155	4179	690,555	4224	226,447	25.45	614,455	4371	37,035	11.44

(RB) for which DE failed and Odd Square problem (OSP) for which DEPC succeeded but both DERL and DE failed. Shekel's Foxhole (FX) is another function where only successful algorithm was DEPC. We present the results for these three problems in Table 1. In Table 1, the first column contains the problem name [7] with its dimension given in brackets; the data under the column fo represents the average number of points per successful run that fell outside the search region. The last row in Table 1 presents the total results (total of averages of fe, fo and cpu) excluding the results for OSP, RB and FX. The total is therefore taken over all functions where all algorithms succeeded in obtaining the global minimum value. The cpu for DE is always worse than the other two and the number of fo for DE is more or less similar to that of DERL, hence these results for DE are not presented in Table 1.

Total results in Table 1 show that DEPC is superior to DERL while DERL is superior to DE. Hence, the new algorithm, DEPC, is much superior to DE. We now compare DEPC with DERL. A comparison using the total results shows that DEPC is superior to DERL by 11%, 55% and 87% in terms of fe, cpu and fo, respectively. It is therefore clear that DEPC was able to reduce the infeasible trial points of DERL by 87% resulting in much lower cpu time for DEPC. In terms of sr, DEPC secured 147 more successes than DERL. Therefore, the total results establish the superiority of DEPC over the other two. It can be seen from Table 1 that for the RB problem DEPC was inferior to DERL both in terms of fe and cpu. However, the solutions of the RB problem obtained by DEPC were identical to the optimal solution to at least four decimal digits. On the other hand, the quality of the optimal solutions of RB obtained by DERL in many runs were inferior to those obtained by DEPC. We observed that the solution of RB obtained by DE was identical to the optimal solution to one decimal digit in each of the 100 independent runs.

7. Conclusion

In this paper we have identified a number of drawbacks of the DE algorithm and suggested some modifications in order to reduce these drawbacks. The modified version was tested on a large set of problems. Numerical results have shown that the new version is considerably better than its previous versions in terms of the number of function evaluations, success rate and cpu time. Another advantage of the new algorithm, DEPC, is that it reduces the number of infeasible trial points considerably.

The direct search type methods such as the DE methods have been designed to solve optimization problems that are non-differentiable and noisy, or they have no exactly known mathematical expressions. These types of problems arise naturally in many practical applications where the function values are dependent on simulation. These functions are very expensive to evaluate. Therefore, the use of the new algorithm is totally justified as it requires much less function evaluations than the previous versions of DE.

Further research is underway in developing more efficient hybrid DE global optimization method for large dimensional problems.

Acknowledgement

The author thanks Professor L.P. Fatti of the School of Statistics and Acturial Science, University of the Witwatersrand, Johannesburg, for his help in deriving the probability density function of the mutated point.

Appendix A. Problems PRD and SBT

A.1. Periodic problem (PRD)

$$\min_x f(x) = 1 + \sin^2 x_1 + \sin^2 x_2 - 0.1\exp(-x_1^2 - x_2^2)$$

$$\text{s.t.} -10 \leqslant x_1, \quad x_2 \leqslant 10.$$

There are 49 local minima all with minimum values 1 and global minimum located at $x^* = (0,0)$ with $f(x^*) = 0.9$.

A.2. Schubert problem (SBT)

$$\min_x f(x) = \prod_{i=1}^{n}\left(\sum_{j=1}^{5} j\cos((j+1)x_i + j)\right)$$

$$\text{s.t.} -10 \leqslant x_i \leqslant 10, \quad i \in \{1,2,\ldots,n\}.$$

Our tests were performed with $n = 2$. The number of local minima for this problem (given n) is not known but for $n = 2$, the function has 760 local minima, 18 of which are global with $f(x^*) \approx -186.7309$. All 2-dimensional global minimizers are listed in Table 2.

Table 2
Global optimizers for Schubert problem

x^*				
(−7.0835, 4.8580),	(−7.0835, −7.7083),	(−1.4251, −7.0835),	(5.4828, 4.8580),	(−1.4251, −0.8003),
(4.8580, 5.4828),	(−7.7083, −7.0835),	(−7.0835, −1.4251),	(−7.7083, −0.8003),	(−7.7083, 5. 4828),
(−0.8003, −7.7083),	(−0.8003, −1.4251),	(−0.8003, 4.8580),	(−1.4251, 5.4828),	(5.4828, −7.7083),
(4.8580, −7.0835),	(5.4828, −1.4251),	(4.8580, −0.8003),		

References

[1] R. Storn, K. Price, Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces, Journal of Global Optimization 11 (1997) 341–359.

[2] M.M. Ali, A. Törn, Population set based global optimization algorithms: Some modifications and numerical studies, Computers and Operations Research 31 (10) (2004) 1703–1725.

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

[4] K. Price, An introduction to differential evolution, in: David Corne, Marco Dorigo, Fred Glover (Eds.), New Ideas in Optimization, McGraw-Hill, London, 1999, pp. 79–108.

[5] R.V. Hogg, E.A. Tanis, Probability and Statistical Inference, fifth ed., Prentice Hall, New Jersey, 1997.

[6] M.M. Ali, A. Törn, Topographical differential evolution using pre-calculated differential, in: G. Dzemyda, V. Saltenis, A. Zilinskas (Eds.), Stochastic and Global Optimization, Kluwer Academic Publisher, Dordrecht, 2002, pp. 1–17.

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

[8] L. Özdamar, M. Demirhan, Experiments with new stochastic global optimization search techniques, Computers and Operations Research 27 (9) (2000) 841–865.

[9] P. Kaelo, Some population set based methods for unconstrained global optimization, Ph.D. thesis, School of Computational and Applied Mathematics, Johannesburg, 2005.